BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Master of Software Engineering

THESIS

Identifying Non-Functional Requirements from Mobile Apps User's Reviews using Deep Learning

---

Author: Nasry Alladaa (1185270)

Supervisor: Dr. AbuAlsoud Hanani

November 19, 2023

BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Master of Software Engineering

Identifying Non-Functional Requirements from Mobile Apps User's Reviews using Deep Learning

تحديد المعلومات بالمتطلبات الغير وظيفية من مراجعات مستخدمي تطبيقات الهاتف المحمول باستخدام التعلم العميق

Committee:

Dr. Abualsoud Hanani , Birzeit University.

Dr. Ahmad Abusnaina , Birzeit University.

Dr. Aziz Qaroush, Birzeit University.

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

February 13, 2024

Identifying Non-Functional Requirements from Mobile Apps User's Reviews using Deep Learning

---

Thesis

Author : Nasry Alladaa

Approved by the thesis committee:

Dr. Abualsoud Hanani : (Chairman of the Committee)

Dr. Ahmed Abusnaina : (Member) (Member)

Dr. Aziz Qaroush : (Member) (Member)

Date of Defense:

February 13, 2024

Declaration of Authorship

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

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

Abstract

Nowadays, smartphones are so essential without them life seems impossible. The number of useful information carried by users reviews every day is huge. The traditional way of extracting requirement-related information from a huge number of reviews is not possible. Hence, Natural Language Processing (NLP) and Machine Learning (ML) techniques were used to solve this problem and make it easier for the requirement engineer to benefit from the large amount of user reviews.

In this thesis, some of the state-of-the-art techniques in NLP were applied for automatically classifying user's reviews into requirement-related information such as usability, reliability, performance, security, functional requirements, and others. Some of the word embedding vectorization methods such as GPT3, Word2Vec, GloVe, FastText, BERT, Dbert were used to represent the user's text reviews. The resulting feature vectors were used to extract the target software requirement information with some of the traditional machine learning classifiers, such as Random Forest, Naïve Bayes, and Artificial Neural Network (ANN) and some deep learning classifiers such as Deep Neural Network (DNN), and Convolutional Neural Network (CNN).

In order to build and test the proposed systems, a dataset collected and used in a previous study conducted by Al-Kilani was used. This dataset consists of 1061 annotated user's reviews for apps in the healthcare domain. To make the dataset larger and wider, another 1693 reviews were collected and manually annotated form apps in different domains.

A set of experiments were conducted using Al-Kilani dataset and the extended dataset, The extended dataset includes Al-Kilani dataset and 1693 newly

collected reviews. To make the results comparable with the previous results achieved by Al-Kilani, some of the experiments were conducted to recognize three major classes (reliability, usability, and performance), and the others are designed to recognize six classes (reliability, usability, performance, security, functional requirements, and others).

For the three classes experiments, the results show that the deep learning-based techniques (SL CNN, ML CNN, DNN) outperform the traditional techniques. The best accuracy achieved by the Random Forest classifier trained and evaluated on the TF-IDF features is 58%, compared with 82% achieved by the DNN classifier trained on GPT3. Similarly, the deep learning techniques outperform the traditional techniques in the six classes experiments, with the best accuracy of 56% compared with 49% achieved by the random forest.

المستخلص

في هذه الأيام، أصبحت الهواتف الذكية مهمة للغاية وتبدو الحياة بدونها مستحيلة. عدد التعليقات والمراجعات لتطبيقات الهاتف المحمول التي ينتجها المستخدمون يوميًا ضخم. تحمل هذه المراجعات معلومات مفيدة عن هندسة المتطلبات لتحسين جودة التطبيقات. الطريقة التقليدية لاستخراج المعلومات المتعلقة بالمتطلبات من عدد كبير من المراجعات غير ممكنة. لتجاوز هذه المشكلة تم استخدام تقنيات حديثة اذ وهي استخدام مفهوم معالجة اللغات الطبيعية (NLP)والتعلم الآلي (ML)الذي أدى إلى تسهيل استفادة مهندس المتطلبات من هذا الكم الهائل من مراجعات المستخدمين.

في هذه الأطروحة، تم تطبيق بعض التقنيات الحديثة في معالجة اللغات الطبيعية (NLP) لتصنيف مراجعات المستخدم بشكل أتوماتيكي إلى معلومات ذات صلة بالمتطلبات مثل سهولة استخدام التطبيق، الموثوقية، أداء التطبيق، درجة الأمان أثناء استخدام التطبيق، إضافة إلى المتطلبات الوظيفية للتطبيق وغيرها. تم استخدام بعض أساليب توجيه تضمين الكلمات مثل GPT3 وWord2Vec وGloVe وFastText وBERT وDbert لإعادة تمثيل وزيادة العبارات المشابه (متجهات) لمراجعات التطبيقات الخاصة بالمستخدم. تم استخدام هذه المتجهات الناتجة لاستخراج معلومات متطلبات البرنامج المستهدف باستخدام بعض مصنفات التعلم الآلي التقليدية مثل Random Forest وNaïve Bayes واستخدام مصنفات التعلم الآلي الغير تقليدي مثل ANN, DNN, CNN.

من أجل بناء واختبار الأنظمة المقترحة، تم استخدام مجموعة البيانات التي تم جمعها واستخدامها في دراسة سابقة أجراها الكيلاني. تتكون مجموعة البيانات هذه من 1061 مراجعة مستخدم مصنفة لتطبيقات في مجال الرعاية الصحية. ولجعل مجموعة هذه البيانات أكبر وأوسع، تم جمع 1693 مراجعة أخرى من تطبيقات مختلفة في مجالات مختلفة وتصنيفها يدويًا.

تم إجراء مجموعة من التجارب باستخدام مجموعة بيانات الكيلاني والبيانات التي تم جمعها والتي تشمل مجموعة بيانات الكيلاني و1693 مراجعة. ولجعل النتائج قابلة للمقارنة مع النتائج السابقة التي حققها الكيلاني، أجريت بعض التجارب للتعرف على ثلاث فئات رئيسية (الاعتمادية، وسهولة الاستخدام، والأداء)، والبعض الآخر صممت للتعرف على ستة فئات (الاعتمادية، وسهولة الاستخدام، والأداء والأمن والمتطلبات الوظيفية وغيرها).

بالنسبة لتجارب الفئات الثلاثة، أظهرت النتائج أن التقنيات القائمة على التعلم العميق (SLCNN, ML CNN, DNN) تتفوق على التقنيات التقليدية. أفضل دقة حققها مصنف Random Forest الذي تم تدريبه وتقييمه على ميزات TF-IDF هي 58%، مقارنة بنسبة 82% التي حققها مصنف DNN المدرب على .GPT3وبالمثل، تتفوق تقنيات التعلم العميق على التقنيات التقليدية في تجارب الفئات الستة، بأفضل دقة بلغت 56% مقارنة بـ 49% التي حققتها Random Forest .

Acknowledgements

Firstly, I would like to acknowledge and give my warmest thanks to my supervisor Dr. Abualsoud Hanani who made this work possible. His guidance, continuous support, and advice carried me through all stages of writing this thesis. In addition, I would like to thank the faculty at my wonderful and beloved university, Birzeit University.

My sincere thanks also go to my colleagues who have been with me through this special experience. Appreciation and acknowledgment for the volunteers who assisted me during the manual classification process in this thesis.

I must give special thanks to all my managers in my previous company Jawwal and to my current company AXSOS Academy for providing support and bringing the weight of their considerable experience and knowledge to this thesis.

Finally, I must express my deep gratitude to my family and friends for providing me with unfailing support and continuous encouragement throughout this work. This accomplishment would not have been possible without you. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Over the last few years, with the rapid spread of smartphones in worldwide markets, mobile devices have become an essential part of our daily lives. This caused exponential growth in mobile application development with a wide set of diverse requirements. A good understanding of these new computer capabilities has gained a lot of attention in the software engineering domain to keep existing users and attract new users by satisfying their needs. There are many approaches to eliciting users' needs in the requirement elicitation phase before starting development. In addition, we can extract requirements the functional and non-functional requirements in other ways after launching the app through user reviews. The review of mobile applications is considered an important resource of information for both developers and business owners. Software developers can early fix bugs, extract new features, or enhance existing features for the new release of mobile apps, so that we can keep evolving all the time. Some of the mobile apps have more than thousands of reviews posted on the different app stores, such as Google Play Store for Android users or App Store for IOS

users.

Extracting requirement-related information (functional, non-functional, and others) from the huge amount of user reviews automatically has been investigated in the previous studies. Various traditional techniques in the Natural language Processing (NLP), such as Bag of Words (BoW) and Term-Frequency – Inverse Document Frequency (TF-IDF), and Machine Learning (ML), such as Random Forest (RF), and Naïve Bayes (NB), are successfully used for identifying some types of the functional and non-functional requirements from the user reviews. However, state-of-the-art techniques based on deep learning technologies have not been investigated for this task. Deep learning techniques include word embeddings, which are based on the pre-trained models (e.g. Word2Vec, BERT, D-BERT, GloVe, GPT3, FastText, etc), and deep neural networks classifiers such as Convolutional Neural Network (CNN).

1.2 Problem statement

Nowadays and in the era of communication and information technology, many of the traditional manual methods for collecting information have been changed. For example, the traditional methods for getting the software requirements from the users have been significantly changed. The number of software app users has increased dramatically across the globe. On the other hand, the quantity of information produced and published on the internet has increased exponentially. Hence, inspecting for useful information related to the requirement engineering using traditional methods becomes impossible. Therefore, using NLP and machine learning techniques is the solution. In this thesis, some of the most recent techniques in the NLP and machine learning models are used to recognize the requirement-related information from the app's reviews. Given the text

review, the system cleans it before computing representative numerical features, which will be fed later to the machine learning classifier that predicts the type of requirement-related information that exists in the given review.

1.3 Objectives

The main objectives of this study can be summarized in the following three directions:

1. Extend Al-Kilani dataset to include more user reviews of more mobile applications from different domains such as education, telecommunication, banking, financial services, diet and nutrition, etc. This helps in two directions: increasing the dataset allows studying the effectiveness of the proposed techniques for recognizing more requirements, and applying deep learning techniques that require a huge dataset.

2. Apply the effectiveness of the state-of-the-art techniques in the NLP, such as Word2Vec, D-bert, BERT, Glove, and GPT3, for recognizing software requirements from the mobile apps user reviews and compare them with the traditional ones included in the reference study [19].

3. Apply traditional in machine learning (Random Forest, Naive Bayes) for classifying user reviews into the specified requirements classes.

4. Apply deep learning techniques and more specifically deep neural networks and convolutional neural networks.

1.4 Research Questions

By the end of this thesis, we intend to present answers to the following possible research questions:

RQ1: How well the word embedding pre-trained models can improve the accuracy of identifying requirement-related information from the mobile apps user reviews, compared with the traditional techniques?

RQ2: How well the deep learning-based classifiers (such as DNN and CNN) can accurately recognize the requirement-related information from the mobile apps user reviews, compared with the traditional classifiers?

RQ3: How effective are the traditional and deep learning techniques in recognizing the most common six software requirements compared with the three classes' classifications?

1.5 Contributions

This thesis includes the following contributions:

1. Extending AlKilani Dataset by collecting more users reviews from multiple different applications on Google Play.
2. Extending the number of used classes from three classes in Alkilani Dataset (Reliability, Security, and Performance) to six classes (Usability, Performance, Reliability, Compatibility, Security, and Functional Requirements).
3. The collected reviews for the new dataset were gathered from different domains (Education, Online Meetings, Social Media, Finance, Telecommunication, Health and Fitness).

4. Using deep learning techniques for feature extraction and classifications such as CNN.

1.6 Thesis structure

This thesis is structured as follows: in Chapter 2, a detailed review of some recent and related studies were reviewed. Chapter 3 presents a comprehensive overview of the research methodology, where, Chapter 4 provides a detailed description of the conducted experiments and the obtained results with some discussion. The conclusion and the future work are explained in Chapter 5.

Chapter 2

Literature Review

Mobile apps that have better reviews achieve better ranking within stores and increase the visibility for the end-users [25]. Several studies explained how the users' review classification and analysis could be conducted and the advantages of user review analysis for the app's success [8], [23], [25]. The literature review was grouped into two parts the first one is about traditional and machine learning approaches below, and the second part is about deep learning techniques with a comprehensive overview of conducted experiments and studies.

Maleej and Nabil et al [24] mentioned that a large number of user reviews brings to the surface the necessity to automate user reviews classification. By evaluating and comparing different classifiers, user reviews can be categorized into four general groups. The first category is bugs report, which states the defects in the application. Secondly, feature requests, which are user suggestions to add new capabilities to the app that could be found in other apps. The third one is user experience reviews, which reflect the helpfulness of the app while users are experimenting with the app. The last category is user ratings, which is the Star representation of the review, which usually holds the least information [24].

Maleej and Nabil et al [24] contributed in their paper three areas. Providing probabilistic techniques and heuristics based on star review, text length, and linguistics. In addition, insights on data accuracy, and lastly, the design and usage of review analytics tools [24].

Pagano and Maalej et al [27] investigated through an empirical study important potentials in the requirement-engineering phase using user-driven methods. The study focuses on feedback usage, content, and impact. Feedback usage can be divided into feedback frequency i.e when and how users make feedback and feedback meta-data indicates the rating, feedback length, and helpfulness. The second aspect is feedback content, which investigates the topics provided in the reviews and their types. It also looks for patterns in the reviews of the same type. Lastly, it studies the impact of feedback in the sense of how it affects other users' decisions to use the app and how it might affect others' ratings. The research method was done in two phases. The first is data preparation by collecting data from app distributors like AppStore. The data consisted of different available application categories divided equally on paid and free apps. By using a scraping tool, the list of reviews was collected for each app and was connected to application metadata and stored in MySQL database [27]. The second phase was data analysis, which answers the research questions in the paper. For feedback usage, some descriptive statistics were used for analysis and hazard factors exclusion. For feedback content, two different researchers manually classified a randomly selected data set. For feedback impact, after combining metadata and content another statistical model was applied to the resulting data [27]. However, the challenge arises in understanding how app stores can be a channel between users and developers, how developers can benefit from reviews for more understanding to the user, and lastly what tools can be used to analyze these reviews [27].

Iacob and Harrison et al [19] discussed that mobile application user's reviews are considered as an important source to get out with new valuable ideas for a specific mobile app. The idea may be a new one or a modification on an existing app. The huge amount of user's reviews made it a challenge for developers or applications owners to figure out the trending feature upon user review, and the feature that should be re-designed to perform better. To overcome these challenges Iacob and Harrison et al [19] suggested a prototype in their paper called MARA Which is Mobile Applications Review Analyzer that can help to identify new features from online users' reviews for specific apps. This prototype design is divided into three parts: first, get online users review for targeted application, secondly mine the retrieved reviews by identifying complete sentence or fragment from a sentence (pre-defined linguistics rules), finally summarize the content of feature and present it into GUI.

Chandy and Gu et al [9] also did a comparison between the decision tree model and Latent class graphical model using a dataset exported and labeled from the IOS app store. The main goal of this comparison is to classify the spam review on the app store by spam or malicious developers. Yang and Pedersen et al [33] perform an empirical study to compare five selection methods of statistical learning of text categorization.

Chen and Zhang et al [10] discussed the steps of Ar-miner framework that is used to extract the most informative user review from the large pool. Ar-miner is made of five sequential steps. The first step the preprocessing, which works on raw review data that converts it to sentence-level review then text-level review. In text level review the reviews are cleaned from non-alphanumeric characters, all characters are changed to lowercase, eliminated whitespaces, and words are stemmed to the root format. The second step is filtering the review database

generated from preprocessing through trained machine learning that will automatically eliminate non-informative reviews. Then comes the next step, which is 'grouping' that combined reviews into groups that are more relevant to each other. The machine learning technique used s topic modeling that assigns multiple topics to each review. The fourth step is ranking which is a flexible and extensible model that measures the group score to indicate group importance and instance score to indicate review importance. Higher group and instance scores indicate more importance of the feature [10].

Lu and Liang et al [23], combined machining learning algorithms such as Naïve Bayes, J48, Bagging with classification techniques such as BoW, TF-IDF, CHI, AUR-BoW to classify user reviews into non-functional requirements such as reliability, performance, usability, portability, functional requirement, and others. User reviews of iBook and WhatApp mobile apps were used for this experiment for two popular and different stores of applications and platforms by using textual review and getting semantic from review using Augmented User Review (AUR) [23].

The upcoming papers are all deep-learning based used for different aims for example sentiment analysis, text classification, and question-answering. Yang and Wang et al [32] proposed a feature extraction model for E-commerce product reviews using Deep learning and sentiment lexicon. The model name is SLCABG. The base of this model is the sentiment lexicon with Convolutional Neural Network (CNN) and attention-based Bidirectional Gated Recurrent Unit (BiGRU). To enhance the sentiment features in the reviews sentiment lexicon and BERT were used and to extract the main sentiment features and classify the reviews CNN and GRU were used. The used dataset reached 100000 user reviews from very popular e-commerce websites in China. The result of this experiment showed the sentiment lexicon with CNN and BiGRU can effectively improve the

performance of sentiment analysis of user reviews of accuracy 93.5%. Yang and Wang mentioned that increasing the dataset will gradually improve the results. This model can help e-commerce merchants improve the quality of their services and attract customers through the obtained feedback.

Stawati and Nurdkholis et al [30] conducted a study for online popular transportation service application user reviews. This study aims to know the opinions of many people who use public transportation. The base of this model is Word2vec for feature extraction and word embedding with support vector machine algorithm (SVM). The result of this research showed that the accuracy value is 89

Umer and Imtiaz et al [31] proposed a framework for long and short-text classification. This model base is Fast Text for word embedding with CNN. The used dataset contains seven popular benchmarks such as Amazon Twitter, Yelp, US Airline, and Yahoo... The amount of user reviews was 100k. The result of the experiment showed an accuracy of 95% which verifies that Fast text with CNN increased the accuracy of classifying long and short text.

Li, Huang, and Ren et al [20] proposed a model named "AFKF" for medical text classification that helped clinics and hospitals in decision support systems which is an automated tool that improves patient care management. Medical text classification was a big challenge for the authors because of medical abbreviations and terminologies. The base of this model is Doc2Vec, BERT, ALBERT, RoBERTa for word embedding of the medical text and RNN and deep learning classifier CNN for classifying the sentence and the documents. The dataset was collected from hospitals on EMR texts (HICH). It contains 3700 real samples. The results were that AFFK outperformed other traditional techniques such as Random forest. The highest accuracy achieved by CNN is 90%.

Lokus and Stogiannidis et al [22] proposed a model for the classification of

text in the financial domain using a dataset called Banking77 with 77 different classes containing around 13,000 examples. The results of state of art model show that GPT-3.5 and GPT-4 as feature extraction techniques can outperform fine-tuned, non-generative models.

Below table 2.1 provided a comprehensive overview of the studied and visited literature reviews. The summary and comparison as below:

TABLE 2.1: Literature review summary and comparison

Authors	UR	Accuracy	Techniques	Classes	Task Classification
[30]	7000	89%	W2V & SVM	2	SA
[32]	100000	93.5%	BERT & CNN	2	SA
[8]	5422	82%	RF	5	UR
[31]	100k	95%	Fast Text & CNN	5	English news
[20]	3700	90%	Doc2Vec, Bert & CNN	4	Medical terms

Abbreviations in the table "SA" stands for Sentiment analysis, "UR" stands for User review classification, and "TB" stands for Technical Debt classification.

Minaee and Kalchbrenner et al [26] provided a comprehensive overview of text classification-based experiments with deep learning techniques such as feature extraction, wording embedding techniques and classifiers. The experiments include sentiment analysis, text classification, and question answering. Table 2.2 shows the accuracy of deep learning used techniques for text classification experiments only. For the results in the table, Text GCN achieved the lowest accuracy meanwhile BERT-large achieved 99.32%.

TABLE 2.2: Accuracy of Classification Models on News Categorization, and Topic Classification Tasks

Method	Accuracy
Text GCN	67.61
Simplfied GCN	88.50
Char-level CNN	90.49
CCCapsNet	92.39
LEAM	92.45
CapsuleNet B	92.50
Deep Pyramid CNN	93.13
ULMFiT	94.99
L MIXED	95.05
BERT-large	99.32
XLNet	95.51

Chapter 3

Research Methodology

The main objective of this study is to extract the most common software requirements from mobile app user's reviews such as usability, performance, reliability, compatibility, security, functional requirements, and others. By using these users' reviews written by users of sixteen selected mobile applications from the Google Play store. Representative features are extracted from the text reviews using natural language processing (NLP) techniques. Some of the recent machine learning and deep learning techniques are used for classification. The proposed methodology can be divided into several pipelined stages, as follows.

3.1 Dataset description :

As mentioned earlier, we started with the available dataset that was collected and used in the previous work conducted by Al Kilani et al [8], which is specified in the healthcare domain. The dataset was extended for the same domain and included user reviews from different domains such as users reviews from businesses and telecommunication, educational, financial, social media, and communication mobile apps to achieve a balanced dataset that includes reviews for

the considered nonfunctional requirements. The extended dataset was collected from Google play using an open-source tool named Web Harvy [1].

3.1.1 Al-Kilani dataset

The dataset of Al Kailani et al was collected using Chrome selenium driver. This software is mainly used for user interface automation testing [1]. The authors benefited from it to retrieve users' reviews. Around 90,000 users reviews were retrieved, all of them written in English. The dataset was retrieved from ten selected mobile applications in the healthcare domain. Retrieved data include the application name, rating and review description, username, and date of submitting a review. A number of volunteers, with experience in the field, were asked to label a randomly selected set of reviews in terms of sentiment (negative, positive, or neutral). In addition, for each review from the selected reviews, they were asked to select one of five software requirement categories (bug, new feature, performance, security, usability). Data analysis showed that 5422 reviews were classified at least one time by one volunteer or by the first author according to category labeling, whereas, 1406 reviews were labelled by two experts. Table 3.1 below shows some statistics about the Al-Kilani dataset.

TABLE 3.1: Statistics about the Al Kilani dataset.

	Statistics about the Al Kilani dataset		
Classes	Req classification counts	Unique reviews	No. audited reviews auditors
Bug	2758	1513	867
Usability	839	666	142
New Feature	1170	748	308
Performance	555	465	72
Security	100	75	17
Total	**5422**	**3467**	**1406**

[1] https://www.webharvy.com/

3.1.2 Extended dataset

Al-Kilani dataset has some limitations, in terms of data size, number of software requirements with sufficient data, and that all of the used app are from one domain, healthcare. This motivates us to collect more data and extend Al-Kilani to overcome these limitations. The first step to extract user reviews for mobile applications from different domains is to select a group of commonly used app with sufficient user reviews. The selection criteria is based on popular apps by users of well-known companies and organizations and the huge number of users' reviews. Moreover, these apps had shown a great boom in the last two years, especially during the corona pandemic and lockdowns. During this period, people worldwide started to find out to get things done at the same time keep social distancing. According to Google trends [2] , the terms "Coursera", "Udemy - Online Courses", "ZOOM", "Webinar", "Google Meet", "Microsoft Teams", "WhatsApp Messenger", "Skype", "PayPal Mobile Cash", "NetDania", "Investing.com", "MyFitnessPal" had shown an increase in Google searching engine worldwide. Sixten mobile applications from six domains have been selected to extract users' reviews shown in Table 3.2 below.

Overall count of users reviews were gathered from Google Playstore to show the huge amount of available users review for the selected apps, which reflects the usage, and popularity of apps. From the above Table 3.2, it is clear that WhatsApp messenger had the highest overall user's reviews with approximately 153M user reviews. Skype had second place with 11M overall user's reviews. The lowest overall users review was for Ana Paltel app with 4,5K users reviews.

Most of the selected mobile applications for the extended dataset have a rating of more than 3 out of 5. Investing.com app has a higher rating of 5 out of 5.

[2]https://trends.google.com/trends/?geo=PS

TABLE 3.2: Sixteen selected mobile applications per domain.

	16 mobile applications per domain.	
App Name	Overall Users reviews	Domain
Coursera	129,326	Education
Udemy - Online Courses	337,930	Education
ZOOM Cloud Meetings	3,398,354	Online Meetings
GoToWebinar	120,115	Online Meetings
Google Meet	1,985,037	Online Meetings
Microsoft Teams	4,434,701	Online Meetings
WhatsApp Messenger	152,876,497	Social Media
Skype - free IM video calls	11,416,141	Social Media
NetDania Forex Trader	24,377	Finance
PayPal Mobile Cash	2,352,111	Finance
Investing.com	479,496	Finance
Bank of Palestine	12,224	Finance
Ana Paltel	4,426	Telecommunication
My Account (Jawwal)	32,033	Telecommunication
Home Workout - Bodybuilding	71,428	Health and Fitness
MyFitnessPal	2,462,465	Health and Fitness

The average rating of all apps is 4.1 out of 5 which means that users generally are satisfied with using these apps. The overall rating of 16 selected mobile apps is shown in Figure 3.1 below.

The total retrieved user reviews are 1,718 reviews all written in English. The user's reviews were distributed into multiple different domains as mentioned previously. The percentage of user reviews per domain can be shown below in the pie chart shown in Figure 3.2. Health & Fitness domain has the higher percentage of total extracted reviews with 28% that equals 480 users reviews, then finance domain with 25% which equals 430 users reviews, business domain with 16% which equals 276 users reviews. Education, Online meetings, and Social media domains have percentages of 11%, 10%, 10% in order which equal less than 180 user reviews for each.

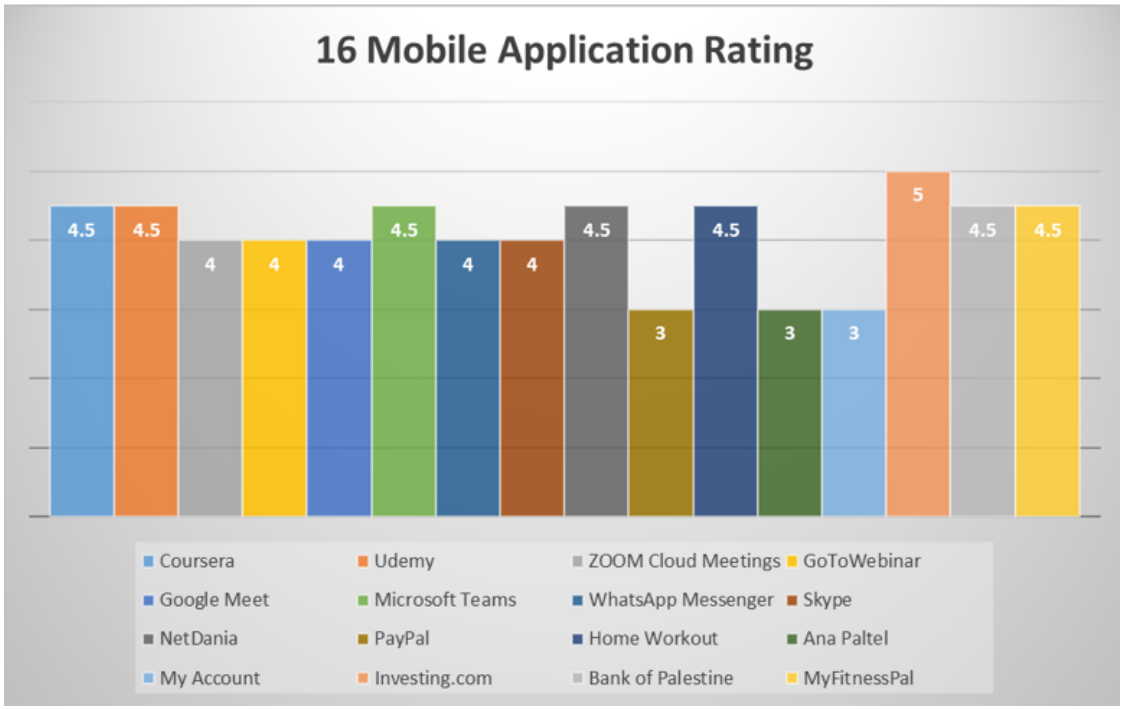

FIGURE 3.1: The selected mobile applications with their overall rating.

User's reviews of seven applications are 160 users' reviews for each. The average per application is 107 users' reviews. Nine apps are above the average meanwhile seven apps are less than 107 user's reviews. The distribution of retrieved users' reviews per application can be shown in Table 3.3.

Our study will be conducted on English user review only. Accordingly, we removed user reviews, which, were written in other languages. Moreover, user reviews with emojis, rating only, and empty descriptions were removed.

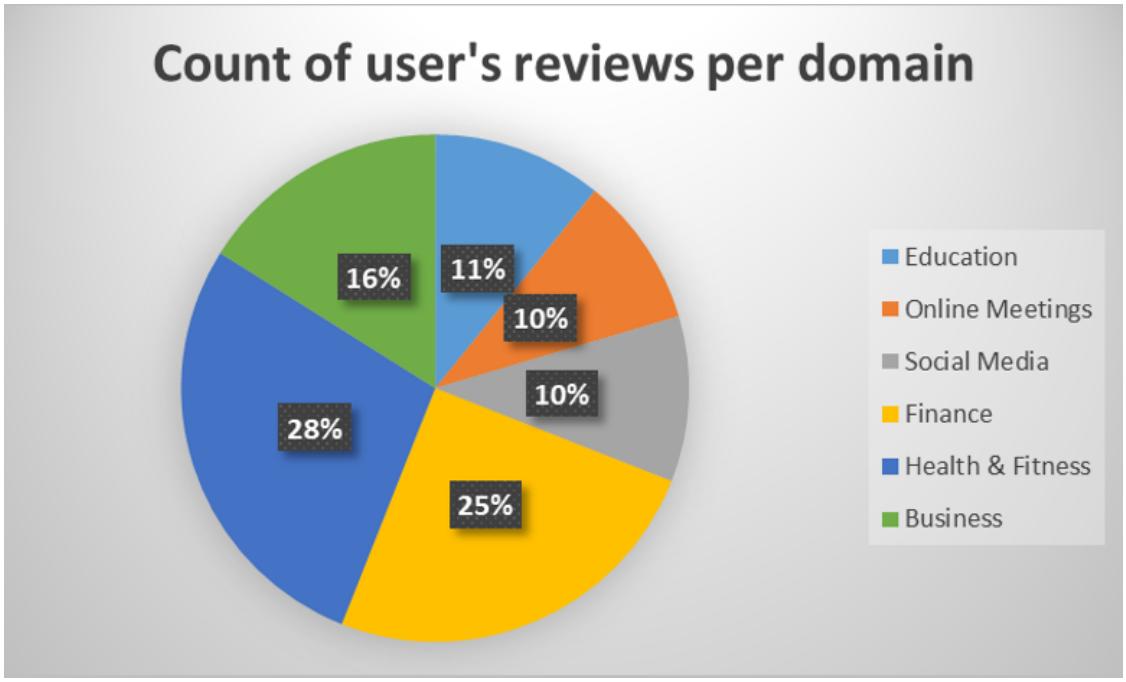

FIGURE 3.2: Count of extended user reviews dataset per domain.

3.1.3 Data annotation

The retrieved user's reviews from the selected mobile apps that are mentioned earlier need to be labelled, similar to the Al-Kilani dataset as described in [8], so they can be used to extend it. The data annotation was performed manually through the following phases. In the first phase, the same classification method used by Lu and Liang et al [23] was used in the data annotation process. In addition to the author, three experts in software engineering participated in the data annotation process. Two of them have Bachelor in computer science with twelve years of experience in software engineering and information technology in the Palestinian market. The third participant is a master student in software engineering program at Birzeit University with proven seven years of experience as a software developer. Each participant manually classified a randomly selected set of user reviews independently, based on ISO 25010 standards of

TABLE 3.3: Count of extended user reviews dataset per application.

Applications	Count of User Reviews
Coursera	26
Udemy - Online Courses	160
ZOOM Cloud Meetings	31
GoToWebinar	12
Google Meet	42
Microsoft Teams	80
WhatsApp Messenger	88
Skype	93
NetDania Forex Trader	160
PayPal Mobile Cash	110
Home Workout - Bodybuilding	160
Ana Paltel	160
My Account (Jawwal)	160
Investing.com	160
Bank of Palestine	160
MyFitnessPal	116

non-functional requirements [14].

The definitions of the non-functional requirements are elaborated for the reviewers based on standards ISO 25010. In case of disagreements on some classified reviews, the author and participants discussed and resolved the disagreement together.

The main reason for choosing the following non-functional requirement (reliability, compatibility, security, performance, usability, portability) in this study is that they are related to the external quality of the system as they can be noticed by mobile application users.

Some non-functional requirements are related to the internal system quality such as documentation, maintainability, functional suitability. Most people who may be interested in this kind of requirements may be team leaders, software architects, software developers, or project managers based on sponsors or project

owners. Moreover, they will not communicate through user reviews to express their requests. Accordingly, all similar kinds were excluded from this study. For these reasons in the manual annotation, we considered the following categorization of users' reviews; reliability, compatibility, security, performance, usability, portability, learnability.

In the second phase, and after exporting and preparing the user review, all the user reviews were imported into a database, and published on a specially designed website [3] and Github [4] , to make it easier for the reviewers to do the annotation. The website consists of three webpages main, registration, and classification. The main page includes the instructions and information provided for the reviewers. It is useful to perform the annotation task correctly and to provide a better understanding of intended non-functional requirements with examples. Figure 3.3 and Figure 3.4 below show a screenshot from the website. The flowchart depicted in Figure 3.3 describes the manual annotation process.

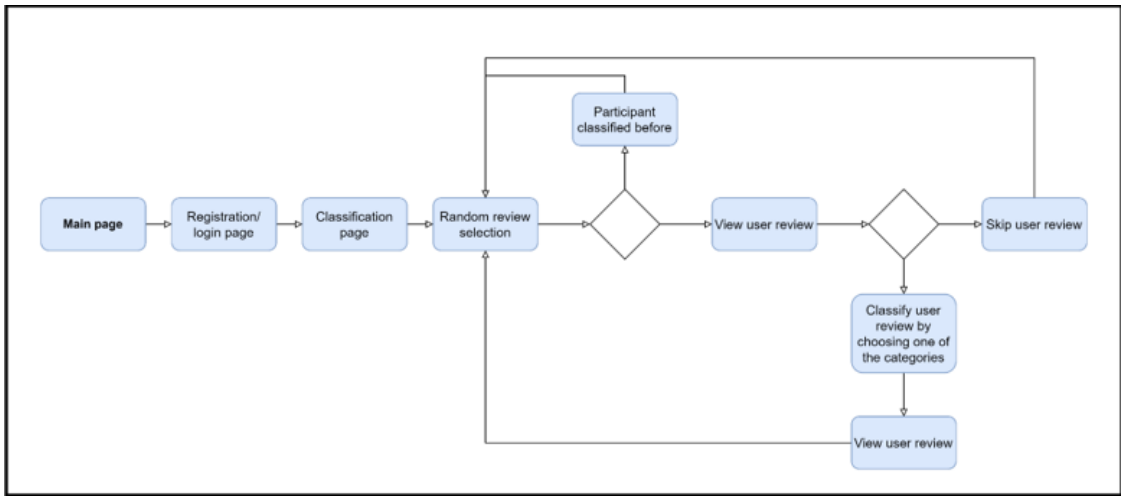

FIGURE 3.3: Manual users review annotation flowchart.

[3]https://nasrisameerladaa.com/
[4]https://github.com/NasriLadaa/Nasri$_{S}ameer_{T}hesis_{S}WEN860$/

The followings are the considered software requirements categories in the annotation process through the website:

1. **Usability:** "Degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" [29]. Example: "I have to navigate into three pages before reaching my appointment page"

2. **Reliability:** "Degree to which a system, product or component performs specified functions under specified conditions for a specified period time" [29]. Example: "After clicking on start call button the app crashes"

3. **Portability:** "Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another" [29]. Example: "On my tablet, it worked fine but on the phone, nothing appeared"

4. **Performance:** "Performance relative to the number of resources used under stated conditions" [29]. Example: " I have to wait 10 seconds to see all my notifications"

5. **Learnability:** "Degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk, and satisfaction in a specified context of use" [29]. Example: "I used many similar apps all of them have a quick tutorial to get started

6. **Compatibility:** "Degree to which a product, system or component can exchange information with other products, systems or components, and/or

perform its required functions while sharing the same hardware or software environment" [29]. Example: "My live match is not synced with FIFA official schedule"

7. **Security:** "Degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization" [29]. Example: "This app allows weak password"

8. **Functional Requirements:** Things that the system shall allow, be able to do, may be able to do. Example: "Can you add Call ambulance button so we can make emergency calls faster"

9. **Others:** Include emotional expressions and not related functional or non-functional requirements. Example: "Cool", "I like this App."

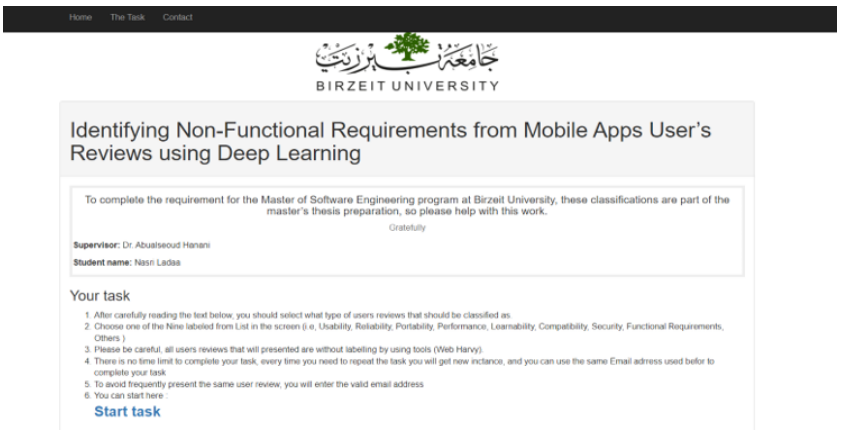

FIGURE 3.4: Website main page - Part 1.

The second page is the registration page as shown in Figure 3.6. It is used for registering the expert's details such as the email, job title, and years of working experience. On the classification page, as shown in Figure 3.7, a user review is

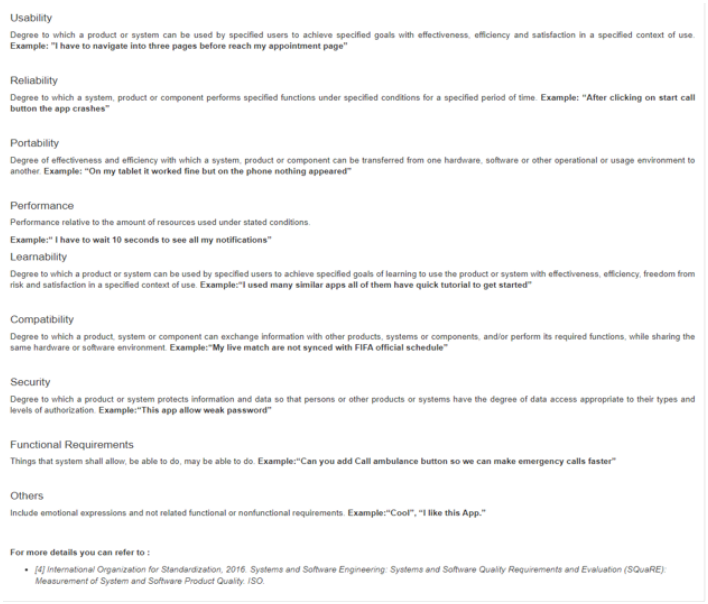

FIGURE 3.5: Website main page - Part 2.

displayed on the page with the specified requirements categories, where, the reviewer read the given user review and select the most appropriate requirement category and then proceed to the next review and so on. The expert participant can skip any unclear review by clicking the skip button. Classified user reviews are not appeared again for the same expert participant. Skipped user reviews may be displayed another time for classifying based on the random function that selects users' reviews from the review in the database. The same user review can be classified multiple times by different participants. The database entity relational diagram and website pages are shown in Appendix A.

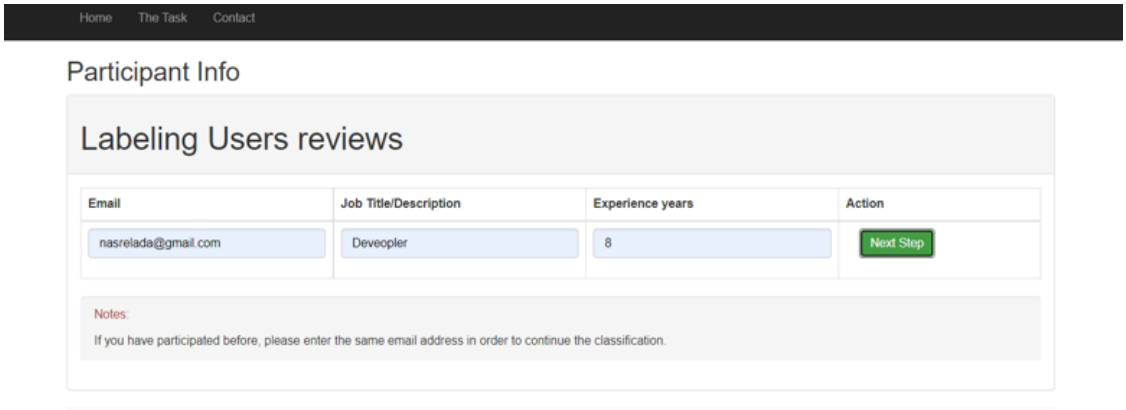

FIGURE 3.6: Register and log in to start the task.

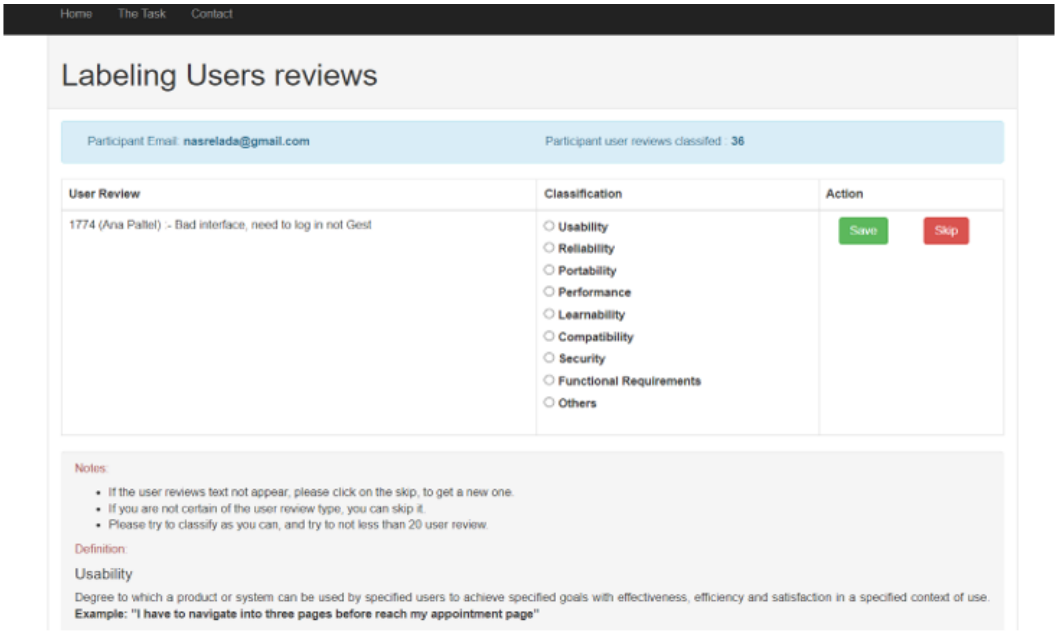

FIGURE 3.7: Classification page with an example

To check the reliability of the annotation process and measure the agreement degree among the experts the Kappa statistical test [13] is conducted and the K - Cohen's kappa coefficient value is calculated. More details about the Kappa test and its result are illustrated in the subsequent sections.

3.1.4 Manual annotation results

Al-Kilani dataset [8] already contains 5422 classified user reviews out of 8,884 reviews. The classified users review categories are bug, new feature, performance, security, usability. Bugs and new feature labeling are considered as functional requirements, where the others are non-functional requirements. According to that, the number of functional requirements from the old dataset is 1443 review. For the new gathered dataset, the first author and experts classified and reviewed all user's reviews. Table 3.4 summarize the results of manual annotation.

TABLE 3.4: First author user's review and old dataset classifications.

	First author user's review and old dataset classifications.		
Dataset	Al Kilani Dataset	New dataset	Total
Total No of reviews	8,884	1,718	10,602
Classified	5422	1,693	7,115
Reliability	0	432	432
Compatibility	0	45*	45
Security	49	87	136
Performance	475	214	689
Usability	537	259	796
Portability	0	58*	58
Learnability	0	27*	27
Functional Requirement	1443	256	1,699
Others	0	315	315

*From the new dataset the number of user reviews that classified as Compatibility is 45, as Learnability 27 and as Portability 58. By comparing the number of average other classified class. It was agreed to drop those classes from the conducted experiments.

TABLE 3.5: All datasets classification.

	All datasets classification.		
Classes	Al Kilani DS	Merged DS 3 classes	Merged DS 6 classes
Security	49	136	136
Usability	537	796	796
Performance	475	689	689
Reliability	0	0	430
Functional Requirement	0	0	256
Others	0	0	315
Total	1061	1612	2622

3.1.5 Kappa Test

To mitigate the chance of creating a biased dataset, we conduct a group session for three participants. Two of them have Bachelor in computer science with twelve years of experience in software engineering and information technology in the Palestinian market. The third participant is a master student in software engineering program at Birzeit University with proven seven years of experience as a software developer. The group session was conducted online using Zoom and was divided into four sections. Each section took around one hour and a half. The first part is a comprehensive overview and background about the topic. The second part is an explanation of the quality indicators of a software system and non-functional requirements.

By computing Cohen's kappa, we may assess the degree of agreement between the experts and the author. The Cohen's Kappa coefficient, which determines the percentage of agreement that is chance-corrected, is a frequently used approach to assess the level of inter-rater agreement for categorical scales. The coefficient's outcome is scaled between -1 and +1, with a negative value denoting agreement that is worse than chance, a zero denoting agreement that is exactly chance, and a positive number denoting agreement that is better than chance.

The agreement is stronger if the value is nearer to +1. The degree of agreement between two observers—an author and experts—and five categories—a requirement, a design fault, a test, and some documentation—is computed. We made use of kappa calculator online [5] . According to Fleiss [12], we reached a degree of agreement measured between the author and experts of +0.8. Values greater than +0.75 are described as high agreement. The test was conducted and results are fine.

3.2 Research approach

This study was constructed to contain a sequential phase. The first phase should be ready in order to be able to move into the next phase and so on until the last phase. Phase 1 uses users reviews that we discussed in the data description 3.2. Phase 2 performs an important task, the preprocessing activities such as tokenization, text cleaning, and normalization. Phase 3 focuses on using feature engineering and word embedding methods, and then training a classification model using machine learning. Finally, evaluate the results of the conducted experiments. Figure 3.8 shows an overview of the system design.

3.3 System design

This section contains multiple sequential phases as below:

[5]https://www.graphpad.com/quickcalcs/kappa1/

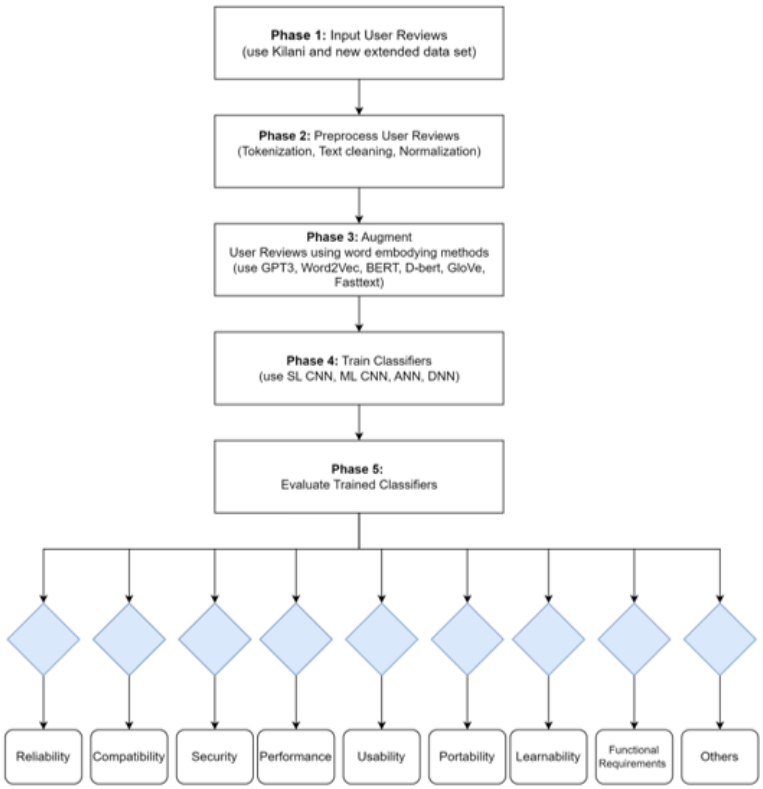

FIGURE 3.8: System design overview

3.3.1 Preprocessing

This phase focuses on words inside the user's review or sentences that carry the meaning. User's reviews are written in natural language, so their semantic meaning can be affected by noises inside sentences. To keep the important words and remove the noise words, these sentences should go through multiple pipeline steps. The output of this phase is an input to the machine-learning algorithms, which affects the performance of the overall system. To perform this phase some NLP techniques [6] [7] were used as follow:

[6]https://www.nltk.org
[7]https://spacy.io/

3.3.2 Tokenization

The objective of tokenization is to split the user's review text into multiple units by white spaces named tokens. These tokens should have semantic meanings. Tokenization could be paragraphs, sentences, and/or words.

Tokenization is not regular. It is based on semantics and meaning. For example, the word "didn't" in the user review will be tokenized into "did" and "not". In regular tokenization, it will be considered as one unit and it will not be tokenized.

3.3.3 Text cleaning

This step aims to remove all undesired and unwanted content inside the user review, by applying the below techniques:

1. **Punctuations marks removal** Using regular expressions, a rule was defined to remove all punctuations inside user review.

2. **Stopping words removal** Removing stop words helps in decreasing the training time of the model. Moreover, it helps in improving the performance and accuracy of classification, keeping meaningful tokens, and decreasing the size of the dataset. A natural langue-processing library was used to remove stop words called NLTK toolkit [8]. The followings are examples of English stop words; 'a', 'about', 'for', 'from', 'have', 'against', 'it', ' 'am', ' 'and', 'any', 'are', "aren't", 'as', 'at', 'be, ...

3. **Non-alphabetic removal** Removing numbers and special characters that do not give any useful meaning or semantic.

[8] https://www.analyticsvidhya.com/

3.3.4 Normalization

The objective of the normalization step is to transform words into uniform sequence by applying the stemming that reduces word's inflection and returning it to root base and lemmatization that removes affixes from words.

3.4 Features engineering

The second stage of our methodology is extracting pertinent data from user reviews and representing it in a way that is suited for machine learning. In our method, we converted words into numeric features using two different ways. The first method uses NLP techniques like Bag-of-Words (BoW) and TF-IDF which rely on syntax to describe the grammatical structures or body of rules that constitute a language. The second method, known as the word embedding method, focuses on the semantics of words and considers their meanings as well as how to group words together to make sense while also taking into account syntactical rules. These technique were used along with the following five other techniques in our system: Word2vec, Fasttext, Bert, D-Bert, Glove, and GPT3.

3.4.1 Syntactic vectorization methods

1. **Bag-of-Words:** A straightforward representation technique known as "bag-of-words" or "term frequency" (TF) uses the frequency of terms in a given document. After preprocessing the user reviews, a list of vocabulary from every remark is created for comprehending BOW, for instance, with each review represented as a numeric vector. Each vector has a length equal to the number of words in the vocabulary list. Every item in the vector that corresponds to a word in the vocabulary list will have a value equal to the frequency of that word in that specific remark, or zero if the term

doesn't appear in that review. In the matrix with dimensions (number of user reviews * size of vocabulary list), all the vectors were reshaped.

2. **TF-IDF:** Term frequency-inverse document frequency (TF-IDF): TF-IDF and BOW vary primarily in that they consider the weight of the words, whereas BOW just considers the frequency of words across a text when constructing the vector.

3.4.2 Word embedding methods

In natural language proceeding (NLP) word embedding is converting words into a numeric vectors. These vectors represent the word's meaning. Words that are closer in the meaning are expected to be similar/close in the vector space. This phase focuses on extracting useful features from user's reviews. The following are the most common word embedding pre-trained models, which are used in the proposed system.

1. **Word2Vec:** is a method [9] used to implement bag of words and skip-gram architectures that are used in different natural language processing applications, as shown in Figure 3.9. The input for is text corpus that will be processed to a words vector. The method will construct a vocabulary from the training set and then it learns the vector representation. Further investigation for the vector can be done by measuring the distance between the specified word and the words around it. This method was created by Google in 2013. It was trained on a news dataset from Google of about 10 billion words. Word2Vec can combine similar words vectors.

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

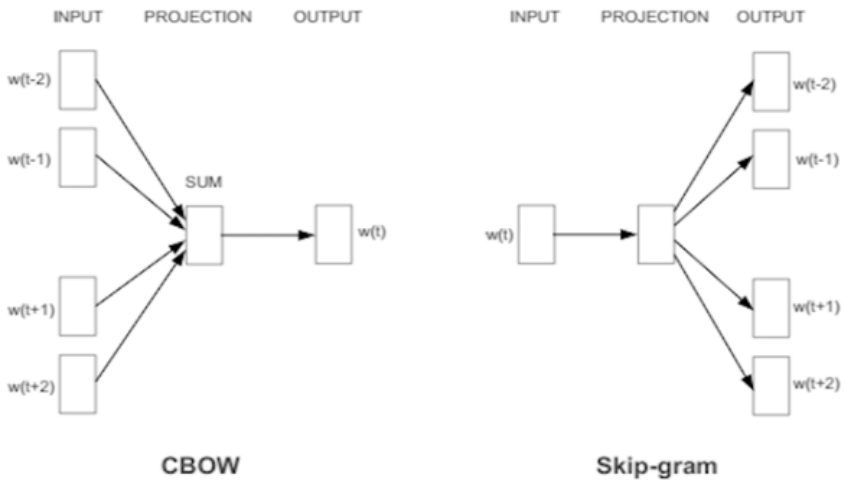

FIGURE 3.9: Architecture of Word2Vec [15].

2. **GloVe:** It is an unsupervised machine-learning algorithm that processes word-word occurrences in a corpus that are represented in linear substructures of the word vector. The architecture of the Glove model is depicted in Figure 3.10. Training GloVe can be computationally expensive initially especially for large corpora to populate the word-word matrix. The training is done on non-zero occurrences of the word-word matrix which becomes faster after populating the matrix as non-zero occurrence words are much smaller than the words in the original corpus [2].

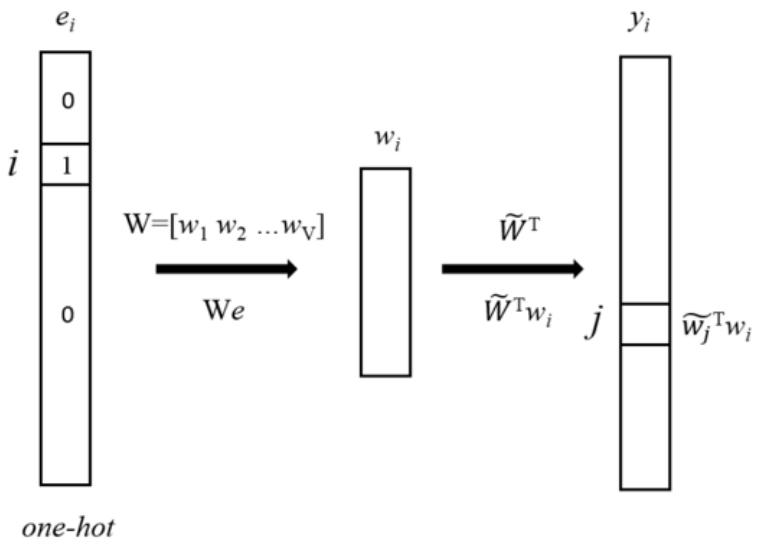

FIGURE 3.10: GloVe Architecture [21].

3. **BERT:** It stands for Bidirectional Encoder Representations from Transformers (BERT). It is a fast pre-trained unsupervised machine-learning model developed by Google that relies on a transformer to learn the contextual relationship between words. It is bi-directional as it looks on the left and the right of the words opposing traditional methods that investigate sequences of words one direction at a time [3], as shown in Figure 3.11.

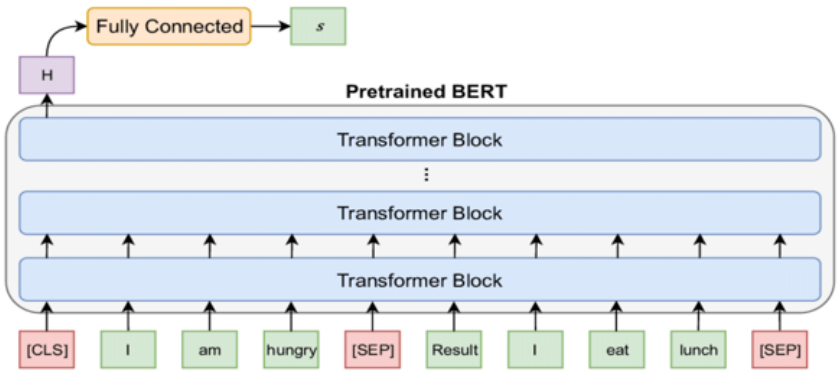

FIGURE 3.11: BERT architecture [17].

4. **D Bert:** it is a model aims to identify the semantic relationships between the head and tail entities in a set of sentences {x1, x2, . . . , xn}. A bidirectional transformer model that can represent words in unannotated corpus by using position embeddings, segment and token. It uses the output hidden vector as the semantic representation of the text and a symbol for text classification. The model as shown in Figure 3.14 can be optimized by adjusting the model parameters. The semantic text representation of the model output can depict the nature of the language and facilitate the subsequent for specific NLP tasks. Using pre-training model adopts, the input sentence is a single sentence at the beginning of this sentence will be added and a special token for the beginning and end of each entity at the beginning is represented [18].

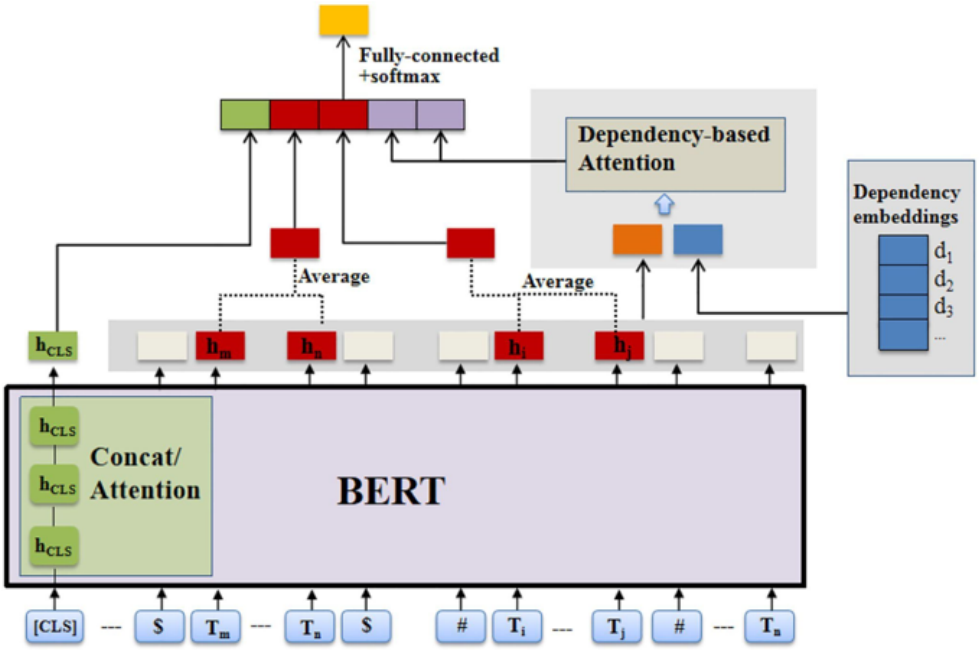

FIGURE 3.12: D-bert model architecture [18].

5. **GPT3:** It is an abbreviation for Generative Pre-trained Transformer 3 that

aims to produce the human-like text of simplifying it through using deep learning. It is established in 2015 by a research laboratory for artificial intelligence known as OpenAI. The goal of it is to benefit humanity through developing and promoting friendly AI and ensure Artificial General Intelligence (AGI) [14]. GPT3 is considered an autoregressive language model designed to generate a sequence of words starting from machine input which is called prompt [14], see Figure 3.13. The model was trained based on a very large unlabeled dataset from different websites such as Wikipedia and other sites in multiple languages. The First GPT version release was in 2018 the model was trained using 110 million learning parameters. Learning parameters are values used during model training in neural networks. GPT2 trained on 1.5 billion and Now, GPT3 trained on 175 billion [14]. Training activities include Chat boot, translation. Grammar correction, answer questions [14]. GPT3, which uses an autoregressive language model, is used to get words similarity and to augment users' reviews [14]. Classification techniques will be used to calculate the weight and to train and evaluate the classifiers.

6. **FastText:** It is an open-source library designed by the Facebook AI research lab that allows the building of scalable applications for text classifications, as represented in Figure 3.14. It uses successful concepts of natural language processing such as a bag of words and n-grams. In addition, fast text can be used for text representation and can work with multiple languages such as Czech, German and Spanish by using the language's morphological structure [5].

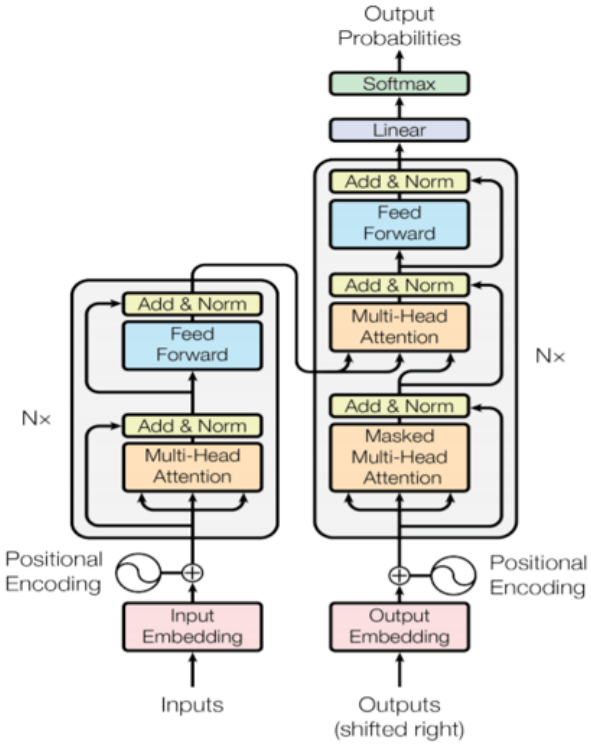

FIGURE 3.13: GPT3 model architecture [4].

3.5 Machine Learning Classifiers

In this thesis, three traditional classification algorithms; random forest, Naïve Bayes, and ordinary artificial neural networks were used with the two traditional features BOW and TF-IDF, for classifying the target categories of the software requirements from the user reviews. In addition, three classification methods based on the deep learning, DNN ANN,, SL-CNN, and ML-CNN were used with the word embedding features such as word2vec, BERT, D-bert, Glove, etc.

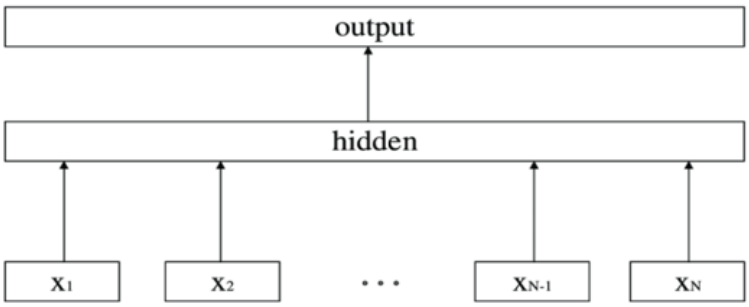

FIGURE 3.14: FastText architecture [34].

3.5.1 Traditional machine learning

1. **Random Forest (RF)** It is a data mining technique to solve problems associated with the classification. In general, the growing ensembles of trees and deciding the class type by voting improve the accuracy of the classification. Random Forest grows each classification and regression tree from a random vector [4]. RF prediction is determined by the majority of trees voting after analyzing their output. Error merges to a limited value, as overfitting is less likely to happen when adding trees with random forests. RF achieves higher accuracy by reducing the bias by creating trees without pruning and by reducing correlation by randomization of variables at each node. The growing and voting process of RF is a multi-step process. The first is using around two-thirds of the training set to grow RF trees and the last third is calculated Out of Bag error (OOB). The second step is choosing the number of variables from the variables pool. The third step is to try multiple values to choose a minimal OOB error [4]. The performance of each tree is tested using the OOB dataset. While growing trees RF estimates unbiased error using the OOB data set. In addition, OOB is used to measure the significance of variables in RF classification [4] [11]

2. **Naïve Bayes** The Bayes theorem is the foundation of the NB machine-learning algorithm. In NLP applications, NB is one of the most well-known and widely used supervised machine learning classifiers. It is used in our study to define the category of technical debt and serves as the starting point for all of our trials.

3.5.2 Deep learning classifiers

1. **Convolutional Neural Networks (CNN) (Single and Multiple Layers)**

   CNNs are a subset of neural networks that were initially developed for computer vision tasks that involved deep learning and were largely utilized for image recognition and classification. Today, CNN is a cutting-edge text categorization method. Depending on the picture resolution, it accepts an input image as a 3-dimensional array. The image's height and breadth corresponded to two array dimensions. The color of the pixel (RGB) is the third dimension. Convolutional, pooling, and fully linked input layers make up the three primary levels of the CNN architecture [16], as shown in Figure 3.15.

   Convolution layer: The first layer in the CNN architecture is convolution. By applying a filter to the input data (picture), with regard to its dimensions, this layer aims to extract features. Convolution learns visual characteristics from tiny input data squares, preserving the link between pixels.

   Pooling layer: Usually, a convolution layer is added before the pooling layer. This layer's goal is to increase the amount of characteristics. This makes the convolutional layer's dimensions smaller in order to obtain the

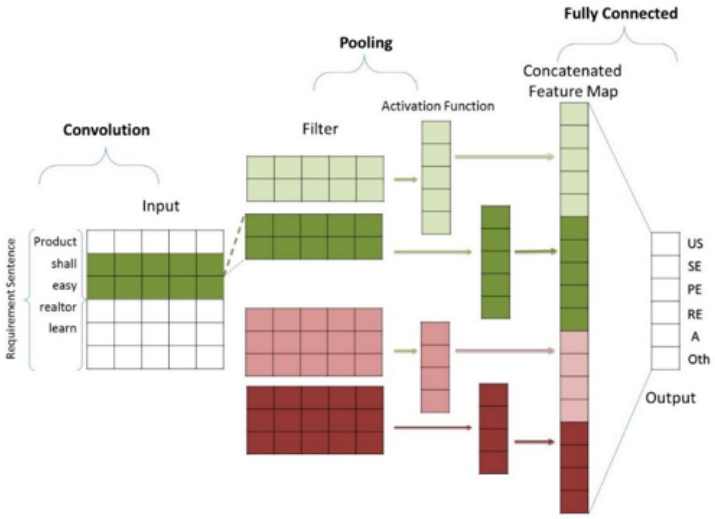

FIGURE 3.15: CNN Architecture [29]

most crucial data. In order to minimize the dimensionality of the data, the pooling layer mostly employs max or average pooling.

Fully connected layer: In CNN architectures, fully connected layers are often discovered at the end, using the results of convolution and pooling layers to forecast the best label and determine the class rankings.

We employed two architectural networks for the CNN: a single hidden layer of basic CNN and multiple hidden layers of complicated CNN. The number of layers distinguishes the two networks. While the complicated CNN has three layers, the single-layer CNN only uses one layer for the convolutional and pooling layers. As a result, the parameter reset was corrected as follows: Filter count: 128; class count: 3 and 6; stride: 1; dropout: 0.2; batch size: 32; maximum epochs: 20. Filter size for layers in order: [2, 3, 4]. To obtain the ideal number of training epochs and avoid the overfitting issue. When a monitored metric stops improving, the model stops learning. We utilized callback 10 as an early stopping parameter in the fit

model; the callback will halt training if the validation loss does not improve for five successive epochs. "Relu" and "Softmax" are the activation functions, while "Adam" is the optimizer. The trainable option is set to false and our own embedding matrix is handed in as the weights parameter since we are not training based on our own embedding and the pretrained model is being used.

2. **Artificial neural network (ANN):** Artificial Neural Networks (ANN) contain a large number of interconnected nodes (neurons) that convert a weighted sum of inputs into an output of 0 or 1. The weighted sum is transferred to output using sigmoidal transfer functions [4]. ANN can have input notes matching the number of i outs, while the output notes number is determined according to the number of classes. When a neural network consists of multi-layers it is called a multi-layer perceptron. Input nodes are interconnected with hidden layer nodes that are connected to an output node. A weight is assigned to each connection in ANN [4].

3. **Deep Neural Network (DNN)** A DNN is a group of neurons arranged in a series of layers, where each layer's activations are fed into the next layer's neurons, which then do a straightforward mathematical calculation (such as a weighted sum of the input followed by a nonlinear activation). The network's neurons work together to accomplish a complicated nonlinear mapping from input to output. Error backpropagation is a method for modifying the weights of each neuron to learn the mapping between the input and output from the data. Simply described, DNN is an artificial neural network (ANN) having numerous layers between the input and output layers, or it is a feedforward multilayer neural network design. To

conduct classification on related, unlabeled data, we will train it using a collection of labeled data.

Below Figure 3.16 summarizes the main three phases in system design in our study that we've already discussed in previous sections.

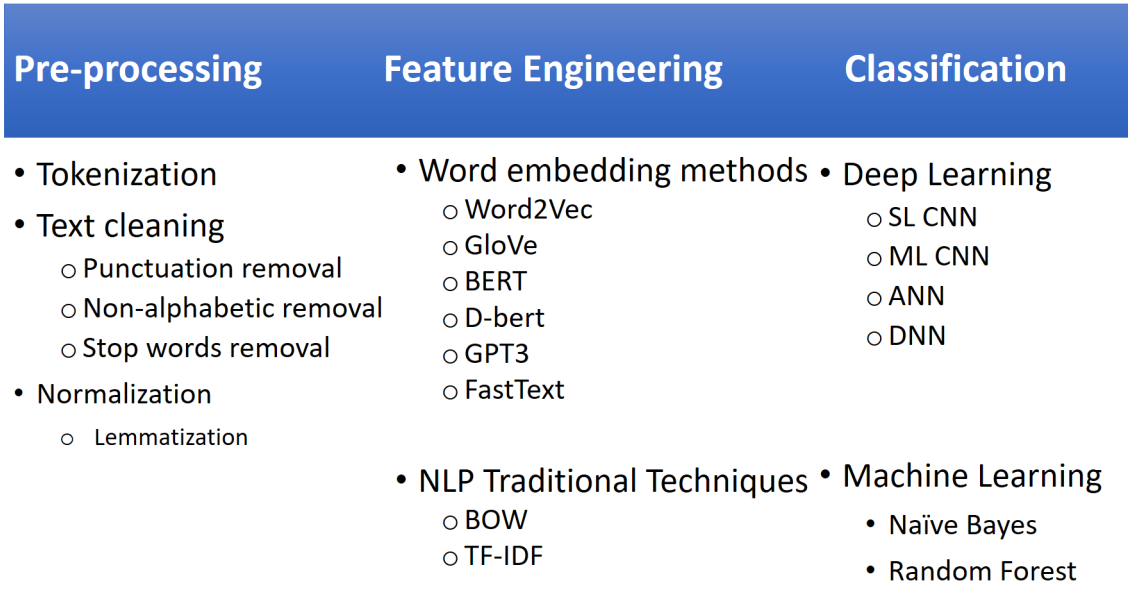

FIGURE 3.16: System design main phases.

3.6 Evaluation metric

According to our technique, the train and test subsets of the dataset are randomly divided to evaluate the system. The test set is solely used to evaluate the performance of the classifiers, whereas the training set is utilized to train ML classifiers. It is important to note that the test dataset (holdout dataset) has never been utilized in training.

We used a 80:20 train to test split as a general guideline. Which is the standard rule for handling small datasets. There are 5422 labelled reviews in Al-Kilani (4337 train, and 1085 test), and 6113 labelled reviews in the extended dataset (4890 train, and 1223 test).

Accuracy, precision, recall, and F1-score measures, which are the most typical classification performance metrics, were employed in all trials. Measure the proportion of properly identified requirements to all true positive and false positive predictions using a precision metric (positive prediction value). Precision measurement is described in Equation 3.1 [28]. The number of correctly categorized needs was indicated by TP ("true positive"). The term "false positive" (FP) indicates the number of incorrectly categorized cases.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

While recall indicates the proportion of pertinent user review that were successfully connected. Depending on the measurement's goal, both metrics – accuracy and recall – have different levels of significance. The recall equation is shown in Equation 3.2 [28]:

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

The weighted harmonic average of recall and accuracy is known as the F-measure. As a result, both false positives and false negatives are considered in the F1-score computation. F1- score can be expressed as shown in Equation 3.3:

$$F - score = \frac{2.Precision \times Recall}{Precision + Recall} \tag{3.3}$$

Chapter 4

Experiments and Results

4.1 Experimental setup :

Google Collaboratory or Colab [1] was used to conduct all of the presented experiments in this chapter. Colab is a cloud service that supports GPU processors that allow running Python code on the cloud. Table 4.1 shows processor specification was used for all experiments.

TABLE 4.1: Environment setup

	Environment setup
Type	Specification
CPU model	Intel ® Xeon(R) CPU @ 2.20GHz
Cache size	56320 KB
RAM	13.3 GB
Disk	69 GB
GPU	Tesla T4
OS	Ubuntu 18.04.5 LTS

The data is split into 20% testing and 80% training in all conducted experiments. Random State 1, epochs 30, batch size 32, verbose 1, Patience 3 to stop training if value loss has not improved in 3 epochs. Class weight percentage

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

for balancing the dataset classes. For the Deep Learning experiments, we used DNN, ANN, BLSTM, SLCNN, and MLCNN across multiple phases The details of the parameters are n_estimators which indicates the number of trees in the forest. 100 trees were considered. class_weight is "balanced" which indicates the “balanced” mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data. Random Forest classifier settings and parameters were n_estimators=100, class_weight="balanced”, random_state=1. All the conducted experiments can be found on Github [2].

4.2 Preprocessing :

The textual data (user reviews) should be cleaned and prepared before using it in the experiments. To achieve this a set of preprocessing steps mentioned earlier in the methodology chapter were used in all conducted experiments as the following.

4.2.0.1 Text cleaning:

This is the first step in preprocessing pipelines. This step eliminates irrelevant data by defining and removing English stop words, removing non-alphabetic characters, numbers, linebreaks, and tabs from the collected datasets. It can be performed using a popular natural language toolkit in Python called NLTK library [6].

4.2.0.2 Tokenization

Tokenization is the process of splitting the sentence (user review) into small units called tokens (words) and then removing the punctuation marks. The

[2] https://github.com/NasriLadaa/Thesis$_E xperiments_2$023$_S WEN_N asri.git$

NLTK library [6] was used for text tokenization in all of the presented experiments.

4.2.0.3 Normalization:

Text normalization includes three subs steps. First, convert the text of the users' reviews to lowercase. Second, find each word's Part Of Speech (POS) tags. The third step and the last one is Lemmatization which returns the words to their original root using the WordNet library [7].

4.3 Feature engineering

In the presented experiments, the following feature extraction techniques were used to convert the text words into a format that can go along with machine learning and deep learning classifiers.

- Bag of Words (BOW)
- TF-IDF
- GPT3 vectorization
- Word2Vec vectorization
- GloVe vectorization
- FastText vectorization
- BERT vectorization
- D-BERT vectorization

4.4 Experiments with the Traditional Techniques

This section presents the results of all conducted experiments using traditional techniques for feature extraction and machine learning. The weighted average of recall, precision, F1-score, and accuracy were used to represent the classification performance in all experiments. Those techniques were trained on the training data and evaluated through the test data of Al-Kilani and the extended dataset.

4.4.1 Experiment set 1: Al-Kilani dataset (three classes)

This set of experiments includes two traditional classifiers, random forest, and Naïve Bayes, trained and evaluated on BOW and TF-IDF features extracted from the Al-Kilani dataset. It is worth mentioning here that the three classes of the AL-Kalani dataset are performance, usability, and reliability. The results of these four experiments are shown in Table 4.2.

TABLE 4.2: Al-Kilani dataset experiments performance metrics with traditional classifiers.

	Experiments with Al-Kilani dataset (3 classes)			
	Random Forest			
	Precision	Recall	F1-score	Accuracy
TF-IDF	0.57	0.58	0.57	0.58
BOW	0.56	0.56	0.56	0.56
	Naïve Bayes			
TF-IDF	0.60	0.51	0.57	0.55
BOW	0.58	0.52	0.54	0.52

According to the results in Table 4.2 and the graphical representation in Figure 4.1, the results are very close, with the Random Forest classifier and TF-IDF giving the best accuracy. Which are upper-performing Bag of Words with Random Forest and TF-IDF and BOW with Naïve Bayes. The Lowest accuracy was

0.15 for using TF-IDF with Naïve Bayes.

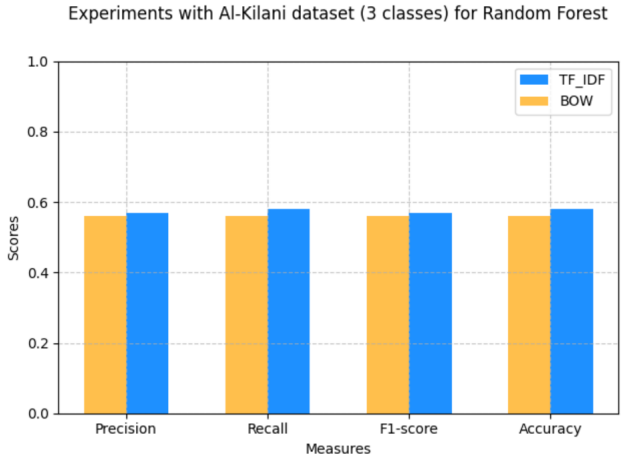

FIGURE 4.1: The results of traditional machine learning (RF) with TD-IDF and BOW for Al-Kilani dataset.

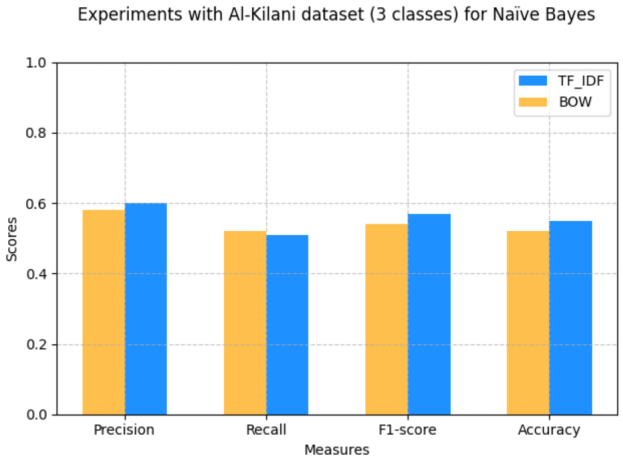

FIGURE 4.2: The results of traditional machine learning (Naïve Bayes) with TD-IDF and BOW for Al-Kilani dataset.

4.4.2 Experiment set 2: Extended Dataset (Al-Kilani + New datasets)

In this set of experiments, the same two traditional classifiers used in experiments set 1 described earlier, are re-trained and re-evaluated on the merged dataset, with the same number of classes.

TABLE 4.3: Merged dataset experiments performance metrics with traditional classifiers.

	Extended Dataset			
	Random Forest			
	Precision	Recall	F1-score	Accuracy
TF-IDF	0.66	0.64	0.63	0.64
BOW	0.66	0.64	0.63	0.64
	Naïve Bayes			
TF-IDF	0.67	0.35	0.39	0.35
BOW	0.65	0.60	0.61	0.60

According to the results in Table 4.3, the results for this set are approximately similar. Using TF-IDF and BOW with Random Forest gave the same accuracy of 0.64. BOW with Naïve Bayes gave 0.60 accuracy, which is so near to 0.64. The combination of TF-IDF with traditional classifier Naïve Bayes gave the lowest accuracy score of 0.35 noting that in experiment set 1, it was also the lowest among all other experiments.

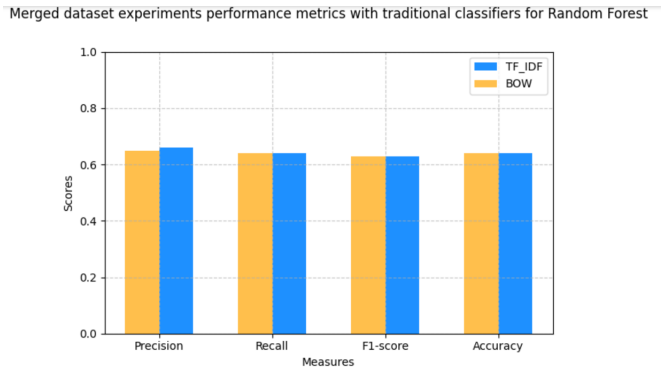

FIGURE 4.3: The results of traditional machine learning (RF) with TD-IDF and BOW for the Merged dataset.

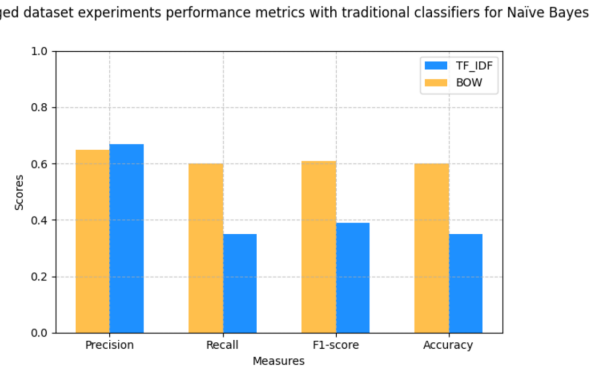

FIGURE 4.4: The results of traditional machine learning (Naïve Bayes) with TD-IDF and BOW using extended dataset.

4.4.3 Experiment set 3: Al-Kilani Dataset (6 classes)

In this set of experiments, the same four experiments are repeated on the Al-Kilani dataset, but, this time with six classes (Usability, Reliability, Performance, Security, Functional, and Others), instead of three. The results are presented in Table 4.4 and Figures 4.5, 4.6 .

TABLE 4.4: Al-Kilani dataset experiments performance metrics with traditional classifiers.

	Al-Kilani Dataset (6 classes)			
	Random Forest			
	Precision	Recall	F1-score	Accuracy
TF-IDF	0.53	0.49	0.47	0.49
BOW	0.46	0.44	0.41	0.44
	Naïve Bayes			
TF-IDF	0.52	0.38	0.4	0.38
BOW	0.50	0.48	0.48	0.48

According to the results in Table 4.4, the highest accuracy is 0.49 for the system which is using TF-IDF and Random Forest. The results of other experiments in this set are in the same range. TF-IDF and Naïve Bayes experiment final output result is the lowest for the third time among experiments in this set also among experiments set 1 and 2. The experiments set 1, 2, and 3 indicate among

all conducted experiments with the three datasets that the highest accuracy can be achieved by combining TF-IDF and Random forest for the traditional techniques.

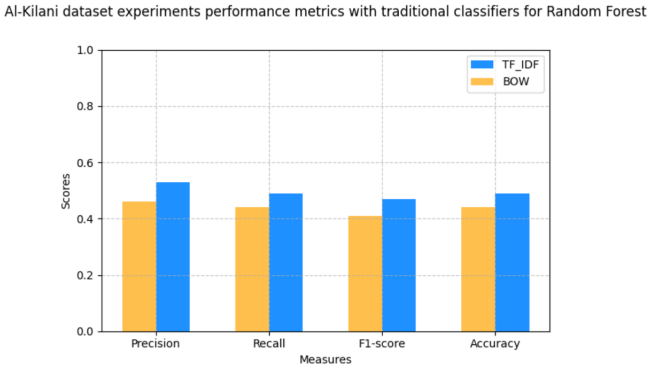

FIGURE 4.5: The results of traditional machine learning (RF) with TD-IDF and BOW for Al-Kilani dataset.

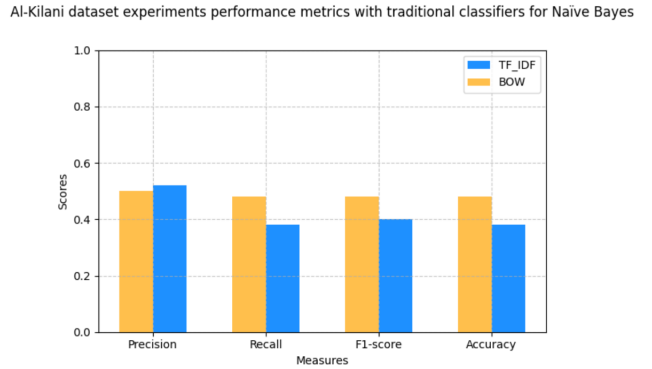

FIGURE 4.6: The results of traditional machine learning (Naïve Bayes) with TD-IDF and BOW for Al-Kilani dataset with 6 classes.

4.5 Experiments using Deep Learning

In this set of experiments, five deep-learning classifiers were used; ANN, DNN, and the two configurations of CNN, Single-Layer CNN (SL CNN) and Multiple-Layer CNN (ML CNN). Six types of features were used with the deep learning classifiers. These features are extracted using pre-trained deep-learning language models such as W2V, Glove, Fasttext, BERT, D-BERT, and GPT3.

4.5.1 Experiment set 4: Al-Kilani Dataset (three classes)

In this sub-section, the results of the experiments that use pre-trained models with deep learning techniques are presented. Al-Kilani dataset with the three classes, performance, reliability, and usability, were used to train and evaluate these systems.

As shown from the results in Table 4.5, the GTP3 pre-trained model with the ANN classifier achieved the highest accuracy (0.82) among all other experiments in this experiment set. The lowest accuracy, 0.7, was achieved by GloVe pre-trained model. Note that the accuracy of the other experiments in this set ranges between 0.7 and 0.82. It is also noticed from these results that the ANN with the six pre-trained model feature types significantly outperforms similar systems with traditional techniques and features.

TABLE 4.5: The results of ANN and DNN classifiers using Al-Kilani dataset with six classes.

	Al-Kilani Dataset (3 classes)			
		ANN		
	Precision	Recall	F1-score	Accuracy
W2V	0.71	0.71	0.71	0.71
GloVe	0.72	0.7	0.71	0.7
BERT	0.81	0.76	0.76	0.76
GPT3	0.82	0.82	0.82	0.82
FastText	0.75	0.71	0.72	0.71
D-bert	0.78	0.77	0.77	0.77
		DNN		
	Precision	Recall	F1-score	Accuracy
W2V	0.69	0.64	0.66	0.64
GloVe	0.68	0.62	0.63	0.62
BERT	0.71	0.67	0.66	0.67
GPT3	0.81	0.7	0.71	0.7
FastText	0.73	0.49	0.46	0.49
D-bert	0.76	0.75	0.75	0.75

Secondly, DNN experimented with W2V, Glove, Fasttext, BERT, D-BERT, and GPT3. According to the results in Table 4.5, The highest accuracy was provided by using D-bert with a DNN of 0.75. The lowest accuracy was 0.49 for Fast Text and DNN. For W2V, GloVe, BERT, and GPT3 the results were between 0.62 - 0.7, which is somehow near the highest, result experiment in this dataset. D-bert with DNN can take second place accuracy-wise after GPT3 with ANN from the above table.

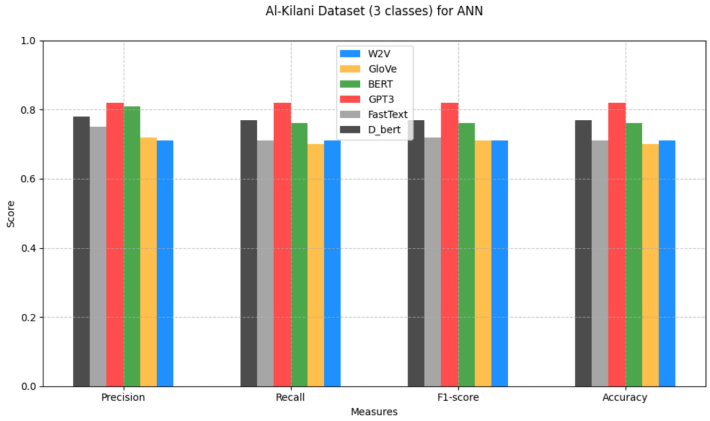

FIGURE 4.7: The results of ANN with pre-trained models for the Al-Kilani dataset.

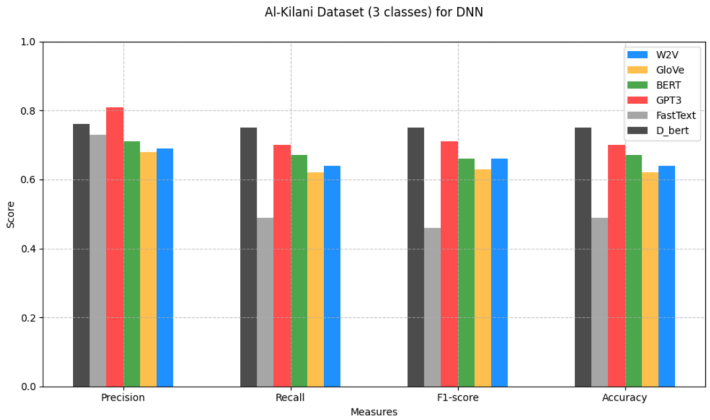

FIGURE 4.8: The results of DNN with pre-trained models for the Al-Kilani dataset.

Single Layer and Mutiple CNN experimented with W2V, Glove, Fasttext, BERT, D-BERT, and GPT3. The results were as below:

TABLE 4.6: SL CNN and ML CNN performance metrics with all pre-trained models for Al-Kilani dataset.

	Al-Kilani Dataset (3 classes)			
	SLCNN			
	Precision	Recall	F1-score	Accuracy
W2V	0.7	0.62	0.64	0.62
GloVe	0.71	0.68	0.69	0.68
BERT	0.62	0.63	0.58	0.63
GPT3	0.79	0.4	0.32	0.4
FastText	0.76	0.61	0.65	0.61
D-bert	0.51	0.33	0.26	0.33
	MLCNN			
	Precision	Recall	F1-score	Accuracy
W2V	0.68	0.68	0.68	0.68
GloVe	0.7	0.61	0.62	0.61
BERT	0.73	0.54	0.53	0.54
D-bert	0.75	0.74	0.72	0.74

According to the results in Table 4.6, GloVe with Single layer CNN performs the highest accuracy 0.68 note that W2V, BERT, and FastText perform from 0.62 (+1) (-1). The lowest accuracy was for GPT3 and D-BERT which performed 0.40, 0.33 sequentially. The result for using same feature engineering with ML-CNN was a little bit different as D-BERT performed the highest accuracy of 0.74. Meanwhile W2V and GloVe results was similar to SL-CNN the lowest accuracy performed by BERT (0.54).

ML-CNN and GPT3, FastText provided the lowest result among all experiments in all experiment sets therefore they were removed from the results. GPT3 with ANN provides the highest accuracy result.

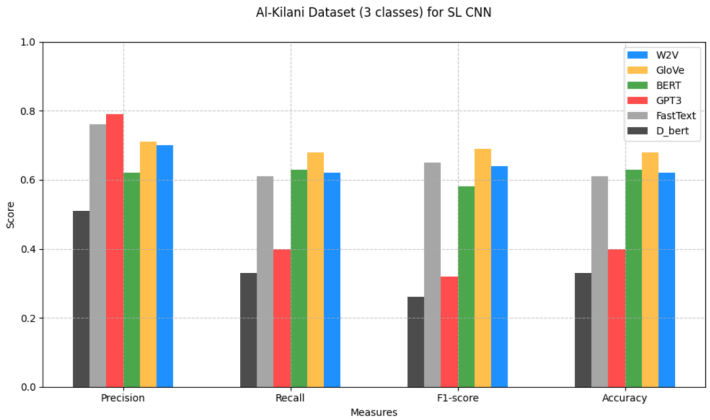

FIGURE 4.9: The results of SL-CNN with pre-trained models for the Al-Kilani dataset.

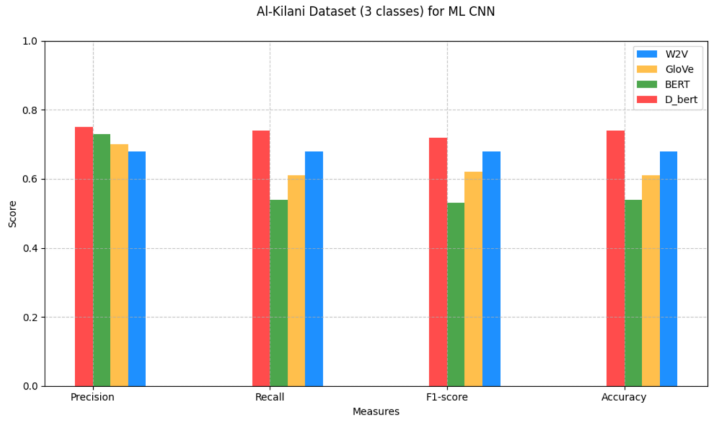

FIGURE 4.10: The results of ML-CNN with pre-trained models for the Al-Kilani dataset.

4.5.2 Experiment set 5: Extended Dataset

In this section, the results of using pre-trained models with the deep learning classifiers that were mentioned above. The used dataset is a merged dataset between the Al-Kilani dataset and an extended dataset on the three main classes (Performance, Reliability, and Usability).

ANN and DNN experimented with W2V, Glove, Fasttext, BERT, D-BERT, and GPT3. The results were as below:

According to the results in Table 4.7, For using ANN with feature engineering. W2V and GPT3 perform the highest accuracy 0.74. BERT and D-bert provide 0.70, 0.72 sequentially. Fast Text provided a result near to bert and D-bert of 0.66. The lowest accuracy provided by GloVe of 0.64. For using DNN, Bert acquired the highest accuracy over other feature engineering techniques of 0.73. This results is also near for using ANN with Bert. Other feature engineering techniques with DNN for this set provided accuracy between 0.61 - 0.69.

TABLE 4.7: ANN and DNN performance metrics with all pre-trained models for the extended dataset.

	Extended Dataset			
	ANN			
	Precision	Recall	F1-score	Accuracy
W2V	0.74	0.74	0.74	0.74
GloVe	0.67	0.64	0.64	0.64
BERT	0.72	0.7	0.7	0.7
GPT3	0.75	0.74	0.75	0.74
FastText	0.7	0.66	0.66	0.66
D-bert	0.73	0.72	0.72	0.72
	DNN			
	Precision	Recall	F1-score	Accuracy
W2V	0.69	0.65	0.65	0.65
GloVe	0.62	0.61	0.61	0.61
BERT	0.74	0.64	0.64	0.64
GPT3	0.72	0.69	0.69	0.69
FastText	0.71	0.61	0.62	0.61
D-bert	0.74	0.73	0.73	0.73

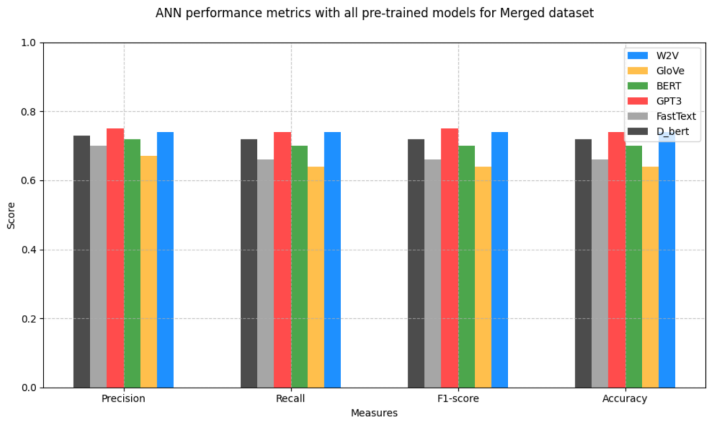

FIGURE 4.11: The results of ANN with pre-trained models for the extended dataset.

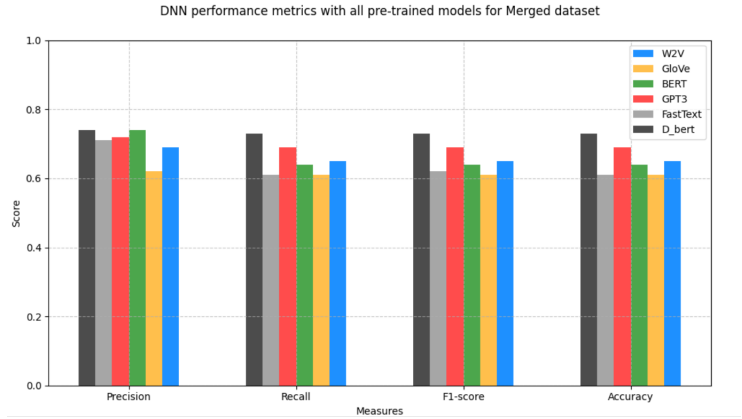

FIGURE 4.12: The results of DNN with pre-trained models for the extended dataset.

According to the results in Table 4.8, using SL CNN with W2V, Fast Text, and D-bert provided the highest accaucy of result 0.68. Meanwhile, GloVe provided a result that is near to W2V, Fast Text and D-bert results which is 0.65. Using SL-CNN with Bert provided low results. The accuracy of using GPT3 with Single CNN was 0.12. It was removed from the results table because it is too low. For using Mutiple layer CNN with BERT acquired the highest accuracy of the result of 0.71 which indicates clearly that BERT can perform better with ML CNN over SL CNN. GPT3 also provided high result of 0.70. Fast Text provided the lowest

result of 0.58. W2V, GloVe and D-bert were in the same range 0.62 (+) (-) 1.

TABLE 4.8: SL CNN and ML CNN performance metrics with all pre-trained for extended dataset.

	Extended Dataset			
	SLCNN			
	Precision	Recall	F1-score	Accuracy
W2V	0.7	0.68	0.69	0.68
GloVe	0.68	0.65	0.65	0.65
BERT	0.12	0.35	0.18	0.35
FastText	0.7	0.68	0.69	0.68
D-bert	0.7	0.68	0.67	0.68
	MLCNN			
	Precision	Recall	F1-score	Accuracy
W2V	0.66	0.61	0.62	0.61
GloVe	0.65	0.62	0.62	0.62
BERT	0.71	0.71	0.71	0.71
GPT3	0.75	0.7	0.71	0.7
FastText	0.63	0.58	0.58	0.58
D-bert	0.68	0.63	0.63	0.63

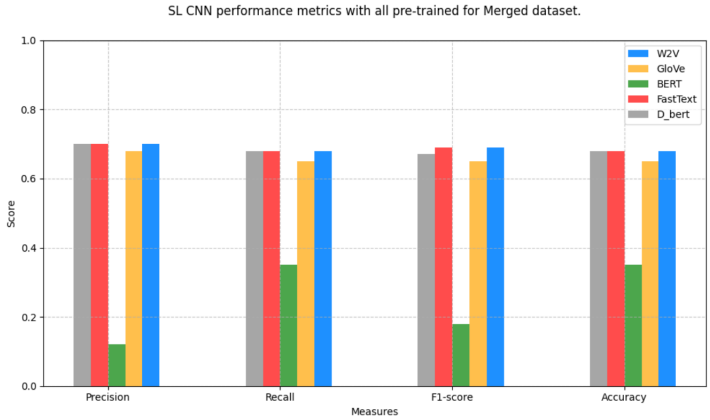

FIGURE 4.13: The results of SL CNN with pre-trained models for the extended dataset.

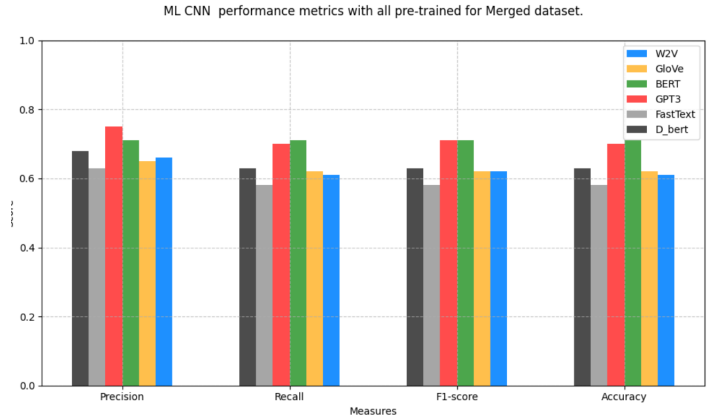

FIGURE 4.14: The results of ML CNN with pre-trained models for the extended dataset.

4.5.3 Experiment set 6: Al-Kilani Dataset (6 classes)

In this section, the results of using pre-trained models with the deep learning classifiers that were mentioned above. The used dataset is a Al-Kilani dataset on the six main classes (reliability, usability, performance, security, functional requirements, and others)

According to the results in Table 4.9, The overall acauccy for all the feature engineering techniques are lower than other experiments set. Using ANN with

GPT3 provided the highest accuracy for result of 0.56. Other techniques provided similar results of accuracy from 0.41 - 0.49.

TABLE 4.9: ANN and DNN performance metrics with all pre-trained models for Al-Kilani dataset (6 classes).

	Al-Kilani Dataset (6 classes)			
		ANN		
	Precision	Recall	F1-score	Accuracy
W2V	0.51	0.49	0.49	0.49
GloVe	0.43	0.41	0.41	0.41
BERT	0.49	0.49	0.49	0.49
GPT3	0.58	0.56	0.55	0.56
FastText	0.45	0.41	0.41	0.41
D-bert	0.49	0.49	0.48	0.49
		DNN		
	Precision	Recall	F1-score	Accuracy
W2V	0.43	0.39	0.33	0.39
GloVe	0.37	0.38	0.35	0.38
BERT	0.49	0.44	0.42	0.44
GPT3	0.44	0.42	0.38	0.42
FastText	0.39	0.34	0.32	0.34
D-bert	0.44	0.44	0.43	0.44

FIGURE 4.15: The results of ANN with pre-trained models for the Al-Kilani dataset (6 classes).

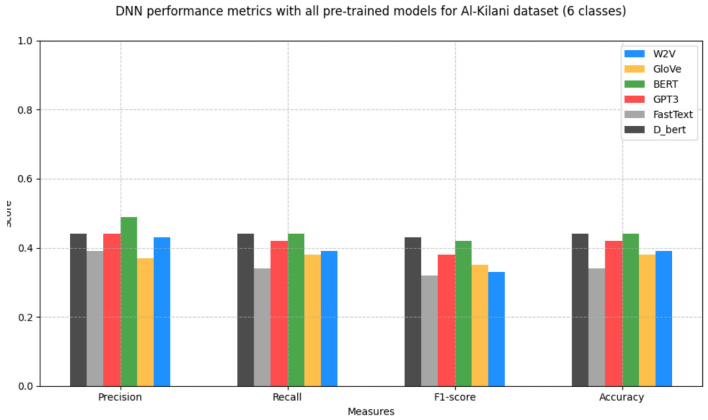

FIGURE 4.16: The results of DNN with pre-trained models for the Al-Kilani dataset (6 classes).

According to the results in Table 4.10, The highest results were acquired by using Fast Text and Single Layer CNN with 0.48 accuracy. W2V and SL CNN accuracy was near to FastText with 0.46. The Lowest accuracy was 0.34 for using Bert and SL CNN together. The overall accuracy for other feature engineering techniques was between 0.39 - 0.41. Regarding Multiple-layer CNN (ML CNN) the highest accuracy was 0.49 for using bert and ML CNN. The lowest accuracy was 0.33 for Fast Text. The overall accuracy for other feature engineering techniques was between 0.43 - 0.45.

TABLE 4.10: SL CNN and ML CNN performance metrics with all pre-trained for Al-Kilani dataset (6 classes).

	Al-Kilani Dataset (6 classes)			
	SLCNN			
	Precision	Recall	F1-score	Accuracy
W2V	0.47	0.46	0.45	0.46
GloVe	0.46	0.41	0.39	0.41
BERT	0.3	0.34	0.29	0.34
GPT3	0.51	0.39	0.4	0.39
FastText	0.52	0.48	0.48	0.48
D-bert	0.43	0.43	0.42	0.43
	MLCNN			
	Precision	Recall	F1-score	Accuracy
W2V	0.5	0.43	0.4	0.43
GloVe	0.44	0.44	0.44	0.44
BERT	0.48	0.49	0.47	0.49
GPT3	0.42	0.43	0.4	0.43
FastText	0.34	0.33	0.33	0.33
D-bert	0.45	0.45	0.43	0.45

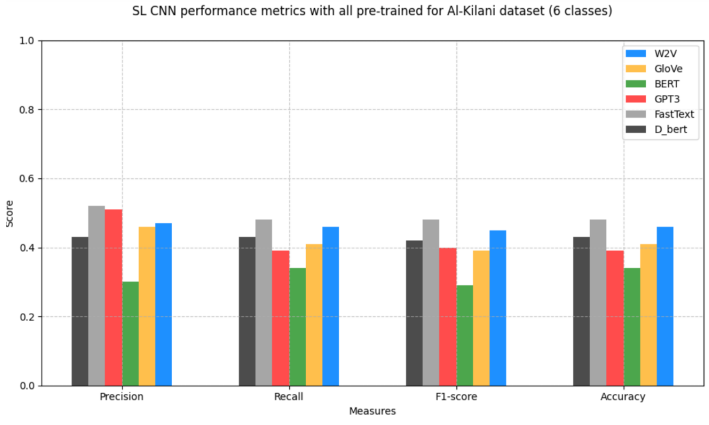

FIGURE 4.17: The results of SL CNN with pre-trained models for the Al-Kilani dataset (6 classes).

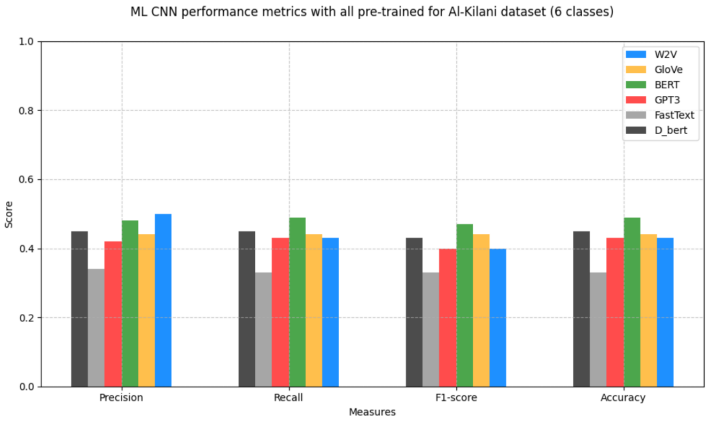

FIGURE 4.18: The results of ML CNN with pre-trained models for the Al-Kilani dataset (6 classes).

4.6 Comparison with Al-Kilani baseline system

As stated earlier in the introduction chapter, one of the main objectives of this thesis is to study how well the deep learning techniques for feature extraction and machine learning classification in recognizing the most common requirements from the user's reviews, compared with the traditional techniques used by Al-Kilani et. al. [8]. Therefore, we used the dataset and the system described in Al-Kilani's work as a baseline for our proposed systems. In our systems, we tried to incorporate the deep learning techniques in the feature extraction and classification stages. In addition, the dataset was extended by adding more user reviews form different domains and for different requirement classes. Table 4.11, 4.12 summarizes the accuracies of the best systems of each experiment sets described earlier in this chapter. It also includes the best accuracies of the results obtained by Al-Kilani with random forest and Naïve Bayes classification algorithms. Backing to RQ1 in chapter one we observed that using word embedding techniques doesn't provide significant improvement over the traditional techniques. The reason for that can be because Al Kilani used different parameters and settings for the experiment such as depth and numbers of trees. The reason also could be because we used different domains in our dataset meanwhile Alkilani used only healthcare user reviews.

Backing to RQ2 below Table 4.11 we observed that using three classes merged dataset the ANN classifier provides a higher accuracy of 0.75 over other classifiers. Secondly, ML CNN provides an accuracy of 0.74 and DNN provides an accuracy of 0.73. We notice also that using deep learning techniques results are a little bit more effective than traditional techniques compared to results in the Al Kilani system.

TABLE 4.11: Comparison with Al-Kilani results

	Al-Kilani Systems				
	Accuracy	Precision	Recall	F-score	No of classes
Naïve Bayes	0.68	0.68	0.69	0.68	3
Random Forest	0.72	0.73	0.72	0.69	3
	Our proposed systems				
	Accuracy	Precision	Recall	F-score	No of classes
Naïve Bayes	0.67	0.65	0.60	0.61	3
Random Forest	0.64	0.66	0.64	0.63	3
ANN	0.75	0.74	0.75	0.74	3
DNN	0.73	0.74	0.73	0.73	3
SL-CNN	0.68	0.7	0.68	0.69	3
ML-CNN	0.74	0.74	0.74	0.74	3
Naïve Bayes	0.48	0.48	0.48	0.52	6
Random Forest	0.49	0.53	0.49	0.47	6
ANN	0.56	0.58	0.56	0.55	6
DNN	0.44	0.49	0.44	0.42	6
SL-CNN	0.48	0.52	0.48	0.48	6
ML-CNN	0.49	0.48	0.49	0.47	6

For RQ3 we observed that using deep learning techniques are more effective than traditional in recognizing the three and six software requirements but for the most six classes mentioned previously, the accuracy is less as shown in Table 4.12. The result for RQ2 and RQ3 can be justified because of number of user reviews is not enough. Making the user reviews dataset larger will consolidate the result. Also using user reviews with no constant number of words, we had some user reviews with 30 words meanwhile others included 3 words only.

TABLE 4.12: Comparison of traditional techniques based on accuracy.

	Al-Kilani Dataset (6 classes)	Extended Dataset	Al-Kilani Dataset (3 classes)
		Random Forest	
TF-IDF	0.49	0.64	0.58
BOW	0.44	0.64	0.56
		Naïve Bayes	
TF-IDF	0.38	0.35	0.55
BOW	0.48	0.6	0.52
		ANN	
W2V	0.49	0.74	0.71
GloVe	0.41	0.64	0.7
BERT	0.49	0.7	0.76
GPT3	0.56	0.74	0.82
FastText	0.41	0.66	0.71
D-bert	0.49	0.72	0.77

Chapter 5

Conclusion and future work

5.1 Conclusion

In this thesis, we presented our proposed system for automatic classification of apps user's reviews extracted from Google play store. Around 10,600 reviews were retrieved from many different applications from different domains. A round 2,622 reviews were manually annotated by four software engineering experts, indicating their confidence for each annotation. Some of the commonly used traditional classifiers such as Naïve Bayes, Random Forest, and ordinary ANN were used with the most common traditional feature extraction techniques such BoW and TF-IDF were used for recognizing some of the software requirement classes from the user's reviews. The traditional techniques were compared with some of the common deep learning algorithms such as DNN and two configurations of the CNN with some of the word embedding features such as W2V, BERT, D-bert, Glove, Fasttext, GPT3, and others. A set of experiments were conducted using Al-Kilani dataset and the extended dataset, which includes Al-Kilani dataset and the collected 1693 reviews. To make the results

comparable with the previous results achieved by Al-Kilani, some of the experiments were conducted to recognize three major classes (reliability, usability, and performance), and the others are designed to recognize six classes (reliability, usability, performance, security, functional requirements, and others). For the three classes experiments, the results show that the deep learning based techniques outperform the traditional techniques. The best accuracy achieved by the Random Forest classifier trained and evaluated on the TF-IDF features is 58%, compared with 82% achieved by the DNN classifier trained on GPT3. Similarly, the deep learning techniques (SL CNN, ML CNN, DNN) outperform the traditional techniques in the six classes experiments, with the best accuracy of 56% compared with 49% achieved by the random forest.

5.2 Future Work

The research of adopted NLP techniques and ML algorithm into requirement classification is still continuing. Two primary direction can be investigated to extend our work. First, we plan to expand the number of adopted software requirement categories, such as (mutability, solubility, etc.. Second direction, we can continually investigate the effectiveness of other ML approaches in recognizing software requirements from user's reviews such as recurrent neural network (RNN) and fusion multi models.

Chapter 6

Appendix A

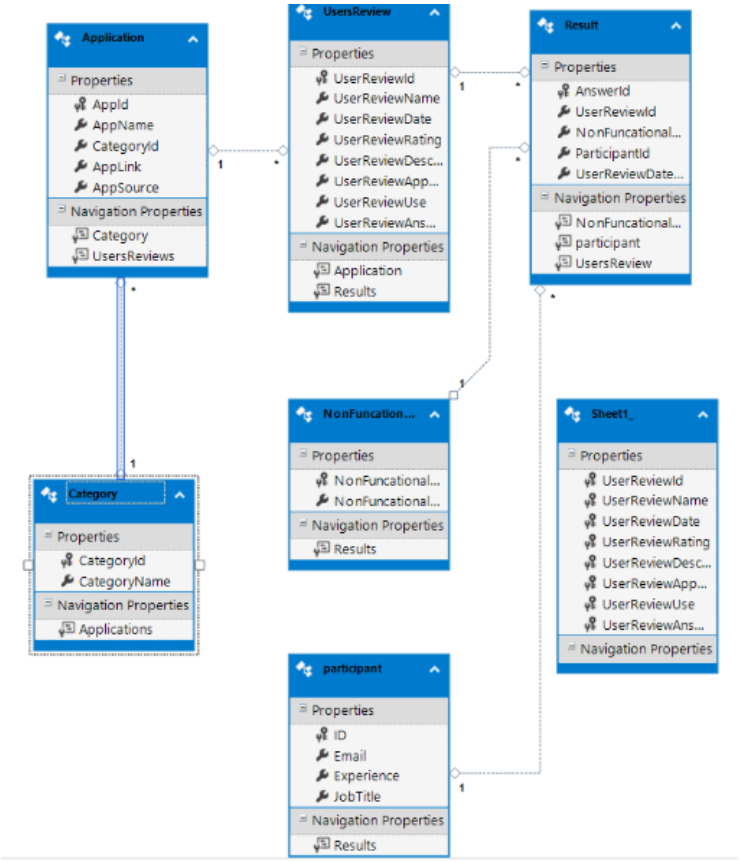

FIGURE 6.1: UML Diagram

Bibliography

[1] In: *Selenium automates browsers. https://www.selenium.dev/. (Accessed on 26/12/2022).*

[2] In: *GloVe: Global Vectors for Word Representation. https://nlp.stanford.edu/projects/glove/. (Accessed on 04/01/2022).*

[3] In: *TensorFlow code and pre-trained models for BERT. https://github.com/google-research/bert. (Accessed on 04/01/2022).*

[4] In: *The GPT-3 Architecture. https://dugas.ch/artificial*$_{c}uriosity/GPT_{a}rchitecture.html.(Accessedon$

[5] In: *English word vectors. https://fasttext.cc/docs/en/english-vectors.html. (Accessed on 04/01/2022).*

[6] In: *NLTK python main Library , https://www.nltk.org/ [Accessed on May 20, 2023].*

[7] In: *Wordnet python Library, https://www.nltk.org/howto/wordnet.html [Accessed on May 20,2023].*

[8] Nadeem Al Kilani, Rami Tailakh, and Abualsoud Hanani. "Automatic classification of apps reviews for requirement engineering: Exploring the customers need from healthcare applications". In: *2019 sixth international conference on social networks analysis, management and security (SNAMS)*. IEEE. 2019, pp. 541–548.

[9] Rishi Chandy and Haijie Gu. "Identifying spam in the iOS app store". In: *Proceedings of the 2nd Joint WICOW/AIRWeb Workshop on Web Quality*. 2012, pp. 56–59.

[10] Ning Chen et al. "AR-miner: mining informative reviews for developers from mobile app marketplace". In: *Proceedings of the 36th international conference on software engineering*. 2014, pp. 767–778.

[11] Nejdet Dogru and Abdulhamit Subasi. "Traffic accident detection using random forest classifier". In: *2018 15th learning and technology conference (L&T)*. IEEE. 2018, pp. 40–45.

[12] Joseph L Fleiss and Jacob Cohen. "The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability". In: *Educational and psychological measurement* 33.3 (1973), pp. 613–619.

[13] Joseph L Fleiss, Bruce Levin, Myunghee Cho Paik, et al. "The measurement of interrater agreement". In: *Statistical methods for rates and proportions* 2.212-236 (1981), pp. 22–23.

[14] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences". In: *Minds and Machines* 30 (2020), pp. 681–694.

[15] Giuseppe Futia et al. "Training neural language models with sparql queries for semi-automatic semantic mapping". In: *Procedia Computer Science* 137 (2018), pp. 187–198.

[16] Alex Graves and Alex Graves. "Long short-term memory". In: *Supervised sequence labelling with recurrent neural networks* (2012), pp. 37–45.

[17] Mutian He et al. "On the role of conceptualization in commonsense knowledge graph construction". In: *arXiv preprint arXiv:2003.03239* (2020).

[18] Yuan Huang et al. “D-BERT: Incorporating dependency-based attention into BERT for relation extraction”. In: *CAAI Transactions on Intelligence Technology* 6.4 (2021), pp. 417–425.

[19] Claudia Iacob and Rachel Harrison. “Retrieving and analyzing mobile apps feature requests from online reviews”. In: *2013 10th working conference on mining software repositories (MSR)*. IEEE. 2013, pp. 41–44.

[20] Jie Li et al. “A novel medical text classification model with Kalman filter for clinical decision making”. In: *Biomedical Signal Processing and Control* 82 (2023), p. 104503.

[21] Chang Liu et al. “Semantic features based N-best rescoring methods for automatic speech recognition”. In: *Applied Sciences* 9.23 (2019), p. 5053.

[22] Lefteris Loukas et al. “Breaking the bank with chatgpt: Few-shot text classification for finance”. In: *arXiv preprint arXiv:2308.14634* (2023).

[23] Mengmeng Lu and Peng 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, pp. 344–353.

[24] Walid Maalej and Hadeer Nabil. “Bug report, feature request, or simply praise? on automatically classifying app reviews”. In: *2015 IEEE 23rd international requirements engineering conference (RE)*. IEEE. 2015, pp. 116–125.

[25] Walid Maalej et al. *On the automatic classification of app reviews*. 2016.

[26] Shervin Minaee et al. “Deep learning–based text classification: a comprehensive review”. In: *ACM computing surveys (CSUR)* 54.3 (2021), pp. 1–40.

[27] Dennis Pagano and Walid Maalej. “User feedback in the appstore: An empirical study”. In: *2013 21st IEEE international requirements engineering conference (RE)*. IEEE. 2013, pp. 125–134.

[28] David MW Powers. “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”. In: *arXiv preprint arXiv:2010.16061* (2020).

[29] IOf Standardization. “Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): Measurement of System and Software Product Quality”. In: *ISO, Geneva, Switzerland* (2016).

[30] Styawati Styawati et al. “Sentiment analysis on online transportation reviews using Word2Vec text embedding model feature extraction and support vector machine (SVM) algorithm”. In: *2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*. IEEE. 2022, pp. 163–167.

[31] Muhammad Umer et al. “Impact of convolutional neural network and FastText embedding on text classification”. In: *Multimedia Tools and Applications* 82.4 (2023), pp. 5569–5585.

[32] Li Yang et al. “Sentiment analysis for E-commerce product reviews in Chinese based on sentiment lexicon and deep learning”. In: *IEEE access* 8 (2020), pp. 23522–23530.

[33] Yiming Yang and Jan O Pedersen. “A comparative study on feature selection in text categorization”. In: *Icml*. Vol. 97. 412-420. Nashville, TN, USA. 1997, p. 35.

[34] Rui Zhu, Delu Yang, and Yang Li. “Learning improved semantic representations with tree-structured lstm for hashtag recommendation: An experimental study”. In: *Information* 10.4 (2019), p. 127.