FACULTY OF ENGINEERING AND TECHNOLOGY MASTER OF SOFTWARE ENGINEERING

Detecting and classifying Software Bugs and Requirements in Arabic Mobile App Reviews

Author:

Qutaiba Mustafa

Supervisor:

Dr. Mustafa Jarrar

This thesis was submitted in fulfillment of the requirements for the degree of Master of Science in Software Engineering from the Faculty of Graduate Studies at Birzeit University, Palestine

Aug 10, 2021

FACULTY OF ENGINEERING AND TECHNOLOGY

MASTER OF SOFTWARE ENGINEERING

Master Thesis

Detecting and classifying Software Bugs and Requirements in Arabic Mobile App Reviews

اكتشاف وتحليل متطلبات البرمجيات واخطائها في تطبيقات الهواتف الذكية باللغة العربية

Author:

Qutaiba Mustafa (1165486)

Supervisor

Dr. Mustafa Jarrar

Committee

Dr. Mustafa Jarrar

Dr. Radi Jarrar

Dr. Adel Taweel

This thesis was submitted in fulfillment of the requirements for the degree of Master of Science in Software Engineering from the Faculty of Graduate Studies at Birzeit University, Palestine

Aug 10, 2021

جامعة بيرزيت

BIRZEIT UNIVERSITY

Approved by the thesis committee:

Dr. Mustafa Jarrar, Birzeit University

Dr. Radi Jarrar, Birzeit University

Dr. Adel Taweel, Birzeit University

Date approved:

Declaration of Authorship

I, Qutaiba Mustafa, declare that this thesis titled "Detecting and classifying Software Bugs and Requirements in Arabic Mobile app Reviews" 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's 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 primary 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.

Signed:

____

Date:

____

Abstract

Collecting software requirements typically involve users through interviews, focus groups, and workshops. Recently, software operators started to collect software requirements, users' feedback, and bug reports from reviews and feedback systems. However, The users usually do not pay attention to review applications after downloading them, and regular users may not know how to report bugs to the software operator, and they might write many useless reviews.

Application distributed systems (known as App stores) allows users to submit their reviews on the app they downloaded in several forms: score rating, like or dislike, and text reviews. Recent studies show that the text reviews could include informative information for the app developers, such as potential software bugs, user requirements, sentiments about the application, or some ideas for improvements.

In this thesis, I propose an automatic method to analyze and classify user text reviews into five main classes: Software bugs, Software users' requirements, Nonfunctional software requirements, not clear (which includes all the reviews that I think it is informative for the app stockholders) and not relevant. I built a corpus of about 10k Arabic reviews collected from 5 different applications and classified them into 33 classes. Then, I fine-tuned an Arabic pre-trained BERT model (state-of-the-art Deep learning architecture for NLP) using the corpus and conducted three experiments on three different versions of the dataset where two versions were generated by grouping the original 33 classes into five classes and three classes. The model achieved 99%, 94%, 95% accuracy in the three different experiments.

ملخص

جمع متطلبات البرمجيات عادة ما يتضمن إشراك مستخدمي هذه التطبيقات لأخذ تغذيتهم الراجعة ليتم تعديل وتطوير هذه التطبيقات بما يتناسب مع احتياجات مستخدميها. قديما، كان يتم جمع تغذية المستخدمين الراجعة من خلال إجراء مقابلات مع المستخدمين أو من خلال عقد حلقات نقاش وتفكير. ومع النمو الكبير في تطوير تطبيقات الهواتف الذكية وازدياد أعداد مستخدمي هذه التطبيقات بشكل هائل، فقد أصبح من الصعب استخدام الطرق التقليدية لجمع احتياجات المستخدمين وأخذ تغذيتهم الراجعة بالحسبان، فقد توجه مطورو هذه التطبيقات إلى استخدام أنظمة جمع التغذية الراجعة من المستخدمين، ومع العدد الكبير في أعداد المستخدمين وتنوع لغاتهم ولهجاتهم فإنه من الصعب على مطوري التطبيقات مراجعة وتحليل الأعداد الكبيرة من تغذية المستخدمين الراجعة من خلال هذه الأنظمة.

تعد متاجر التطبيقات أحد أشهر أنظمة جمع تغذية المستخدمين الراجعة لتطبيقات الهواتف الذكية، حيث يستطيع مستخدمو التطبيقات من خلال هذه المتاجر تقديم تغذيتهم الرجعة لمطوري التطبيق والمستخدمين الآخرين بعدة أشكال منها: تقييم التطبيق بعدد من النجوم، إبداء إعجابهم بهذا التطبيق أو عدمه أو من خلال كتابة تغذيتهم الراجعة على شكل نص بلغتهم. وقد وجدت بعض الدراسات الحديثة أن تغذية المستخدمين النصية على متاجر التطبيقات قد تحتوي على العديد من المعلومات المفيدة لمطوري هذه التطبيقات، فقد يشير المستخدمون من خلال هذه النصوص إلى أخطاء في التطبيق أو اقتراحات وأفكار وظيفية أو غير وظيفية لتطوير التطبيق.

وفي هذا البحث العلمي، أعتزم استخدام تقنيات الذكاء الصناعي وأنظمة الشبكات العصبونية المدربة لبناء أداة تقوم بتحليل الأعداد الهائلة من تغذية المستخدمين الراجعة النصية المكتوبة باللغة العربية بشكل آلي وتصنيفها إلى خمس فئات رئيسة: الأخطاء البرمجية، متطلبات المستخدمين، متطلبات المستخدمين غير الوظيفية، النصوص العامة التي لا تحتوي أي معلومات مفيدة، النصوص التي قد تحتوي معلومات مفيدة، ولكنها غير واضحة وتحتاج الى التواصل مع المستخدمين لفهمها. لقد قمت بجمع مدونة نصية تحتوي على حوالي عشرة الاف تغذية راجعة من ٥ تطبيقات مختلفة وقمت بتصنيف هذه النصوص إلى ٣٣ فئة محددة. وقمت بعد ذلك بضبط نظام بيرت للتعلم العميق لكي يستطيع فهم وتصنيف التغذيات الراجعة بطريقة آلية. لقد قمت بإجراء ٣ تجارب مختلفة على المدونة النصية حيث ان التجربة الأولى كانت باستخدام ال ٣٣ فئة، اما في التجربة الثانية فقد قمت بتصنيف المدونة النصية الى ٥ فئات محددة، والى ٣ فئات أخرى في التجربة الثالثة، وحصلت بذلك على دقة وصلت الى ٩٩٪ و٩٤٪ و٩٥٪ في الثلاث تجارب.

Acknowledgments

First and foremost, I am deeply grateful to my parents, brothers, and sisters for their unlimited support and courage at every stage of my life.

I want to extend my sincere thanks to my supervisor Dr. Mustafa Jarrar for his insightful guidance, comments, suggestions, and patience at every stage of this research. I would like to also thank Dr. Mohamad Khalilia for his comments, suggestions, and feedback in Fine-tuning BERT.

Special thanks to all the teaching crew of the Software Engineering master program at Birzeit University for their continuous efforts and support. Furthermore, my appreciation goes to my friends and colleagues at both university and work who supported me at all times.

Table of Contents

References ... 92

List of Figures

List of Tables

List of Equations

List of Abbreviations

BERT: Bidirectional Encoder Representations from Transformers.

NLP: Natural Language Processing.

MLM: Masked Language Modeling.

AUROC: Area Under the Receiver Operating Characteristic curve.

API: Application Programming Interface.

iOS: iPhone Operating System.

AI: Artificial Intelligence.

ML: Machine Learning.

GPU: Graphics Processing Unit.

CPU: Central Processing Unit.

TPU: Tensor Processing Unit.

RAM: Random Access Memory.

SDLC: Software Development Life Cycle.

NSP: Next Sentence Prediction.

NER: Named Entity Recognition.

SVM: Support Vector Machine.

TF-IDF: Term Frequency-Inverse Document Frequency.

LDA: Latent Dirichlet Allocation.

KNN: K-Nearest Neighbors.

NB: Naïve Bayes.

CUDA: Compute Unified Device Architecture.

OSCAR: Open Super-large Crawled Aggregated corpus.

OSIAN: Open-Source International Arabic News corpus.

TP: True Positive.

TN: True Negative.

FN: False Negative.

FP: False Positive.

Chapter 1

Introduction

Software reviews are an essential part of the software development life cycle (SDLC). They help software owners and developers validate the software's quality and make sure user's needs are fulfilled. They also help in improving the software and focusing on the most features used by the users. However, in App Store reviews, several significant difficulties can limit the ability of the software analysts to analyze the software reviews. First, many reviews are added to the app stores every day, which requires a large amount of effort to be analyzed. A recent study found that iOS users submit on average 22 reviews per day per app [1]. Top-rated apps such as Facebook get more than 4000 reviews per day [2]. Second, the text reviews vary widely, which makes it challenging to analyze and classify the reviews.

1.1 Introduction and Motivation

According to App Annie statistical report in 2020 [3], by the end of 2019 and with over 2.7 billion smartphone users, there were 204 billion downloads for mobile apps worldwide (grown by 45% since 2016), 120 billion dollars were spent on the mobile application by the consumer, users spend 3.7 hours in average using mobile application per day. The statistics also show that the number of applications on the primary two App stores is over 5 million applications (2.2 available on the Apple App Store, and 2.8 available on Google Play Store). The vast increase in the mobile application industry introduced a high level of competition. Therefore, app review plays a crucial role in the success of the mobile application by helping the application stockholders improving the application, understand the user needs and discover any existing issues that bother the application users.

A group of researchers in 2017 [4] conducted a survey study to describe and compare the areas of research that have been done on analyzing different aspects of the app stores such as API, Feature, Releases, Security, and mining the reviews in app stores by searching specific terms in the following search repository: Google Scholar, Scopus, JSTOR, ACM, IEEE, and arXiv. Their study shows strong growth in the number of published papers related to the app stores in 2015 compared with the past few years.

Collecting customer feedback helps the application stockholders improve their application and provide a better user experience; it also helps increase the user's retention by valuing the users' opinions and making the application fulfill their needs. Customer feedback helps the application developers identify their roadmap and priorities the app requirements to increase customer satisfaction, which also helps them improve the app marketing by focusing on the features and improvements requested and needed by the customers.

Analyzing software reviews manually is a complex process [5] due to several reasons:

- Expensive: users add millions of reviews daily, which requires many resources to analyze and filter their reviews.
- Time Consuming: many of the reviews do not contain any valuable information for the software analysts; filtering these reviews might take much time.
- Languages, dialects, and culture differences: App stores allow the users to add reviews in any language or dialect. Which makes it harder for the software stockholders to understand these reviews, which come from millions of people worldwide in different dialects and cultures.

The Arabic language has an enormous number of users on the internet [6]. Reviews in the Arabic language in App stores are expected to be informative regarding software requirement engineering [7]. Some of these reviews could automatically report bugs in large and standard software systems used by Arab users. I chose to work with the Arabic language due to two factors. First, Arabic language analysis is of growing importance due to its already large-scale audience. Second, the Arabic language is challenging due to multiple dialects and diversity, which resulted in fewer tools available currently to analyze, mine, and classify Arabic texts compared to other languages. The Arabic language is striking because of its history, the strategic importance of its people, the region they occupy, and its cultural and literary heritage [8].

This thesis propose an automatic method to analyze and classify Arabic reviews from 5 applications in 5 different business domains, the applications were selected from Google Play and App Store based on the high number of Arabic reviews added to them according to Appbot[1] (a tool to analyze app stores reviews). 10K reviews were collected and will be manually analyzed and annotated with one or more classes of each review. Then I will apply the proposed automatic method to these reviews to identify the performance of the automated method against the manual classification.

1.2 Research objectives and Problem statement

Due to the complexity in the manual analysis of the mobile app's reviews and the difficulties in analyzing the Arabic language [9] [10] [11], I propose this automatic method to help the app stakeholders to analyze the Arabic user reviews on their application in an efficient, accurate, and cost-effective way. The app owners will see the reviews classified into different classes, which will help them

[1] *https://appbot.co/*

understand the user's needs, fix the undiscovered bugs, and improve the application roadmap to satisfy user's requirements and achieve success.

The main objectives for this research are:

- Identifying how the App Store reviews could be meaningful and useful for the application Quality Assurance and Requirements engineers by extracting only the reviews that contain some potential bugs or user needs and comparing these reviews to the total number of reviews.
- Understands and analyzes users' reviews into classes, which will help the application owners to improve the application and satisfy the user's needs.
- The currently available tools and research on analyzing and classifying Arabic text are limited due to the complexity of the language and the diversity in its dialects. In this research, I will focus on the Arabic language and dialects and measure the performance of the proposed automated method.

1.3 Research Overview

The rest of this thesis is divided into the following:

- **Background**: A review of the theory and tools and behind this thesis, in this section, I will cover several topics, including Neural networks, some optimization algorithms used in this thesis, BERT Architecture [12], and some AI and ML concepts.
- **Related Work**: This chapter focuses on the other studies related to the fields I focus on in this thesis; this includes other papers about collecting user's requirements, analyzing Arabic language, and other papers about Neural networks in general and BERT models in particular. In this chapter, I also

the areas where improvements can be made and discuss the limitations of these studies.

- **Data collection, analysis**: This chapter describes and discusses the data collected in this research regarding methodology, quality, and quantity. All the manual classification and classes will be described in this section, along with examples.
- **Research Methodology And Experiment Setup**: This chapter describes how this thesis will answer the research question by describing the experiments I did and link each one to a research question. In addition to that, this chapter will also describe the experiment setup.
- **Experiments Results and Analysis**: This chapter presents the results of the experiment.
- **Conclusion and Discussion**: In this chapter, I will discuss how the results from the previous section answered the research question. In addition, I will discuss threats to validity and discuss the future work and some recommendations.

1.4 Research Activities

To ensure the diversity in the reviews and user's requirements, reviews were collected from different Arabic business domains and different app stores stores (the stores of the most common mobile operating systems: IOS and Android [13]). The following activities will also be carried out:

- Annotate the dataset manually and build a corpus of 33 classes.
- Grouping the 33 classes into five main classes (Software bugs, Non-functional requirements. User's requirements, not relevant, and not clear –

see section 4.2 for more details) and prepare a different version of the dataset where the classes are the main five groups.

- Grouping the 33 classes into three main classes: Informative reviews (reviews that can be meaningfully useful for app Quality Assurance and requirements Engineers to identify potential bugs or user needs), Uninformative reviews, and not clear – see section 4.2 for more details) and prepare a different version of the dataset where the classes are the main five groups. The results of this experiment (see section 6.6) answer the first research question of whether the Apps reviews can include some helpful information for the App stockholders or not.
- Setup the experiment and conduct it for the three versions of the dataset (the main dataset annotated with the 33 corpus, the main dataset annotated with the 5 grouped classes and the main dataset annotated with the 3 grouped classes). Each experiment was conducted several times with different configurations to get the best results.

All the experiments were conducted using CoLab[3] (Colaboratory) under the same circumstances with the exact GPU resource specification and the same RAM limitation, see section 5.1 for details.

[3] https://colab.research.google.com/notebooks/intro.ipynb

Chapter 2

Background

2.1 App distribution platform

App distributed platforms are the electronic software distributed markets for several mobile devices like smartphones and tablets [9]. According to App Annie statistical report in 2020 [3], by the end of 2019 and with over 2.7 billion smartphone users, there were 204 billion downloads for mobile apps worldwide (grown by 45% since 2016), 120 billion dollars were spent on mobile applications by the consumer, users spends 3.7 hours on average using mobile applications per day. App Annie statistical report also shows that the number of applications on the primary two App stores is over 5 million applications (2.2 available on the Apple App Store, and 2.8 available on Google Play Store [3]).

Analyzing the users' reviews and feedback on the apps in the app platforms is a necessary process to improve the app development and increase user satisfaction [5]. Some of these reviews could contain informative information for the app developers. This research focused on classifying the reviews into Users requirements, Non-functional requirements, and Software bugs to make it easier for app developers to analyze these reviews and take action.

2.2 Software requirements and bugs

Creating and collecting software requirements is a complex task as it consists of several processes such as elicitation, analysis, specification, validation, and management. It is one of the primary and vital stages in any software development process where high-quality and precise requirements help mitigating the financial risks and keeps the project on the specified road map [2].

Software requirements are usually divided into three types:

- Business Requirements: This includes the high-level goals and objectives of the software [14].
- Users' requirements: This describes what the user needs the software to do, user requirements are usually collected from the users, or it comes in a specific user requirements document where the user of the software signs this document [15].
- System requirements: This describes the specifications of the software that must meet both business and user requirements. It can be functional or nonfunctional [15] [16]:
    - Functional: which describes how the software must be functioning and the features needed to achieve the goal of the software and satisfy the user's needs.
    - Nonfunctional: which describes the quality attributes of the system.

In this research, I collected the data from the user reviews from 2 different public app platforms (Google play and App Store). As I was not aware of the software features in detail and the development road map, I will be focusing on two types of requirements (user requirements and Non-functional requirements) as these are the common and relevant types of requirements one can find in app stores reviews [17] [18].

2.2.1 User requirements:

User requirement is one of the keys in human-centered design as it describes the basis for a good design and its evaluation [19]. User requirements specification is typically collected and documented during the validation process of the software [20]. However, it is crucial for app developers to align and validate the initial user requirements with the users' feedback they get from the actual system users. Collecting users' feedback and transferring them to requirements helps to improve product development, marketing, and operation [21].

2.2.2 Nonfunctional requirements:

Nonfunctional requirements (NFRs) define system attributes such as reliability, security, performance, scalability, maintainability, and usability. They serve as restrictions or constraints on the design of the system. NFRs are just as critical as functional requirements as they ensure a good user experience and ease of use operating the software. It can also help improve users' trust when they feel safe and secure while using the software [22].

NFRs covers different aspects of the software:

- Operational aspects: such as Security, availability, integrity, accessibility, usability reliability, safety, efficiency, and reliability.
- Revisional aspects: such as maintainability, flexibility, scalability, and modifiability.
- Transitional aspects: such as portability, reusability, and installability.

2.2.3 Software bugs:

"A software bug is a flaw, failure, error or fault in a computer software or system that causes it to return unexpected or incorrect results." [23] Software bugs are usually a result of human errors or mistakes during the development process, they should be identified and fixed during the testing phase of the SDLC, but sometimes they can go through the development process and appears to users after deployment. Software bugs can vary from a slight effect on the user to catastrophic effects in critical software.

Software bugs that pass through the testing phase are usually hard to detect after deployment from the app developers [23] as they can happen in particular and complicated scenarios or can only happen in specific circumstances like different environments. It is crucial to have a reporting mechanism in the software to allow the users to report any issues or bugs they face while using the application. It helps the developers identifying and fixing them according to their priorities and impacts.

2.3 Neural network:

Like humans, artificial intelligence (AI) systems were not born perfect; they need to learn and adapt by taking in information or data continuously, process it, and keep it for future use, All AI parts are inspired by the human mind, but Neural network is the clearest example for that; it was inspired by the billions of neural and trillions of synapses in the human brain as shown in Figure 2.1, Neural network is part of the deep learning science [24].

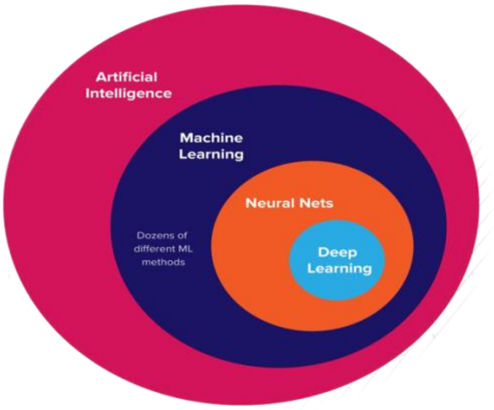

Figure 2. 1: Neural Network position among other sciences[4].

Neural networks are a collection of connected nodes or units called neurons where each node can send signals and information to the other node, the signal transmitted between the nodes the numbers and can move from node to another forward or backward. Neural network nodes and edges have weights usually used to adjust the learning process by having thresholds in neurons to identify the learning rate and improve it. Typically, neurons are packed into layers. The first

layer is called the input layer, which receives the inputs, the last layer in the neural network is called the output layer, which has the final prediction for the output, and in between, there are several hidden layers to do the computation.

How it works

To explain how NNs works in a simple example, let us assume I am building a neural network to differentiate between circle, triangle, and rectangle. The first step is to split the shape image into pexels and feed them to the input layer. Each node

[4] *Source: https://serokell.io/blog/ai-ml-dl-difference*

in the input layer is connected to one or more nodes from the second layer (one of the hidden layers) through channels where each channel is assigned to a numeric value called weight, the sum of each channel per node will be sent to the corresponding node, each node in the second layer has a value called bias. The summation result from this value with the summation from the channels per node identifies if a node should be activated or not. Only the active nodes transmit data to the subsequent layers in a process called Forward Propagation, which eventually will be sent to the output layer as probabilities.

The neural network also has the output fed to it, the final predictions in the output layer are compared with the actual value, and in case of any error, the information will be sent back to the hidden layer, and based on this information, the weight can be adjusted in what is called Backward Propagation, the same forward, and backward propagation, each forward and backward propagation called EPOC, and usually number of Epocs is one of the significant factors to improve the model quality.

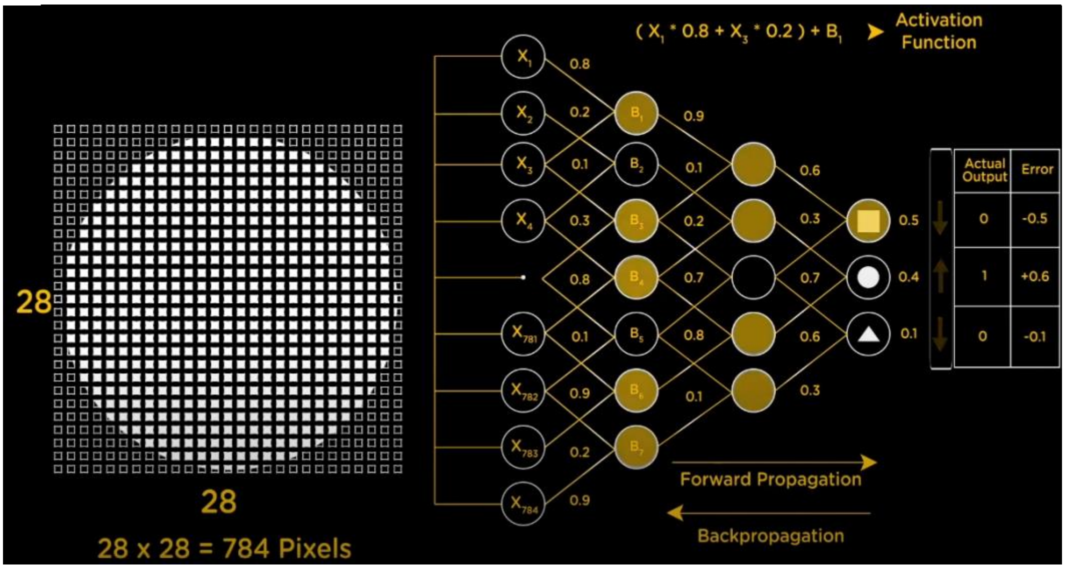

Figure 2. 2: NNs simple example.[5]

NNs are time and resources consuming; sometimes, it is hard to train a neural network model with low resources because an EPOC is too big to feed to the computer at once. Batches could solve this issue sometimes by splitting one Epoc into several smaller batches, where each batch holds a total number of examples in the train data set (called batch size).

Iteration represents the number of batched are required to complete on Epoc.

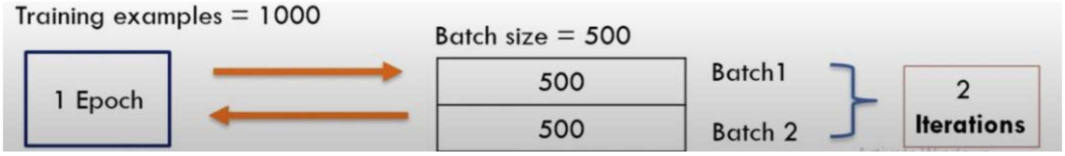

Figure 2. 3: Explain Epocs, batches, and iterations.[6]

[5] *Source: https://www.youtube.com/watch?v=bfmFfD2RIcg&t=257s*

[6] *Source: https://www.youtube.com/watch?v=K20lVDVjPn4&t=2s*

There are several applications on the neural network these days, such as:

- Facial recognition: smartphones these days have cameras and apps that can predict your age. This is an application of a neural network where it separates the face from the background and then correlating the lines and spots on the face to predict the age.
- Forecasting: neural network can be trained to understand the patterns and predict the possibility of a specific event (such as a rise in stock prices) with high accuracy.
- Music composition: Neural network can also learn patterns in music and train itself to compose a fresh tune.

2.3.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) was published recently in 2018 by researchers from Google AI [25]. It caused a revolution in machine learning science by presenting a state-of-the-art result in several NLP tasks, including Questing answering, natural language interface, and others [25].

The key innovation in BERT is applying Bidirectional training of transforms, unlike previous efforts which took the text sequence either from right to left, or from left to right, or combined right-to-left and left-to-right training, the models present a deeper meaning of the language context and flow compared to the previous Unidirectional models, BERT also introduces a novel technique called Masked Language Modeling "MLM" to allow bidirectional training models [12].

How BERT works?

As opposed to previous directional models, which reads the sentence from right to left or left to right, the transformers encoders in BERT reads the entire sequence of words at once (bidirectional), which allows the model to learn the context of a word based on what is surrounding the word in all directions.

BERT uses the transformers encoders to replace the words with mase and embedding them into vectors, which eventually will be processed in a neural network. The prediction goal could be a challenge for context learning in the directional model. To overcome this challenge, BERT used two training strategies:

- Masked LM (MLM): 15% of the words in each sentence are replaced with the token [mask] before feeding them into BERT; BERT tries to predict the original word depending on the other non-masked word in the sentence and using the classification layer on top of the encoder, BERT predicts and assign probabilities to the possible word for the masked token, where the validation loss on BERT takes only the masked words and ignore the other non-masked one, which also makes it slower than the other directional model.

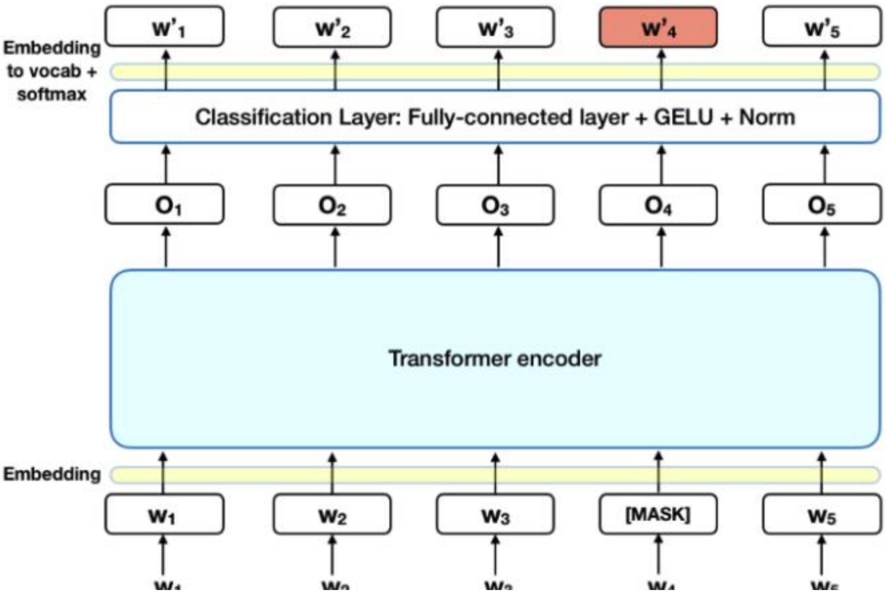
Embedding to vocab + softmax
W'1
W'2
W'3
W'4
W'5
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5

Figure 2. 4: BERT Masked LM (MLM).[7]

- Next sentence prediction (NSP): in the process of training, BERT receives pairs of sentences as input to learn and predict if the second sentence in the pair is the subsequent sentence in the original document. *50% of the pairs are in pairs* as they are in the original document, and the other 50 are paired randomly where the model should predict that they are not subsequent and separate them. To distinguish between sentences, the model processes the input before starting the training process by adding a [CLS] token at the beginning of the first sentence and a [SEP] token at the end of each sentence. To predict if the two sentences are connected, BERT transfers the CLS tokens' output into a 2*1 shaped vector using a classification layer and then calculates the probability of the IsNextSequence with SoftMax.

[7] *Source: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270*

Fine-tune BERT:

BERT can be used in several language tasks:

- Classification tasks: This can be done similarly to NSP by adding a classification layer on top of the transformers.
- Question Answering tasks (Q&A): This can be done by training BERT on two extra vectors that identify the beginning and end of the answer.
- Named Entity Recognition tasks (NER): This can be done by feeding the output vector of every token into a classification layer that predicts the entity label.

2.4 Logistics regression:

Linear classifier classifies the data based on a linear combination of input features by separating data using a line or plane. Linear classifiers can only be used when the data can be split linearly. Perceptron, logistics regressions, and SVM are the primary three algorithms in Linear Binary Classifiers.

While perceptions output only a Boolean result on where the input feature should be, logistics regressions pass the weighted linear combination of the input feature through a sigmoid function which returns a result between 0 and

1; this value indicates where the value should be on the plane. The probability of classification of points very close to the plane is close to 0.5 [26].

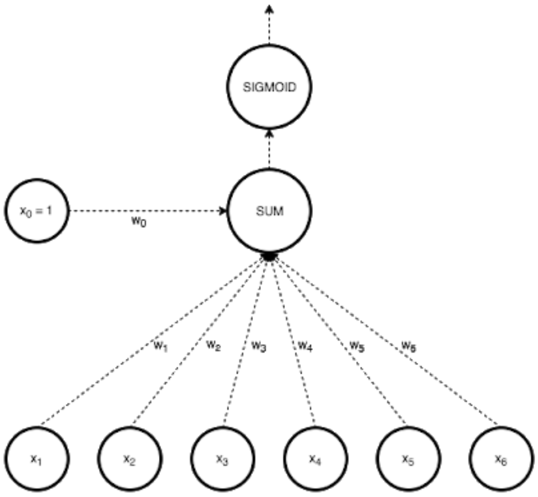

Figure 2. 5: Logistics regressions Linear classifier.[8]

[8] *Source: https://sites.google.com/site/machinelearningnotebook2/classification/binary-classification/linear-classifiers*

2.5 Warmup steps:

In the case of a highly differentiated dataset or unbalanced classes in the dataset, the model can suffer from early Overfitting. Warmup steps are a way to reduce the primacy effect of the early training examples by focusing on the learning rate during these steps and modify it per iteration. Let us say the target learning rate is p and the warmup period is n, then the first batch iteration uses 1p/n as its learning rate, and the second batch iteration uses 2p/n and so on till the model hits its nominal rate at iteration n. This means that the first iteration gets only 1/n of the primacy effect. This does a reasonable job of balancing that influence.

2.6 Learning rate:

Learning rate is a hyperparameter that controls how much the model should be changed according to the error after each iteration (when the weights are changed). It is usually a tiny positive number between 0.0 and 0.1. Choosing the most suitable learning rate is a challenging and experimental configuration as a low learning rate can cause a very long training process, and a high rate can cause speedy and unreliable results.

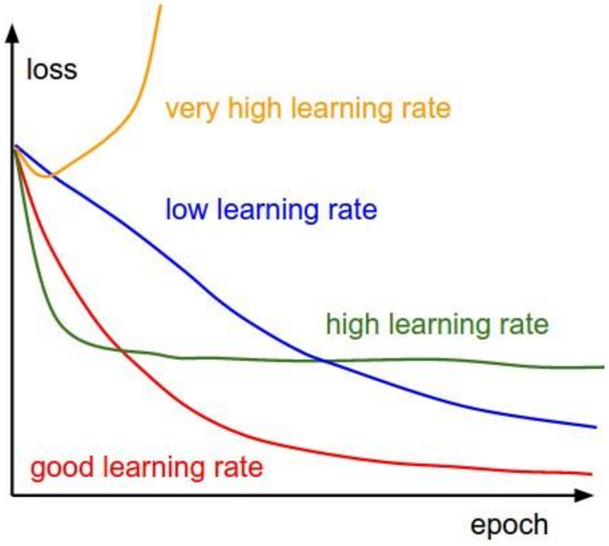

Figure 2. 6: Explain the effect of high/low learning rate.[9]

2.7 Early stopping:

Early stopping is a concept in machine learning that refers to stop training the model before it starts suffering from the over-fitting; I can use early stopping by splitting the dataset into a train, test, and validation dataset and keep an eye on both test and validation loss after each epoch where the model should be stopped if the accuracy of validation in decreasing the training accuracy is increasing (the model is overfitting). Overfitting means the model is working

[9] *Source:* https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

perfectly on your training set but not on other datasets. The model is closer to the perfect point when the training loss is close to the validation loss.

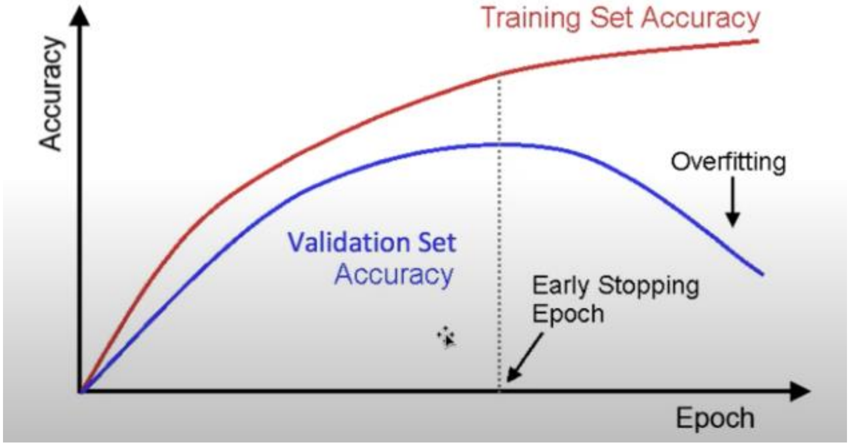

Figure 2. 7: Explain the concept of Early stopping.[10]

2.8 AdamW:

Adam optimizer first introduced in 2014 with a simple and intuitive idea: why should we use the same learning rate for every parameter while some parameter needs to move faster and further than others, some studies after Adam released shown 200% speed in training, but later in 2017 Ilya Loshchilov and Frank Hutter [27] pointed that the way weight decay is implemented in Adam is wrong, and proposed a simple solution called AdamW, and provided some charts to show the improvements in AdamW above Adam along several Epocs.

[10] *Source:* https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html

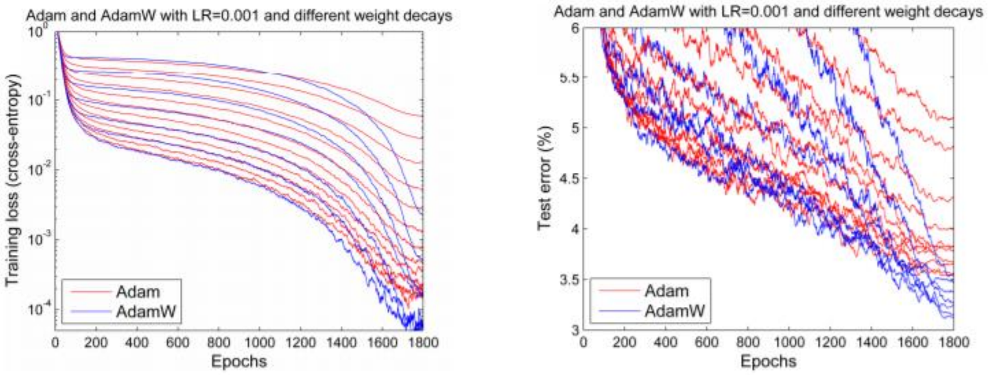

Figure 2. 8: The difference between Adam and AdamW.[11]

2.9 Sigmoid function:

Sigmoid is a mathematical function with the characteristics of an S-shaped or sigmoid curve, and it is a bounded and differentiable function that takes input values and transforms them into a number between 0 and 1 or -1 and 1 (called probability). Logistics Regression is one of the most common cases where the sigmoid function is used where it takes input values and transform them into probabilities between 0 and 1 according to the formula 2.1.

$$\begin{aligned} S(x) &= \frac{1}{1+e^{-x}} \\ &= \frac{e^x}{e^x+1} \end{aligned}$$

Equation 2. 1: Sigmoid Formula.

Where x represents the input value.

[11] *Source: https://www.fast.ai/2018/07/02/adam-weight-decay/*

2.10 Colab:

Collaboratory, or "Colab" for short [12], is a product from Google that allows anybody to write and execute Python code through the client's browser. Colab is very suitable for machine learning, data analysis, research, and education. Colab presents its copy of the jupyter notebook service, which requires no installation or setup to use. Colab also provides free access to computation resources such as CPUs, GPUs, TPUs. Although it provides free access to resources, it still has limitations, where the resources in Colab are shared between the users with some limitations per user. Colab also provides a premium edition for users who are interested in more reliable and better resources.

[12] *https://colab.research.google.com/notebooks/intro.ipynb*

Chapter 3

Related Work

In this chapter, a detailed review of the previous related work is presented. In order to make this section more organized and readable, and since I could not find any previous work for analyzing Arabic reviews from App platforms using BERT architecture, this chapter will be divided into three sections to cover the related work from 3 sides: The first section will present the work related to analyzing reviews from Google Play and App Store regardless the language of the reviews or the techniques used to analyze these reviews, the second section will cover the previous work related to analyzing Arabic regardless the techniques or the origin of the text being analyzed. The last section of this chapter will cover the previous work that is related to using the BERT architecture with more focus on the work that overlaps with the other two sections (the Arabic language and the app platforms reviews).

3.1 App platform reviews analysis

Several machine learning and natural language processing techniques were used to classify a text into different classes in literature. For example, Al Kilani, Tailakh, and Hanani in [5] used four different machine learning algorithms to classify App stores review into five categories: Bugs, Usability, new features, performance, and security. They retrieved around 90000 English reviews on ten different applications from google play. They employed ten software experts to do the manual classification, and they were able to conduct their experiment on 7500 reviews. For each review, the software experts were asked to assign a class among the mention five categories and to support their decision with a confidence level (high, medium, low) in order to train the classifiers on the confidence levels and compare the performance of the systems trained using classes with high, medium, and low confidence. Naive Bayes Multinomial was used as their baseline technique which shown the best performance among other techniques (Naive Bayes, Random Forest, and Support Vector Machine). They were able to improve the results concerning accuracy metrics by employing some other features like sentiment analysis. Some NLP techniques (such as n-grams modeling, TF-IDF features, and noise removal) were also used in the pre-processing data stages to improve the accuracy.

Similarly, Maalej and Nabil in [28] used several probabilistic techniques to classify app reviews into four types: bug reports, feature requests, user experiences, and ratings. They also used some metadata (such as star rating, tense, and text length) to improve the accuracy of their results. They used String Matching and Document classification (Bag of words) as the primary classifier and two NLP techniques: Stop-words removing and lemmatization. Their model achieved 97% accuracy (comparing with the manual classification) after employing the metadata in their model.

Moreover, Otoom, Sara, and Maen in [29] achieved an average accuracy of 93.1% using the Support Vector Machine classification algorithm to classify the newly reported bugs into two classes: corrective and perfective bug reports. They collected the bug reports data set from three open-source projects and fed these bugs into their automated classifier after they classified them manually. They also used some NLP techniques such as Tokenization and stemming steps to improve the accuracy of their automated tool. Among the three machine learning techniques they used (Naïve Bayes: average accuracy of 92.9%, Random Trees: average accuracy of 89.6%, SVM: average accuracy of 93.1%), SVM was the most accurate technique in their case.

Pagano and Maalej conducted a similar empirical study in 2013. They analyzed over one million reviews from Apple AppStore to study how and when the users add feedback, inspect the feedback content, and study its impact on the user's community. They found that feedback is added chiefly after the new releases, and they usually contain user experience, bug reports, and feature requests [30]. Another study was conducted in 2013 to extract new changes in requirements from users' comments in third-party applications [31].

In 2014, Guzman and Maalej conducted a study on 7 Apps from Apple AppStore and Google Play to identify the fine-grained app features by extracting the user's sentiment about the identified features and give them a general score, their automated approach has a precision of 0.59 and recall of 0.51 (compared with manual analysis), and the extracted features in their approach were coherent and relevant to requirements evolution tasks [32].

Yang and Liang in [33] presented an approach to classify user reviews from a popular app iBook in the Apple App Store into functional and non-functional requirements. They used several retrievals and NLP techniques such as TF-IDF and regular expressions to build their classifier. They also investigated the cost and returned for the proposed approach, and the results show a relatively stable recall, precision, and F-measure when selecting an appropriate size of sampled reviews.

Iacob and Harrison in [34] designed a prototype named MARA (Mobile App Review Analyzer) to support the process of analyzing a large number of reviews to extract the functional requests on a mobile application. Using their model, they were able to show that 23.3% of the reviews submitted on the App stores are functional feature requests. They analyzed 136,998 reviews using Linguistic rules and several NLP techniques to extract the feature requests.

They used LDA (Latent Dirichlet Allocation) model to assign topics from their corpus to each request.

Chaochang Chiu in [35] investigated the analysis of the Chinese review in some android game apps. He extracted 207,048 reviews of 4,268 games and analyzed them, including different factors like the game type and attributes. He found a high dependency between the type of the game and the gender of the user; males and females have differing opinions on game attributes.

3.2 Arabic language classification and analysis

Despite the difficulties in analyzing the Arabic language [36] [37], there are several kinds of research where Arabic language texts were analyzed and classified. For example, researchers in [38] build an opinion mining model that accepts an Arabic social media comment as input and identifies whether that comment is subjective or objective, positive or negative, and strong or weak. They built several algorithms to achieve their desired results, such as the Subjectively algorithm, Strength/Intensity Algorithm, and Polarity Algorithm. They used 66% of their dataset as training data and 34% as testing data. Finally, there were able to achieve the following accuracy using different algorithm:

- Subjectively analysis (Naïve Bayes – the most effective technique): 93%.
- Polarity Evaluation (K-NN Classifier – the most effective technique): 90%.
- Intensity Evaluation (Naïve Bayes – the most effective technique): 96.9%.

Different dialects are one of the most common difficulties in analyzing and classifying Arabic text [39]. However, several studies have been published to remove this difficulty by proposing automated tools and models that use some NLP techniques such as Lemmatization. For example, Jarrar and a group of research in [40] [39] presented Curras, an annotated corpus consisting of 55960 tokens for the Arabic Palestinian dialect and rich morphological and semantic information.

Ashraf, Yasmin, and Anas in [41] introduced a new dataset, and they named it HARD (Hotels Arabic Reviews Dataset), a large dataset with more than 370K Arabic reviews from the booking site. In order to examine their dataset, they have applied six machine learning classifiers to test the polarity and the rating of the reviews. The SVM and logistics regressions classifiers produced the best results, ranging from 94 to 97% for polarity and 72 to 75% for rating. They have also used the constructed lexicon on their dataset and achieved 89% accuracy.

A group of researchers in [42] presented an experiment to categorize Arabic text automatically. They have selected five main categories from popular Arabic datasets and generated three different versions of the dataset: stemmed, root-based stemmed, and light10. They applied four machine learning classifiers of the three datasets, SVM, NB, Decision Tree, and KNN. They got the best result on the light10 dataset using the SVM classifier, which achieved 98.4% accuracy using the RapidMiner tool, and 98% using the Weka tool.

Asma, Leila, and Abdesselam in [7] collected a dataset of 50 reviews from the Google Play Store. They have collected reviews only in French, Arabic, and Algerian dialects and analyzed these reviews to train a model that can predict sentiment analysis using two approaches, machine learning-based and Lexicon-based based. They achieved 80% using the Lexicon-based approach and 72%

using the machine learning approach, where the SVM classifier achieved the best results.

Sufyan, Omar, Bilal, and Khaled in [43] proposed an Arabic Aspect-Based Sentiment Analysis (ABSA) that combines both lexicon with rule-based models to classify the Arabic reviews from the governmental mobile apps after classifying them manually and five a sentiment score for each review. Their experiment approved that applying rules setting showed some improvements in both accuracy and f1 score, where the accuracy increased by 6% and the f1 score showed 17% improvement.

A recent study by Ahlam and Maha in [44] showed that many Arabic Apps in Google play are miscategorized. They refer to this for two reasons: the misunderstanding of Google play categorization schema or the lack of categories available for Arabic Apps. After collecting 13279 Arabic apps in Google play from different domains like Education, Family, Books, Lifestyle and classifying those apps using the LDA algorithm, they came up with this statement and predicted each app's category.

3.3 BERT and AraBERT

BERT [12] is a paper published in 2018 by researchers at Google AI, and it caused a storm in the machine learning community; it presented some extortionary results in several NLP tasks like Question Answering, Entity Recognition, and Natural Language Inference and others.

The key innovation in BERT is applying bidirectional training of transforms, unlike previous efforts which took the text sequence either from right to left, or from left to right, or combined right-to-left and left-to-right training, the models present a deeper meaning of the language context and flow compared to the previous single directional models, BERT also introduces a novel technique called Masked LM "MLM" to allow bidirectional training models [12]. AraBERT appeared for the first time in 2020; it is a pre-trained language model based on Google’s BERT architecture and uses the same BERT-Base config.

Till the moment of writing this research, the last AraBERT version is 2. AraBERT version 2 was trained on a large dataset (77 GB) with around 8.6 billion words and 200 million sentences. AraBERT v2 comes with two different pre-segmented models: AraBERTv0.2-base and AraBERTv0.2-large [45]. AraBERT will be used in this research, and previous work on AraBERT will be focused on in this section.

Recently, a group of researchers in [46] proposed a model to detect Hate Speech and offensive language. They have used different versions of the AraBERT model and fine-tune it on a total of 10K tweets. They labeled the data with HS and NOT_HS (HS stands for Hate Speech) and OFF and NOT_OFF (OFF stands for offensive language), the divided the dataset into training (70%), testing (30%) and validation (10%), and fine-tuned the model in two approaches: multi-labels and multi-tasks, they have found that the results of the multi-task AraBERT model outperformed the multi-label mode where the multi-task model showed 90.15% Macro-F1 for the offensive language and 83.41% for the Hate speech. Meanwhile, the best version of the multi-label model showed 89.55% Macro-F1 for the offensive language and 80.81% for the Hate speech. One of the main challenges they have faced is the imbalanced classes in the training data where they have 6489 tweets labeled as NOT HS and only 350 labeled as HS, they tried to solve this imbalance in different methods like using the weight loss function or by resampling the data, but none of these methods showed any improvement in their experiment.

Dalya and Malak in [47] used AraBERT to build that detects sarcasm tweets, and they named it sarcasmDet. They have used the prelabeled Shared Task on Sarcasm and Sentiment Detection in Arabic from WANLP 2021 [48] as their dataset and used different versions of the AraBERT model with one single Boolean class identifies in the tweet is sarcastic or not, and they have achieved 0.5989 F1-score for the large version of AraBERT and 0.3993 for the base version.

An interesting paper was published recently by Ahmad, Nada, and Ammar in [49], where they propose an effective to fight the COVID-19 Infodemic tweets. They used the NLP4IF 2021 [50] dataset and classified its 2556 tweets manually into seven labels indicates if the tweet has false information, verifiable claims, harmful, needs verification, harmful to society, requires governmental attention, or has some interesting information for the general public, their model achieved 67.7 accuracy using the base version of AraBERT.

Anshul in [51] proposed an approach to identify Arabic dialects in Nuanced Arabic Tweets Using Farasa Segmentation and AraBERT, he used a prelabeled training dataset that contains a total of 21000 tweets, validation, and test dataset contains 5000 tweets, he has applied both large and base AraBERT and achieved 0.433 accuracy as the best result.

Abdullah, Eric, and Abdulrahman in [52] used a prelabeled dataset called QurSim to binary classify pairs of verses provided by the dataset to check if the pairs are semantically related or not. They have used different versions of AraBERT, where the best result they achieved from AraBERTv0.2 with 92% accuracy.

Chapter 4

Data collection and analysis

This chapter describes how I got the reviews from the app platforms and how they are classified manually into three different. It also describes the corpus built and how the classes are categorized into users' requirements, bugs, and non-functional requirements.

4.1 Data collection

Up to my knowledge, there are no available annotated/classified datasets for the Arabic reviews of applications from google play and the Apple App Store. I also could not use any available APIs to fetch the reviews from Google Play or Apple App Store. However, I was able to modify and use some of the open-source crawling tools to collect a dataset from both Google Play and the App Store. This process took much time between modifying the tools to satisfy the needs and doing some workaround to prevent browser crashing or hanging through the crawling process due to the massive number of reviews in some applications. However, I was able to modify this tool with techniques to fetch a large number of reviews, including: (AppId, review author, review date, score rating, review text ...etc.).

In this research, I focused only on the Arabic reviews, so I picked five applications from the most common application in the Arabic region from various categories. I collected over 90000 reviews, but I reduced that number to 10000 reviews (the most recent reviews from each app) to do the manual classification. I noticed that the number of reviews on Google play is much larger than the one in Apple App Store.

App name	Domain	Reviews collected	
		App Store reviews	Google play reviews
Altibbi	Health	132	4000
Careem	Car booking	800	1000
Shahed	Video streaming	621	1000
Waze	Navigation and live traffic	447	1000
Yamsafer	Hotels and flight booking	0	1000
Totals:		2000	8000
		10000	

Table 4. 1: The list of Arabic applications used to create the dataset.

4.2 Data manual classification and categorization

I[13] started the manual classification process by reading each review and assign it to one of the classes if its suites any of the classes in the corpus; if not, I extend the corpus with a new class and assign the review to it[14]. The criteria I used to classify the reviews and build the corpus are depending on the repeatability of issues and requirements in app's reviews where I added a new class for an issue or requirement only if it's repeated by 10 or more users in different reviews:

- Repeated meaningful issue: this means that 10 or more people reported it as a potential meaningful issue or bug to a software tester.

[13] A master student in software engineering with over 6 years of experience as a software engineer.

[14] 2% of the corpus was discussed in a face-to-face with an external expert. The aim of this meeting was not to measure agreement or disagreement on the classification but to have an external opinion about understanding reviews which I took into account while classifying the rest of the corpus.

- Repeated meaningful requirements: it’s a user need that frequently repeated (in 10 or more reviews) and identified meaningfully useful to a software tester and/or requirement engineer, by the expert annotator.

Although these criteria were practical in classifying the corpus, but there might be some reviews that are vague. By the end of this process, I was able to classify around 10K reviews and build the corpus with 33 classes that were found arbitrarily as they are patterns in the reviews. Table 4.2 shows the classes, description, example, and the count for each class in the dataset.

Class	**Description**	**Example**	**Count**
Not relevant	• Describes all the comments that did not have any informative information for the develops. • It’s not a repeated meaningful issue or requirement that need to be resolved.	الله يعطيكم العافية خدمة والله مميزة ومشكور للكبتن احمد الصبحي الله يعطي العافية ويحفظة	7138
Issues appeared after updates	• Describe all the bugs specifically after updates, and also some feature removed with the new updates. • Repeated meaningful potential issues that needs to be resolved by app testers.	xr التحديث الأخير لا يعمل على اجهزة ايفون	120
Missing business features (user suggestions)	• Describes all the domain feature that are request by users. • Repeated meaningful requirement that needs to be added to the app according to user’s needs.	البرنامج ممتاز ولكن لو يضيفون خاصية عداد القسمه طوال الرحلة يكون ممتاز	116
Annoying ads	• Describes the user requests to remove or reduce the number of ads on the app. • Repeated meaningful potential issues that needs to be resolved by app testers.	حرفيا بدون مبالغه اذا حاب تتابع حلقه مدتها ساعه ٤٥ دقيقه بالقليل اعلانات	143

UI/UX issues/suggestions	• Describes all the issues with the app user interface or user experience, or some user suggestions. • Repeated meaningful potential issues/requirements related to UI/UX that need to be resolved/Added.	تطبيق بواجهة ومميزات فقيرة ، صعوبة الوصول الى الرحلات السابقة	17
Notifications	• Describes all the issues related to app notification or some feature requested by the user related to notifications. • Repeated meaningful potential issues that need to be resolved by app testers.	دائما يظهر انه يوجد اشعارات ولكن عند الدخول للتطبيق لايوجد اشعارات	13
Availability	• Describes all the issues related to app availability. • Repeated meaningful potential issues regarding the app availability that needs to be resolved.	السلام عليكم مشرفو التطبيق لدي مشكلة هو انني لا استطيع ان اسأل لانني لست من دول الخليج انا من شمال افريقيا والتطبيق يلزمه رقم مبايل من بلدان المحددة في التطبيق اريد حل لهذه المشكلة والسلام عليكم	195
Performance	• Describes all the issues related to app performance. • Repeated meaningful potential issues regarding the app performance that needs to be resolved.	برنامج جميل لكنه بطيئ بعض الشيئ	137
Work offline	• Describes all the reviews, suggestions, or complaints about app functionality in offline mode. • Repeated meaningful requirement that needs to be added to the app according to user's needs.	ناقصه تنزيل الخرائط بدون نت	37
Bugs related to the app features	• Describes all issues user faced in domain features. • Repeated meaningful potential issues regarding the app functional features that needs to be resolved.	سيء اطلب فنادق في مكه يطلعلي فنادق القاهره 7 كيلومتر من مكه وجع اعتمر بالقاهره ؟ وش ذا	180

Compatibility	• Describes all the bugs and user requests for the app compatibility with other third parties. • Repeated meaningful potential issues/requirements related to app compatibility that need to be resolved/Added.	اتمنى الدعم على Apple TV	113
Issues related to the app languages	• Describes all the reviews where users complaining about issues in languages, or request supporting new language. • Repeated meaningful potential issues regarding the app supported languages that needs to be resolved.	يحتاج الى تحسين كبير للواجهة العربية لانها تحتوي على كلمات انجليزية كثيرة وانعكاس للواجهة الانجليزية ويحتاج الى توضيح الازرار اكثر خصوصا عند الغاء الطلب	55
Political and racism reviews	• Describes all reviews related to political and racism. • Repeated meaningful comments from users about Political and racism issues.	تطبيق عنصري يتماشى مع سياسات الاحتلال الصهيوني..	147
Suddenly crash	• Describes all reviews mentioning the App stopped or crashed. • Repeated meaningful potential issues regarding the app crashing issue that needs to be resolved.	ممكن تلقولي حل كل ما ادخل تطبيق شاهد يقولي تم توقيف تطبيق شاهد لييييش ايش المشكلة اطلب منكم تحلون هذا المشكلة بلييز بلييز بليييز بلييز	125
Privacy	• Describes all reviews related to user's privacy. • Repeated meaningful potential issues regarding the app privacy that needs to be resolved.	التطبيق يقوم بنسخ ما اكتبه جرب انك تنسخ كلام وتضع لصق التطبيق سينسخ النص الذي نسخته انا. تطبيق سيء ينتهك الخصوصية	16
Usability	• Describes all reviews related to app ease of use. • Repeated meaningful potential issues/requirements regarding the app	التطبيق صعععععععععب وأحسه حوسة مرة غير مرتب الحين طالبة طلب بس ضيعت وينه ماعرفت شلون ألقاه	46

	Usability that needs to be resolved/added.		
Lack of instructions and guidance	• Describes all reviews related to lack of information or guidance. • Repeated meaningful potential issues/requirements regarding the app instructions that needs to be resolved/added.	حملت التطبيق لكن كيف الاشتراك والبدء بسؤال الطبيب وهل هو مجان وماهو الكوبون اللي يريد مني ادخاله	171
Internet connection and coverage issues	• Describes all reviews related app connection to internet while the phone is connected. • Repeated meaningful potential issues regarding the app connectivity that needs to be resolved.	الان البرنامج لا يعمل (بقول ليك غير متصل بالانترنت بالر غم من توفر الشبكة بالجهاز	69
Business limitations	• Describes all reviews related to app feature limitations. • Repeated meaningful requirements regarding the app limitations that needs to be added.	ياريت زيادة عدد الحروف في كتابة الشگوي .. غير 👌 گده تطبيق ممتاز	10
Human and technical support issues	• Describes all reviews related to human's agent and technical support. • Repeated meaningful potential issues regarding the app support issues that needs to be resolved.	مش برتاح مع كريم اسلوب غير محترم من الشركة و موظفين خدمة العملاء و شكله ان العملاء كمان هنقول مفيش كريم افضل	273
High cost	• Describes all reviews related to user complaining about app cost. • Repeated meaningful potential issues/requirements regarding the app cost that needs to be resolved/added.	الاسعار تكلفته غالية ولا تقولي شركات اتصال ودولي اصلا كثير من اتصالات النت مجانا بس الاطباء الي يبيعوا معلومات سوو خير وانفعوا العالم مع الاسف انا كنت استعمل البرنامج وبعد ميشتغل مثل قبل ولا يجاوب الا بفلوس ومسحته	193
Subscription	• Describes all reviews related to subscription issues.	اول شي ارسل لي انو اشتراكي خلص وهو لسا ما خلص وسويت اشتراك ثاني من اسبوع وما تفعل	208

	• Repeated meaningful potential issues/requirements regarding the app subscription that needs to be resolved/added.	وليا اسبوع اتواصل على الواتس والايميل لكن يقولون انه مفعل وهوا ما يشتغل اسوا خدمة عملاء !!!!	
Payment issues & voucher	• Describes all reviews related to Payment issues. • Repeated meaningful potential issues/requirements regarding the app payment and voucher issues that needs to be resolved/added.	اذا لغيت المشوار ليش تنزل لي 7 ريال في المشوار الثاني	267
App content issues	• Describes all reviews where users are complaining about app content. • Repeated meaningful potential issues regarding the app content issues that needs to be resolved.	جيد ولكن عرض المسلسلات او البرامج الوثائقية يتم عرضها بعد عرضها ع التلفزيون .. نحن من اشتركنا بشاهد نريد عروض حصريه قبل العرض ع التلفاز	105
App size	• Describes all reviews related to app size. • Repeated meaningful potential issues regarding the app size issues that needs to be resolved.	التطبيق صار ثقيل ولايفتح ويعلق	11
Login, registration, password reset, and activation issues	• Describes all reviews related to users Authentication. • Repeated meaningful potential issues regarding the app authentication panels issues that needs to be resolved.	فبه مشكلة عدم امكانية اعادة الرقم السري ان اخطأت في الايميل	138
Download & installation issues	• Describes all the issues users are facing in downloading and installing the app. • Repeated meaningful potential issues regarding the app download and installation issues that needs to be resolved.	مش رضي يتحمل يوصل 90 ويوقف	10

No Free edition	• Describes all the reviews where users are complaining about no fermium version of the app. • Repeated meaningful potential issues/requirements regarding the app freemium edition that needs to be resolved/added.	كان الحلو فيه انه مجاناً وجوده عاليه بس الحين صار بافلوس يعني نظام نتفلكس	52
Communication issues	• Describes all the reviews where users are complaining communication issues. • Repeated meaningful potential issues regarding communication issues that needs to be resolved.	لا يوجد رقم أو طريقة تواصل بين الشركة ١- والزبائن. لا يوفر كريم رقم مجاني لحجز التكسي ٢-.	232
Supportability	• Describes all the reviews related to app supportability. • Repeated meaningful potential issues/requirements regarding the app supportability that needs to be resolved/added.	stc ليش في السعوديه ما يشغل مع مستخدمين شركه حاولت كثير السداد عبر رقمي من اس تي سي لكنه يرفض التعرف على الرقم	42
High internet consumption	• Describes all the reviews related to the app high internet consumption. • Repeated meaningful potential issues regarding the app internet consumption issues that needs to be resolved.	شفط ابو النت ٤٠ قيقا راحت في يومين	17
Security	• Describes all the reviews related to the app security. • Repeated meaningful potential issues/requirements regarding the app security that needs to be resolved/added.	يوجد ثغرات في أمان التطبيق حيث يمكن للكابتين التلاعب و إلغاء الرحلة دون معرفه الراكب و إرسال سياره اخرى دون إعلام الراكب بما حدث حيث يتم الاتصال به من الشخص المزيف المرسل من قبل كابتن كريم و بعد الركوب و اثناء السير نتعرض للتهديد و الابتزاز و تحصيل مبالغ ماليه رهيبة و سوف تدفع تحت التهديد و إنهاء الموقف و قد تكون الخساره اكثر من ذلك فقد تتعرض للخطف و الموت و تجاره الأعضاء	11

Not clear	• Describes all the reviews where the users were complaining about some issues, but it was not clearly mentioned • It's a repeated issue or requirement, however, what's need to be resolved is not clear.	دائما اخطاء في الطلب سيء جدا جدا جدا مع العلم انا عميل ذهبي لديهم	142

Table 4. 2: The original 33 classes with their description and example on each class.

After the manual classification for the 33 classes, I found that some of the classes do not have many reviews to support them during the training process, so I generated two different datasets from the original one with different labeling. The first one is by categorizing the 33 classes into five parent classes:

- Software bugs: 1370 reviews.
- Non-functional requirements: 1725 reviews.
- Users Requirement: 845 reviews.
- Not Relevant: 7138 reviews.
- Not clear: 142 reviews.

To address all the research objectives mentioned in section 1.2, I have created another third dataset to find how informative Google Play and App Store reviews can be. The third dataset had only three classes:

- Informative reviews: 2720 reviews.
- Uninformative reviews: 7138 reviews.
- Not clear: 142 reviews.

Figure 4.1 shows all the classes, and how I generated the second and third data set from the first dataset classes where the green nodes represent the classes of the third dataset, the blue ones represent the classes in the second dataset, and the gray nodes represent the original classes in the first dataset.

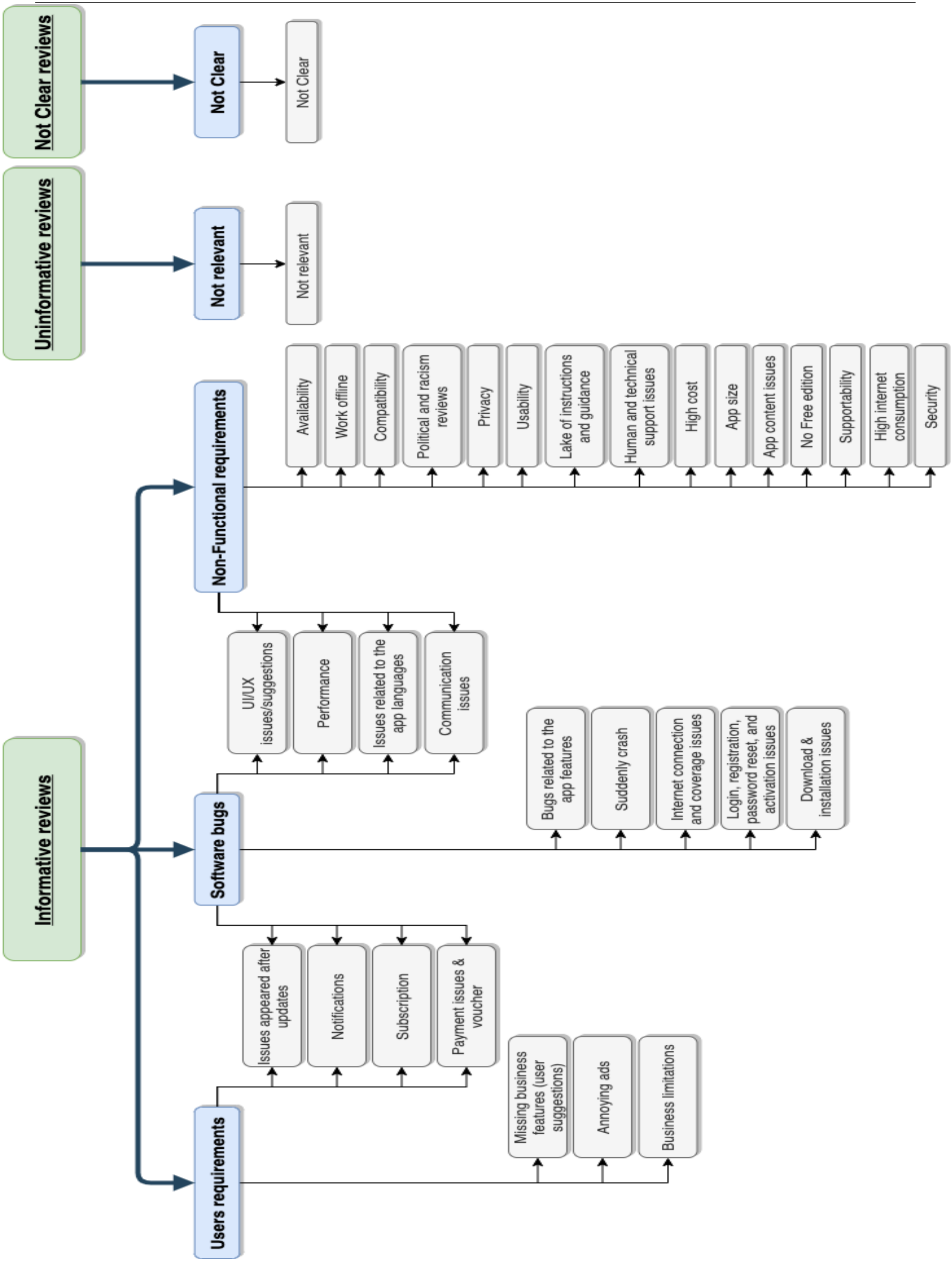
Informative reviews
Uninformative reviews
Not Clear reviews
Users requirements
Software bugs
Non-Functional requirements
Not relevant
Not Clear
Missing business features (user suggestions)
Annoying ads
Business limitations
Issues appeared after updates
Notifications
Subscription
Payment issues & voucher
Bugs related to the app features
Suddenly crash
Internet connection and coverage issues
Login, registration, password reset, and activation issues
Download & installation issues
UI/UX issues/suggestions
Performance
Issues related to the app languages
Communication issues
Availability
Work offline
Compatibility
Political and racism reviews
Privacy
Usability
Lake of instructions and guidance
Human and technical support issues
High cost
App size
App content issues
No Free edition
Supportability
High internet consumption
Security
Not relevant
Not Clear

Figure 4. 1: Categorizing the original classes into two other different versions of classification.

Chapter 5

Research Methodology And Experimental Setup

In order to fulfill the objectives of this research which were introduced in the first chapter, Reviews in Arabic language were collected from Google play and App store and annotated with the 33 classes mentioned earlier. In this chapter I will explain how the data were pre-processed and will show the tools used in the experiments.

5.1 Environment setup

Due to the massive number of recourses and computation power need to fine-tune a BERT model, Collaboratory was a good available solution to run the experiments due to its easy usage and high computation power. However, the Free version of Colab has many limitations (especially RAM) that prevented the model from completion due to memory crashes and lack of resources. The solution was the premium version of Colab with a more considerable amount of RAM and access to larger GPUs and TPUs.

Three Colab projects were prepared to run three experiments with the three datasets mentioned earlier, and I made sure that all the three experiments run on the same hardware resources and the same BERT configurations, which will be presented later in this chapter. I have also connected Colab projects to Google drive to save all the logs, results, and training checkpoints for later use to predict and evaluate the results after the training process.

In all the experiment, it was hard to get a chance to run the experiment on Tensor processing units (TPUs) even with the pro version to the limited number of TPUs and the high demand and usage on them, so I ran all the experiments with the exact Nvidia CUDA GPU specifications:

- GPU version: Tesla P100 PCI-E
- Memory: 16280 MiB
- Driver Version: 460.32.03
- CUDA Version: 11.2

5.2 AraBERT

Although Google introduced their multilanguage model with the BERT architecture and it supports the Arabic language. However, researchers found that the performance could improve with a pre-trained BERT model on specific language, which was the case in AraBERT where the authors pre-trained BERT on a large set of Arabic words and outperformed the original Multilanguage model after fine-tuning their model on several NLP tasks such as NER, question answering and sentiment analysis [53].

AraBERT appeared for the first time in 2020 [45]. It is a pre-trained language model based on Google's BERT architecture and uses the same BERT-Base config. Till the moment of writing this research, the last AraBERT version is 2. AraBERT version 2 was trained on a large dataset (77 GB) with around 8.6 billion words and 200 million sentences. AraBERT v2 comes with two different pre-segmented models: AraBERTv0.2-base and AraBERTv0.2-large [45]. The first model will be used in this research due to the enormous number of resources and time needed to fine-tune the second large model.

AraBERT Dataset combines several datasets such as the Arabic Wikipedia dump, the 1.5B words Arabic corpus, OSCAR, the OSIAN corpus, and the Assafir news articles. It consists of more than 8.6 billion words before using the Farasa segmentation [53]; Farasa is an Arabic segmenter. Segmentation involves breaking Arabic words into their constituent clitics [54].

AraBERT comes with a well-implemented preprocessor to process the Arabic text before starting the training to improve the model's final results. AraBERT preprocessor performs the following operations on the text [45]:

- Stripping the words from the Arabic Tashkeel.
- Stripping the words from the Arabic Tatweel.
- Replace URLs and emails with Arabic tokens.
- Remove any HTML markups.
- Remove non-digits repetition (removes any characters or special characters that are repeated more than two times).
- Remove emojis.
- Insert whitespace before and after all non-Arabic digits or English Digits and Alphabet and the two brackets.
- Replace slash with a dash.

After reviewing several BERT pre-trained models, AraBERT achieved the best performance among all. Thus, it will be used in this research along with its text preprocessor.

5.3 Used tools

In order to achieve the desired results for the experiments, many libraries were used to prepare and preprocess the data and to work along with the BERT to provide the best results:

- PyArabic[15]: is an open-source python library that provides the essential functions to manipulate Arabic text and letters, such as removing diacritics, detecting Arabic letters, and stripping Arabic words (removing Tashkeel). This library was required and used heavily by the AraBERT to perform the text preprocessing and helps in internal functionalities.
- Torch[16]: Torch is an open-source scientific computing framework with excellent support for AI and machine learning algorithms and models that requires a lot of computation power. It has many utilities and machine learning algorithms, and optimizers that utilize the computation power and make the process of running complex deep-learning projects easy and fast. Among all the utilities and algorithms in torch, the following were used in this research:
    - Linear: Is the classifier that will serve as a way to get the output of BERT model and convert them into the classes we want to predict [55].
    - Sigmoid: As mentioned earlier in the background section, I have used the sigmoid from the torch with the logistics

[15] https://pypi.org/project/PyArabic/

[16] https://pytorch.org/

    regression Linear classifier to get the probabilities per class [56].

    - BCE Loss function: It stands from Binary Cross-Entropy. It is used in this research to measure the error in each class by combining it with the probabilities from the sigmoid function.
    - Dataset: An abstract class in a torch that torch DataLoader heavily uses, it is used in this experiment with custom implementation for the length and the getItems functions to prepare the dataset and feed it to the DataLoader.
    - DataLoader: a torch class that uses the Dataset class and provides many functionalities on the data. In this research, it is used to fetch the data from the dataset in the form of batches and feed it to BERT. It is also used to identify the number of workers used per batch.
- Pytorch lightning[17] is an open-source python library developed by William Falcon. It is a library built on top of a torch and designed to make research projects more scalable and quicker to iterate. The following PyTorch lightening functionalities were used in this research:
    - Metrics: several metrics were used to evaluate the results, such as accuracy, F1, precision, recall, and AUROC. AUROC stands for Area Under the Receiver Operating Characteristic Curve. The curve is created by plotting the true positive rate (TPR) vs. the false positive rate (FPR),

[17] https://www.pytorchlightning.ai/

where the closest curve to one is the better results. In this research, I use it to measure the performance of each class.

  - Early Stopping: a function that takes the monitoring matrix and another numeric parameter to stop the training if the monitoring matrix did not improve after a number of Epocs, which is identified by the second numeric parameter (called: patience). In this research, I used the validation loss as the monitoring matrix and five as the patience.
  - Model Checkpoints: training the model is both resource and time-consuming process, and it is frustrating to restart this process in case of any interruption for the training process. Checkpoints can be used to save a checkpoint after training each Epoc to come back and resume the model training process from the last saved checkpoint.
  - Tensor Board Logger: This tool provides several measurements and visualization during the training process; it provides and visualizes some important matrices like accuracy. It is used in this research to provide some crucial graphs for each class.

- Transformers[18]: is an open-source library provided by hugging-face, which provides thousands of pre-trained models to perform tasks on texts such as classification. It is used in this research to help in fine-tuning the AraBERT model. The tools used in this research are:
  - AdamW: implements AdamW optimizer to adjust and optimize the learning rate during the training.

[18] https://huggingface.co/transformers/

  - Auto tokenizer: it is used to create a model that is an instance of the BERT Model from a specific pre-trained model. In our case, it was the AraBERT version 2 model.
  - Get linear schedule with warmup: a function that uses the AdamW optimizer, the number of the warmup steps, and the number of the training steps to prepare the schedular with a learning rate the decreases in early from the initial learning rate set in the optimizer to 0, after a warmup period during which it increases linearly from 0 to the initial learning rate set in the optimizer.
- Sklearn[19]: an open-source library that contains many efficient tools for statistical modeling and machine learning, including clustering, regression, and classification, the following tools from Sklearn were used in this research:
  - Train test split: a function used for splitting data arrays into two subsets, testing and training data. It was used twice in this research to split the datasets first to train dataset (0.7) and test dataset (0.3) and used a second time to split the test dataset and generate a validation dataset (0.1) and leave the other 0.2 for the test dataset.
  - Classification report: a function used to print a text report that shows the main classification metrics (precision, recall, f1-score, support, macro avg, micro avg, and weighted avg) for each class.

[19] *https://scikit-learn.org/stable/*

- Matplotlib [20] : an open-source python library that works like MATLAB, used in this research to visualize the results of the experiments, and provide some graphs.
- TQDM[21]: an open-source library, the name was derived from the Arabic word (تقدم), which mean progress in English, it is used in this research to visualize the progress of training per Epoc.

5.4 Code explanation

The experiment code is written on Colab, and the link can be found in the Appendix. The Colab project contains explanation and documentation for each step and is supported with examples and charts. However, in this section, I will explain the main steps in the code at an abstract level.

The first step after linking the Colab project to a google drive account and reserving the resources in Colab is importing the utilities needed for the experiment, which was explained in the previous section.

The second section in the code starts with loading the dataset from Google drive and split it into three datasets randomly, training dataset (70%), which will be used to train the model, test dataset (20%), which will be used to test and evaluate the model and validation dataset (10%) which will be used to fine-tune the hyperparameters of the model during training such as learning rate and optimizers.

The third section of the code starts with identifying the AraBERT model version used for the experiment and uses its text pre-processor to pre-process

[20] https://matplotlib.org/
[21] https://tqdm.github.io/

the reviews in the training, test, and validation datasets. It also pulls the classes from the dataset and finds the number of reviews for each class to detect any unbalanced classes in the training dataset and take samples of the unbalanced classes.

The first step in the fourth section of the code identifies the tokenizer used in the experiment from the pre-trained AraBERT model, followed by some examples of how the tokenization works, and explains the input ids and attention mask concepts. It also shows some examples of how each sentence is wrapped with the CLS and SEP tokens. The last step in this section iterates over the training dataset and finds the number of tokens per sentence and visualizes that in a graph. The maximum number of tokens per sentence in the training dataset was 370 tokens; this number will be fed to the model to limit the number of tokens and improve the model performance.

The sixth section of the code is all about preparing the datasets in a flexible way where they can be fed to the model in batches by using the torch lightning dataset and data loaders I explained in the previous section, many examples were provided on how the data can be fetched in batched and shows the input ids, attention masks and label for a sample review. This section also contains a step to download the AraBERT model and prepare it for the next section. It also has some steps to show the number of hidden layers and the shape of a sample model. The last step in this section is for identifying the number of Epocs that will be used, the batch size, and instantiating an instance of the data loader class.

The model section has the main class of the model, which is a lightening module with overriding the following hooks:

- Init: in this function, I assign the BERT model (the AraBERT instance from the previous section), the classifies (the logistics regression linear classifier will be used as discussed earlier), training and warmup steps (will be calculated later and passed to the model as parameters) and the criterion which will be the BCELoss function.

- Forward: is the core step of the model where it takes the input ids and attention mask for a review, pass them to the BERT model and apply the linear classification, sigmoid on the BERT results to get the probabilities for each class; it also uses the loss function to find the loss for each class.
- Training step: this is the training in the module where it takes a batch of reviews and passes the reviews one by one to the forward step; this step returns the outputs and the loss for the whole batch. It also logs the training loss per batch to keep an eye on it during the training.
- Validation step: this step is similar to the training step except that it takes batches from the validation dataset and calculates and logs the validation loss.
- Test step: the testing step is also the same as the training and validation step except that it takes its baches from the test dataset and calculates and logs the test loss.
- Training Epoch end: This hook gets fired at the end of each Epoc. It combines the prediction for each label from all batches into one array and calculates the AUROC for each label.
- Configure optimizers: this hook holds the optimizer (AdamW) and the warmup steps configurations to adjust the learning rate and improve the performance model.

The training section starts with clearing all previous logs and identifying the folders where the logs will be saved to link them to the tensor board. Checkpoints are also defined in this section to reduce the time and effort by allowing the model to resume training from the last saved Epoc if any interruption happened. Once the loggers and checkpoints are ready, I can start

instantiating a trainer instance from the lightning module explained in the previous section and start training the model by passing the model and data module to the fit function in the trainer. During the training process and after each Epoc, the Epoc number, number of steps, and the validation loss will be printed for the whole Epoc, and a new checkpoint will be saved. After each Epoc, the early stopping will check if the validation loss is improving from the last Epoc or not, it will continue the training if it is improving, or it will stop the training process if the loss did not improve for the past specific number of Epocs (the number is 5 in this experiment). The final step of this section is to find the training loss for the last saved Epoc (which has the best validation loss). The closes the test loss to the validation loss, the better the model.

The Prediction section loads the model from the last saved checkpoint (the Epoc with best validation loss) and provides two examples of providing any random review to the model to get the probabilities predictions for each class.

The last section of the code is all about evaluating the model's performance by using the reviews in the test dataset to get the model predictions for each review and compare with the actual values and calculate all the needed matrices. The evaluation results will be shown in detail in the next chapter.

Chapter 6

Experiments Results and Analysis

After preparing the datasets, the experiment environment, and code, this chapter will present the results of the three-experiment performed and discuss the results:

- Experiment I: Performed on the original dataset with 33 classes.
- Experiment II: Performed on a generated dataset of 5 classes: User's requirements, Non-functional requirements, Software bugs, not clear and not relevant.
- Experiment III: Performed on a generated dataset of 3 classes: Informative, no clear, and not relevant.

6.1 Hyperparameter values

Hyperparameters are a set of parameters or configurations that controls the learning process. After reviewing the theory behind these parameters and some similar experiments, I found that some of these parameters are experimental parameter where the best value of the parameter can be found by running the experiments with different values and monitor the results, an example for that is the learning rate where I ran the experiment with five different values of learning rate (the most used values in the other previous research) to find the best result in this experiment.

Max token count is an essential parameter in BERT where it identifies the amount of memory needed to be reserved to handle one review. If this value is not identified, the BERT will take the default value of 512 tokens and truncate all the tokens above 512. Due to the memory limitation in this experiment, I found that most of the review in the dataset have less than 200 tokens, only a few of them have above 200, and the maximum token length was 370 token which was used in the experiment as shown in Figure 6.1 where the horizontal axes represent the number of tokens while the vertical axes represent the number of reviews,

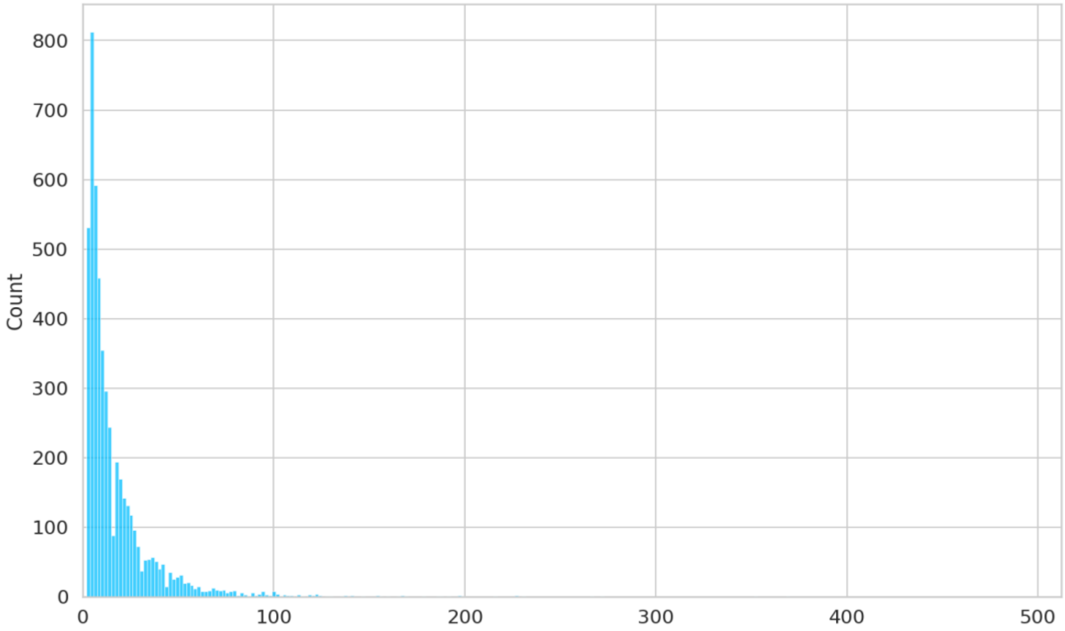

Figure 6. 1: The number of tokens vs review count.

The number of Epocs is a crucial factor in BERT, where a small amount of Epocs can produce a lousy accuracy and learning, and many Epocs could produce an overfitting model. In this research, I used Early stopping to stop the training process before the model begins to over-fit. After few experiments, I noticed that the model usually keeps improving till 5 to 15 Epocs, and then it begins overfitting. 15 Epocs was the number of Epocs in this research, but only 9 Epocs reached before the model stopped.

The batch size value is highly recommended to be a power of 2; this is related to the alignment between the virtual and the physical processors where the physical process is usually a power of 2 and using a different number for virtual processors can lead to bad performance. In this research, and due to the limitation in processing and RAM resources, the value of 8 was selected to be the batch size.

The learning rate might be the most crucial hyperparameter in the model; the linear with the warmup schedular was the most recommended schedular for BERT models where the models learning rate starts improving during the specified warmup steps until it reaches the specified initial learning, then the learning rate starts going down from the initial rate till it reaches zero during the training steps. The total number of steps per Epoc is the number of the reviews in the dataset, while the number of the total steps in the model is the number of steps per Epoc multiplied by the number of Epoc specified for the model. After reviewing the theory and the previous work, 3 and 5 was the number of steps that were usually used as the warmup steps and achieved the best results, and both numbers were tested in this experiment along with different learning rates (1e-5, 2e-5, 3e-5) and five warmup steps with 2e-5 achieved the best results. Figure 6.2 shows an example of the linear learning rate with warmup steps behaves with a hundred total training steps and two warmup steps:

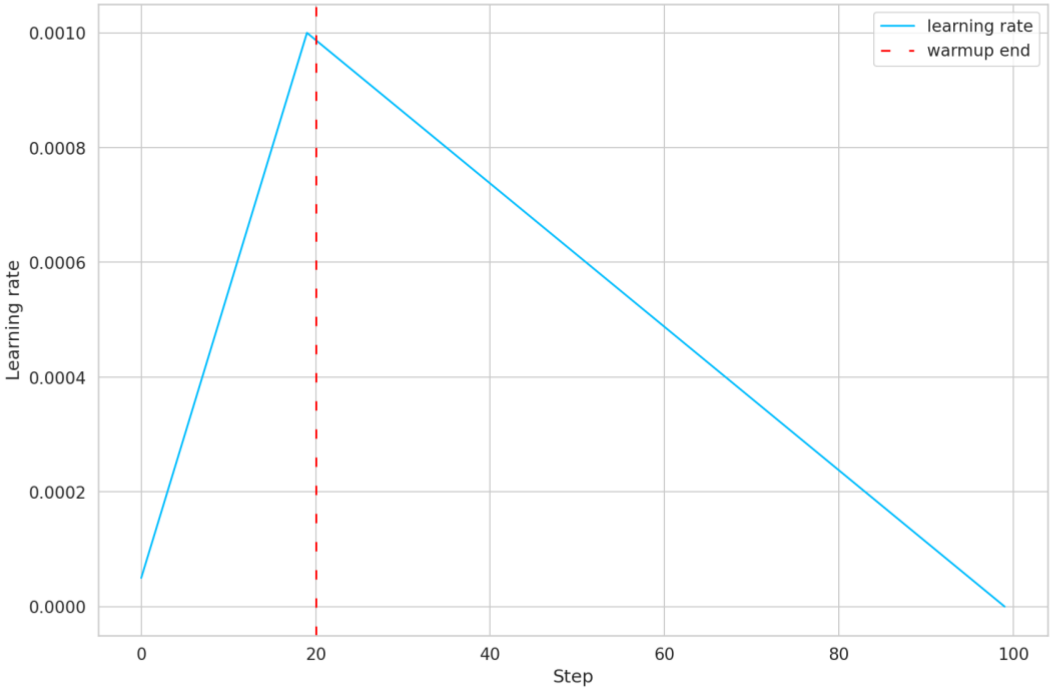

Figure 6. 2: Warmup and learning rate graph.

Early stopping is used in this experiment to prevent the model from overfitting, stopping the training process when the model is not improving for a specific number of Epocs. After reviewing the previous similar experiment and perform out an experiment with different values, I found that usually after a specific number of Epocs, the validation loss stops improving, and making sure that the validation loss will not get improved after, 5 EPOCs were used to monitor if there is any improvement can be found on the validation loss. The best Epoc is saved in this experiment as a checkpoint where it can be loaded and used later as the best result the model can produce. Table 6.1 shows a summary of the hyperparameters used in this research for all experiments.

Hyperparameter	value
Max token count	370
Epocs	50
Batch size	8
AdamW learning rate	2e-5
Total training steps	30800
Warmup steps	6160
Early stopping monitor	Validation loss
Early stopping patience	5
Checkpoint save the top value	1

Table 6. 1: The hyperparameters values used in all experiments.

6.2 Datasets and text preprocessing and Tokenization

6.2.1 Splitting dataset and solve the unbalanced class issue

In order to get the hyperparameters works to optimize the model, a validation dataset was added along with the train and test datasets with a portion of 10% of the original dataset. The total number of reviews in the original dataset was 10000 reviews, and table 6.2 shows the number of reviews used for train, test and validation datasets.

dataset	train	test	validate
count	7000	2000	1000

Table 6. 2: Train, test and validate datasets.

After analyzing the classes in the training dataset, one unbalanced class was found in all three experiments, which is the not relevant class. The three figures below (6.3, 6.4, 6.5) show the number of reviews per class in each experiment.

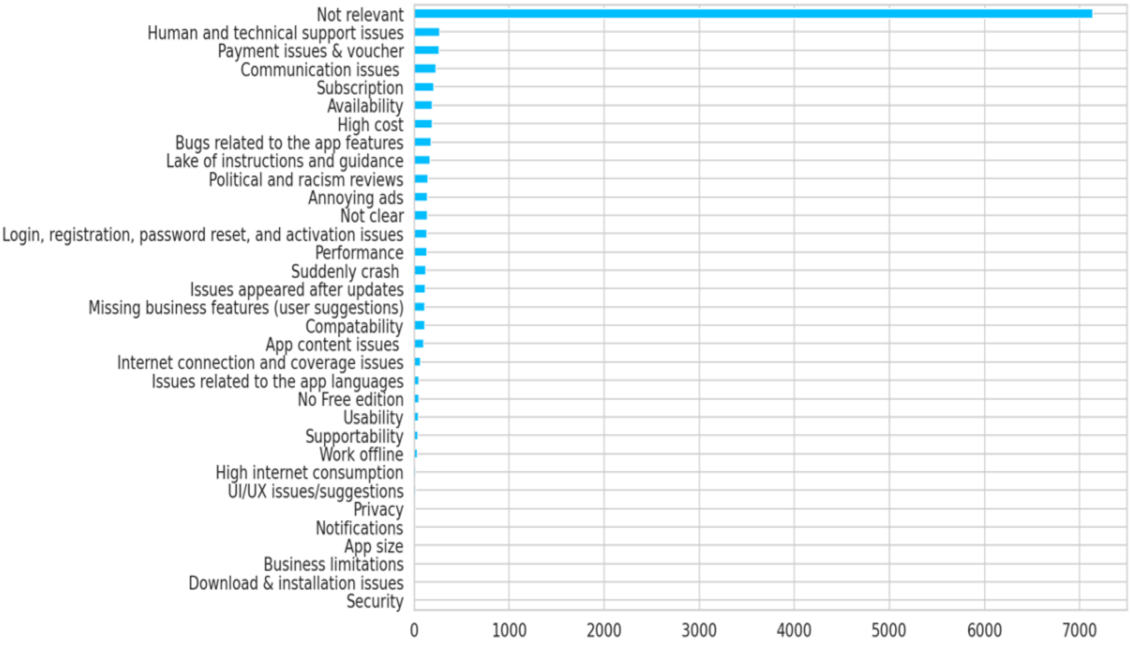

Figure 6. 3: The reviews distribution in the train dataset of Exp. I.

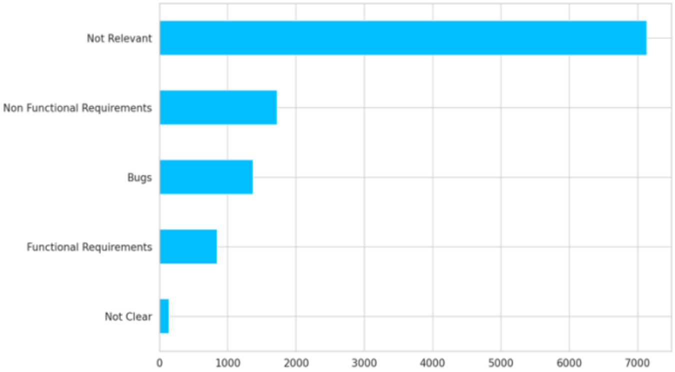

Figure 6. 4: The reviews distribution in the train dataset of Exp. II.

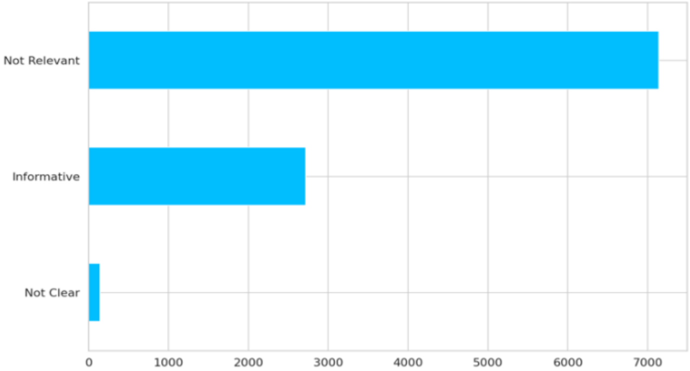

Figure 6. 5: The reviews distribution in the train dataset of Exp. III.

Imbalanced classes can cause severe issues in the model performance where the model can simply guess and always classify any review to the unbalanced class and get high accuracy [57]. For example, if 95% of the reviews belong to the not relevant class, then the model can predict any review to be not relevant and get an accuracy of 95%. There are many techniques to solve unbalanced class issues, such as changing the performance metric, changing the algorithm, and different resampling techniques [57].

- Changing the performance metric: Accuracy could not be the best metric to use when there is an unbalanced class in the dataset as it could be misleading. Different other evaluation matrices were used in this research and will be presented later in the following sections of this chapter:
    - Precision: precision measures the model exactness where it represents the number of true positives divided by all positive predictions. Lower precision means a higher number of false positives.
    - Recall: recall measures the model completeness where it represents the number of true positives divided by all positive values in the test dataset. Low recall means a higher number of false negatives.
    - F1 score: the weighted average of precision and recall.

  Changing the algorithm: Changing the classifying algorithm could be a solution to resolve the imbalanced classes issue, but not in this research where the Linear classifier gave the best results.
- Resampling techniques: two types of resampling can be used to balance the number of reviews assigned to each class.

- Oversampling: This can be done by replicating the minority classes as much as needed to reach a close ratio of data points for each class.
- Under-sampling: This can be done by removing some of the reviews assigned to the majority classes as much as needed to reach a close ratio of data points for each class. The under-sampling technique was used in this research on the training dataset by considering only 3000 reviews of the not relevant class and remove the others.

Table 6.3 shows the number of reviews per dataset in Experiment I after resampling the not-relevant class. The same technique was used in all the experiments.

dataset	**train**	**test**	**validate**
count	5026	2000	1000

Table 6. 3: Train, test and validation datasets after resampling.

6.2.2 Text preprocessing and tokenization

Data Preprocessing is an essential step as it improves the data quality and improves the extraction of meaningful text from the data. In this research, I used the pre-processor introduced by AraBERT (explained in detail in the previous chapter). It was highly recommended to use the processor provided by the model itself as the model was trained on a large amount of text preprocessed by the same pre-processor.

BERT was trained using the word piece tokenization, which means that one word can be broken into more than one sub-word. A whole sentence in BERT is represented by one vector before being fed to the classifier where the first token in the vector represents the whole vector; to achieve this, a [CLS] token will be added as the first token for each review. [SEP] token will also be added at the end of each review to inform the model about the sentence end. Meanwhile, a [PAD] token is added to fill the rest of the vector if the number of tokens in a vector is less than the specified max token size. [UNK] is another token in BERT, representing the tokens that are new to the model (the model was not trained on this token and did not recognize it). UNK stands for the unknown token, while this problem is known as the out of vocabulary problem (OOV).

During the training process of the BERT model, each token will be assigned to a unique id, so when the model is being used and fine-tuned as a pre-trained model, all the vectors need to be generated as a vector of ids where each it represents its corresponding token. An attention mask is a mask that will be used if the token length is smaller than the specified max token length. It is the mask that is typically used for attention when the dataset has varying length sentences.

Table 6.4 shows an example of a sentence with a specified max token length of 12 where the first row shows the original sentence, the second row shows how the AraBERT pre-processor stripped the word from Taskeel, Tatweel and removing the duplicate characters and emojis. The third line shows the tokens that AraBERT generated from the sentence, while the last two lines represent the generated token ids and attention mask.

original	ارجـــو إصْلَاح التَّطْبِيق هههههههههه .... 😀
Pre-processed	ارجو إصلاح التطبيق هه . .
Tokenized - vector shape: 1 * 12	['[CLS]', 'ارجو', 'إصلاح', 'التطبيق', 'ه', '##ه', '.', '.', '[SEP]', '[PAD]', '[PAD]', '[PAD]']
Input ids - vector shape: 1 * 12	[2, 43104, 6714, 11818, 138, 195, 20, 20, 3, 0, 0, 0]
Attention mask – vector shape: 1 * 12	[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]

Table 6. 4: Example of AraBERT pre-processor and tokenizer.

6.3 Model training:

Figure 6.6 is an example from the first experiment that shows the executed Epocs against the total number of steps where the horizontal axes represent the number of steps while the vertical axes represent the number of Epocs. It is also important to mention that this graph is a visualization of only the Epocs

considered in the training process, and it does not show the Epocs that were ignored by the Early stopping.

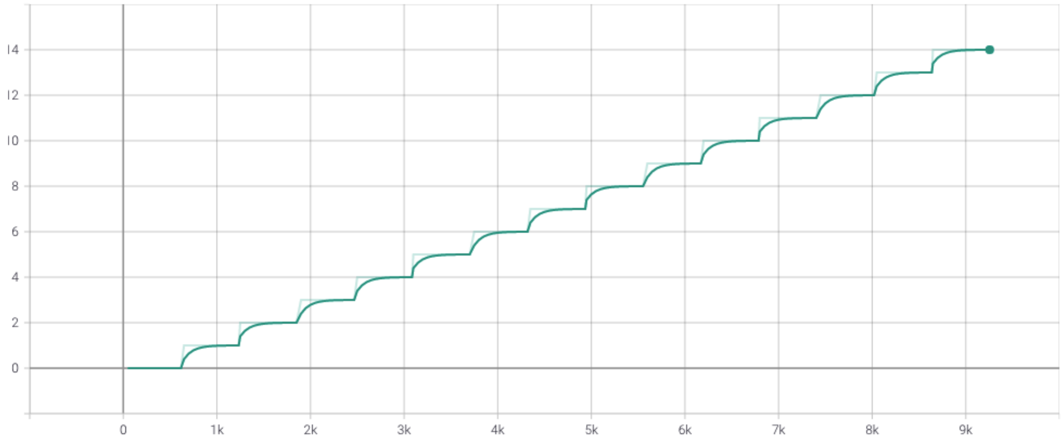

Figure 6. 6: The number of Epoc against the number of steps.

As the model started training, logs were configured to print the validation loss after each Epoc to monitor the validation loss and the early stopping during the training process. Table 6.5 summarizes the Epocs executed and the validation loss after each Epoc for the three conducted experiments. It is noticeable that all experiment started with a relatively high validation loss Epoc and started improving after each Epoc. 14 Epocs were Executed for the first experiment while the validation loss kept improving till the ninth Epoc, the next 5 Epocs were executed, but none of them provided a better value of validation loss. Hence, the model stopped learning and considered the ninth Epoc is the end of the model training process. 10 Epocs were executed for both the second and the third experiments, and they both got the best result and stopped training at the fifth Epoc.

EPOC #	Global steps	Validation loss		
		Experiment I	Experiment II	Experiment III

0	616	0.24660	0.30627	0.27402
1	1233	0.12045	0.21443	0.20466
2	1850	0.08107	0.20298	0.14673
3	2467	0.06350	0.19601	0.13667
4	3084	0.05338	0.14396	Not the best
5	3701	0.04579	0.13933	0.11721
6	4318	0.03581	Not the best	Not the best
7	4935	0.03261	Not the best	Not the best
8	5552	0.02999	Not the best	Not the best
9	6169	0.02869	Not the best	Not the best
10	6786	Not the best	Not the best	Not the best
11	7403	Not the best	-	-
12	8020	Not the best	-	-
13	8537	Not the best	-	-
14	9254	Not the best	-	-

Table 6. 5: Validation loss per Epoc for the three experiments.

After training the model, the test loss can be used compared to the validation loss to evaluate the model. Table 6.6 shows the final test and validation loss for each experiment.

Experiment	**Experiment I**	**Experiment II**	**Experiment III**
Test loss	0.03244	0.13505	0.11721
Val loss	0.02869	0.13933	0.11210

Table 6. 6: Test and validation loss for the three experiments.

6.4 Experiment I result

6.4.1 AUROC per class

Aria Under the Receiver Operating Characteristic (AUROC) is one of the most important matrices to measure the model performance, especially in multi-class cases, it is also known as the AUC-ROC curve, where Aria Under Curve (AUC) represents the measure of separability or degree, and the AUC represents the probability. AUROC tells how the model can distinguish between multiclass. The higher AUROC, the better the model is in distinguishing between the classes. AUROC is plotted with the True positive rate (TPR) on the vertical axes against the false positive rate (FPR) on the horizontal axes. The AUROC value is between 1 and 0, where 1 represents the perfect model [58].

Figure 6.7 shows an example of a perfect model where the model can entirely distinguish the two classes without overlapping. Meanwhile, the two classes in the second example (figure 6.8) show that the two classes overlap by 30%, which means that the model has a chance of 70% to distinguish between the two classes, which means that the AUROC result for each class is 0.7.

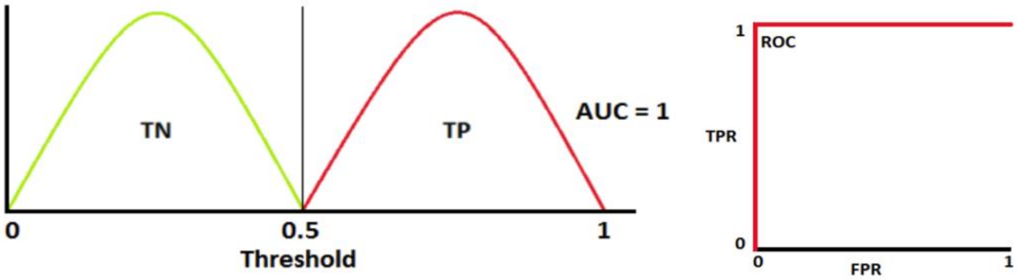

Figure 6. 7: An example of a perfect AUROC curve.

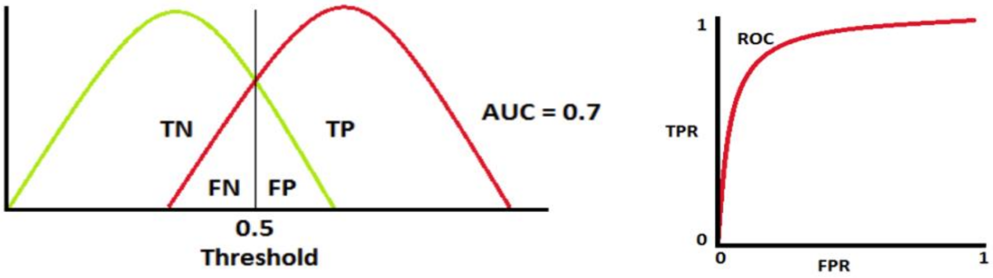

Figure 6. 8: An example of a 0.7 AUROC curve.

Table 6.7 shows a summary of the AUROC values achieved in the first experiment for each class. Most of the classes have good AUROC values above 0.9, except two classes got less than 0.9. These classes were focused on by re-visiting the reviews that represent these classes in the dataset. It was found that most of the reviews assigned to these two classes are written and described with general words in Arabic that could also be used in other classes. Furthermore, most of the reviews assigned to these two classes are also assigned to other classes (have multi-labels) because it was unclear whether these reviews belong to a specific class during the manual classification process.

Class	**AUROC**
Not relevant	0.9838
Issues appeared after updates	0.9453
Missing business features (user suggestions)	0.8831
annoying ads	0.9880
UI/UX issues/suggestions	0.9898
Notifications	0.9969
Availability	0.9871
Performance	0.9573
Work offline	0.9996
Bugs related to the app features	0.9588

Compatability	0.9836
Issues related to the app languages	0.9998
Political and racism reviews	0.9993
Suddenly crash	0.9912
Privacy	0.9944
Usability	0.9532
Lack of instructions and guidance	0.9507
Internet connection and coverage issues	0.9998
Business limitations	0.9751
Human and technical support issues	0.9758
High cost	0.9686
Subscription	0.9838
Payment issues & voucher	0.9867
App content issues	0.9560
App size	0.8673
Login, registration, password reset, and activation issues	0.9958
Download & installation issues	0.9300
No Free edition	0.9044
Communication issues	0.9753
Supportability	0.9963
High internet consumption	0.9326
Security	0.9149
Not clear	0.9129

Table 6. 7: AUROC values per class in Exp. I.

Figure 6.9 shows the AUROC curve for only one class which is the not relevant class from the first experiment. The rest of the graphs for all the other classes can be found in the Colab project in Appendix A.

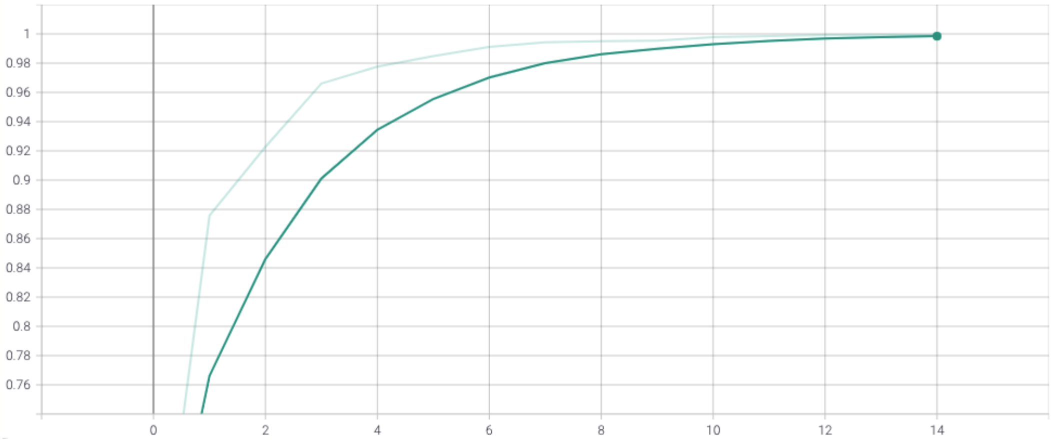

Figure 6. 9: Exp. I, the not relevant class AUROC curve.

6.4.2 Train vs validation loss

The validation and training loss comparison is an essential step since it is one of the primary keys to identify if your model is Overfitting or Underfitting, where the higher train loss indicates a possibility of underfitting. In contrast, a higher validation loss indicated the possibility of an overfitting model. Figure 6.10, 6.11 and Table 6.8 shows a comparison between the validation and training loss in this experiment at the best Epoc of the model, the difference between the two values is less than 0.6% for the training loss (underfitting) which indicates a good model comparing with the other similar experiments.

Matrix	**At the best Epoc #9**
Validation loss	0.02869
Training loss	0.0351
difference	**0.00641**

Table 6. 8: Train vs Validation loss in Exp I.

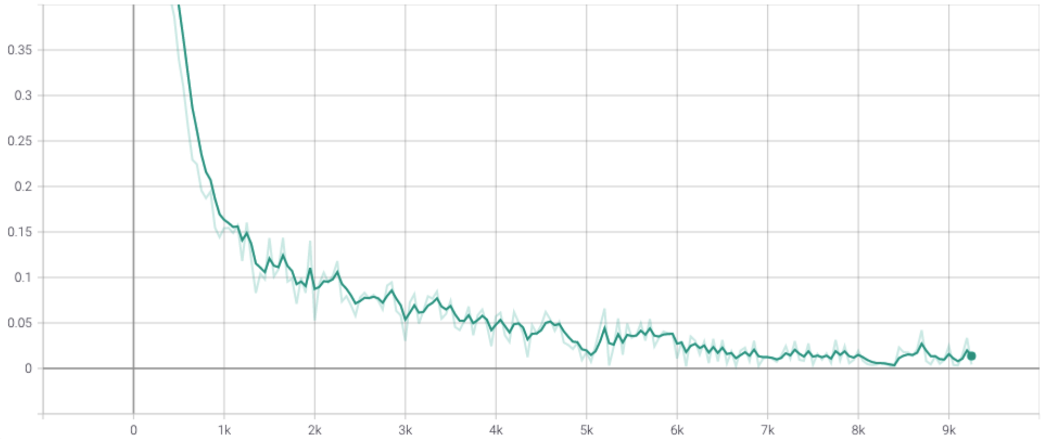

Figure 6. 10: Exp. I, training loss curve.

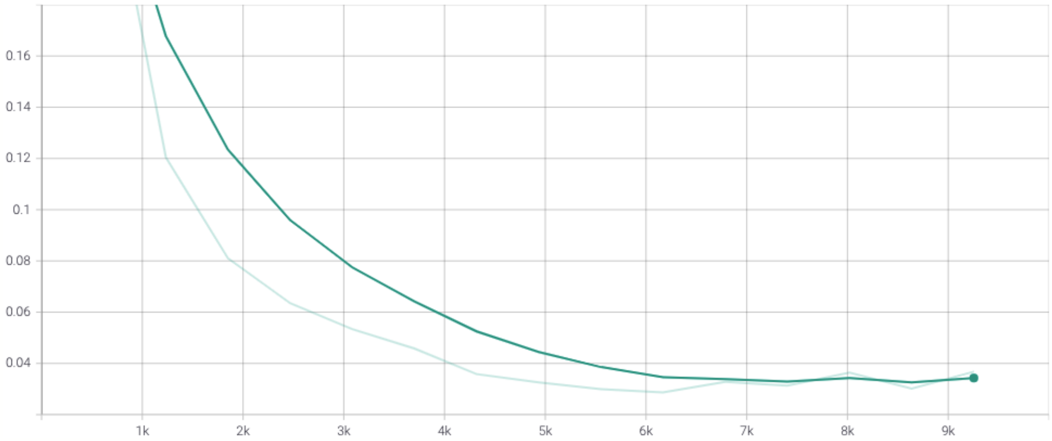

Figure 6. 11: Exp. I, validation loss curve.

6.4.3 Prediction

After training the model, I have used the saved checkpoint of the last saved Epoc (best result) to feed the model with random Arabic sentences and let the model predicts the classes for them. Table 6.9 shows the example and the top three classes predicted by the model, along with their probabilities.

Text	**Prediction**	
	Class	**Probability**
تطبيق جيد جدا بارك الله فيكم وشكرا	Not relevant	0.9906
	Issues appeared after updates	0.0009
	Missing business features (user suggestions)	0.0007
التطبيق سيىء مبارح نزلتو ومش راضي يشتغل معي	Not clear	0.8425
	Not relevant	0.0730
	Compatability	0.0444
التطبيق فيه مشاكل بطيء وبضلو يعلق وفيه كثير مشاكل ما بدعم لغة عربية وكمان تطبيق يحارب المحتوى الفلسطيني	Issues related to the app languages	00.6598
	Performance	0.0705
	Political and racism reviews	0.0561

Table 6. 9: Model predications in Exp I.

6.4.4 Experiment result

The final results of the model per class are shown in table 6.10. The last column shows the number of reviews from the dataset used to support its corresponding class and produce the results. I noticed that some of the classes have a precision, accuracy, and F1 of zero; most of these classes have less than six reviews to support them. Usually, the small number of reviews is not enough to train the model for a specific class and produce results, so these classes with less than six reviews are not reliable and cannot be counted on. The solution for such cases is to remove these classes from the classes set or enrich the dataset with more reviews to increase the number of reviews labeled with these classes.

The final result for the whole model is presented in table 6.11.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Not relevant	0.93	0.98	0.95	1455
Issues appeared after updates	0.25	0.07	0.11	30
Missing business features (user suggestions)	1.00	0.08	0.15	24
annoying ads	0.92	0.96	0.94	23
UI/UX issues/suggestions	0.00	0.00	0.00	3
Notifications	1.00	0.25	0.40	4
Availability	0.82	0.90	0.86	40
Performance	0.58	0.72	0.64	25
Work offline	0.50	1.00	0.67	5
Bugs related to the app features	0.29	0.72	0.41	36
Compatability	0.47	0.70	0.56	23
Issues related to the app languages	0.79	1.00	0.88	11
Political and racism reviews	0.72	1.00	0.84	21
Suddenly crash	0.22	0.92	0.36	12
Privacy	0.00	0.00	0.00	1
Usability	0.50	0.10	0.17	10
Lack of instructions and guidance	0.33	0.47	0.39	34
Internet connection and coverage issues	0.75	1.00	0.86	6
Business limitations	0.00	0.00	0.00	3
Human and technical support issues	0.39	0.89	0.54	56

High cost	0.60	0.69	0.64	35
Subscription	0.61	0.69	0.65	45
Payment issues & voucher	0.50	0.90	0.64	61
App content issues	0.34	0.55	0.42	20
App size	0.00	0.00	0.00	2
Login, registration, password reset, and activation issues	0.61	0.77	0.68	26
Download & installation issues	0.00	0.00	0.00	3
No Free edition	0.24	0.57	0.33	7
Communication issues	0.68	0.78	0.72	54
Supportability	0.50	0.71	0.59	7
High internet consumption	0.00	0.00	0.00	6
Security	0.00	0.00	0.00	1
Not clear	0.27	0.52	0.36	27
micro avg	0.76	0.88	0.82	2116
macro avg	0.45	0.54	0.45	2116
weighted avg	0.80	0.88	0.83	2116
samples avg	0.85	0.90	0.86	2116

Table 6. 10: Results per class in Exp I.

Matrix	**Accuracy**	**Precision**	**Recall**	**F1**
Value	0.99	0.92	0.80	0.86

Table 6. 11: Final results in Exp I.

6.5 Experiment II result

For the second experiment, the same model, hyperparameters, and optimization techniques were used; the only difference between the two experiments is the number of classes.

6.5.1 AUROC per class

Table 6.12 show the AUROC results for each class in the second experiment, and figures 6.13, 6.14, 6.15, 6.16, 6.17 shows the AUROC curve for each class where the horizontal axes represent the number of Epocs while the vertical axes represent the AUROC value, note that the values in table 6.12 represent the value on the curve at the best Epoc which was the fifth Epoc in this experiment.

Class	**AUROC**
Functional Requirements	0.9795
Non Functional Requirements	0.9607
Bugs	0.9687
Not Relevant	0.9839
Not Clear	0.8291

Table 6. 12: AUROC values per class in Exp. II.

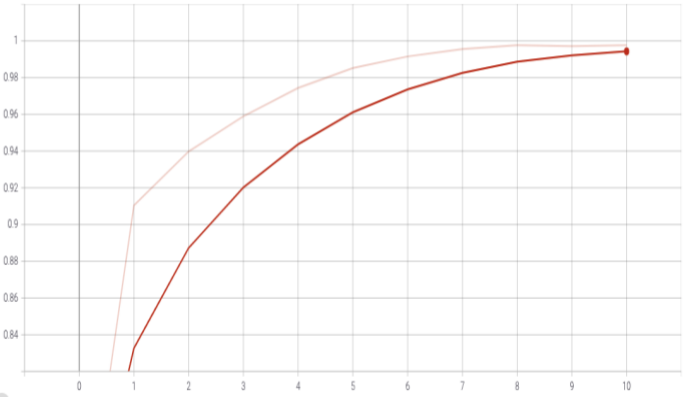

Figure 6. 13: Exp. II, AUROC curve for the Bugs class.

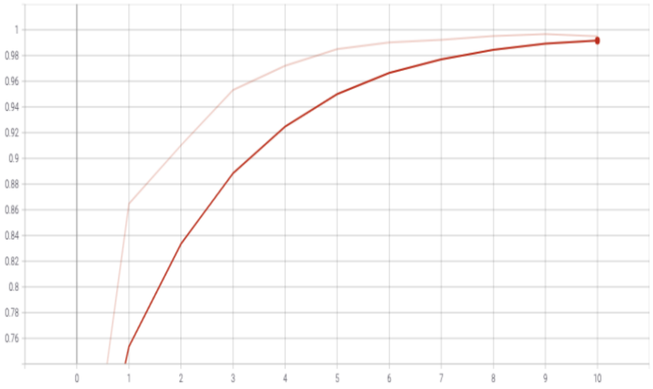

Figure 6. 12: Exp. II, AUROC curve for the Users requirements class.

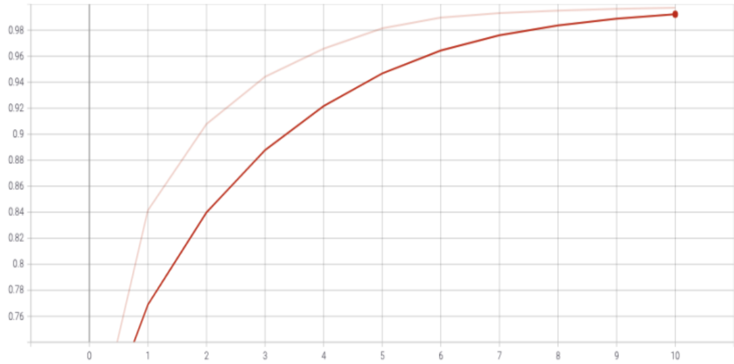

Figure 6. 14: Exp. II, AUROC curve for the non-functional requirements class.

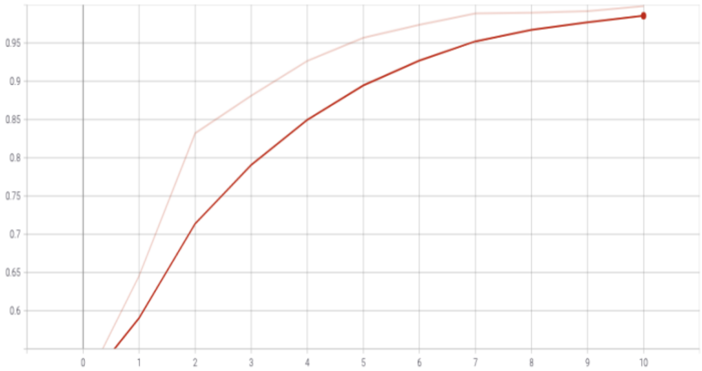

Figure 6. 15: Exp. II, AUROC curve for the not clear class.

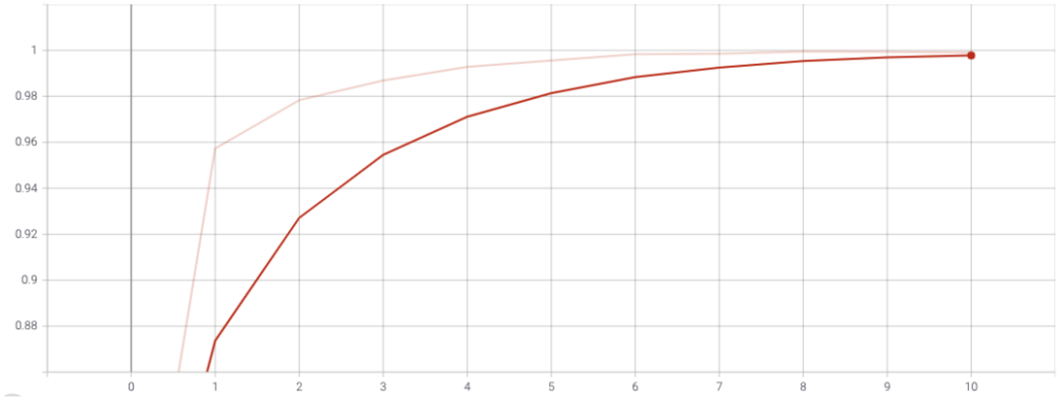

Figure 6. 16: Exp. II, AUROC curve for the not relevant class.

6.5.2 Train vs validation loss

Table 6.13 shows the validation and the training loss in the second experiment, while figures 6.18, 6.19 show the validation and train loss graphs.

The validation loss in this experiment is higher than the training loss, which means the model is overfitting by 9.6%.

Matrix	**At the best Epoc #5**
Validation loss	0.13933
Training loss	0.0432
difference	**0.09613**

Table 6. 13: Train vs Validation loss in Exp II.

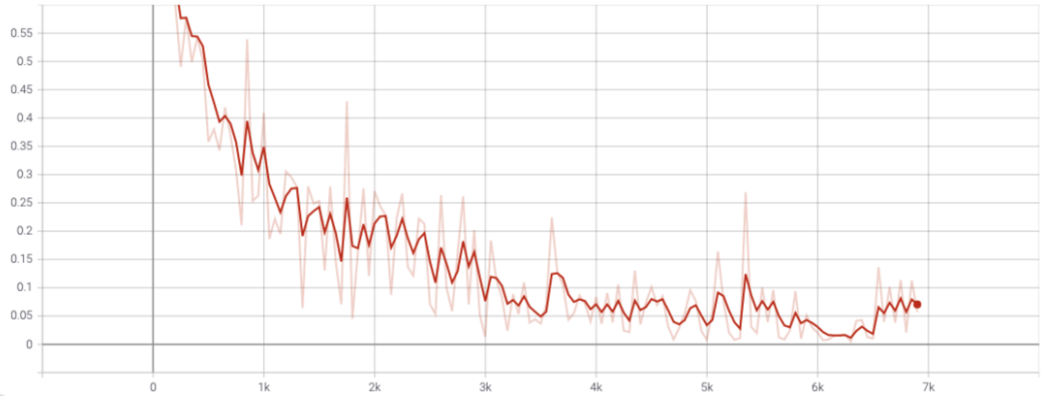

Figure 6. 17: Exp. II, the training loss curve.

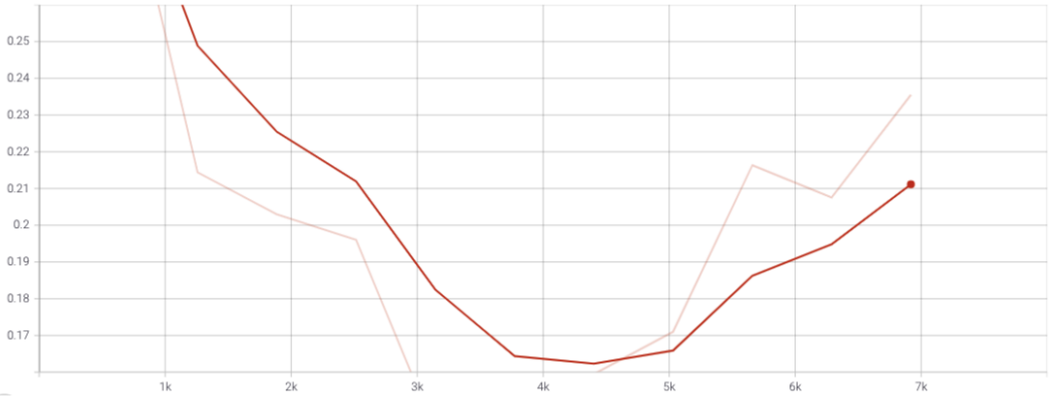

Figure 6. 18: Exp. II, the validation loss curve.

6.5.3 Prediction

As mentioned in the first experiment, I have used three sentences and fed them to the model and see the model's predictions. Table 6.14 shows the model results.

Text	**Prediction**	
	Class	**Probability**
تطبيق جيد جدا بارك الله فيكم وشكرا	Not relevant	0.9956
	Not Clear	0.0055
	Non Functional Requirements	0.0052
التطبيق سيىء مبارح نزلتو ومش راضي يشتغل معي	Not clear	0.7098
	Not relevant	0.2053
	Bugs	0.1740
التطبيق فيه مشاكل بطيء وبضلو يعلق وفيه كثير مشاكل ما بدعم لغة عربية وكمان تطبيق يحارب المحتوى الفلسطيني	Non Functional Requirements	0.9804
	Bugs	0.4194
	Not Clear	0.0552

Table 6. 14: Model predications in Exp II.

6.5.4 Experiment result

Table 6.15 shows the matrix score per class, while table 6.16 shows the final matrices results of the whole model. I noticed that the final Recall and F1-score were improved compared to the first experiment, while Accuracy and Precision were better in the first experiment than this one.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Functional Requirements	0.78	0.75	0.77	181
Non Functional Requirements	0.81	0.68	0.74	333
Bugs	0.82	0.68	0.74	275
Not Relevant	0.96	0.96	0.96	1455
Not Clear	0.33	0.15	0.21	27
micro avg	0.91	0.86	0.88	2271
macro avg	0.74	0.64	0.68	2271
weighted avg	0.90	0.86	0.88	2271
samples avg	0.89	0.88	0.88	2271

Table 6. 15: Results per class in Exp II.

Matrix	**Accuracy**	**Precision**	**Recall**	**F1-score**
Value	0.94	0.90	0.86	0.88

Table 6. 16: Final results in Exp II.

6.6 Experiment III result

For the third experiment, the same model, hyperparameters, and optimization techniques were used. The only difference between the experiments is the number of classes.

6.6.1 AUROC per class

Table 6.17 show the AUROC results for each class in the third experiment, and figures 6.21, 6.22, 6.23 shows the AUROC curve for each class where the horizontal axes represent the number of Epocs while the vertical axes represent the AUROC value, note that the values in table 6.17 represent the value on the curve at the best Epoc which was the fifth Epoc in this experiment.

Class	**AUROC**
Informative	0.9821
Not Relevant	0.9834
Not Clear	0.9247

Table 6. 17: AUROC values per class in Exp. III.

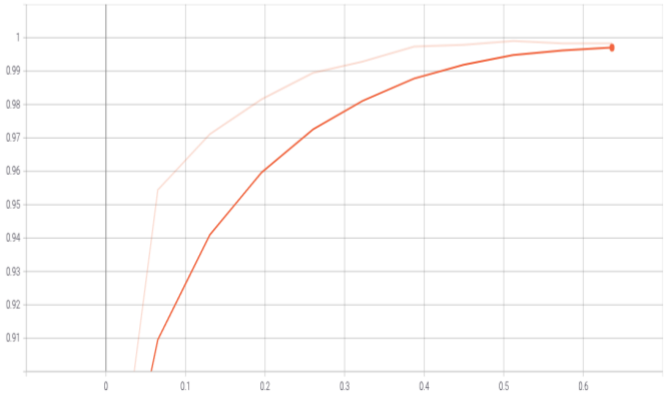

Figure 6. 19: Exp. III, AUROC curve for the informative class.

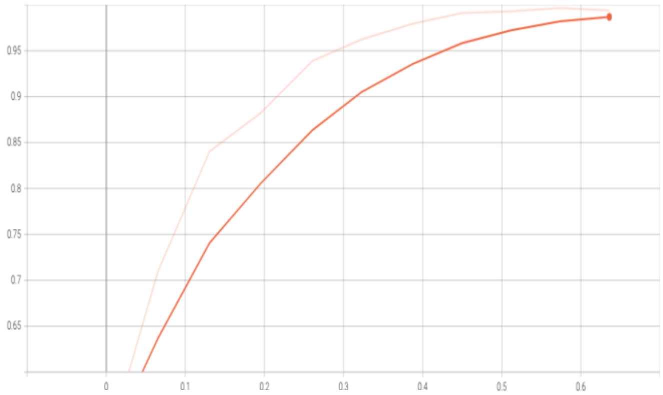

Figure 6. 20: Exp. III, AUROC curve for the not clear class.

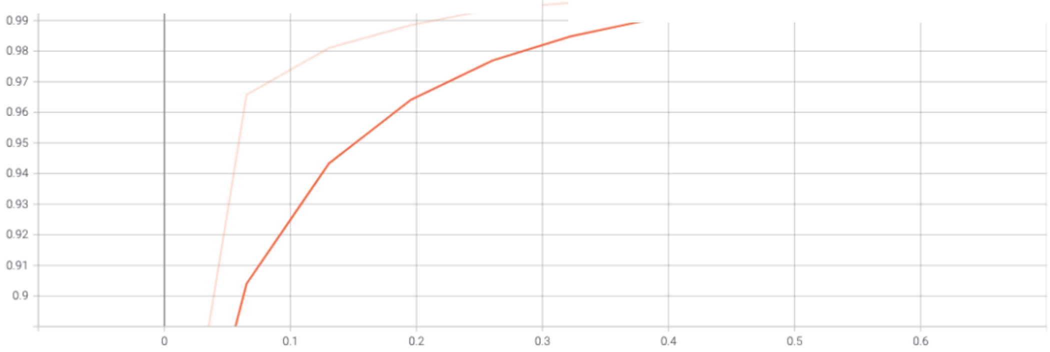

Figure 6. 21: Exp. III, AUROC curve for the not relevant class.

6.6.2 Train vs validation loss

Table 6.18 shows the validation and the training loss in the second experiment, while figures 6.24, 6.25 show the validation and train loss graphs. The validation loss in this experiment is higher than the training loss, which means the model is overfitting by 8.5%.

Matrix	**At the best Epoc #5**
Validation loss	0.11721
Training loss	0.03171
difference	**0.0855**

Table 6. 18: Train vs Validation loss in Exp III.

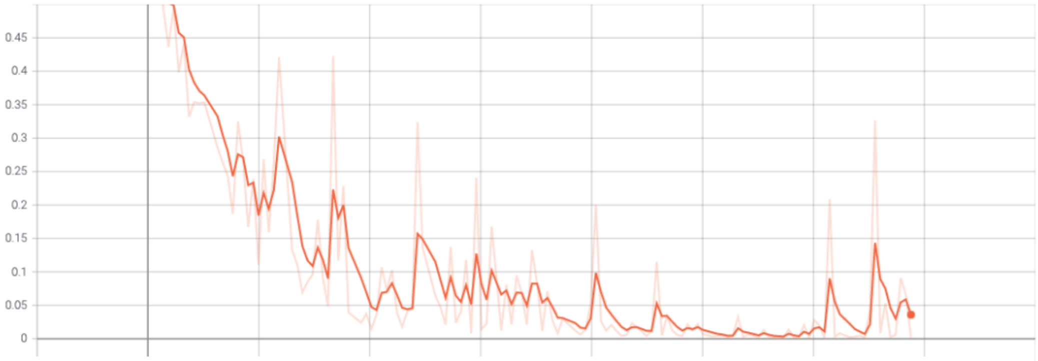

Figure 6. 22: Exp. III, the training loss curve.

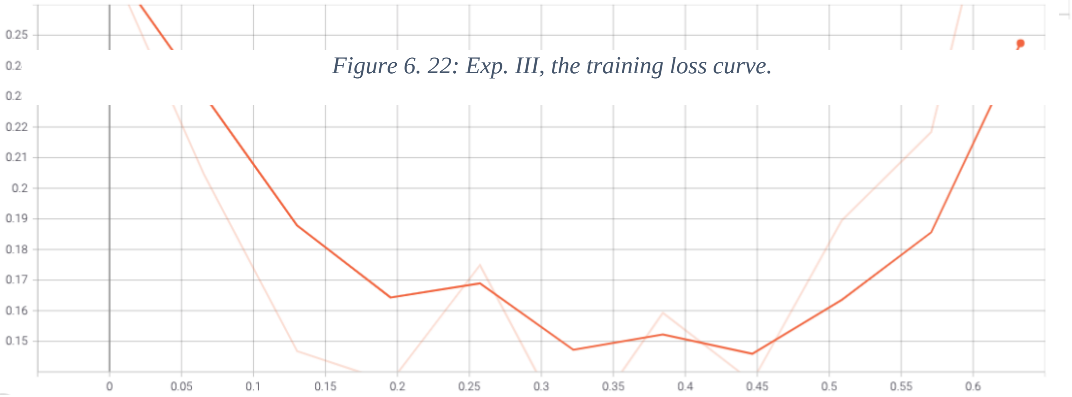

Figure 6. 23: Exp. III, the validation loss curve.

6.6.3 Prediction

Like the first and second experiments, I used three sentences, fed them to the model, and saw the model’s predictions. Table 6.19 shows the model results.

Text	**Prediction**	
	Class	**Probability**
تطبيق جيد جدا بارك الله فيكم وشكرا	Not relevant	0.9973
	Not Clear	0.0030
	Informative	0.0023
التطبيق سييء مبارح نزلتو ومش راضي يشتغل معي	Not Clear	0.6601
	Informative	0.2958
	Not Relevant	0.1505
التطبيق فيه مشاكل بطيء وبضلو يعلق وفيه كثير مشاكل ما بدعم لغة عربية وكمان تطبيق يحارب المحتوى الفلسطيني	Informative	0.9877
	Not Clear	0.0088
	Not relevant	0.0082

Table 6. 19: Model predications in Exp III.

6.6.4 Experiment Results

Table 6.20 shows the matrix score per class, while table 6.21 shows the final matrices results of the whole model. After re-visiting the first and second experiment results, this experiment outperformed the second experiment by Accuracy, Precision, Recall, and F1-score while it also outperformed the first experiment by Precision, Recall, and F1-score.

Class	**Precision**	**Recall**	**F1-score**	**Support**
Informative	0.89	0.88	0.88	518

Not Relevant	0.97	0.96	0.96	1455
Not Clear	0.41	0.26	0.32	27
micro avg	0.94	0.93	0.93	2000
macro avg	0.76	0.72	0.72	2000
weighted avg	0.94	0.93	0.93	2000
samples avg	0.93	0.93	0.93	2000

Table 6. 20: Results per class in Exp III.

Matrix	**Accuracy**	**Precision**	**Recall**	**F1-score**
Value	0.95	0.94	0.92	0.93

Table 6. 21: Final results in Exp III.

Chapter 7

Conclusion

7.1 Conclusions

This thesis presents fully automated approach that helps mobile app developers classify users' feedback on their applications into different classes. Three different experiments were conducted in this thesis using a dataset of 10K reviews collected from different apps belongs to different business domains.

The First experiment aims to classify the reviews to a corpus of 33 classes that describe a different set of users' requirements, potential software bugs, and some non-functional requirements and achieved an accuracy of 99% and 86% F1-score while the model was underfitting by only 0.06%. The second experiment aims to classy the reviews into five main classes: not relevant, not clear, user requirement, non-functional requirement, and software bugs, and achieved accuracy of 94% and 88% F1-score while the model was overfitting by 9.6%. The third experiment aims to classify the reviews into three parent classes: Informative reviews, not relevant and not clear, and achieved an accuracy of 95% and 93% F1-score while the model was overfitting by 8.5%.

During the manual classification process and after conducting the three experiments, it was found that the reviews on App platforms can be beneficial and could hold some vital information about missing feature or critical issues, or even to get the app developers attention to some new issues appeared after an App update for example. I also found that it is hard to manually analyze and classify every single review added to the App due to the enormous number of reviews, the number of reviews that are not informative and could be misleading, the differences in users' dialects, the typos in user's feedback, the long reviews that refer to different information and many other reasons. This accurate automated approach saves a lot of time and effort for the app developer to improve their application and align it with their users' needs.

7.2 Future work

In the future, the intent is to enrich the data with extra reviews from a different domain with different classes to improve the model performance. On the other hand, this model can be hosted on a server and exposed for external usage in the form of APIs or even a web application where the app developers can identify their application and get an analysis of the reviews continuously. It can also be integrated with another tool to continuously scrap all the new reviews from specific apps and feed them to the model to get the results for the App developers. Continuous learning is also an important feature that can be implemented to continuously feed and train the model and convert it to an expert model.

Other Ideas can also be implemented in the future, such as integrating the model with other open-source sentiment analysis tools and link the analysis of the users' reviews in this model to their sentiments.

Appendix A

- **Dataset:**

Available at:
https://drive.google.com/file/d/1m2i7kjAiLFsH7u41Gg8vyyD6Etl70K7c/view?usp=sharing

- **Colab Experiments:**

Experiment I is available at:
https://colab.research.google.com/drive/1adkyK7i3R0TDycdOKQT168gYk7ykeVRF?usp=sharing

Experiment II is available at:
https://colab.research.google.com/drive/1jQW9wUPi5nIH7OHzsExQkCPEM_HPR7sU?usp=sharing

Experiment III is available at:
https://colab.research.google.com/drive/10Ep1CjpOkDMId0XX_HdGaJNX1aFWdjD4?usp=sharing

References

[1] M Harman, Y Jia, Y Zhang, "App store mining and analysis: MSR for app stores,," *2012 9th IEEE Working Conference on Mining Software Repositories (MSR),* pp. pp. 108-111, doi: 10.1109/MSR.2012.6224306, 2012.

[2] E Guzman, W Maalej, "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews.," *IEEE 22nd International Requirements Engineering Conference (RE),* 2014.

[3] A. A. Content., "The State of Mobile in 2020: How to Win on Mobile - App Annie Content.," 2020. [Online]. Available: https://www.appannie.com/en/insights/market-data/state-of-mobile-2020/.

[4] W. Martin, F. Sarro, Y. Jia, Y. Zhang and M. Harman, "A Survey of App Store Analysis for Software Engineering," *IEEE Transactions on Software Engineering, 43(9),* pp. pp.817-847, 2017.

[5] N Al Kilani, R Tailakh, A Hanani, "Automatic Classification of Apps Reviews for Requirement Engineering: Exploring the Customers Need from Healthcare Applications," *Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS),* 2019.

[6] N. Farra, E. Challita, R. A. Assi and H. Hajj, "Sentence-Level and Document-Level Sentiment Mining for Arabic Texts," *IEEE International Conference on Data Mining Workshops.,* 2010.

[7] A Chader, L Hamdad, A Belkhiri, "Sentiment Analysis in Google Play Store: Algerian Reviews Case. In: Chikhi S., Amine A., Chaoui A., Saidouni D., Kholladi M. (eds) Modelling and Implementation of Complex Systems.," *MISC 2020. Lecture Notes in Networks and Systems, vol 156. Springer, Cham.,* 2021.

[8] A Shoukry, A Rafea, "Sentence-level Arabic sentiment analysis". *International Conference on Collaboration Technologies and Systems (CTS),* 2012.

[9] S AlOtaibi, MB Khan, "Sentiment Analysis Challenges of Informal Arabic Language", *The International Journal of Advanced Computer Science and Applications,* pp. Vol. 8, No. 2, PP. 278- 284, 2017.

[10] K. Darwish, N. Habash, M. Abbas, H. Al-Khalifa, H. Al-Natsheh, H. Bouamor, et al., "A Panoramic Survey of Natural Language Processing in the Arab World", *Communications of the ACM,* Vols. Vol. 64 No. 4, , pp. Pages 72-81, 2021.

[11] D. Abuaiadah, D. Rajendran and M. Jarrar, "Clustering Arabic Tweets for Sentiment Analysis", *The 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications,* pp. Pages(499-506), 2017.

[12] J Devlin, MW Chang, K Lee, K Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805,* 2018.

[13] Statcounter, "Mobile Operating System Market Share Worldwide," Global Stats, 08 2021. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[14] W. contributors, "Business requirements," Wikipedia, 6 2021. [Online]. Available: https://en.wikipedia.org/wiki/Business_requirements.

[15] J. Parker, "Business, User, and System Requirements," 2012. [Online]. Available: https://enfocussolutions.com/business-user-and-system-requirements/.

[16] altexsoft, "Functional and Nonfunctional Requirements: Specification and Types," 2018. [Online]. Available: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/.

[17] Jha, N., Mahmoud, A. Mining non-functional requirements from App store reviews. Empir Software Eng 24, 3659–3695 (2019). https://doi.org/10.1007/s10664-019-09716-7.

[18] Jha N., Mahmoud A. (2017) Mining User Requirements from Application Store Reviews Using Frame Semantics. In: Grünbacher P., Perini A. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2017. Lecture Notes in Computer Science, vol 10153. Springer, Cham. https://doi.org/10.1007/978-3-319-54045-0_20.

[19] U. C. Design, "User Centered Design," [Online]. Available: https://www.interaction-design.org/literature/topics/user-centered-design.

[20] N Bevan, J Carter, J Earthy, T Geis, S Harker, "What are user requirements? Developing an ISO standard", *Conference: HCI 2018: Human-Computer Interaction. Theories, Methods, and Human Issues ,* 2018.

[21] M. Bano, "Aligning services and requirements with user feedback," *IEEE 22nd International Requirements Engineering Conference (RE),* 2014.

[22] I. Scaled Agile, "Nonfunctional Requirements," 2021. [Online]. Available: https://www.scaledagileframework.com/nonfunctional-requirements.

[23] GoodFirms, "What is a Software Bug?," [Online]. Available: https://www.goodfirms.co/glossary/software-bug/.

[24] D. Inc., "Artificial Intelligence 101: Everything You Need to Know To Understand AI," 2017. [Online]. Available: https://medium.com/@diamond_io/artificial-intelligence-101-everything-you-need-to-know-to-understand-ai-8e20fe4bd750.

[25] R. Horev, "BERT Explained: State of the art language model for NLP," 2018. [Online]. Available: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[26] G. M. L. Notebook, "Linear Classifiers," [Online]. Available: https://sites.google.com/site/machinelearningnotebook2/classification/binary-classification/linear-classifiers.

[27] I Loshchilov, F Hutter, "Decoupled weight decay regularization", *arXiv preprint arXiv:1711.05101.*, 2017.

[28] W Maalej, H Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews", *IEEE 23rd International Requirements Engineering Conference (RE),* 2015.

[29] AF Otoom, S Al-jdaeh, M Hammad, "Automated Classification of Software Bug Reports", *9th International Conference on Information Communication and Management - ICICM 2019,* 2019.

[30] D Pagano, W Maalej,"User feedback in the appstore : an empirical study", *In Proc. of the International Conference on Requirements Engineering.,* pp. pages 125–134, 2013, 2013.

[31] LVG Carreno, K Winbladh, "Analysis of user comments: an approach for software requirements evolution", *ICSE '13 Proceedings of the 2013 International Conference on Software Engineering,,* p. pages 582–591, 2013.

[32] E Guzman, W Maalej, "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews", *2014 IEEE 22nd International Requirements Engineering Conference (RE),* 2014.

[33] H. Yang and P. Liang, "“Identification and classification of requirements from app user reviews.," *SEKE,* p. pp. 7–12., 2015.

[34] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews", *Proceedings of the 10th Working Conference on Mining Software Repositories,* p. pp. 41–44., 2013.

[35] C Chiu, RJ Sung, YR Chen, CH Hsiao, "App review analytics of free games listed on google play", *Proceedings of the 13th International Conference on Electronic Business, Singapore,* 2013.

[36] M Jarrar, F Zaraket, R Asia, H Amayreh, "Diacritic-based Matching of Arabic Words", *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP),* 2018.

[37] M Jarrar, H Amayreh, "An Arabic-Multilingual Database with a Lexicographic Search Engine", *24th International Conference on Applications of Natural Language to Information Systems (NLDB 2019),* vol. vol. 11608, pp. pp. 234-246, 2019.

[38] Mohammed et al., 2014 N.A. Mohammed, M.A. Izzat, H.G. Amal, A.W. Heider, M.H. Mohamad, "Opinion Mining and Analysis for Arabic Language", *International Journal of Advanced Computer Science and Applications,* 2014.

[39] M Jarrar, N Habash, D Akra, N Zalmout, "Building a corpus for Palestinian Arabic: a preliminary study. In Proceedings –Arabic Natural Language Processing Workshop", *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014),* pp. (pp. 18-27), 2014.

[40] Jarrar, M., Habash, N., Alrimawi, F. *et al.*, "Curras: an annotated corpus for the Palestinian Arabic dialect", *Language Resources and Evaluation,* 2016.

[41] A Elnagar, YS Khalifa, A Einea, "Hotel Arabic-Reviews Dataset Construction for Sentiment Analysis Application", *Shaalan K., Hassanien A., Tolba F. (eds) Intelligent Natural Language Processing: Trends and Applications. Studies in Computational Intelligence,* vol. vol 740, 2018.

[42] I. Hmeidi et al., "Automatic Arabic text categorization: A comprehensive comparative study", *Journal of Information Science, 41(1),* pp. pp.114-124, 2015.

[43] S Areed, O Alqaryouti, B Siyam, K Shaalan, "Aspect-Based Sentiment Analysis for Arabic Government Reviews", *Abd Elaziz M., Al-qaness M., Ewees A., Dahou A. (eds) Recent Advances in NLP: The Case of Arabic Language. Studies in Computational Intelligence,* vol. vol 874., 2020.

[44] A Fuad, M Al-Yahya, "Analysis and Classification of Mobile Apps Using Topic Modeling: A Case Study on Google Play Arabic Apps", *Complexity, vol. 2021, Article ID 6677413, 12 pages,* 2021.

[45] arabert, "Araber github repository," github, 2020. [Online]. Available: https://github.com/aub-mind/arabert.

[46] M Djandji, F Baly, W Antoun, H Hajj, "Multi-Task Learning using AraBert for Offensive Language Detection", *European Language Resource Association,* 2020.

[47] D Faraj, M Abdullah, "SarcasmDet at Sarcasm Detection Task 2021 in Arabic using AraBERT Pretrained Model", *Association for Computational Linguistics,* 2021.

[48] IA Farha, W Zaghouani, W Magdy, "Overview of the wanlp 2021 shared task on sarcasm and sentiment detection in arabic", *In Proceedings of the Sixth Arabic Natural Language Processing Workshop,* 2021.

[49] A Hussein, N Ghneim, A Joukhadar, "DamascusTeam at NLP4IF2021: Fighting the Arabic COVID-19 Infodemic on Twitter Using AraBERT", *Association for Computational Linguistics,* 2021.

[50] Shaden Shaar, Firoj Alam, Giovanni Da San Martino, Alex Nikolov, Wajdi Zaghouani, and Preslav Nakov, "Findings of the NLP4IF- 2021 Shared Task on Fighting the COVID- 19 Infodemic and Censorship Detection", *In Proceedings of the Fourth Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda,* 2021.

[52] A. Wadhawan, "Dialect Identification in Nuanced Arabic Tweets Using Farasa Segmentation and AraBERT," *arXiv preprint arXiv:2102.09749.,* 2021.

[52] AN Alsaleh, E Atwell, "Quranic Verses Semantic Relatedness Using AraBERT", *Proceedings of the Sixth Arabic Natural Language Processing Workshop,* 2021.

[53] W Antoun, F Baly, H Hajj, "Arabert: Transformer-based model for arabic language understanding" *arXiv preprint arXiv:2003.00104,* 2020.

[54] K Darwish, H Mubarak, "Farasa: A New Fast and Accurate Arabic Word Segmenter", *European Language Resources Association (ELRA),* 2016.

[55] pytorch. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.Linear.html#torch.nn.Linear.

[56] pytorch, Pytorch, [Online]. Available: https://pytorch.org/docs/stable/generated/torch.sigmoid.html.

[57] T. Boyle, Towardsdatascience, 2019. [Online]. Available: https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18.

[58] S. Narkhede, "Understanding AUC - ROC Curve," towardsdatascience, 2018. [Online]. Available: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5.

[59] Interaction Design Foundation, [Online]. Available: https://www.interaction-design.org/literature/topics/user-centered-design.

[60] I. D. Foundation, "User Centered Design," Interaction Design Foundation, [Online]. Available: https://www.interaction-design.org/literature/topics/user-centered-design.

[61] U. C. Design, "User Centered Design," Interaction Design Foundation, [Online]. Available: https://www.interaction-design.org/literature/topics/user-centered-design.