BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Master of Software Engineering

THESIS

Automatic Classifying of Requirements-relevant contents from App Reviews in the Arabic Language

---

Author: Alaa Isaac

Supervisor: Dr. Abualsoud Hanani

March 12, 2022

BIRZEIT UNIVERSITY

Faculty of Engineering and Technology
Master of Software Engineering

Automatic Classifying of Requirements-relevant contents from App Reviews in the Arabic Language

التصنيف التلقائي للبينات المتعلقة بالمتطلبات البرمجية من المراجعات العربية لتطبيقات الهواتف المحمولة

Committee:

Dr. Abualsoud Hanani: (Chairman of the Committee)

Dr. Yousef Hassouneh: (Member)

Dr. Ahmad S. Afaneh: (Member)

A thesis submitted in fulfilment of the requirements

for the degree of Maters in Software Engineering

March 12, 2022

BIRZEIT UNIVERSITY

Automatic Classifying of Requirements-relevant contents from App Reviews in the Arabic Language

Thesis

Author : Alaa Isaac

Approved by the thesis committee:

Dr. Abualsoud Hanani: (Chairman of the Committee)

[illegible]

Dr. Yousef Hassouneh: (Member)

[illegible]

Dr. Ahmad S. Afaneh: (Member)

[illegible]

Date of Defense:

Declaration of Authorship

I, Alaa Isaac, declare that this thesis titled, " Automatic Classifying of Requirements-relevant contents from App Reviews in the Arabic Language" 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 main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

Abstract

The market for mobile application development is continuing to thrive, with billions of users and millions of apps. Collecting software requirements for mobile apps has to cope with this trend, so as for the software to compete in this crowded scene. Therefore, efforts to analyze and mine mobile app reviews for requirements have shown a similar trend of increasing.

Among the billions of mobile users, there are hundreds of millions of Arabic-speaking users. According to our knowledge, this study would be one of the first studies in the field of mining Mobile app reviews for the assistance of Requirements Engineering, to direct its focus on Arabic reviews.

The relevant academic literature on the topics of mining Arabic text and App reviews mining has been studied. Besides the lack of studies on mining Arabic App reviews, the application of the machine learning techniques of deep learning on Arabic text deserves further research and enhancement, especially for improving the handling of the various dialects of Arabic.

A dataset of 7604 Arabic app reviews has been constructed and manually annotated by six experts. Each categorization aims at assisting one or more processes of software requirements engineering. Several deep learning approaches and configurations were tested in the experiments on the dataset. And finally, the results were utilized, in comparison with similar systems, in answering the research questions.

Various configurations of deep neural networks, namely, CNN, LSTM, and BLSTM, were used to classify the app reviews into the considered categories of the software requirement from the Arabic reviews. Furthermore, two word embeddings were utilized, on a pre-trained fasttext word vector, and another Word2Vec word vector, produced by this study.

The sentimental analysis results show that the LSTM classifier with the fasttext word embeddings gives the best F1-score, 79.17%. However, the BLSTM classifier with the fastText embeddings outperforms the other classifiers, with F1-score of 69.83%, when used for identifying the sub-categories of the user perspective main category.

The F1-score of classifying the sub-categories of the intention and topics with the LSTM and using fastText embeddings is 82.68% and 85,02%, respectively. These results outperform the other configurations of the classifiers and word embeddings.

ملخص

من الملاحظ أن سوق تطبيقات الهواتف المحمول في نمو مستمر، يصل إلى المليارات من المستخدمين و الملايين من التطبيقات. من جهة أخرى، فإن عملية جمع المتطلبات البرمجية تتطلب التماشي مع هذا التوجه، لكي تستطيع أن تبقى منافسة. لذا، فقد تنامت عمليات التحليل والتنقيب في مراجعات تطبيقات الهواتف المحمولة.

من أصل مليارات المستخدمين للهواتف الذكية، هناك مئات الملايين من المستخدمين المتحدثين بالعربية. حسب علم الكاتب، فإن هذه الدراسة تعد من أوائل الدراسات المختصة بمجالي التحليل والتنقيب عن المراجعات العربية لتطبيقات الهواتف المحمولة، لأجل تطويعها في مجال هندسة المتطلبات البرمجية.

في هذا الدراسة، تم التركيز على دراسة المؤلفات الأكاديمية المنشورة في مجال التنقيب في النصوص العربية، بالإضافة لمؤلفات التنقيب على المتطلبات البرمجية. لأن دراسة التنقيب عن البيانات العربية من خلال تقنيات التعلم العميق، والذي يعتبر من أساليب تعلم آلة، يحتاج إلى مزيد من التمحيص، خصوصا للتعامل مع أثر تنوع اللهجات العربية.

تم تجميع ٧٦٠٤ مراجعة عربية لتطبيقات الهواتف المحمولة، ومن ثم تصنيفها حسب عدة تصنيفات من قبل ست متخصصين. كل تصنيف استهدف دعم عملية أو أكثر من هندسة المتطلبات البرمجية. تم اجراء عدة تجارب تهدف لتدريب النظام المطروح على مجموعة البيانات، باستخدام عدد من تقنيات وإعدادات التعلم العميق. وتلاه استخراج النتائج وتحليلها ومقارنتها مع أنظمة مشابهة لإجابة اسئلة البحث.

تم استخدام عدد من إعدادات وتقنيات التعلم العميق، بالتحديد CNN, LSTM, BLSTM من أجل تصنيف مراجعات تطبيقات الهواتف المحمولة لعدد من التصنيفات المعتمدة على الدراسات السابقة، والخاصة بهندسة المتطلبات البرمجية. تبعا لذلك تم استخدام تقنيتين لاستنباط متجه الكلمة، بالتحديد fasttext و Word2Vec . في حالة fasttext تم تفعيل نموذج مدرب من قبل نفس الشركة على اللغة العربية. أما في حالة Word2Vec ، فقد تم تدريب نموذج من قبل الباحث على عدد كبير من مراجعات المستخدمين.

حقق استخدام تقنية LSTM بالإضافة لمتجه الكلمة fasttext أعلى نسبة كفاءة حسب تقيم F١ والتي حققت ٧٩٫١٧٪. كما حقق استخدام التقنيتين السابقتي الذكر أعلى نسبة كفاءة حسب تقيم F١ حسب تصنيفي النية

والموضوع، بالتحديد ٨٢,٦٨% و ٨٥,٠٢% على الترتيب. وكانت المقاربة بتحقيق تقنية BLSM بالإضافة لمتجه الكلمة fasttext أعلى نسبة كفاءة حسب تقيم F١، بالتحديد ٦٩,٨٣%، في تصنيف المراجعات حسب منظور المستخدم. بجمع هذه النتائج كانت بالمقارنة مع نتائج التوافيق الباقية من تقنيات التعلم العميق ومتجهات الكلمة في حالة كل تصنيف.

List of Abbreviations

LSTM	Long Short-Term Memory
CNN	Convolutional Neural Networks
BLSTM	Bidirectional Long Short-Term Memory
TP	True Positive
FP	True Negative
FN	False Negative
TN	False Positive
MSA	Modern Standard Arabic
CSV	Comma-Separated Values
TF-IDF	Term Frequency-Inverse Document Frequency
CBOW	Continuous Bag Of Words Model
QA	Quality Assurance

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to express my utmost appreciation to my supervisor, Dr. Abualsoud Hanani, for his dedication and support. His immense knowledge and plentiful experience have encouraged me in all the stages of the research project, and his insightful comments, suggestion, and guidance have been an inspiration in overcoming many difficulties during the research journey.

I want also to thank the faculty members at Birzeit University for their continued support and guidance.

Furthermore, I want to thank my colleagues at Zeva International, who were keen to support my research.

Last but not least, I would like to express my gratitude to my family for their encouragement, patience, and never-ending support.

Chapter 1

Introduction

1.1 Overview and motivation

As the number of mobile apps offered on common app stores exceeds millions[14], there is drastic attention drawn on optimizing apps, in order to compete in this highly congested market. Types of both paid and free apps offered on popular app stores haven't been this diverse. All the business challenges on such market competitors are reflected in the ability of the software development processes to offer dynamics to achieve maximum user satisfaction, in addition to increasingly attract new users. Thus, arises the importance of user feedback is an essential part of the mobile app development process[61].

Mobile app stores offer the tools to let users of apps rate and review them continuously. Additionally, the apps themselves usually encourage users to review and provide feedback on the apps, especially in correlation with new releases or services. Prior studies concluded that a good ratio of such reviews can be precious to software development and maintenance, including bugs, suggested features, and suggested updates to existing features. Therefore, collecting and analyzing app reviews could be an essential contributor to user feedback as an activity of the software development life-cycle.

For an app with a limited number of users, reviews can be inspected manually. However, for an app with tens of thousands of daily reviews, written in dozens of languages, manual inspection seizes to be an option. Therefore, machine learning techniques were applied to textual and non-textual data to try to classify and prioritize user reviews for various uses. Classifying reviews can help detect reviews with credible value for software development uses.

A large chunk of app reviews contains no informative data on how and why users are satisfied or dissatisfied with apps. They may contain vague praise of the app or a simple statement of dissatisfaction. Other reviews are completely silent on users' satisfaction or their lack of. Ratings typically provided in reviews could mismatch the textual content of the review itself. Therefore, classifying reviews stands as essential in extracting data from reviews.

There is a growing interest in applying sentiment analysis techniques and classification of Arabic texts. Several approaches have been introduced to face the challenges of classifying Arabic-based textual content based on linguistic features, machine learning, or both. However, there is yet an attempt to apply classification techniques to analyze Arabic app reviews for software requirements engineering purposes[55].

This thesis proposes to fill this gap, by evaluating the state-of-the-art features extraction and classification approaches that were previously applied in sentiment analysis, on Arabic mobile app reviews, taking into consideration the challenges of analyzing multi-dialect unstructured Arabic text.

In order to assist requirement engineering processes, the proposed framework applies deep learning classification models on the Arabic textual data of mobile app reviews. In that, we reviewed different taxonomies that have been applied in related and previous studies and choose the best to fit the goals of our study.

As the main goal of this thesis is to focus on Arabic app reviews, we explore the challenges that face applying machine learning techniques to the Arabic text, especially when it comes to preprocessing, feature extraction, and classification model construction phases. Taking into consideration the noisy multi-dialect form of the data.

As discussed in the literature[54], classifying user feedback by sentiment, intention, user experience, and topic can assist in achieving software requirements engineering goals. For instance, they can assist in eliciting software requirements, measure product acceptance, requirement prioritization, identify software bugs, suggest software enhancements, and clarify ambiguous software requirements.

We propose a framework for the automatic classification of Arabic app reviews into different categorizations that reflect their suggested uses in software requirements processes. Additionally, we propose a methodology for testing and evaluating our framework and the various proposed models of classification.

Furthermore, the study highlights the method used for data collection, preprocessing, feature extraction of mobile app reviews, in our case, from the Google play store.

1.2 Research objectives

The main aim of this thesis is to propose and implement a framework to enhance analyzing mobile app reviews, written in Arabic, for requirements elicitation. A prototype implementation of the proposed framework has been implemented, and proved its feasibility, to serve as the cornerstone of further research. In addition, we aim to collect a sufficient number of mobile app reviews, in the Arabic language, of mobile apps from different domains. The collected reviews need to be prepared and annotated. The annotation process is done manually by reading each review and assigning it to a specific category of the software requirements. The details of the annotation process and the considered categories of the software requirements are presented in the methodology chapter.

The state-of-the-art techniques in natural language processing and machine learning, particularly deep neural networks, are customized and investigated for the task of classifying requirement-relevant content from the user reviews written in the Arabic language.

1.3 Research questions

In this thesis, we are trying to tackle the following three main research questions:

- RQ1: To what degree do Arabic mobile app reviews contain useful information for software requirements engineering purposes?
- RQ2: Which deep learning architecture can better serve as a classification model for identifying and analyzing requirement-relevant information from the Arabic user feedback?
- RQ3: How accurate the state-of-the-art word embeddings in Arabic, such as word2vec and fastText, can be used for classifying requirement-relevant contents from the user feedback?

1.3.1 Contribution

The major contributions of this thesis can be summarized in the following points:

- Around 7604 Arabic app reviews were extracted from different types of mobile applications from different domains. The collected reviews are manually annotated by the thesis author and other experts in software development. The annotation follows the user feedback classification taxonomies described in Santos et al.[54].

- Two of the state-of-the-art word embeddings techniques, namely word2vec and fastText, were applied for the task of identifying requirement-relevant contents from the app reviews written in the Arabic language. In the case of word2vec, we generated a pre-trained word vector based on the full collected Arabic app reviews, namely 13.5 million reviews.
- Three variants of the deep neural networks, namely CNN, LSTM, and BLSTM, were customized and applied to classify the Arabic app reviews.
- The results of this research including the manual annotation suggest that Arabic Mobile App reviews do contain data that is relevant to Software Requirements engineering. Furthermore, the results of the experiments suggest that the proposed automatic classification solution, including the deep learning techniques applied, can be applied to the problem of Arabic Mobile App reviews, with good accuracy.

1.4 Structure of the thesis

The introduction has provided an overview of this proposal, detailed the research questions, touched on the proposed approach, and has suggested the contribution of the proposed solution.

Chapter 2 overviews key concepts that are essential to understanding the rest of the study.

In Chapter 3, we discuss the background of how analyzing Mobile app reviews can support different software development activities. Additionally, we discuss the relevant literature and state of the art state-of-theour topic of interest.

Chapter 4 outlines the research methodology which has been followed to collect the data, in addition to discussing the preprocessing strategy followed to clean the data. Furthermore, we discuss the research approach to follow, including data analysis, feature extraction, annotation, and learning models. Chapter 5 explores the results of the experiments done to evaluate our framework, with multiple configurations and settings. Then it digs deeper in asses the results, by using widely used metrics, and in comparison with similar systems. Thus, situating the research done in the wider spectrum of research.

Chapter 6 concludes what has been achieved so far in the research and discusses the suggested planning of future research.

Chapter 2

Background

In this chapter, we examine key concepts central to understanding the rest of the study, such as user review, requirements engineering, requirements elicitation, machine learning, and text classification.

2.1 Software requirements engineering

Requirements are usually identified in terms of their characteristics and relation to the product or the process. Fundamentally, not all textual descriptions of a system's features or qualities are to be considered requirements. A good requirement has to be unambiguous, measurable, testable, and achieve a value for the product or process[17].

Following the definition of requirements in this context, requirement engineering applies system engineering to the processes of ”discovering, developing, tracing, analyzing, qualifying, communicating, and managing requirements”[16], which defines the system.

2.2 Crowd-based requirements engineering

Current trends in the practice of software engineering are continuously challenging the traditional Software Engineering was realized. Shortening release cycles, adopting Agile methodologies, and applying Continuous Integration/Continuous Deployment practices, increase the demand for updating Software Requirements Engineering processes[48]. For instance, by taking advantage of analyzing data generated and saved in software systems, Data-Driven Requirements Engineering seeks to both automatically collect requirements and assist other Requirements Engineering processes[35].

Crowd-based requirements engineering is a popular manifestation of data-driven requirements engineering, as it seeks to analyze data from crowd users in order to assist requirements engineering [23]. Mobile app reviews and social media content are among the most used sources to apply this method, though more recent studies have tried to include other types of sources, such as mobile usage data[10].

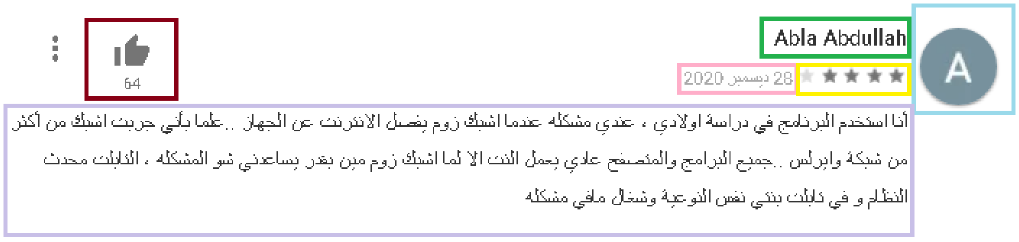
2.3 User feedback and automatic text classification

While mobile development shares most of the aspects of other kinds of software development, it differs in several key aspects. In mobile development, the software is usually run on a diverse type of mobile device, which adds complexity to testing, development, and maintenance[20]. Additionally, the software is delivered incrementally in short release cycles[47]. Furthermore, user feedback, which is an essential part of software development, is obtained mainly from end-users, via app store reviews, blogs, and social media. Hence, mobile development usually applies agile methodologies, and its release cycle reflects such dynamics.

Several decisions have to be made during planning a mobile app release[47]. Requirements are gathered, tested, and prioritized, in addition to the release time frame.

Afterward, based on the planning phase, coding and maintenance activities are conducted. Then the software is tested based on a variety of testing techniques. After making sure that the release is ready for publishing, the software gets deployed to app stores. What precedes is obtaining user feedback. This phase triggers the planning phase of the next release, and so on.

Ultimately, user feedback in the form of analyzed app reviews improves all steps. Classifying reviews in terms of reported bugs, suggested features, and enhancements helps to plan for the next release, specifically on what to work on and in which priority. Similarly, the information provided on reported bugs, in terms of how it is to be reproduced and fixed, enhances development and maintenance phases. Testing is enhanced further by adding test cases that have only been discovered via user app reviews. All in all, analyzed user app reviews provide a unique advancement of the mobile software development cycle.



Abla Abdullah

★★★★☆ 28 ديسمبر 2020

64

أنا استخدم البرنامج في دراسة اولادي ، عندي مشكله عندما اشبك زوم بفصل الانترنت عن الجهاز ..علما بأني جربت اشبك من أكثر من شبكة وايرلس ..جميع البرامج والمتصفح عادي يحمل النت الا لما اشبك زوم مين يقدر يساعدني شو المشكله ، الذابلت محدث النظام و في تابلت بنتي نفس النوعية وشغال مافي مشكله

Figure 2.1: Google Play review example

Generally, a mobile app review consists of a textual review, a star rating, reviewer information, review date, and in some cases other users' rating of the review, as shown in figure 2.1. Essentially, the textual content of a review is the main source of requirements for developers, though other information provides useful features in analyzing reviews. Therefore, automatic text classification is proved to be useful in analyzing reviews and extracting useful information for requirements engineering purposes, as discussed in chapter 3.

2.3.1 Sentimental analysis

What differentiates factual information from opinion is mainly the subjectivity of the latter. Opinions often describe the experiences of people with various sentiments about specific subjects. When it comes to digesting opinions, it's important to realize the identity of the opinion holder, and the time of them giving the opinion. Additionally, the sentiment of an opinion is what underlies it of emotions, feelings and evaluations aspects, and attitudes. A sentiment can be characterized by orientation, intensity, and type[64].

Sentiment analysis aims to recover the main aspects of opinions, in order to achieve a specific goal. These aspects include the aspects of the entity the opinion describes, the time of giving the opinion, the holder of the opinion, and the sentiment. The sentiment description has many forms, especially the three main forms of point-rating, star-rating, and positive, negative, or neutral annotations. Analysis can dig deeper to try to recover the reasoning behind the opinion or the limit of an opinion to a specific case or fragment of the entity. [64]

Fundamentally, the process of sentiment analysis includes five main steps:

1. Data retrieval
2. Data extraction and selection
3. Pre-processing
4. Feature Extraction and selection
5. Sentiment Classification

2.4 Text classification

The basic approach to classifying text into predefined categories is the keyword-based approach, which uses pattern matching to compare the text that needs to be classified with a set of keywords, corresponding to specific categories.

2.5 Arabic text classification

The Arabic language is one of the World's most spoken languages and is considered the native language in more than 25 countries. Native speakers often speak their region's spoken dialect and use Modern Standard Arabic (MSA) in formal settings such as the media and scholarship, use classical Arabic in religious and historical settings. Furthermore, Arabic dialects are also used in writing on social media platforms[59].

Written Arabic is distinguishable by its right-to-left orientation, its non-capitalized 28 letters, diacritical vocalization, and cursive lettering with letters taking multiple shapes of the script[59].

In addition to Arabic written in the Arabic script, the rise of social media has led to a trend of writing Arabic in Latin letters or Arabizi as it referred to[63]. In this form, Arabic letters that are un-mappable to Latin letters are written in the form of numbers or a combination of various letters, and there is no standard or agreed-upon way of writing Arabizi.

One of the main stages in performing textual classification in any language is pre-processing. In Arabic, that is usually includes parsing, text tokenization, stop word removal, stemming, term weighting, and part-of-speech marking.

Parsing is the process of realizing the text in smaller forms, such as sentences, documents, or paragraphs. Text tokenization, on the other hand, aims to divide a given text into tokens of small fragments of letters, words, or tokens. In many textual analysis processes, there are words that are highly frequent and have barely any effect on realizing the text. These are called stop words and are usually removed, in addition to other general numbers, words, and symbols. Removing irrelevant prefixes and suffixes are also applied, and is called stemming, or in another format, lemmatization. Stemming reduces the word into its stem format and is considered more heaving-handed than reducing a word into its lemma, or basic dictionary format[56].

2.6 Artificial neural networks

Among the commonly used techniques are artificial neural networks. Their name reflects the basic idea behind their architecture, which is to mimic learning in biological organisms, even though the analogy is often criticized as poor[1]. The Basic architecture of a neural network consists of input nodes, output nodes, and neurons. A perceptron is the basic building block of neural networks, containing multiple input nodes, one output node, with a bias neutron.

2.6.1 Deep learning

The concept of deep learning builds up upon artificial neural networks, using the idea of adding depth to reduce parameter requirements[1]. Thus, deep learning implements more complex architecture, with hidden layers and stepping functions.

2.6.2 Performance metrics

That concept to be distinguished from performance metrics as used in the study of Software engineering. In a typical classification scenario, there are multiple measures to assess the predictive ability of a classifier. However, it is better to understand these measures by relating them to the concept of confusion matrix, as shown in Table 2.1.

This table abstracts the results of a typical binary classification problem. The elements of a confusion matrix are detailed as follows[53]:

- True Positive(TP) indicates the correct prediction of the presence of the condition.
- False Positive(FP) is the incorrect prediction of the non-presence of a condition.
- True Negative (TN) indicates the correct prediction of the non-presence of a condition.
- False Negative(FN) happens in the case of an incorrect prediction of the presence of a condition.

		Actual	
Predicted		Positive	Negative
	Positive	TP	FP
	Negative	FN	TN

TABLE 2.1: Binary classification confusion matrix

Depending on the confusion matrix elements, multiple performance metrics can be calculated:

Precision

The proportion of the correctly classified positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall

The proportion of the correctly classified positive identifications.

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Accuracy

The ratio of the correctly classified over all classification instances.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{2.3}$$

F$_1$ score

The harmonic mean of Recall and Precision.

$$F_1 score = \frac{2 * Precision * Recall}{(Precision + Recall)} \tag{2.4}$$

Chapter 3

Literature Review

This Chapter details the state of art relating to the topics that this study is interested in, in addition to the gap of knowledge, that this study tries to fill. To make the survey clearer, the chapter was divided into sections.

3.1 Mining app store reviews

Mining app store reviews aims to extract useful information from user app reviews via an automatic process, to be utilized in mobile software development activities. Each review may contain multiple sentences, hence the possibility of multiple extricable pieces of useful information. Other extricable useful information, other than the textual content, includes the score of the review, its date, its language, developer reply, in addition to information about the writer. Information about the app, in terms of category and rating, can also be useful.

As proposed by Harman et al.[26], the app review analysis framework can be divided into three main phases:

1. Raw data extraction from the app store.

2. Parsing raw data according to the app's attributes into features, before parsing textual description.

3. Extracting features from the textual content of each review, including technical information, business information, and customer information.

3.2 Automatic text analysis

Both supervised and unsupervised machine learning algorithms have been used to analyze the textual content of app reviews. Supervised machine learning algorithms are used for purposes of classifying reviews into categories, based on the criteria that the researcher is interested in. One way of classification is to classify reviews to bugs, features, and feature status. Another way is to classify reviews to functional and non-functional requirements, or to several types of non-functional reviews. The training set is trained on a dataset of reviews that are manually classified. The feature vector is represented in the bag of words model, implying that words of the training set are represented into sets of numbers, against their number of occurrences in each review. The result is a mapping function that predicts the category of a review. Furthermore, in case a review consists of multiple sentences, each sentence can be categorized separately.

In contrast, unsupervised machine learning algorithms are used mainly to discover patterns in reviews that are otherwise harder to map manually for the training set. Thus, reviews are grouped in clusters, each of which targets a specific feature or topic. This approach could be used to group reviews to represent specific bugs and features. However, in this approach, human intervention is also inevitable, as clusters have to be interpreted to be useful for software engineering purposes.

In order to put this study in perspective, we examine both mentioned aspects of the literature on their own, then we try to explore studies that are more related to our research.

3.2.1 Classification of App reviews

Previous systematic literature reviews analyzed studies are done regarding this topic. Martin et al.[42], examined studies done on app store analysis for software engineering and found that there are 45 studies conducted on analyzing app reviews between the years of 2010-2015. Another study of the literature focused more on the specifics of mining app reviews[21]. We try here to summarize these findings and focus on a couple of studies that are the closest to our focus.

Past studies focused on producing classification models, to extract useful information about the app. Some studies focused on classifying the reviews to non-functional requirements and functional requirements, or specific types of each category[34, 30, 60, 36, 2]. This approach specifically targets assisting software requirement engineering during planning. Others have taken the approach of classifying app reviews to support requirements engineering during the maintenance phase. Such studies focused on the classification of categories regarding features and bugs[12, 39, 52, 51].

Regarding the second approach, there are multiple categorizations to classify reviews. Chen et al.[12] classified reviews into informative and uninformative. Others adopted different models, including bug reports, feature requests, praise, feature evaluation, and others[25, 24, 52, 51], with slightly different variations.

These Studies have offered multiple feature design approaches. Some studies focused on textual features, while others focused on linguistic and lexical features. Hybrid proposals were also tried. Textual features mostly depend on the Bag-of-Words model[36, 39, 34], Bag-of-characters model[24], and frequency-inverse document frequency (TF-IDF)[52, 51]. In contrast, linguistic features used include part-of-speech[34, 24], constituency parse tree[24], lemmatization[39], among others.

A review could contain multiple sentences and could represent multiple categories. Several studies considered the review as one sentence, therefore rendering the analysis at the review level[39]. Other studies tried to capture multiple sentiments in a review, therefore rendering the analysis at the sentence level[12, 52]. Other studies signified the multi-label aspect of a review, instead of trying to split it[43, 29].

In most cases, studies have used traditional supervised models. That is, to train an algorithm on a labeled set of app reviews, validate it, and then apply it to automatically classify reviews. Such algorithms include SVM[52, 51], Naive Bayes[39], Decision tree[39], and Gradient Boosted Regression tree[13]. Recent studies have tried to escape that paradigm by applying several techniques that have been used on text classification, namely deep learning. A simple Convolutional Neural Network has been proposed to ease the process of labeling[57].

3.3 Arabic sentiment analysis

There has been an increasingly significant number of studies of the application of Sentiment analysis for the Arabic language, and its related applications. Taking advantage of the vast literature of sentiment analysis on the English language, studies on Arabic focus on challenges that are witnessed specifically when dealing with the special nature of Arabic orthography, morphology, and syntax. Other studies try to explore challenging problems faced in other languages, but try to focus on the special way of dealing with it in Arabic.

As the field of study of Arabic Sentiment analysis is vast, multiple comprehensive literature review surveys were conducted in order to analyze it, especially in recent years. [6] et al. reviewed 118 studies written about topics relating to Arabic Sentiment Analysis. They discussed the different preprocessing used, the sources of datasets, the number of documents used in each dataset, the lexicon-based approach or the machine learning techniques applied, the features extracted, in addition to the Language(i.e. MSA only, or Dialects, or both). The survey concluded that applying combinations of

Deep Learning models is more promising than traditional machine learning techniques. Additionally, the survey concluded that the levels of accuracy that resulted from many studies were misplaced, as it compared datasets of different domains.

Oueslati et al.[50], has conducted another significant survey of Arabic Sentiment Analysis. In contrast to [6] et al., they put attention to the main challenges of Arabic Sentiment analysis, such as Arabic dialects and Arabizi, and how those contrasts with analyzing MSA only. They further studied the difference between the lexical-based approach and the corpus-based approach to sentiment analysis. Their findings show that analyzing Arabic text on the concept level is more suitable than the word level, as it is more fitting to handle various dialects.

The lexical-based approach to sentiment analysis offers a solid process of opinion mining. There are multiple notable lexicons that were constructed for Arabic, primarily either by mapping available English lexicons to Arabic or by applying Machine learning techniques. Most of them, however, focus on MSA and neglect dialectal Arabic and Arabizi, therefore minimizing their effectiveness in classifying informal Arabic text[6]. Notable studies that followed this approach are Badaro et al.[7] and Mahyoub et al.[41].

Other studies opted to apply machine learning techniques by manually labeling a dataset, and then constructing a machine learning model in order to perform classification. The preprocessing phase of machine learning was implemented in the literature by performing stopword removal, tokenization generic word removal, punctuation removal, stemming from its various manifestations, spell checking, normalization, and normalizing dialectal text. That is in addition to removing other unwanted textual content, such as URLs, hashtags, and emoticons[50].

As for applying Machine learning techniques, there were many studies and approaches. Social media content received the most attention, especially from Twitter. That means, for most cases, taking into consideration Arab dialects in addition to MSA, although restricting their focus on one or two Arabic Dialects. Other studies focused on content from news sites and online reviews, and such focusing on MSA alone.

As for what were the applied Machine learning techniques, there were also a variety of cases. The three most applied techniques were SVM(Support Vector Machine), KNN(K-nearest neighbor), and NB(Naive Bayes). Additionally, deep learning classification was applied in more recent studies[45]. We will discuss one major study that focused on typical machine learning techniques, and the other two that studied applying deep learning mechanisms.

In order to achieve better results when analyzing informal Arabic text, Maghfour[40] et al. proposed an approach of classifying text into MSA and Arab dialects, taking Moroccan Arabic as a test case, before performing the Opinion mining process. They collected a dataset of Facebook pages and news. The dataset of 9901 comments was manually annotated into sentiment categories, in addition to categorization into either MSA or Moroccan dialect. They compared the performance of one-step sentiment analysis, meaning without taking into account differentiation of the dialect, and a two-step classification, classifying MSA comments apart from Morrocan dialect comments. The features extracted were unigrams and bigrams, in addition to TF and TF-IDF. Furthermore, they experimented with different classifiers, namely NB and SVM. Their results show that two-step classification yielded better results, but not significantly.

Dahou et al.[15], on the other hand, opted to apply deep learning to analyze the text of MSA and Arabic dialects. They experimented with classifying multiple existing datasets of manually annotated Arabic tweets. Their approach applied different architectures that combined Differential Evaluation(DE) with CNN. Their results were promising in enhancing the existing state-of-the-art approaches, acknowledging the added complexity of the classification model.

While sentiment classification received the most attention in the literature, categorizing text-based other classes received some share of studies. Elnaga et al.[19] produced two huge annotated Arabic datasets of news posts. The first is called Sanad, which is a single-label annotated corpus, and Nadia, a multiple-label annotated corpus. The study experimented with nine deep learning architectures, in addition to investigating the use of the word2vec word embeddings model. The experiments yielded very good results for the single-label corpus, and less so for the multi-label corpus.

More recently, and during the final phase of this research, a couple of studies have been published on the topic of mining Arabic App Reviews. Chader et al.[11] focused on sentiment analysis of 50000 Google Play app reviews, both by lexical and SVM as classification techniques. The reviews contained MSA reviews, Algerian-dialect reviews, and french reviews. In the results, the lexical method outperformed machine learning, with 80% over 75%, respectively.

On the other hand, Qutaiba[44] collected 10000 Arabic mobile app reviews, belonging to 5 applications only. Then, the Arabic adaptation of the BERT deep learning technique was utilized to classify app reviews into several categorizations, in order to assess software engineering purposes. The results of the categories yielded an accuracy of 95%.

3.4 Highlight the gap of knowledge

To our knowledge, there are very limited studies on mining Arabic mobile app reviews for facilitating software engineering activities. This study tries to fill that gap. Furthermore, earlier studies on applying deep learning for Arabic Sentiment analysis suggest encouraging results when compared to traditional machine learning techniques. Therefore, further study is needed to achieve better results with minimum complexity.

Chapter 4

Research Methodology

4.1 Proposed system overview

In order to achieve the objectives of this thesis and build a system that can identify some of the software requirements for the user reviews of some mobile apps, an annotated sufficient dataset is needed. There are some available datasets with annotations in different languages and mostly in English, such as that provided by Nadeem Alkailani in his thesis [5]. To our knowledge, there is no available annotated dataset in Arabic language. Some of the available datasets focus on one type of software requirements (functional and non-functional), and some focus on apps in a specific domain, such as healthcare apps which Nadeem's data is focusing on. In this thesis, we intend to expand the study to include mobile apps from different domains and investigate various and a wider range of software requirements, and most importantly, in the Arabic language. Therefore, we decided to collect our own dataset and do the annotation manually. The manual annotation process is described in section 4.2.2. A web application is developed to assist the annotators, who are experts in the field of software engineering, in labeling the selected app reviews. Experts should be chosen from different backgrounds in software engineering development, preferably practitioners. Each expert is asked to label a subset of the dataset. Each subset has to represent the variation of mobile app categories.

Some preprocessing steps are applied to the dataset to enhance the reviews. The pre-processing steps, including text normalization, tokenization, and segmentation, are described in detail in section 4.3.1.

After data preprocessing, useful representative features are extracted and selected from the user reviews.

The final step of the proposed system is to use the produced features to build a machine learning classifier system that can identify the considered software requirements from the user review. The machine learning classifiers consist of constructing variants of deep learning architectures. In order to evaluate the overall system, several experiments are conducted, iterating through different feature extraction techniques, different classification algorithms, and the considered software requirement categories. Additionally, the two algorithms are combined in another experiment, as done in [3]. In order to experiment with the word embeddings, the system is evaluated with two main pre-trained word embeddings; word2vec provided by Google and fastetxt provided by Facebook.

4.2 Dataset

4.2.1 Data Collection

There are two main platforms for collecting Arabic app reviews, namely the Apple app store and Google Play store. Open source tools to scrap app reviews on both platforms are available. However, there is a serious limitation when it comes to collecting Arabic Apple app store reviews, as the platforms allow us to filter reviews by country only. Even filtering app reviews submitted by users belonging to Arabic-speaking countries yielded too many reviews that are not in Arabic. In contrast, open-source tools to scrap the Google Play store contain the option of filtering results by language. Nevertheless, such reviews could be in English written by users with an Arabic Language preference. Taking that into consideration, we chose to work only on Google Play Store reviews.

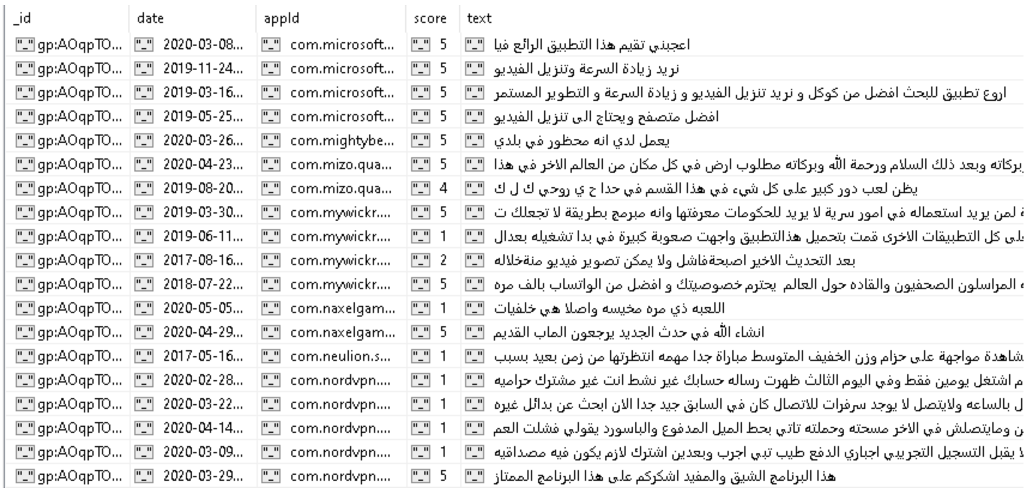
We have collected over 13.5 million raw Arabic reviews belonging to 5465 free mobile apps using an open-source Node.js based scraping tool[49]. Apps were selected from all major application categories, including educational apps, multimedia, medical, in addition to games. The number of included games is 1733. This study doesn't differentiate between games and other types of applications, as the main concern is requirement elicitation.

The data collected about the apps include their basic identifiers, in addition to category, score, description. Similarly, the data collected about the reviews include basic identifiers, with the score, date, and textual content. To avoid complexity, no data about developers' replies were collected, though such data could prove to be beneficial to software life-cycle analysis.

The process of Google play scraper consists of three phases. The first phase aims to fetch identifiers of apps that belong to some category in the Google Play store, whether free or paid, whether gaming or educational. In the second phase, a list of reviews is obtained in pages. And in the third phase, details of reviews are scrapped based on a defined number of pages. To construct the dataset, we iterated through all top free apps of each Google play category, scraping the maximum of 20,000 Arabic reviews of each. Not all apps had that high number, though some had considerably more.

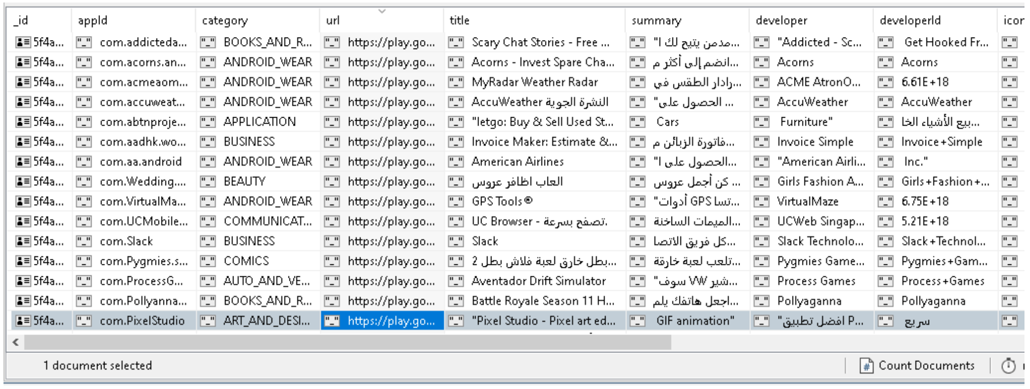
Out of this corpus, we randomly selected 8,000 reviews for 235 apps, belonging to all categories with the same weight. After conducting prepossessing on the collected reviews, many reviews ended up with empty textual content. Additionally, a set of reviews were found annotated with low confidence, and as such, neglected. Therefore, we were left with 7,604 reviews for analysis.

The collected reviews and app details were stored in a MongoDB database. It was preceded by importing all the data into CSV files. The choice of the NoSQL database aims to minimize errors in data collection, as such type of database management systems allows to store the data objects in semi-structured documents instead of structured table rows. This feature helps in transforming data structure on the go. Figure 4.1 shows a sample of app reviews obtained and figure 4.2 shows a sample of app details.



_id	date	appId	score	text
gp:AOqpTO...	2020-03-08...	com.microsoft...	5	اعجبني تقيم هذا التطبيق الرائع فيا
gp:AOqpTO...	2019-11-24...	com.microsoft...	5	نريد زيادة السرعة وتنزيل الفيديو
gp:AOqpTO...	2019-03-16...	com.microsoft...	5	اروع تطبيق للبحث افضل من كوكل و نريد تنزيل الفيديو و زيادة السرعة و التطوير المستمر
gp:AOqpTO...	2019-05-25...	com.microsoft...	5	افضل متصفح ويحتاج الى تنزيل الفيديو
gp:AOqpTO...	2020-03-26...	com.mightybe...	5	يعمل لدي انه محظور في بلدي
gp:AOqpTO...	2020-04-23...	com.mizo.qua...	5	بركاته وبعد ذلك السلام ورحمة الله وبركاته مطلوب ارض في كل مكان من العالم الاخر في هذا
gp:AOqpTO...	2019-08-20...	com.mizo.qua...	4	يظن لعب دور كبير على كل شيء في هذا القسم في حدا ح ي روحي ك ل ك
gp:AOqpTO...	2019-03-30...	com.mywickr....	5	ة لمن يريد استعماله في امور سرية لا يريد للحكومات معرفتها وانه مبرمج بطريقة لا تجعلك ت
gp:AOqpTO...	2019-06-11...	com.mywickr....	1	على كل التطبيقات الاخرى قمت بتحميل هذالتطبيق واجهت صعوبة كبيرة في بدا تشغيله بعدال
gp:AOqpTO...	2017-08-16...	com.mywickr....	2	بعد التحديث الاخير اصبحةفاشل ولا يمكن تصوير فيديو منةخلاله
gp:AOqpTO...	2018-07-22...	com.mywickr....	5	ه المراسلون الصحفيون والقاده حول العالم يحترم خصوصيتك و افضل من الواتساب بالف مره
gp:AOqpTO...	2020-05-05...	com.naxelgam...	1	اللعبه ذي مره مخيسه واصلا هي خلفيات
gp:AOqpTO...	2020-04-29...	com.naxelgam...	5	انشاء الله في حدث الجديد يرجعون الماب القديم
gp:AOqpTO...	2017-05-16...	com.neulion.s...	1	شاهدة مواجهة على حزام وزن الخفيف المتوسط مباراة جدا مهمه انتظرتها من زمن بعيد بسبب
gp:AOqpTO...	2020-02-28...	com.nordvpn....	1	م اشتغل يومين فقط وفي اليوم الثالث ظهرت رساله حسابك غير نشط انت غير مشترك حراميه
gp:AOqpTO...	2020-03-22...	com.nordvpn....	1	ل بالساعه ولايتصل لا يوجد سرفرات للاتصال كان في السابق جيد جدا الان ابحث عن بدائل غيره
gp:AOqpTO...	2020-04-14...	com.nordvpn....	1	ن ومايتصلش في الاخر مسحته وحملته تاتي بحط الميل المدفوع والباسورد يقولي فشلت العم
gp:AOqpTO...	2020-03-09...	com.nordvpn....	1	لا يقبل التسجيل التجريبي اجباري الدفع طيب تبي اجرب وبعدين اشترك لازم يكون فيه مصداقيه
gp:AOqpTO...	2020-03-29...	com.nordvpn....	5	هذا البرنامج الشيق والمفيد اشكركم على هذا البرنامج الممتاز

FIGURE 4.1: Representation of sample of app reviews



_id	appId	category	url	title	summary	developer	developerId	icor
5f4a...	com.addicteda...	BOOKS_AND_R...	https://play.go...	Scary Chat Stories - Free ...	"I مدمن يتيح لك...	"Addicted - Sc...	Get Hooked Fr...	
5f4a...	com.acorns.an...	ANDROID_WEAR	https://play.go...	Acorns - Invest Spare Cha...	انضم إلى أكثر م...	Acorns	Acorns	
5f4a...	com.acmeaom...	ANDROID_WEAR	https://play.go...	MyRadar Weather Radar	رادار الطقس في...	ACME AtronO...	6.61E+18	
5f4a...	com.accuweat...	ANDROID_WEAR	https://play.go...	AccuWeather النشرة الجوية	"الحصول على ...	AccuWeather	AccuWeather	
5f4a...	com.abtnproje...	APPLICATION	https://play.go...	"letgo: Buy & Sell Used St...	Cars	Furniture"	بيع الأشياء الخا...	
5f4a...	com.aadhk.wo...	BUSINESS	https://play.go...	Invoice Maker: Estimate &...	فاتورة الزبائن م...	Invoice Simple	Invoice+Simple	
5f4a...	com.aa.android	ANDROID_WEAR	https://play.go...	American Airlines	"I الحصول على...	"American Airli...	Inc."	
5f4a...	com.Wedding....	BEAUTY	https://play.go...	العاب اظافر عروس	كن أجمل عروس ...	Girls Fashion A...	Girls+Fashion+...	
5f4a...	com.VirtualMa...	ANDROID_WEAR	https://play.go...	GPS Tools®	"أدوات GPS تسا...	VirtualMaze	6.75E+18	
5f4a...	com.UCMobile...	COMMUNICAT...	https://play.go...	UC Browser - تصفح بسرعة.	الميمات الساخنة...	UCWeb Singap...	5.21E+18	
5f4a...	com.Slack	BUSINESS	https://play.go...	Slack	كل فريق الاتصا...	Slack Technolo...	Slack+Technol...	
5f4a...	com.Pygmies.s...	COMICS	https://play.go...	بطل خارق لعبة فلاش بطل 2...	تلعب لعبة خارقة...	Pygmies Game...	Pygmies+Gam...	
5f4a...	com.ProcessG...	AUTO_AND_VE...	https://play.go...	Aventador Drift Simulator	"شير VW سوف...	Process Games	Process+Games	
5f4a...	com.Pollyanna...	BOOKS_AND_R...	https://play.go...	Battle Royale Season 11 H...	اجعل هاتفك يلم...	Pollyaganna	Pollyaganna	
5f4a...	com.PixelStudio	ART_AND_DESI...	https://play.go...	"Pixel Studio - Pixel art ed...	GIF animation"	"افضل تطبيق P...	سريع	

1 document selected

Count Documents

FIGURE 4.2: Representation of sample of app details

4.2.2 Classification taxonomy

Building on the study of Santos et al.[54], we used four main categorizations for the user reviews. They are as follows:

- **Sentiment**: the reviews are classified into positive, negative, or irrelevant(i.e. neutral). This classification corresponds neatly to the categories of praise, complaint, or others, that is witnessed otherwise in the literature[39].
- **Intention**: relevant reviews shall reflect their specific user intentions, which are captured in the following categories:
  - Informing: the review provides information about the app, or a specific feature, whether describing some experience with a feature or noticing a missing feature.
  - Requesting: the review reports a bug, unexpected behavior, failure, or crash.
  - Reporting: the review requests a new feature, an enhancement for an existing feature, or seeks piece of information.
  - Irrelevant: the review does not reflect any of the above-mentioned categories or is irrelevant.
- **Topic**: This categorization reflects the relation between the content of the review and the app or its content, as follows:
  - Product quality: the review is concerned with the app's main features, or its quality attributes, such as security and usability.
  - Product context: the review is concerned with the content of the app, the app itself, or specific updates of the app.
  - Other product-related: the review is concerned with the price of the app, its company, or describes a compliance issue.

  - Irrelevant: the review does not reflect any of the above-mentioned categories or is irrelevant.

- **User perspective**: this categorization reflects the relation between the content of the review and the app or its content, as follows:

  - Quality in use: the review expresses an experience resulting from using the app, or specific features and qualities, whether describing errors or effectiveness, from the perspective of the user.
  - User-oriented perception: the review describes the general user experience using the app, and their emotional reaction, whether satisfaction or frustration.
  - Product-oriented perception: the review describes the users' whole experience of using the app, its services, and content, in terms of likeability, likeability, or support.
  - Irrelevant: the review does not reflect any of the above-mentioned categories or is irrelevant.

Different classification groups are meant for different software engineering purposes, and to assist several requirements engineering processes[54]. Both intention and topic categorizations assist in requirements elicitation. Topic and intention categorizations help in prioritizing requirements. The sentiment, user perspective, and topic categorizations combined help in measuring feature and product acceptance.

4.3 System description

Figure 4.3 shows a basic outlook for the design of our proposed system. Any Mobile app development, with respect to its size or complexity, goes into the steps of release planning, development, testing, and deployment phases. Then feedback is collected, and the cycle is repeated. In order to assist in this cycle, we propose to build a model to analyze mobile app reviews. That the mobile app reviews are collected from mobile app stores, analyzed, and annotated by experts in a specific taxonomy. Then the collected reviews are preprocessed and normalized and have their features extracted. Then, in order to establish a classification model, or update an existing one the model is trained and fine-tuned. After this step is completed, the classification model will be ready to work.

After the previous steps are completed, the reviews of the specific app under development are collected continuously, and classified based on the taxonomy, and fed to the release planning phase, and as such to other phases.

To narrate an example of using the proposed taxonomy, we can use figure 5.1 as an example. The reviews are classified by our model with the corresponding classes. In the release phase, these reviews would be considered as a bug (negative and reporting classes) in an app's quality, based on the user report of being affected by the quality mentioned.

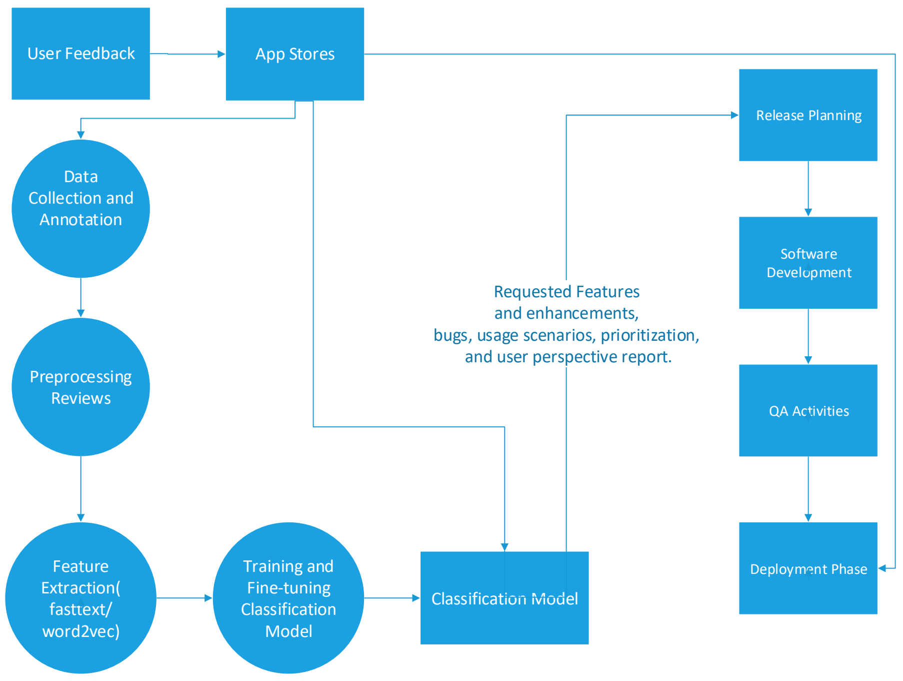

User Feedback
App Stores
Data Collection and Annotation
Preprocessing Reviews
Feature Extraction(fasttext/ word2vec)
Training and Fine-tuning Classification Model
Classification Model
Requested Features and enhancements, bugs, usage scenarios, prioritization, and user perspective report.
Release Planning
Software Development
QA Activities
Deployment Phase

FIGURE 4.3: Proposed system design in a simple mobile app release life-cycle

4.3.1 Data preprocessing

As expected, in any application dealing with highly noisy data, several steps of preprocessing need to be conducted. This process has been done using regular-expressions scripts written in Python programming language. In the following, we summarize the main steps of conducting preprocessing.

- Removing Arabic diacritics: While Arabic diacritics are important for the comprehension of Arabic words, removing them will simplify.

- Removing single-character words.

- Normalizing Arabic Vowels: Replacing final Ya' with Alif Maqsura, Teh Marboota with Ha', and Alef Mamdooda or Alef Mahmooza with Alef.
- Removing Non-Arabic letters: It's not uncommon for Arabic to be transliterated in Latin letters on social media, or as it is called the Arabic chat alphabet. We decided to sacrifice this data for the sake of simplicity. Additionally, it's not rare for Arabic writers to include words written in European languages to describe foreign concepts and things. However, such usages usually have very little to do with the category of the review.
- Removing punctuation marks: By taking this approach, we simplified the review as one sentence, without the ability to classify each sentence of a review.
- Cleaning redundant spaces: Instead of having multiple spaces between words, they were replaced with only one space at a place.
- Removing stop words: Arabic stop words are of no value in the process of review classification. On the contrary, they may lead to less accurate results. Therefore, it's better to eliminate them from the process. A list of Arabic stop words was used[18].
- Normalizing repetition of letters: This step removes redundant repetition of Arabic letters, as shown in Figure 4.4. Such repetition is common in social media platforms to emphasize the word, by generally repeating its vowels, or less commonly, its consonants to display the word at greater length. This phenomenon could be useful in sentiment analysis. Because this study is concerned with text classification, normalizing such words is beneficial.

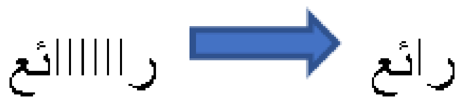


FIGURE 4.4: Normalizing repetition of letters

4.4 Feature extraction

To achieve more efficient results, the features were selected based on their performance with deep learning techniques. Word embeddings have shown good results when used for text classification with deep learning techniques. The idea is to produce vectors representing words[31]. The following two word embeddings techniques are used in our experiments. in both techniques, the provided pre-trained models can be used, and/or the models can be re-trained on a huge dataset, such as the data collected in this study.

- Word2Vec: In itself, Word2Vec is an implementation of deep neural networks that traverses a corpus, generating similarities based on co-occurrences, resulting in vectors that represent words. However, with similar words semantically with least distance[31]. The pre-trained models of the word2vec were trained on huge news data from Google.
- FastText: Similar to Word2Vec, fasttext is a word embedding developed by Facebook, and based on deep neural networks. The pre-trained model of the fastText was trained on a huge collection of Facebook posts.

4.4.1 Deep learning classification models

Based on the literature review conducted in this study, we chose two deep learning models in order to conduct our study, described in the subsequent sections.

Convolution Neural Networks (CNN):

CNN is a type of neural network that was originally designed for deep learning computer vision tasks primarily used in image recognition and classification. Now a day CNN is a state-of-the-art technique in text classification. It takes an input image as a 3-dimensional array based on the image resolution. The height and the width of the image represented 2 dimensions of the array. While the third dimension is the color of the pixel (RGB). CNN architecture is mainly composed of three layers, convolutional layer, pooling layer, and fully connected input layer [37]. Sentences are represented in N x K vectors representations. The first layer applies convolution operations to the vectors with a specific filter size. The next layer collects the result of its precedent and represents it in a long vector. The third step applies dropout operations to the data. And the final layer classifies the data using softmaxer layer. Figure 4.5 shows all the layers of CNNs.

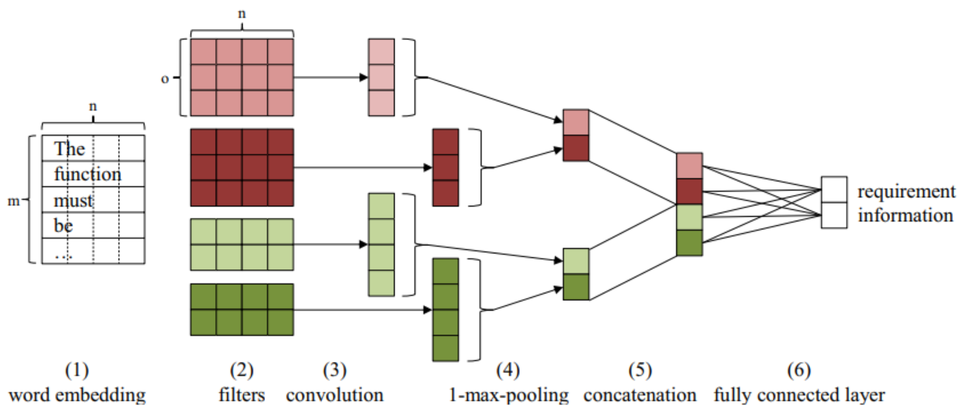


FIGURE 4.5: Convolutional neural network architecture, source: [62]

A major step to enhance the performance of this model requires word embeddings, which result in training an unsupervised machine learning model. To capture the effect of word embeddings, the experiment should find the results of their existence and their absence of the CNNs model.

In our system, we apply CNN model to identify and classify software requirements from the user reviews. The reviews are segmented into words. Each word is converted to vectors using the two feature extraction techniques. Using the Word2Vec and fastText, the reviews are converted into vectors by adopting the semantic meaning.

Long short-term memory (LSTM)

It is considered a type of Recurrent Neural Networks (RNN). The main idea behind it is handling classification problems on sequential data. That is, it remembers data between layers. It consists of two gates, an update gate and a forget gate. Figure 4.6 shows the basic functionality of LSTM.

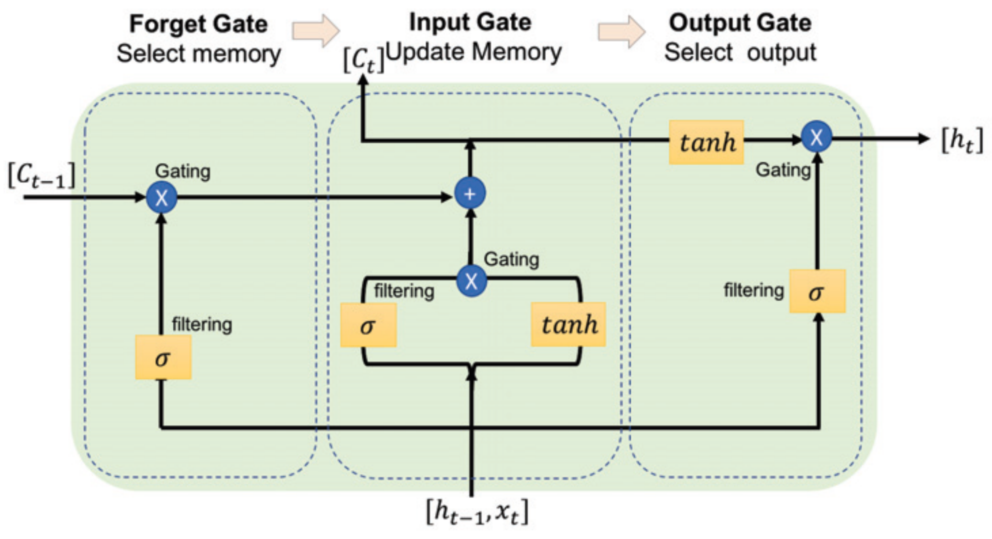


FIGURE 4.6: Long short-term memory architecture, source: [8]

Bidirectional Long Short-Term Memory (BLSTM)

As an enhancement on LSTM, in order to recognize word-dependency over a long body of data, BLSTM was designed [28]. So, BLSTM is the process of making any neural network o have the sequence information in both directions backward (future to past) or forward (past to future). However, in bi-directional, we can make the input flow in both directions to preserve the future and the past information.

4.5 Evaluation criteria

The precision, f1-score, and recall values are calculated for each categorization, in each experiment. More details about these performance measures and how they are calculated are presented in section 2.6.2. In this way, the performance is calculated for each algorithm by averaging the performance of the algorithm in all categories. Furthermore, it is handy to calculate the average run-time of each algorithm to investigate their practicality.

Chapter 5

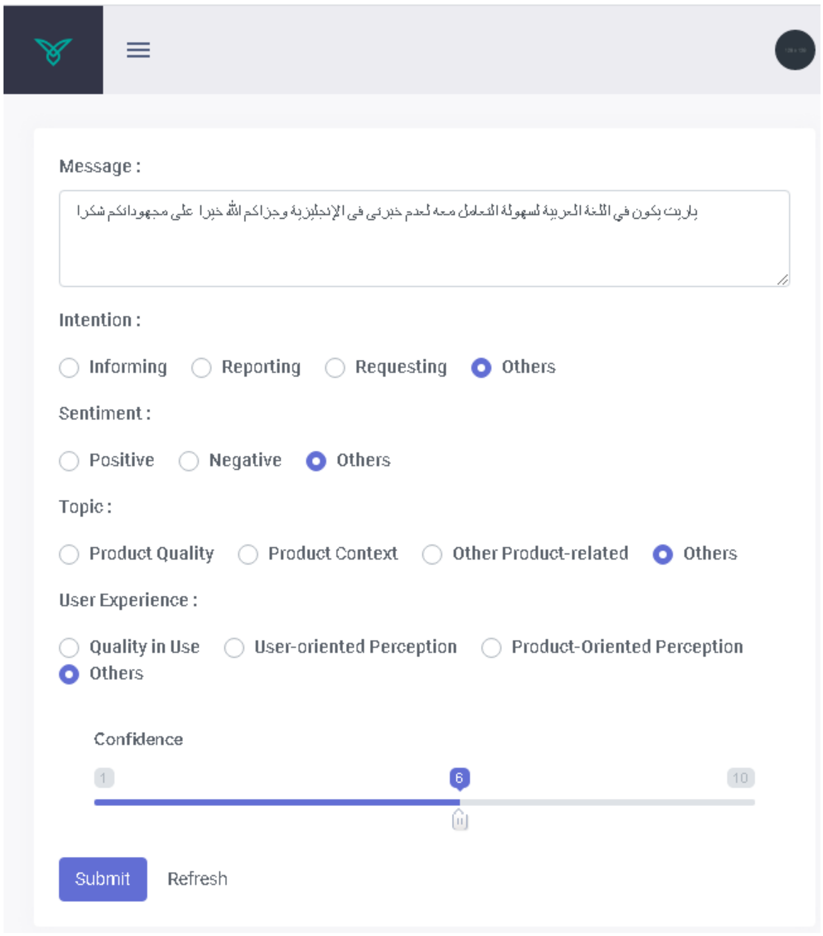
Experimental Results and Evaluation

In this chapter, the manual annotation results are detailed. The conducted experiments and the evaluation results are presented and discussed. The performance of the proposed models is evaluated based on several metrics such as precision, recall, accuracy, and F-measure.The performance of the proposed systems is compared with similar systems and experiments published in related studies, in order to better understand the results in perspective.

5.1 Manual annotation analysis (RQ1)

Accordingly, 7,604 distinct reviews were manually classified into the classification categories mentioned above. The starting number of reviews was 8000. However, the reviews that achieved a low confidence rating, specifically less than 50%, were neglected. In addition, 100 reviews were annotated by two reviewers, in order to perform the cross-validation process.

We developed a website tool for that purpose, with the main screen shown in Figure 5.1. Six reviewers were chosen for the initial classification, in addition to the author of this study. They are software engineering practitioners, with 3-12 years of experience in software development, including five software developers, and one product owner. They have experience in various types of software products, including Mobile development, Desktop App development, and cloud services. A session was conducted to explain the goal of the experiment, explain the classification tool, and identify the categories of classification. The classification was conducted on multiple sessions based on the experts' needs.



Message :

ياريت يكون في اللغة العربية لسهولة التعامل معه لعدم خبرتى فى الإنجليزية وجزاكم الله خيرا على مجهوداتكم شكرا

Intention :

Informing Reporting Requesting [✓ Others]

Sentiment :

Positive Negative [✓ Others]

Topic :

Product Quality Product Context Other Product-related [✓ Others]

User Experience :

Quality in Use User-oriented Perception Product-Oriented Perception [✓ Others]

Confidence

1 6 10

Figure 5.1: Manual classification Application

Each reviewer was guided to enter their confidence level in their manual annotation, as mentioned above. The share of each reviewer in the annotation process was as follows:

1. Reviewer 1(Author): 3500 reviews(43.75%).
2. Reviewer 2: 850 reviews(10.62%).
3. Reviewer 3: 800 reviews(10%).
4. Reviewer 4: 950 reviews(11.87%).
5. Reviewer 5: 900 reviews(11.25%).
6. Reviewer 5: 1000 reviews(12.5%).

Table 5.1, shows a sample result of a manually classified review. Figure 5.2, Figure 5.3, Figure 5.4, and Figure 5.5 respectively show the results of the manual annotation process.

To achieve the maximum number of annotated reviews while making sure the annotation process is accurate as much as possible, instead of cross-validating the whole annotated reviews by more than one expert, a test was done to cross-validate 100 reviews only. The cross-validation process included anotating the same review by exactly two reviewers, for each of 100 reviews, covering all reviewers. The rate of the classification of a review identically by the two experts by category was as follows:

- Sentiment: 88%
- Intention: 83%
- User Perspective: 73%
- Topic: 77%

Review text	Intent	Topic	User Perspective	Sentiment
لما افتحه و اعمل حساب يقول حدث خطأ ما يرجى اعاده المحاولة لاحقا!!	Reporting	Product Quality	Quality in Use	Negative

TABLE 5.1: Sample of manually annotated review

Figure 5.2 shows the manual annotation bar chart of classifying reviews by Intention of the review. It's clear that the plurality of the reviews was not considered irrelevant, that is not having useful information relating to either requesting information, reporting an issue or a bug, or informing about a use case or a comment. However, as shown, the majority of the sampled dataset contains relevant information.

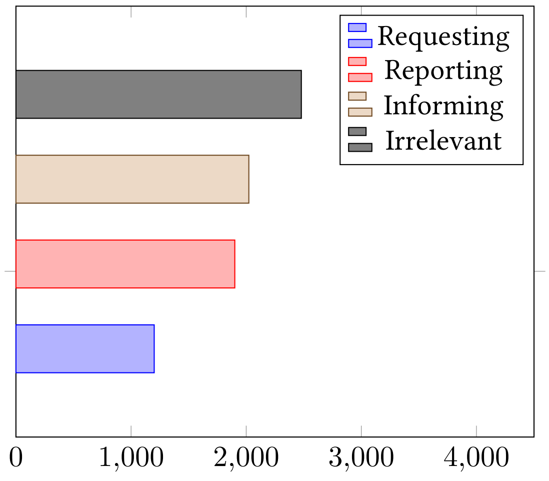


FIGURE 5.2: Manual Annotation Results of Category Intention

Annotating the sampled reviews based on sentiment produced a different picture. Figure 5.2 shows that the majority of the reviews are either considered Negative or Positive by experts.

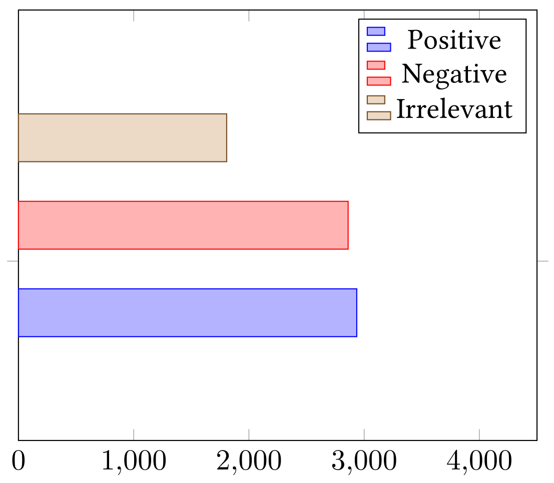


FIGURE 5.3: Manual Annotation Results of Category Sentiment

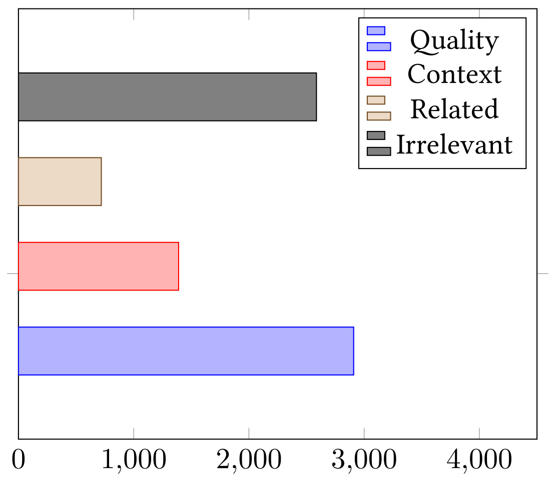


FIGURE 5.4: Manual Annotation Results of Category Topic

Based on expert annotation, the plurality of reviews was classified as Quality related, when classifying them by Topic, as shown in Figure 5.4. As mentioned in Chapter 4, the quality class expresses the app's main functionalities and non-functional quality attributes. Furthermore, in the next ranking, 34% of the sample were considered irrelevant, as not containing topic information. Notably, the results are less distributed across the categories.

Annotating reviews based on user perspective yielded different results, as shown in Figure 5.5. The majority of the reviews were considered by experts as relevant, spreading between the categories of Quality in Use, Product-oriented perception, and User-oriented perception. Only minority reviews were considered irrelevant. The plurality of the reviews was annotated as ”Quality in Use”, as in describing quality in the application, whether satisfaction or dissatisfaction with the app.

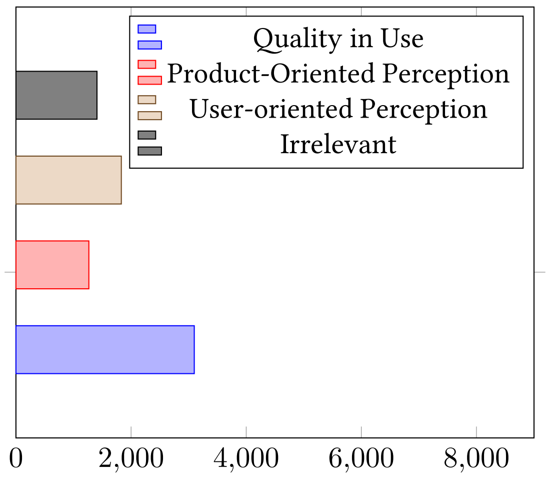


FIGURE 5.5: Manual annotation results of category user perspective

The results of the manual annotation process show clearly that, in all our categories, the majority of reviews were classified as having relative information for the purposes of assisting software engineering activities. Whether it is for measuring product acceptance, understanding a user usage scenario, user priorities, identifying bugs and issue reports, or suggesting enhancements, the annotated app reviews showed promising results.

5.2 Experimental setup

The raw data with the results of manual annotation, as described in section 5.1, was stored in CSV files containing the textual review data, and the label associated with it, for each experiment taxonomy. The data is then randomly split into three subsets, to avoid over-fitting:

- Training subset: Constituting 80% of the data, and is used for training the classifiers. In each epoch of 10 utilized, the classifier is fed the same training data in order to build the classification model.
- Validation subset: Constituting 10% of the data, and utilized in validating the performance of the model, and to tune the settings of the classifier.
- Testing subset: Constituting 10% of the dataset, and used for evaluation. The obtained results with the testing data are compared with the results of the validation data, and this shows how concrete the results are.

The preprocessing steps, described in chapter 4, are applied to the text reviews in the whole dataset, using the text manipulation tools. The resultant text of each review is tokenized into words and their indexes, using the Keras preprocessing package[1], with a vocabulary size of 45000. Furthermore, the label assigned to each review is represented as a binary number reflecting its vector value. It is important to note here that all the experiments were implemented in Python programming language in related environments, utilizing mainly the tensorflow platform[2].

Before digging deeper into the deep learning algorithms and their settings, we will focus on Word Embeddings that were used in the conducted experiments.

5.2.1 Word embeddings

As described in the previous chapters, word embedding is one of the most popular representations of words based on pre-trained models with deep learning techniques. It captures the context of a given word in a text, syntactic and semantic similarity with

[1]Keras. *Keras preprocessing*. URL: https://keras.io/api/preprocessing/ (visited on 11/30/2021).
[2]Tensorflow. *Tensorflow*. URL: https://www.tensorflow.org/ (visited on 11/25/2021).

other words. Our experiments show that building word embedding models using our training dataset only gives worse results compared with the pre-trained models, which are trained on a relatively huge dataset. Therefore, three word embedding models were used in the presented experiments. One is trained on the training subset of the collected data through this thesis, and one pre-trained word vector is provided by Google, called Word2Vec [38], and one is provided by Facebook, called fasttext.

Word2Vec

We tried to look for pre-trained Arabic Word2Vec models, but they were not focused on Arabic App reviews, and our preliminary tests have shown that they performed worse than our model. Therefore, their results will not be included below. For the sake of our experiments, we were able to create a word vector, representing the full length of our pre-processed Arabic app reviews, that is 13.5 million reviews, using the Word2Vec representation with CBOW (Continuous Bag of words) Model. In the results, we will refer to this word embedding as ”Reviews”, which was uploaded online[27].

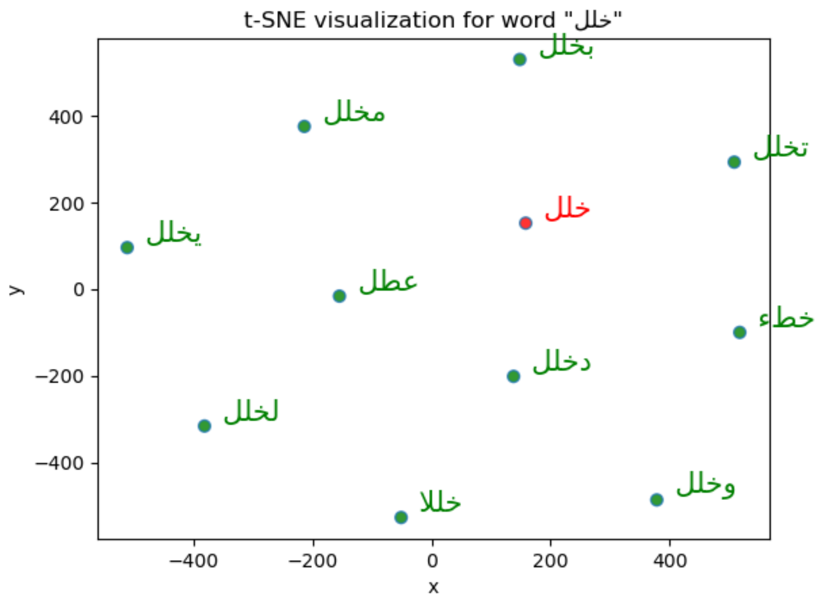


Figure 5.6: Visualizing a snapshot of Word Vector around the Word Ḫll(error)

Figure 5.6 shows the Word2Vec vector visualization of the word "khalal", which means failure, with the representations of the synonym words, dialect variation, unstemmed words, and writing errors. For instance, "khatta'" (error) is very related, and thus closer in representation to "ʿOttol" (mistake). By this, we can see that the Word2vec vectors represent the semantics of the words, so the vectors of similar words are close in the space.

Fasttext

In this case, we used a word-embedding pre-trained model created by the developer team of fasttext themselves, Facebook's AI Research (FAIR) lab[3]. The model was trained on a preprocessed huge crawled Modern Standard Arabic text dataset[22], using the CBOw method, thus suitable for our experiments. The lab offers another pre-trained

[3]Piotr Bojanowski and Edouard Grave. *Fasttext.* URL: https://research.facebook.com/blog/2016/08/fasttext/ (visited on 11/30/2021).

model on Egyptian Arabic text. However, our preliminary tests have indicated that the Modern Standard Arabic version consistently performed better. Therefore, the Egyptian Arabic model results will not be included below.

5.2.2 Deep learning classifiers

In each experiment, a single deep learning architecture, or classifier, was used. The parameters and configurations used for each classifier are as follows:

LSTM

In order to have a baseline system with the LSTM, we use the following hyper-parameters and configurations:

- Embeddings dimension: 300 embeddings. Which limits the state space for reconstruction, resulting in a fair amount of data points, without compromising on performance, and limiting the number of computations.
- Hidden layers: The number of hidden layers was set to 100.
- Randomization: For each epoch, the training data was randomized, in order to allow better optimization of the model.
- Activation function: Softmax was used, as it generalizes the Logistic function for the multi-class space.
- Number of Epochs: 10.
- Embeddings: Instead of generating embeddings from the training data subsequently, we opted to use pre-trained word vectors, as the training data is smaller than to achieve this task without compromising performance.
- Loss function: Our implementation used Categorical Cross-Entropy function, in order to measure the performance of the model during the training phase. This

loss function is the best fit for multi-class classification, as it measures the difference in the distribution of the multi-class predicted, related to the model performance.

- Optimizer: In this case, the Root Mean Squared Propagation was used.
- Layer weights initialization: Random.
- Layer Weight Regularization: Apply penalty to a layer's kernel weight.

BLSTM

In a similar fashion to the case of the LTSM, the baseline BLSTM configurations were as follows:

- Embeddings dimension: 300.
- Hidden layers: 100.
- Randomization: For each epoch, the training data was randomized, in order to allow better optimization of the model.
- Activation function: Softmax.
- Number of Epochs: 10.
- Embeddings: pre-trained.
- Loss function: Categorical Cross-Entropy function.
- Optimizer: the Root Mean Squared Propagation.
- Layer weights initialization: Random.
- Layer Weight Regularization: Apply penalty to a layer's kernel weight.

CNN

CNN differs from the other classifiers, and its parameters are as follows:

- Embeddings dimension: 300.
- Hidden layers: 100.
- Randomization: For each epoch, the training data was randomized, in order to allow better optimization of the model.
- Activation function: Rectified Linear Unit for Hidden layers, and Softmax for non-hidden layers.
- Number of Epochs: 10.
- Embeddings: pre-trained.
- Loss function: Categorical Cross-Entropy function.
- Optimizer: the Root Mean Squared Propagation.
- Convolutional Model: The convolutional model of Kim Yoon was adopted[33] as a reference, as it was designed for text classification. However, setting the parameters as follows:
  - Kernel sizes: 3, 4, 5.
  - Number of filters: 10.
  - Dropout rate: 0.5.
  - Weight regularization (L2): 3.

5.2.3 Model performance metrics

The performance metrics to evaluate the performance of the model during training, testing, and validation were as outlined in Subsection 2.6.2, Precision, Recall, and F1-score.

5.3 Experiments

In this section, we present the results of the conducted experiments based on the various approaches discussed in chapter 4. That includes experimenting with different deep learning classifiers, different word embeddings configurations, and fine-tuning parameters for each classifier. In order to clearly display the results, we divided the experiments based on their concerned taxonomy. In what follows, the results of the classifications of app reviews are displayed, namely by topic, sentiment, intent, and user perspective.

5.3.1 Experiments set 1: classifications reviews by intention classes

In this set of experiments, the proposed methodology is applied to annotated reviews, with considering only the labels with respect to the user intention. That means the four subcategories (informing, reporting, requesting, and irrelevant) of the intention are used as classes. The results of classifying the reviews based on intention classes are shown in table 5.2. Remarkably, the LSTM combined with the fasttext word embedding outperformed the other combinations of classifiers and word embeddings, with an 82.68% F_1-score. However, CNN showed acceptable results combined with the Arabic fasttext word embeddings.

Taxonomy	Intention					
Classifier	**LSTM**		**CNN**		**BLSTM**	
Model	Reviews	MSA	Reviews	MSA	Reviews	MSA
Word Vector	Word2Vec	fasttext	Word2Vec	fasttext	Word2Vec	fasttext
Recall	30.50%	**75.21%**	40.81%	55.07%	44.17%	76.93%
Precision	70.85%	**91.79%**	93.33%	**94.77%**	75.73%	84.23%
F1-score	42.65%	**82.68%**	57.55%	69.66%	55.79%	80.41%

Table 5.2: System performance results of intention classification

5.3.2 Experiments set 2: classifying reviews by sentiment classes

In this set of experiments, the sentimental labels, i.e. positive, negative, irrelevant, are considered as classes. The case of classifying reviews by sentiment classes achieved similar results as before, as shown in table 5.3. The combination of LSTM and the fasttext word vector achieved the best results, with 79.17% of F_1-score. Also notable in this case is that the Word2Vec word vector achieved low recall performance, in almost every case. The precision results when using Word2Vec achieved considerably higher rates than recall.

Taxonomy	Sentiment					
Classifier	**LSTM**		**CNN**		**BLSTM**	
Model	Reviews	MSA	Reviews	MSA	Reviews	MSA
Word Vector	Word2Vec	fasttext	Word2Vec	fasttext	Word2Vec	fasttext
Recall	46.08%	**76.00%**	48.67%	65.31%	41.44%	68.00%
Precision	63.22%	82.61%	**96.24%**	88.28%	55.33%	74.73%
F1-score	53.31%	**79.17%**	64.65%	75.07%	47.39%	71.20%

Table 5.3: System performance results of sentiment classification

5.3.3 Experiments set 3: classifying reviews by topic classes

In the third set of experiments, the labels according to topic classifications, i.e. product quality, product context, other-product related, and irrelevant, are considered as classes. Table 5.4 shows the results of classifying reviews based on their topic classes. Similar to the results of previous categories, the combination of LSTM and fasttext MSA word vector achieved the best results, in this case of all performance measures. It achieved 85.02% F_1-score, 87.14% precision, and 82.99% recall. The difference in performance in this category between classifiers is starker than in previous categories, and in this case, BLSTM achieves better results than CNN, with 78.09% F_1-score.

Taxonomy	**Topic**					
Classifier	**LSTM**		**CNN**		**BLSM**	
Model	**Reviews**	**MSA**	**Reviews**	**MSA**	**Reviews**	**MSA**
Word Vector	Word2Vec	fasttext	Word2Vec	fasttext	Word2Vec	fasttext
Recall	46.85%	**82.99%**	62.45%	71.55%	48.22%	74.62%
Precision	72.28%	**87.14%**	78.01%	84.50%	75.40%	81.89%
F1-Score	56.85%	**85.02%**	69.37%	77.49%	58.82%	78.09%

TABLE 5.4: System performance results of topic classification

5.3.4 Experiments set 4: classifying reviews by user perspective classes

The results of classifying reviews by user perspective sub-classes, i.e. quality in use, user-oriented perception, product-oriented perception, and irrelevant, are shown in table 5.5. The results reflect some agreement with previous results, but also some differences. The combination of BLSTM with fasttext word vector achieved the best performance in this case, with F_1-score of 69.83%. The combination of LSTM and fasttext word embedding achieved comparable results, with 66.03% F_1-score. However, using CNN with Word2Vec, achieved the maximum precision score of 89.05%. That reflects a trend in some of our experiments, of classifiers achieving high precision, albeit low recall results. This observation will be discussed further in the following section.

Taxonomy	**User Perspective**					
Classifier	**LSTM**		**CNN**		**BLSTM**	
Model	Reviews	MSA	Reviews	MSA	Reviews	MSA
Word Vector	Word2Vec	fasttext	Word2Vec	fasttext	Word2Vec	fasttext
Recall	51.11%	65.19%	49.68%	31.47%	33.70%	**65.19%**
Precision	60.04%	66.88%	**89.05%**	85.38%	54.48%	75.18%
F1-score	55.21%	66.03%	66.17%	47.33%	41.64%	**69.83%**

TABLE 5.5: System performance results of user perspective classification

5.4 Discussion (RQ2) & (RQ3)

In Section 5.1, the manual annotation process was discussed in detail. The output of which was fed to the corresponding experiments. In Section 5.3, we detailed the results of the experiments that were conducted. The result represented training and testing a classification based on multiple classifiers(CNN, LSTM, and BLSTM), and multiple feature extraction algorithms, or in this word embedding(fasttext and Word2Vec). The best results in each category, closely matched or exceeded the state-of-art systems of Arabic multi-class text classification. The model generated out of each experiment can be used further to build a production system, capable of classifying app reviews, and integrated into a mobile app development life-cycle.

Based on the taxonomy we followed, or in other words, based on the categories and classes we used, mobile app reviews get fed to the requirements engineering activities. The top activity to be assisted in this case is requirement elicitation. Reviews classified based on sentiment indicate if the review is positive or negative, or simply neither. In the case that the user review is reporting a usage scenario, the sentiment is to be

Intention	Topic	Sentiment	User Perspective	
Informing	Product quality	Positive	Quality in use	The review informs a use case of quality attribute
Requesting	Product context	Negative	User-oriented perception	The review describes a negative feeling about the app content or updates, requesting help
Irrelevant	Irrelevant	Positive	Irrelevant	The review is irrelevant for software engineering uses
Requesting	Product quality	Positive	Quality in use	The review is requesting a new feature or enhancement to existing features or quality attributes.
Reporting	Product quality	Negative	Quality in use	The review reports a bug in a feature or quality attribute, negatively affecting the user experience.

Table 5.6: Example of how to interpret reviews based on taxonomy values in the context of Requirements Engineering

expected as non-negative. In the case that the user review is reporting a bug or an issue, the sentiment is expected to be negative. However, if the review is requesting (intention is classified as requesting), and the sentiment is classified as positive, then this indicates that the user intends to request a new feature or an enhancement. Furthermore, The User perspective category indicates whether the user review is expressing an opinion about product quality, user satisfaction, or frustration, or whether the user is a non-quality attribute of the app. As for the topic category, the classes express either an opinion about a quality attribute, update, or general comments, or an opinion about non-quality issues, such as price or license. Irrelevant in all categories indicate that a review doesn't tell anything beneficial for software engineering for the respective category. However, this interpretation of taxonomy, as shown in Figure 5.6, isn't to be considered a limitation on other interpretations, based on the specific application of the solution.

The results of the experiments in section 5 show clearly that the choice of word embeddings, thus feature extraction, is a significant factor in the performance of the classification model. Generally speaking, the fasttext pre-trained dataset provided by the fasttext project by Facebook achieved higher recall, and as a result, a higher F_1score than the Word2Vec model generated in this study. The Result in many instances is a high-precision low-recall situation. The fasttext differs from Word2Vec by utilizing n-gram representations, as opposed to only word representations. The great difference in numbers, however, should not be only attributed to the difference in the algorithm. Because the Word2Vec word embedding was trained on a large dataset of app reviews, they contain very noisy data, with multi-dialect syntax, but also misspelling, non-normalized instances, and user errors. Figure 5.6 shows an example of the phenomenon. That is in stark difference with the fasttext pre-trained model, which is trained mostly on crawled data from the Arabic Wikipedia and other sites, mostly containing MSA language, with expectedly fewer writing errors, and thus more normalizable.

The stark difference in numbers between using various word vectors is less observable when using different deep learning classifiers. Even though LSTM, especially combined with fasttext, has achieved better F_1-score performance results in three out of the used categories, namely topic, intention, and sentiment, other classifiers were close enough. For instance, in the case of classifying by intention, using fasttext word vector, BLSTM achieved 80.41% F_1-score and LTSM achieved 82.68%, with BLSTM achieving better recall, but lower precision. BLSTM, combined with fasttext word vector, achieved a better F_1-score, with 69.83%, but LSTM was not far behind with 66.03%. There were in this case some instances of high precision-low recall situation, such as when using the combination of CNN and fasttext word vector, in classifying reviews by intention. The F_1-score of 69.66% is achieved in this case, but with the high precision of 94.77%. To improve the recall of this model, a larger annotated dataset has to be used for training.

In comparison with the literature, the performance of the best classification models achieved in this study is quite comparable to the performance of the use of the same classifiers in classifying highly noisy Arabic text, specifically social media content[45]. However, it is noticeable that the task for most of the literature on text classification was sentiment analysis. From among the best results, Nassif et al. [46] tested both CNN, LSTM, and other hybrid deep learning architectures on Arabic Tweets to achieve the task of sentiment analysis. The best F_1-score results for LSTM and CNN were 83.54% and 83.11% respectively. Another case is Alkhatib et al.[4], in which applying text classification on Arabic tweets using CNN achieved F_1-score of 82.6%.

5.5 Threats to Validity

The experiments and analysis presented in this chapter are subject to several types of threats of validity, namely threats to internal validity, external validity, and construct validity.

5.5.1 Threats to internal validity

There are several conditions and presumptions in our experiments that could lead to limiting the validity of our results. During the manual annotation phase, the research was dependent on experts to annotate the textual reviews with corresponding classes based on the taxonomy of each experiment. The annotators have a wide range of expertise, interest in the experiment, confidence in their choices, and are probably biased in their approach to annotation. To mitigate this threat, we added a user confidence parameter for each annotation, in order to measure the experts' confidence in the process and then neglect annotations with low confidence. Furthermore, a sample of 100 reviews was cross-annotated by multiple experts, in order to make sure that the discrepancy between experts is minimum, and the individual bias is not a significant factor. The experiment was also preceded by a group training session, to make sure the experts understood the process and the taxonomy provided. Last but not least, the experts took the time they needed to complete the task, to mitigate the effects of boredom and apathy in the process.

5.5.2 Threats to external validity

The ability to generalize the research outcome, given the utilized dataset and the experimental settings, is limited by many factors. The dataset collected and annotated was crawled from the Google app store. Taking into consideration that there are other mobile stores, we tried to mitigate this effect by collecting reviews from several thousand apps and games, under a wide range of categories, to ensure the results could be generalizable. Furthermore, the sample for training and testing was selected to reflect this diversity and to minimize data bias.

5.5.3 Threats to construct validity

Construct validity is concerned with how valid the performance measures of the experiments are. To mitigate such threats, we used the widely used performance measures in the research (precision, recall, and F_1-score, and which are less susceptible to data or user bias.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Crowd-based requirements engineering presents many opportunities for app developers to take advantage of the wealth of data provided by users, for use in software engineering, and specifically in requirements engineering. In this thesis, we proposed a framework to classify app reviews in Arabic, by closely exploring the vast literature on the topic, and conducting the data collection and preprocessing phases, in order to complete the needed experimentation, testing, and evaluation needed.

We have gathered 13.5 million raw Arabic app reviews for the purpose of constructing the classification model. Five hundred out of them have been sampled randomly. Additionally, with the help of six experts in various fields related to software engineering, the reviews were annotated according to the taxonomy of our framework. After that, the reviews were preprocessed and prepared for the experiment by extracting the features using Word2Vec and fastText techniques.

Various configurations of deep neural networks, namely, CNN, LSTM, and BLSTM, were used to classify the app reviews into the considered main and sub-categories of the software requirement from the Arabic reviews. The sentimental analysis results show that the LSTM classifier with the fasttext word embeddings gives the best F1-score, 79.17%. However, the BLSTM classifier with the fastText embeddings outperforms the other classifiers, with F1-score of 69.83%, when used for identifying the sub-categories of the user perspective main category.

The F1-score of classifying reviews by intention and topics categories with the LSTM and using fastText embeddings, is 82.68% and 85,02%, respectively. These results outperform the other configurations of the classifiers and word embeddings.

6.2 Future work

There are many directions for extending this study. For example, different types of classes related to software requirements can be investigated. The word embedding models can be re-trained or tuned on the training data and then compared with the pre-trained models. More investigation of the neural networks configurations and settings can be studied in future work. Moreover, the effect of the data size on the system performance can be also investigated. Combining different classifiers with different settings together to identify the software requirements is also interesting for future work.

Bibliography

[1] Charu C Aggarwal et al. *Neural networks and deep learning*. Springer, 2018.

[2] Nadeem Al Kilani, Rami Tailakh, and Abualsoud Hanani. "Automatic Classification of Apps Reviews for Requirement Engineering: Exploring the Customers Need from Healthcare Applications". In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE. 2019, pp. 541–548.

[3] Marwan Al Omari et al. "Hybrid CNNs-LSTM Deep Analyzer for Arabic Opinion Mining". In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE. 2019, pp. 364–368.

[4] Manar AlKhatib et al. "A sentiment reporting framework for major city events: Case study on the China-United States trade war". In: *Journal of Cleaner Production* 264 (Aug. 2020), p. 121426. ISSN: 09596526. DOI: `10.1016/j.jclepro.2020.121426`. URL: `https://linkinghub.elsevier.com/retrieve/pii/S0959652620314736`.

[5] Nadeem AlKilani. "Automatic Classification of Apps Reviews for Requirement Engineering (Exploring The Customer's Need from The Healthcare Applications)". MA thesis. 2019.

[6] "Arabic Sentiment Analysis: A Systematic Literature Review". In: *Applied Computational Intelligence and Soft Computing* 2020 (2020). ISSN: 16879732.

[7] Gilbert Badaro et al. "A large scale Arabic sentiment lexicon for Arabic opinion mining". In: *Proceedings of the EMNLP 2014 workshop on arabic natural language processing (ANLP)*. 2014, pp. 165–173.

[8] Anurag Bhardwaj, Wei Di, and Jianing Wei. *Deep Learning Essentials: Your hands-on guide to the fundamentals of deep learning and neural network modeling*. Packt Publishing Ltd, 2018.

[9] Piotr Bojanowski and Edouard Grave. *Fasttext*. URL: https://research.facebook.com/blog/2016/08/fasttext/ (visited on 11/30/2021).

[10] Hong Cao and Miao Lin. "Mining smartphone data for app usage prediction and recommendations: A survey". In: *Pervasive and Mobile Computing* 37 (2017), pp. 1–22.

[11] Asma Chader, Leila Hamdad, and Abdesselam Belkhiri. "Sentiment Analysis in Google Play Store: Algerian Reviews Case". In: *Modelling and Implementation of Complex Systems*. 2021, pp. 107–121. DOI: 10.1007/978-3-030-58861-8_8. URL: http://link.springer.com/10.1007/978-3-030-58861-8%7B%5C_%7D8.

[12] Ning Chen et al. "AR-miner: Mining informative reviews for developers from mobile app marketplace". In: *Proceedings - International Conference on Software Engineering*. 2014.

[13] Adelina Ciurumelea et al. "Analyzing reviews and code of mobile apps for better release planning". In: *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2017, pp. 91–102.

[14] J. Clement. *Number of apps available in leading app stores as of 1st quarter 2020*. 2020. URL: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores (visited on 08/20/2020).

[15] Abdelghani Dahou et al. “Arabic Sentiment Classification Using Convolutional Neural Network and Differential Evolution Algorithm”. In: *Computational Intelligence and Neuroscience* 2019 (2019). ISSN: 16875273. DOI: 10.1155/2019/2537689.

[16] Jeremy Dick, Elizabeth Hull, and Ken Jackson. *Requirements Engineering*. Vol. 42. 4. Cham: Springer International Publishing, 2017, p. 9. URL: http://link.springer.com/10.1007/978-3-319-61073-3.

[17] Jeremy Dick, Elizabeth Hull, and Ken Jackson. *Requirements engineering*. Springer, 2017.

[18] Rehab M. Duwairi. “Sentiment analysis for dialectical Arabic”. In: *2015 6th International Conference on Information and Communication Systems, ICICS 2015*. 2015.

[19] Ashraf Elnagar, Ridhwan Al-Debsi, and Omar Einea. “Arabic text classification using deep learning models”. In: *Information Processing and Management* 57.1 (2020), p. 102121. ISSN: 03064573. URL: https://doi.org/10.1016/j.ipm.2019.102121.

[20] Rita Francese et al. “Mobile app development and management: results from a qualitative investigation”. In: *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE. 2017, pp. 133–143.

[21] Necmiye Genc-Nayebi and Alain Abran. “A systematic literature review: Opinion mining studies from mobile app store user reviews”. In: *Journal of Systems and Software* 125 (2017), pp. 207–219.

[22] Edouard Grave et al. “Learning Word Vectors for 157 Languages”. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[23] Eduard C Groen, Joerg Doerr, and Sebastian Adam. "Towards crowd-based requirements engineering a research preview". In: *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer. 2015, pp. 247–253.

[24] X. Gu and S. Kim. """What Parts of Your Apps are Loved by Users?" (T)". In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2015, pp. 760–770.

[25] Emitza Guzman et al. "User feedback in the app store". In: *Proceedings of the 40th International Conference on Software Engineering Software Engineering in Society - ICSE-SEIS '18*. August. ACM Press, 2018, pp. 13–22.

[26] Mark Harman, Yue Jia, and Yuanyuan Zhang. "App store mining and analysis: MSR for app stores". In: *IEEE International Working Conference on Mining Software Repositories* (2012), pp. 108–111.

[27] Alaa Isaac. *Arabic App Reviews Word2Vec Word Embeddings*. URL: https://www.kaggle.com/aiaarisaac/arabic-app-reviews-w2v.

[28] Auliya Rahman Isnain, Agus Sihabuddin, and Yohanes Suyanto. "Bidirectional Long Short Term Memory Method and Word2vec Extraction Approach for Hate Speech Detection". In: *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)* 14.2 (2020), pp. 169–178.

[29] Ajay Kumar Jha, Sunghee Lee, and Woo Jin Lee. "An empirical study of configuration changes and adoption in Android apps". In: *Journal of Systems and Software* 156.July (2019), pp. 164–180.

[30] Nishant Jha and Anas Mahmoud. *Mining non-functional requirements from App store reviews*. Vol. 24. 6. Empirical Software Engineering, 2019, pp. 3659–3695.

[31] John D Kelleher. *Deep learning*. Mit Press, 2019.

[32] Keras. *Keras preprocessing*. URL: https://keras.io/api/preprocessing/ (visited on 11/30/2021).

[33] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *CoRR* abs/1408.5882 (2014). arXiv: 1408.5882. URL: http://arxiv.org/abs/1408.5882.

[34] Zijad Kurtanovic and Walid Maalej. "Mining User Rationale from Software Reviews". In: *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017* (2017), pp. 61–70.

[35] Sachiko Lim, Aron Henriksson, and Jelena Zdravkovic. "Data-Driven Requirements Elicitation: A Systematic Literature Review". In: *SN Computer Science* 2.1 (2021), pp. 1–35.

[36] Mengmeng Lu and Peng Liang. "Automatic classification of non-functional requirements from augmented app user reviews". In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. 2017, pp. 344–353.

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

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

[39] Walid Maalej and Hadeer Nabil. "Bug report, feature request, or simply praise? On automatically classifying app reviews". In: *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. IEEE, 2015, pp. 116–125.

[40] Mohcine Maghfour and Abdeljalil Elouardighi. "Standard and dialectal Arabic text classification for sentiment analysis". In: *International Conference on Model and Data Engineering*. Springer. 2018, pp. 282–291.

[41] Fawaz HH Mahyoub, Muazzam A Siddiqui, and Mohamed Y Dahab. “Building an Arabic sentiment lexicon using semi-supervised learning”. In: *Journal of King Saud University-Computer and Information Sciences* 26.4 (2014), pp. 417–424.

[42] William Martin et al. “A survey of app store analysis for software engineering”. In: *IEEE Transactions on Software Engineering* 43.9 (2017), pp. 817–847.

[43] Stuart McIlroy, Nasir Ali, and Ahmed E. Hassan. “Fresh apps: an empirical study of frequently-updated mobile apps in the Google play store”. In: *Empirical Software Engineering* 21.3 (2016), pp. 1346–1370.

[44] Qutaiba Mustafa. “Detecting and classifying Software Bugs and Requirements in Arabic Mobile App Reviews”. MA thesis. Birzeit University, 2021.

[45] Ali Bou Nassif et al. “Deep learning for Arabic subjective sentiment analysis: Challenges and research opportunities”. In: *Applied Soft Computing* (2020), p. 106836.

[46] Ali Bou Nassif et al. “Deep learning for Arabic subjective sentiment analysis: Challenges and research opportunities”. In: *Applied Soft Computing* 98.December (2021). ISSN: 15684946.

[47] Maleknaz Nayebi, Bram Adams, and Guenther Ruhe. “Release Practices for Mobile Apps–What do Users and Developers Think?” In: *2016 ieee 23rd international conference on software analysis, evolution, and reengineering (saner)*. Vol. 1. IEEE. 2016, pp. 552–562.

[48] Nan Niu et al. “Requirements engineering and continuous deployment”. In: *IEEE software* 35.2 (2018), pp. 86–90.

[49] Facundo Olano. *google-play-scraper*. 2019. URL: https://github.com/facundoolano/google-play-scraper (visited on 07/25/2021).

[50] Oumaima Oueslati et al. “A review of sentiment analysis research in Arabic language”. In: *Future Generation Computer Systems* (2020).

[51] Sebastiano Panichella et al. “ARdoc: App reviews development oriented classifier”. In: *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering* 13-18-Nove (2016), pp. 1023–1027.

[52] Sebastiano Panichella et al. “How can i improve my app? Classifying user reviews for software maintenance and evolution”. In: *2015 IEEE 31st International Conference on Software Maintenance and Evolution, ICSME 2015 - Proceedings* September (2015), pp. 281–290.

[53] Claude Sammut and Geoffrey I Webb. *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[54] Rubens Santos, Eduard C Groen, and Karina Villela. “A Taxonomy for User Feedback Classifications.” In: *REFSQ Workshops*. 2019.

[55] Rabab E. Saudy et al. “Use of Arabic Sentiment Analysis for Mobile Applications’ Requirements Evolution: Trends and Challenges”. In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*. Vol. 639. 1. ACM Press, 2018, pp. 477–487.

[56] M. Sawalha, E. Atwell, and M. A. M. Abushariah. “SALMA: Standard Arabic Language Morphological Analysis”. In: *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. 2013, pp. 1–6.

[57] Faiz Ali Shah, Kairit Sirts, and Dietmar Pfahl. “Simplifying the classification of app reviews using only lexical features”. In: *International Conference on Software Technologies*. Springer. 2018, pp. 173–193.

[58] Tensorflow. *Tensorflow*. URL: https://www.tensorflow.org/ (visited on 11/25/2021).

[59] Kees Versteegh. *Arabic language*. Edinburgh University Press, 2014.

[60] Chong Wang et al. “Augmenting App Review with App Changelogs: An Approach for App Review Classification.” In: *SEKE*. 2019, pp. 398–512.

[61] Anthony I. Wasserman. “Software engineering issues for mobile application development”. In: *Proceedings of the FSE/SDP Workshop on the Future of Software Engineering Research, FoSER 2010* (2010), pp. 397–400.

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

[63] Mohammad Ali Yaghan. ““Arabizi”: A contemporary style of Arabic Slang”. In: *Design issues* 24.2 (2008), pp. 39–52.

[64] Jun Zhao, Kang Liu, and Liheng Xu. *Sentiment analysis: mining opinions, sentiments, and emotions*. 2016.