Automated Extraction of Non-Functional Requirements (NFRs) From App Reviews to Facilitate Requirements Elicitation Process

by

Bisma Rehman(18F-0818)

Masters of Science in Computer Science

A thesis submitted
in partial fulfillment of the
requirements for the degree of
Masters of Science in
Computer Science at the
National University of Computer and Emerging Sciences

Department of Computer Science

National University of Computer and Emerging Sciences

Chiniot-Faisalabad, Pakistan, 2020

Plagiarism Undertaking

I take full responsibility of the research work conducted during the Masters Thesis titled Automated Extraction of Non-Functional Requirements (NFRs) From App Reviews to Facilitate Requirements Elicitation Process. I solemnly declare that the research work presented in the thesis is done solely by me with no significant help from any other person; however, small help wherever taken is duly acknowledged. I have also written the complete thesis by myself. Moreover, I have not presented this thesis (or substantially similar research work) or any part of the thesis previously to any other degree awarding institution within Pakistan or abroad.

I understand that the management of National University of Computer and Emerging Sciences has a zero tolerance policy towards plagiarism. Therefore, I as an author of the above-mentioned thesis, solemnly declare that no portion of my thesis has been plagiarized and any material used in the thesis from other sources is properly referenced. Furthermore, the work presented in the thesis is my own original work and I have positively cited the related work of the other researchers by clearly differentiating my work from their relevant work.

I further understand that if I am found guilty of any form of plagiarism in my thesis work even after my graduation, the University reserves the right to revoke my Masters degree. Moreover, the University will also have the right to publish my name on its website that keeps a record of the students who plagiarized in their thesis work.

____

Bisma Rehman

Date: __________________

Declaration of Authorship

I, Bisma Rehman, hereby state that my Masters thesis titled Automated Extraction of Non-Functional Requirements (NFRs) From App Reviews to Facilitate Requirements Elicitation Processis my own work and it has not been previously submitted by me for taking partial or full credit for the award of any degree at this University or anywhere else in the world. If my statement is found to be incorrect, at any time even after my graduation, the University has the right to revoke my Masters degree.

____

Bisma Rehman

Date: ____________________

“*Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.*”

Dave Barry

Abstract

With the recent paradigm shift towards mobile applications, user reviews are considered one of the most important source of information about an app. Anlaysis and classification of the reviews to extract information has emerged as significant challenge. The competitive mobile app market result in functionally similar applications over the app distribution platforms. Mobile applications receive user feedback in the form of reviews that can be helpful in identifying apps with better functionality and also help developers to extract useful information pertaining to user requirements, bugs, desired extended features and description of user experience associated with existing features. App review analysis has recently emerged as an active area of research in software engineering considering the immensely large user base and potential benefits of automated feature and bug extraction. Recently, several research studies have been conducted to mine and categorize user-reviews into actionable software maintenance requests, including feature requests and bug reports. However, existing literature have mostly focused on mining functional aspects and analysis of user reviews for extracting and synthesizing Non-functional requirements NFRs is largely neglected. To achieve user satisfaction and to survive in the app market, addressing these requirements is essential. Therefore, in this research we formulate this problem as a Multi label classification problem and propose a classification model. Representative features are used to train a model for classification. Use of feature extraction methods to train a classification model may lead to divergent results, which implies the need for a careful selection of these methods. Several recent studies have emphasized that basic state-of-the-art feature extraction methods are useful for significant results. Therefore, this research employs these methods to evaluate the performance of most widely used machine learning algorithms for multi label review classification. In addition, we have used word2vec embedding for proposed MLRC-CNN model using ten thousands reviews extracted from fifty Android apps. This is the first study of its kind in mobile app review classification that classify reviews into multiple categories of NFRs using CNN classifier. Considering the promising results and significance of this model, it can be exploited as a classification model for other user feedback analysis platforms.

Acknowledgements

First and foremost, I have to thank Almighty Allah for countless blessings. Then I have to thank my parents for their love and support throughout my life. Thank you both for giving me strength to reach for the stars and chase my dreams. My brothers, and my entire family deserve my wholehearted thanks as well. I would like to sincerely thank my supervisor, Dr. Khubaib Amjad Alam, for his guidance and support throughout this research, and especially for his confidence in me. I would also like to thank all friends who stood by me through the ups and downs of life. . . .

Contents

Declaration of Authorship **i**

Abstract **iii**

Acknowledgements **v**

1 Introduction **1**

1.1 Challenges Pertaining to App Reviews Analysis 3

1.2 Problem Formulation 5

1.2.1 App Reviews Classification 6

1.2.2 Multi-Label Reviews Analysis 7

1.3 Research Objectives 7

1.4 Research Questions 8

1.5 Research Approach 9

1.6 Thesis Structure 10

1.7 Summary 11

2 Literature Review **12**

2.1 Requirement Engineering Process 12

2.1.1 Requirement Elicitation Process 14

2.1.2 Requirement Elicitation for Mobile App Development . . . 16

2.1.3 Software Quality Measures 17

2.2 Automatic Text Analysis 18

2.2.1 Machine Learning for Reviews Classification 20
2.3 App Reviews Analysis 26
2.3.1 Reviews Classification 26
2.3.2 Multi-label Reviews Analysis 28
2.4 Model Evaluation 30
2.5 Summary 33

3 Mobile App Reviews Analysis and Classification 34
3.1 Systematic Literature Review Protocol 35
3.1.1 Research Questions 36
3.1.2 Search Terms 36
3.1.3 Studies Selection 37
3.2 Data Extraction 37
3.3 Results and Discussion 37
3.3.1 Type of Information mined from Feedback 37
3.3.2 Machine Learning Approaches for Mobile App Reviews . . . 40
3.3.3 Text Pre-processing Approaches 42
3.3.4 Application Features and Reviews Features Extraction . . 43
3.3.5 Evaluation Metrics 45
3.3.6 Conclusion 47
3.4 Research Gap Analysis 48
3.5 Summary 56

4 Research Methodology 57
4.1 Research Methodology and Design 58
4.2 Design Science Research 59
4.2.1 Identification of Problem and Motivation 59
Systematic Literature Review Process 60
4.2.2 Defining Objectives of a Solution 61
4.2.3 Design and Development 62

4.2.4 Demonstration and Evaluation 63
4.3 Data Collection Process 63
4.4 Data Analysis Techniques and Methods 64
4.4.1 Finding a Suitable MARC Method 68
Number of Alternatives and Motivation 68
Algorithms for Comparative Study 68
4.4.2 Justification for Text Pre-processing Methods 69
4.4.3 Justification for Feature Extraction Methods 69
4.4.4 Justification for Model Selection 70
4.5 Mapping Research Question, Research Methods and Outcomes . . 70
4.6 Summary 71

5 Analyzing Android Health App Reviews to Mine Non-Functional Requirements 72

5.1 Introduction 73
5.2 Research Approach 73
5.2.1 Data Collection 74
Selection of apps and app category 74
Selection of user reviews 74
5.3 Manual Classification 75
5.3.1 Results and Discussion 75
5.4 Automated Classification 77
5.4.1 Multi-label Classification 77
5.4.2 Text Pre-processing 79
Feature Extraction 79
Bag-of-Words 79
Term Frequency - Inverse Document Frequency 81
5.4.3 Classifiers 81
5.4.4 Evaluation strategies 83

5.4.5 Classification Setup 83

5.4.6 Results and Discussions 84

5.5 Threats to Validity 87

5.6 Conclusion 88

5.7 Summary 89

6 Proposed Classification Model 90

6.1 Architecture of the Proposed Model 90

6.2 Preliminary Concepts 92

6.2.1 Word Embedding 92

6.2.2 Convolutional Neural Network 92

Neural Network 92

Convolution 93

Pooling 93

Activation Function 93

6.2.3 Classification Process 93

6.3 MLRC-CNN Analysis and Design 94

6.3.1 Evaluation Metrics 97

6.4 Summary 97

7 Implementation and Evaluation 99

7.1 Results of Dataset Analysis 99

7.2 Results of the MLRC-CNN classifier 106

7.3 Summary 108

8 Conclusion 109

8.1 Achieving Research Objectives 110

8.2 Contribution and Significance 112

8.3 Limitations and Future Work 113

A SLR Protocol 135

A.1 Electronic Databases 135

A.2 Searched Strings over each database 135

A.3 Inclusion and Exclusion Criteria 136

A.4 Quality Assessment Criteria 136

A.5 Prisma Flow 137

List of Figures

1.1 Research Methodology 9

2.1 Organization of Chapter 2 13
2.2 Requirement Elicitation Process 14
2.3 Elicitation 15
2.4 Overview of Mobile App Development Release Cycle 19
2.5 Presence of Requirements in a Single Review 19
2.6 Encoding Example 20
2.7 Machine Learning Approaches 21
2.8 Working of SVM[69] 23

3.1 SLR Taxonomy 35

4.1 Peffers design-science research methodology 60
4.2 Systematic process of literature review 61
4.3 Selected Quality Attributes of ISO-25010 Model 65
4.4 Choice of SLRC or MLRC 69

5.1 Reviews distribution over the requirement categories 76
5.2 Category wise Reviews Distribution over Applications 76
5.3 Illustration of the Multi-label One-vs-Rest Classifier 78
5.4 Bag-of-Words Representation 80
5.5 Automated Machine Learning Analysis Results 86

6.1 Architecture of the Proposed Approach 91
6.2 Generic Process Flow of Reviews Classification Approach 94
6.3 MLRC-CNN Architecture 95
6.4 Activity flow of MLRC-CNN 96

7.1 Shopping App Reviews Distribution 101
7.2 communication App Reviews Distribution 102
7.3 Education App Reviews Distribution 103
7.4 Health App Reviews Distribution 104
7.5 Entertainment App Reviews Distribution 105
7.6 All Five Categories App Reviews Distribution 106

A.1 Prisma Flow 138

List of Tables

3.1 SLR Research Questions 36
3.2 Focus of Data extraction form 38
3.3 Description of the studies included in analysis 39
3.4 Summary of Studies Analysed in SLR 50

4.1 List of Selected Categories and Applications 64
4.2 Mapping of objectives, methods and Outcomes 71

5.1 Applications selected for comparative study 74
5.2 Classification performance of the models based on unigram features for all Requirement types 85

7.1 Dataset Description 100
7.2 Initial Results of MLRC-CNN 107
7.3 MLRC-CNN Results over the Dataset of Ten Thousands Reviews . 108

List of Abbreviations

ARAC	App Review Analysis and Classification
BoW	Bag of Words
CNN	Convolutional Neural Network
DL	Deep Learning
DT	Decision tree
iOS	iPhone Operating System
LR	Logistic Regression
MAD	Mobile Application Development
MARC	Mobile App Review Classification
ML	Machine Learning
MLRC	Multi Label Review Classification
NB	Naive Bayes
NLP	Natural Language Processing
OvA	One vs All
OvR	One vs Rest

POS	Parts Of Speech
RE	Requirement Engineering
ReLU	Rectified Linear Unit
RF	Random Forest
RL	Reinforcement Learning
RO	Research Objective
RQ	Research Question
SD	Software Development
SDLC	Software Development Life Cycle
SLR	Systematic Literature Review
SML	Supervised Machine Learning
TF-IDF	Term Frequency-Inverse Document Frequency
SVM	Support Vector Machine
SPM	Standard Performance Measures

Chapter 1

Introduction

Smartphones existed before the app store release, but not until 2008 when users could genuinely take advantage of their extra computing power and ensuring versatility by allowing app download. Customized and even commercial off-the-shelf apps had been available prior to the release of app stores; however, these stores had some variations, including variety, availability, user-submitted-content material, ease-of-use and compatibility. Therefore, these tempting variations and worldwide accessibility of smart devices such as mobile phones and tablets have led to a rise in need for applications to provide support for these devices [1, 2]. To fulfil growing market demands, application distribution platforms such as Google-Play store, Apple-App store, and Windows-Phone store have appeared as a new on-line platforms. The popularity of these on-line platforms is rapidly growing, since March 2020, the number of Google-Play store apps with 4.5/5 stars or higher rating on the average, were approximately 4.6 million [3]. The app distribution platforms have focused on reducing the limitations for small scale business and commercial app developers by adopting open business approach. Therefore, lowering the entry barriers has raised competition in mobile app market.

Survival in market competition depends on customer satisfaction so that decisions of app release should base on the complete knowledge of the user's preferences and demands . Considering customer preferences minimize not only the failure risk but also helps in app market survival [4, 5, 6]. App Stores allow users to download, use and express their opinion about the app, which is in form of a textual review and star rating. Reviews contain technically useful information that can be further utilized to understand the user's requirements and needs [7, 8]. Recent analysis has revealed that reviews are helpful for developers to resolve issues and identify requirements to meet the end user's expectations [9, 10, 11, 12, 13]. The manual analysis is impractical for a large number of reviews, received every day. Therefore, research on app-review-mining has followed the Text Mining approaches to distil valuable information from the reviews. One of the generally used Text Mining approaches for locating valuable information in user reviews, is the training of an ML-based classification model, aimed at automatically assigning categories to the textual reviews according to their content [1]. For example, to classify reviews into the categories such as feature request, feature evaluation, bug report and others, a classification model is trained that help developers obtaining useful information from reviews [14, 15, 16]. However, some researchers attempted to extract Non-Functional Requirements (NFRs), and sentiments conveyed towards them in user feedback, to understand user's opinion [17, 18, 19, 20, 21, 22]. It is crucial in terms of accuracy of the method used to automatically extract app quality information when analysing users' opinion about the app quality. Automatic extraction of NFRs is a challenging problem due to the variability of users' expression about quality aspects of an app in user reviews. Up to now, various approaches have been used for automatic extraction of NFRs from user feedback. However, they differ in terms of datasets or evaluation strategies, which makes the accuracy comparison of these methods impractical.

Moreover, few studies have paid attention to mine NFRs from mobile app reviews[17, 18, 19, 23]. NFRs define a set of quality characteristics that a software system should exhibit. NFRs express various quality attributes such as usability, reliability, compatibility, performance, and security. These quality attributes enforce various constraints over the development process, and addressing them is essential as they help in attaining market viability along with customer satisfaction [24].

In contrast with the previous discussion, little attention to research efforts on NFRs mining from mobile app user reviews can result in vague understanding of NFRs. Unlike, explicitly stated technical attributes such as functional requirements, quality attributes such as NFRs defined implicitly and latent within the text [25]. Therefore, this challenge draws attention to a more comprehensive analysis of reviews to identify issues and requirements, considering the potential meaning that a review text signifies. This thesis aimed at automatic extraction of NFRs from mobile app reviews, training of classification models to classify user reviews under multiple-labels, facilitating requirement elicitation process.

1.1 Challenges Pertaining to App Reviews Analysis

In comparison to the process of Software Development, the process of mobile application development is different, because of unstable user requirements and limited capabilities of mobile devices. In addition to the device limitations, the rapid mobile application development makes the overall lifecycle faster, than the other desktop application software. Software development starts with Requirement Engineering (RE) that influence the smooth execution of a software development process and of absolute importance [26]. Efficient collection of requirements not only smooth the development process but also derive to efficient and cost effective systems [27]. Therefore, it is worth knowing what a term requirement actually means and what it is. IEEE defines requirement as "a condition or capability needed by a user to solve a problem or achieve an objective" [28]. Thus, satisfying user needs and meeting user objectives is a quality measure of a software product, to ensure that a software product will meet those quality measures, any Software Development process has an RE practice. Detail of the activities that are part of this RE process are discussed in chapter 2. Therefore, the very first activity of any RE process is Requirement Elicitation, process of getting information directly from the users who will use the system. Generally, Requirement Elicitation process is categorised into requirements gathering, classification, prioritization, and specification. Rigorously following elicitation process support later stages of software development [29]. However, MAD has some limitations that restrict the adoption of traditional SDLC. Short term development and various device specifications including processor power, screen size, memory size, touch sensitivity and cross device compatibility pose more challenges for mobile app developers [30].

Mobile apps are globally available, which provoke that these challenges are difficult to quantify during testing. That emanate the need of Crowd-Sourced RE, a practice in which large-number of users involved in the activities of RE [31]. These users are anonymous volunteers, huge in numbers and their involvement in activities can take various forms. For instance, information that is generated by users, freely and openly accessible for requirement specialists to utilise it for requirement elicitation. Conversely, this information may aid in distributed problem-solving, finding workarounds in apps, which may form the requirements for a successive release of an app. Massive number of users involvement gives an opportunity to Software Development (SD) organizations to get more adequate requirements, helps attaining just-in-time RE, which in turn entails a significant potential to minimize the RE process cost. Crowd sourcing in RE have a promising structure, employed through businesses and draws much research interest, is the utilization of feedback received through a massive a number of unknown users of the software systems within a specified time frame in an RE activity, which includes elicitation, validation, or prioritization. Up to now, previous research indicates that explicit and implicit user response is a primary source for RE specialists to identify new as well as changed requirements and to further choose what functionalities to improve, add or abandon [10].

Therefore, with the recent paradigm shift towards mobile applications, users report different issues and suggestions on an App Distribution Platforms such as Google-Play store and Apple-App store, in the form of User Reviews. The developers have to identify the reported problems and suggestions by analysing and classifying the large corpus of reviews. Generally, reviews are classified under the categories of Feature Request, Bug Report and sentiments such as positive or negative review category [14, 16, 32]. These categories are valuable when the concern is to improve the functionality of an app but when the concern is quality and user satisfaction then categorization of reviews into NFRs is essential and challenging.

1.2 Problem Formulation

User reviews mining and analysis have emerged as significant challenges for application developers and app users/vendors as the recent paradigm shifted towards mobile applications. The competitive mobile app market have millions of apps available over the mobile application stores and these apps have billions of user reviews. This excess amount of user reviews cause difficulty in analyses and interpretation of the reviews manually for information extraction. When extracting information from large number of reviews automatically, training of classification models is performed using lexical features [33]. However, the significance of these models for multi-label text classification is not clear. This thesis address this issue and evaluates the performance of classification models. The approaches used for NFRs extraction from app reviews have applied on different datasets or evaluation strategies, so that their results are not comparable. To deal with this issue, first in this thesis, we evaluate the performance of basic feature extraction methods on labelled reviews dataset and the performance of supervised ML methods is investigated. To perform review classification, previous studies [34, 35, 36, 37, 38] implies more effort to extract relevant features indicated in user reviews of an app.

However, this strategy is at risk of extracting many irrelevant features because reviews are in text format and contains extra information, which is insignificant to acquire as app features. To address this challenge, this thesis perform review classification without explicitly using any feature extraction methods by training a convolutional neural network model. The problem context is identified in the following areas: App reviews classification, and Multi-label review classification, approaches are explained in details.

1.2.1 App Reviews Classification

Classification models for app review analysis, automatically assign classes to reviews, keeping in view the contents of the review text [1]. To train and evaluate these models, data of labelled reviews is required, where the manual classification of the reviews into different categories have done by the humans. In addition to the quality of labelled data for training a model, the classification accuracy depends on the set of features identified from review text [33].

In previous studies, review classification models have been developed using variable complexity level for textual features, from basic to advanced linguistic features [33, 39]. For example, extraction of lexical features, additionally known as BoW is simple and straightforward approach, involves frequency count of each word. However, to extract importance of these features in a document requires counting of every word, this approach known as TF-IDF. Training of classification models that consider lexical features is comparatively fast and their language adaptation is easy.

This thesis evaluates the performance of ML based classification models with variable degree of complexity of feature extraction. In case that ML based classification model with simple BoW could perform better, compared with ML based TF-IDF classification model than is hard to justify the use of more complex lexical feature extraction techniques to classify review sentences.

1.2.2 Multi-Label Reviews Analysis

Most of the prior studies performed classification of app reviews as a single or multi class [37, 40, 39, 41, 42]. Whereas, most of the users point to more than one issues in a single review, identifying and classifying multiple categories of single review, considering a complete review is a multi-label problem. To address this challenge, we performed multi-label classification using problem transformation approach for ML based models.

In addition to multi-label categorization problem of reviews, the feature engineering for review classification can cause false effort of feature extraction, considering unrelated information present in the user reviews. To address this problem, a new domain of ML models has been introduced in recent years, which automatically learn features from labelled data, known as Deep Learning [39]. Specifically, Convolutional Neural Network (CNN), a model of DL, has performed effectively for numerous text classification problems. However, CNN performance for multi-label app reviews dataset has not yet known. Therefore, this thesis perform multi-label classification of mobile app reviews using CNN model. First, we labelled the dataset according to the defined ISO25010 model into multiple categories of NFRs and a Functional Requirement, then design a CNN model to get information from reviews. Moreover, our approach aid the developers to identify the requirements from mobile app user reviews, facilitating requirement elicitation process.

1.3 Research Objectives

This research mainly focuses on the Multi-Label Review Classification problem in the context of mobile apps. To resolve this problem, following objectives are intended to be achieved:

- RO1: To gain insight in the state-of-the-art classification methods that need to be addressed in the context of app review analysis.

- RO2: To evaluate the performance of successfully employed ML models for multi-label review classification.
- RO3: To automatically categorise reviews into multiple labels using CNN model

1.4 Research Questions

For the problems identified in the previous section, we have formulated the following research questions (RQ) to guide our research.

- Research Objective 1: To gain insight in the state-of-the-art classification models that need to be addressed in the context of app review analysis.
  - RQ1: what are the most investigated classification approaches for app reviews analysis?
    - RQ1.1: to identify most widely used text pre-processing techniques for review classification?
    - RQ1.2: to identify the existing feature extraction methods in review classification?
    - RQ1.3: to identify the most investigated algorithms to train a review classification model?
- Research Objective2: To evaluate the performance of successfully employed ML models for multi-label review classification.
  - RQ2: How do feature extraction methods influence the performance of ML classifiers for app reviews?
    - RQ2.1: To evaluate the performance of simple Bag-of-Words and TF-IDF approach on multi-label review dataset?

- Research Objective3: To automatically categorise reviews into multiple labels using CNN algorithm.
  - RQ3: How to automatically categorise reviews into multiple categories of NFRs using CNN?
    - RQ3.1: to validate the model on mobile app reviews dataset?

1.5 Research Approach

The whole research process is divided into four distinct phases and a dedicated chapter (chapter 4) provides the detailed research methodology and steps involved in the individual phases. Here a brief overview of the four constituent phases according to the design science approach defined in [43] is provided. Figure 1.1 depicts these constituent phases. The first phase comprises of the literature review

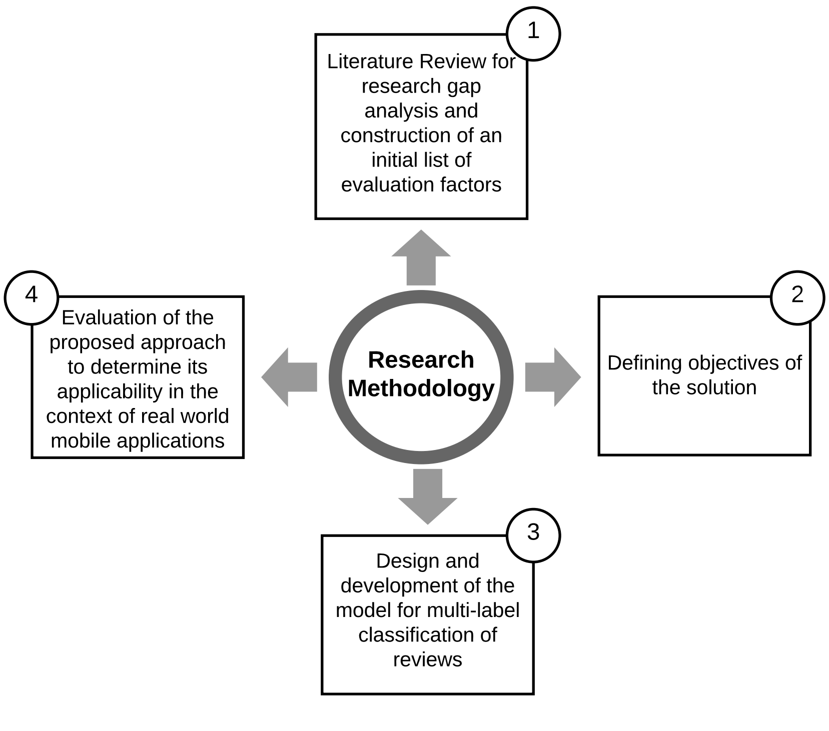

1
Literature Review for research gap analysis and construction of an initial list of evaluation factors
2
Defining objectives of the solution
3
Design and development of the model for multi-label classification of reviews
4
Evaluation of the proposed approach to determine its applicability in the context of real world mobile applications
Research Methodology

Figure 1.1: Research Methodology

and preliminary research processes. A literature review is conducted from multiple perspectives. This includes a general overview and discussion on multiple App Review Analysis and Classification (ARAC) methods. Subsequently, a comprehensive overview of the most commonly used ARAC methods and their hybridization is provided as part of the first phase of the research methodology. Lastly, a detailed Systematic Literature Review (SLR) is conducted with a particular focus on Mobile App Review Classification (MARC) methods. This SLR not only aimed at identification of the existing classification methods, but also served as the basis for the development of the proposed classification approach. First phase is concluded with the identification of the research gap and a comprehensive list of potentially relevant classification methods for ARAC. During the second phase of the research methodology, ROs and RQs are formulated that have to be answered. These objectives and research questions define the focus and scope of the current research. The third phase is related to the design and development of the classification models. This phase includes collection of dataset and pre-processing of the data, where the data is labelled and vectorized for classification by employing suitable process.

In the fourth phase, proposed classification model is evaluated through a case study on the real-world applications. Finally, a qualitative comparison with the prominent classification models, provides a clear picture of the shortcomings in the existing models and highlights the novelty and contribution of the proposed classification model.

1.6 Thesis Structure

This chapter of introduction has described the context of the thesis, defined research objectives, and outlined research questions of the thesis. chapter 2 provides a general overview of literature on review analysis methods, multi-label review analysis approaches, commonly used ARAC methods and their hybridization. chapter 3 3 provides all related work, summarizing to rank our work in the area of app review analysis and classification. chapter 4 provides a discussion on the general research methodology comprising of five distinct and successive phases, each of which having constituent activities, is described in separate sections. chapter 5 compares the performance of supervised machine learning models using lexical feature extraction methods. chapter 6 presents the design and development of the proposed classification approach, with detailed working principles and procedures of the constituent steps. chapter 7 reports the implementation of the proposed classification approach and detailed results of a case study on real-world Google play applications. chapter 8 offers the concluding statements and indicate potential future research directions.

1.7 Summary

This chapter presented the introduction of this thesis as well as the overview and the background of the problem to provide preliminary ground for the research. This chapter includes specified problem areas, formulated research questions, and research objectives to be achieved over the course of this research. The next chapter will comprehensively review existing literature on app reviews.

Chapter 2

Literature Review

First this chapter explains the necessary background of the problem, then a review of existing literature on review analysis methods is discussed. The discussion starts with some fundamental concepts of requirement engineering in section 2.1 that include requirement elicitation process, and requirement elicitation for mobile app development. Subsequently, a comprehensive discussion on different automated review analysis methods is provided in the section 2.2. Section 2.3 highlights the fundamental concepts of app review analysis including classification and multi-label review analysis presented by different studies. Figure 2.1 illustrates the organization of the this chapter.

2.1 Requirement Engineering Process

RE is defined as a systematic method to specify and manage requirements considering the following objectives defined in [29]:

- Understanding, documenting, and managing related requirements systematically as well as attaining an agreement among stockholders.

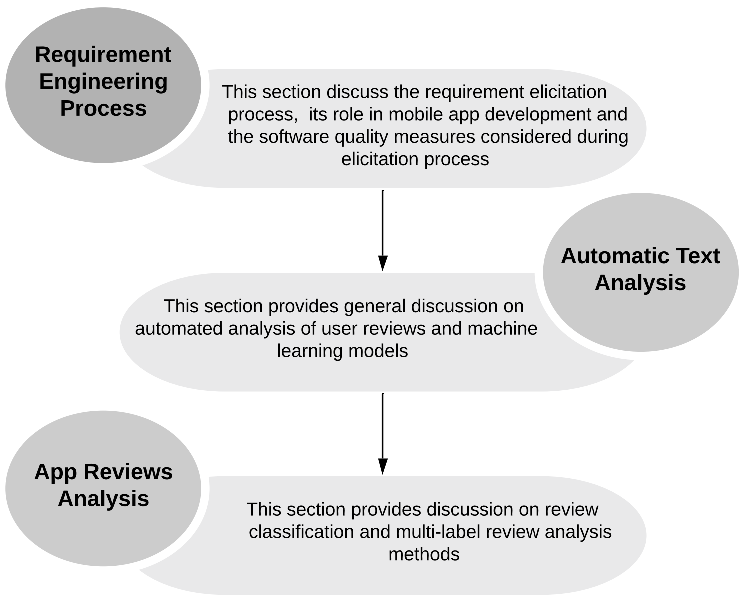

Requirement Engineering Process
This section discuss the requirement elicitation process, its role in mobile app development and the software quality measures considered during elicitation process
Automatic Text Analysis
This section provides general discussion on automated analysis of user reviews and machine learning models
App Reviews Analysis
This section provides discussion on review classification and multi-label review analysis methods

Figure 2.1: Organization of Chapter 2

- To reduce the risk of stockholders' dissatisfaction with the delivering system, specifications and management of requirements is necessary after understanding and documenting all the wants and needs of the stockholders

The four main activities to meet these ends are as follows:

Elicitation

During the activities of requirement elicitation, various methods are adopted for collection of requirements from different sources and specifically from stockholders, to clearly define requirements in more detail.

Documentation:

Defining elicited requirements properly, throughout the documentation activity. Various methods, using conceptual models or natural language are exploited to document requirements.

Validation and negotiation

To assure the completion of the defined quality criterion, validation and negotiation of documented requirements is essential.

Management

This activity is orthogonal to the rest of the RE activities, where structuring requirements is necessary by any means, to make them use-able by various roles, to manage stability after accommodating of changes in requirements and to assure their execution. Detailed discussion of requirement elicitation process and its significance in mobile application development is provided in this section. However, rest of the RE process activities are out of the scope of this thesis, not discussed in detail.

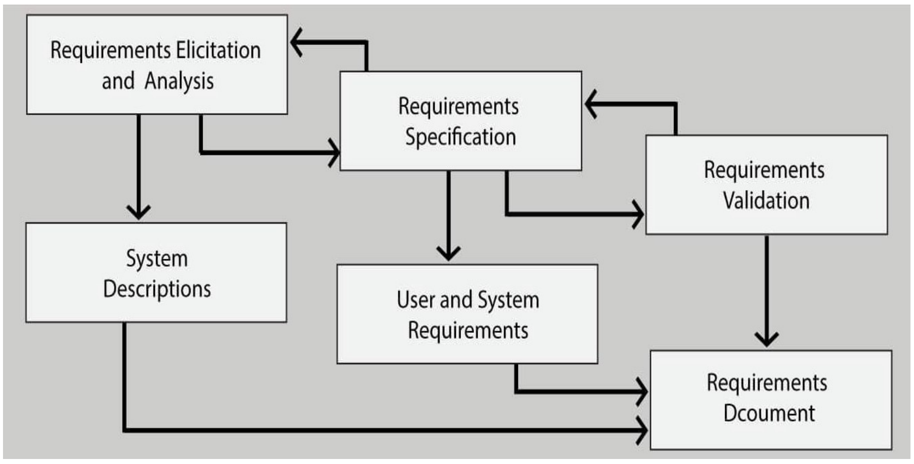

Requirements Elicitation and Analysis
Requirements Specification
Requirements Validation
System Descriptions
User and System Requirements
Requirements Dcoument

Figure 2.2: Requirement Elicitation Process

2.1.1 Requirement Elicitation Process

Among the most crucial steps in the SE project the one is requirement elicitation, process of getting information directly from system users. Overall process is described in the following steps:

- Identification of related requirement sources.
- Inquire about user's needs

- Examine the collected information; analyse the issues, inconsistencies, or any implications
- Be sure about the understanding and completeness of the requirements of users
- Synthesize the selected statements of complete requirements

Elicitation techniques provides the complete understanding of the stakeholders' mind, conscious, unconscious, and subconscious requirements. Moreover, selection of techniques to elicit requirements depends on the environment and type of the system, no universal approach is present that elicit requirements completely from all types of systems [44]. Some of the elicitation techniques are illustrated in the Figure 2.3.

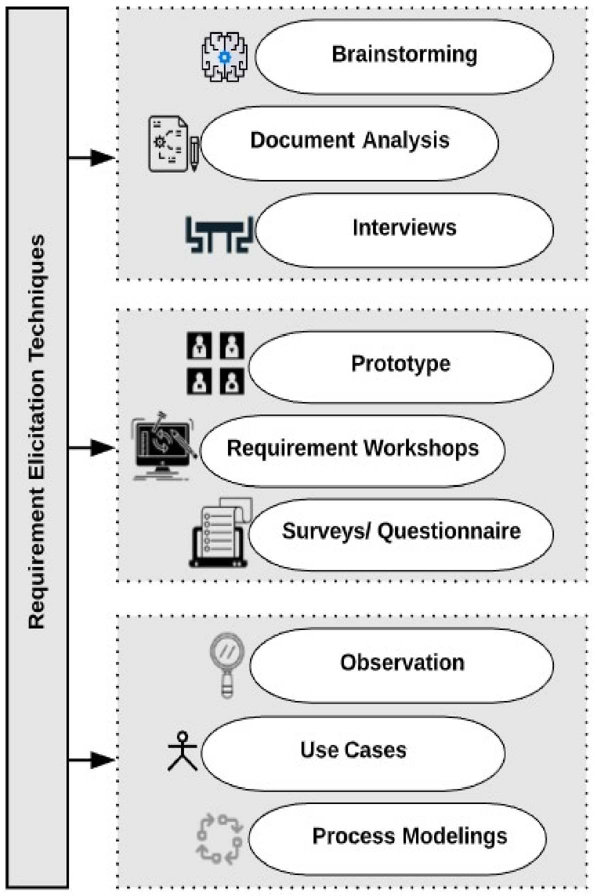


FIGURE 2.3: Requirement Elicitation Techniques

2.1.2 Requirement Elicitation for Mobile App Development

Among various influencing factors to the choice of elicitation techniques, the most influencing factor is the desired level of detail of requirements [45, 46]. Therefore, document centric approaches can be utilized well to elicit detailed requirements, including the strategies that make use of existing documents, considering the fact of extracting adequate level of detailed information from these documents [47]. Since, the intention of commercial mobile application developers is to appease a wide range of users rather than satisfying a specific set of users, gathered requirements should be detailed and reflect the expectations of more users. As mentioned, document centric requirement elicitation techniques result in more fine grained requirements. However, the agile and rapid properties of mobile app development process show less importance to a detailed documentation. In spite of the fact, the existence of any informative document about the app, is the property of a developer or an organization, originating a challenge of choosing a document centric approach as requirement elicitation method.

Therefore, the challenge of the choice of requirement elicitation technique for mobile application development can be addressed using the feedback of the existing applications, available on the app distribution platforms such as Google-Play store, in the form of User Reviews. Studies in literature introduced, the analyses of user feedback for gathering requirements as requirement elicitation technique [48, 49]. Dag et al. introduced the Baan requirements management process, which finds the relationships between feedback comments and business requirements (objectives) [50, 51]. Some studies addressed the extraction of requirements from feedback review comments of smartphone applications such as, Fu et al. introduced a system to perform analyses on App Store reviews and identifies requirements by topic modelling. The system also classifies feedback comments into individual functions [22]. Guzman et al. introduced an automated method that help developers to refine, combine, and analyse app reviews, and making use of an algorithm that find collocations to extract requirements from reviews. Than the requirements are combined into more meaningful groups of high-level features, using topic modelling [42]. Moreover, continuous evaluation of users' expectations and needs is a key to maintain a competing dominance in app marketplace [33]. To achieve this, user feedback can be incorporated into the process of requirement gathering Figure 2.4, improve the quality of an app [15]. specifically, user opinion has been considered as an important source of information in the previous studies[37, 40, 52, 53, 54, 47] that could be useful to improve app development activities, published in the form of 'user reviews' to app marketplace.

2.1.3 Software Quality Measures

The first phase of a software development release cycle is Requirement Elicitation, which is a process of collecting technical attributes of a software system along with the fulfilment of its quality attributes. However, quality of a software system is determined after its development using a standard quality model. These standard quality models are defined by "International Organization for Standardization" (ISO). Different versions of these models proposed at different times, the latest product quality model is ISO25010, proposed in 2011. This model describes two aspects of product quality evaluation, the first one is "Product Quality Model" and the second is "Quality in Use Model". Product quality incorporates features, that make the product free from defects and deficiencies, have tendency to meet customer needs and improve products to satisfy consumers. Moreover, 'Quality in use' defined as users' perspective of the product quality, consider the end results of the usage of a software product instead of the software properties, it is a collective effect of the quality characteristics of a software product [55]. These quality attributes contain NFRs that are usually overlooked during RE phase, due to the main focus on getting functional features of a system. Functional features are explicitly defined while non-functional attributes are latent within the text, which can lead to the lack of effective and efficient non-functional requirement elicitation and modelling methods and vague understanding of NFRs [25]. Moreover, various characteristics of mobile apps including battery life, screen size and multitasking impose a special set of NFRs on the operational attributes. In addition, ubiquity of mobile devices and emergence of new technologies has stimulate various constraints on apps' usability, reliability, compatibility, security and performance [15]. In the same way, NFRs' related information can be beneficial for both requirement elicitation and testing activities such as Step 2 and Step 3 (see green line in Figure 2.4). Moreover, information related to NFRs, mined from user reviews could be benefited to improve app quality before its release. As has been noted, NFRs directly impact user experience and latent within the review text. Therefore, it is challenging for a software developer to get aware of user's concerns and to integrate those concerns into app development process.

2.2 Automatic Text Analysis

A great number of user reviews in the form of text are being published to mobile app marketplace each day. Therefore, an automated approach can get relief from the manual effort of important information extraction from user reviews, support various MAD release cycle activities [51]. Example of a user review, containing useful information for the improvement of an app, shown in fig: reviewexample.

A typical automated approach practiced to extract relevant information from reviews text of users is primarily based on SML [14, 48, 56, 23, 57].

In SML, algorithms learn from the training pairs of input-output and map the samples according to the learned mapping function, the training set considered as an example is $R = (x^i, y^i)^{N_1}$ It would contain N number of user reviews, where each review considered as a whole sentence and a target label is assigned manually to the review, that is (x^i, y^i) where y^i denotes categories or labels into which we want to classify sentences such as NFRs and Functional Requirement. For the training of a classification model, the dataset is transformed into the representation of zero

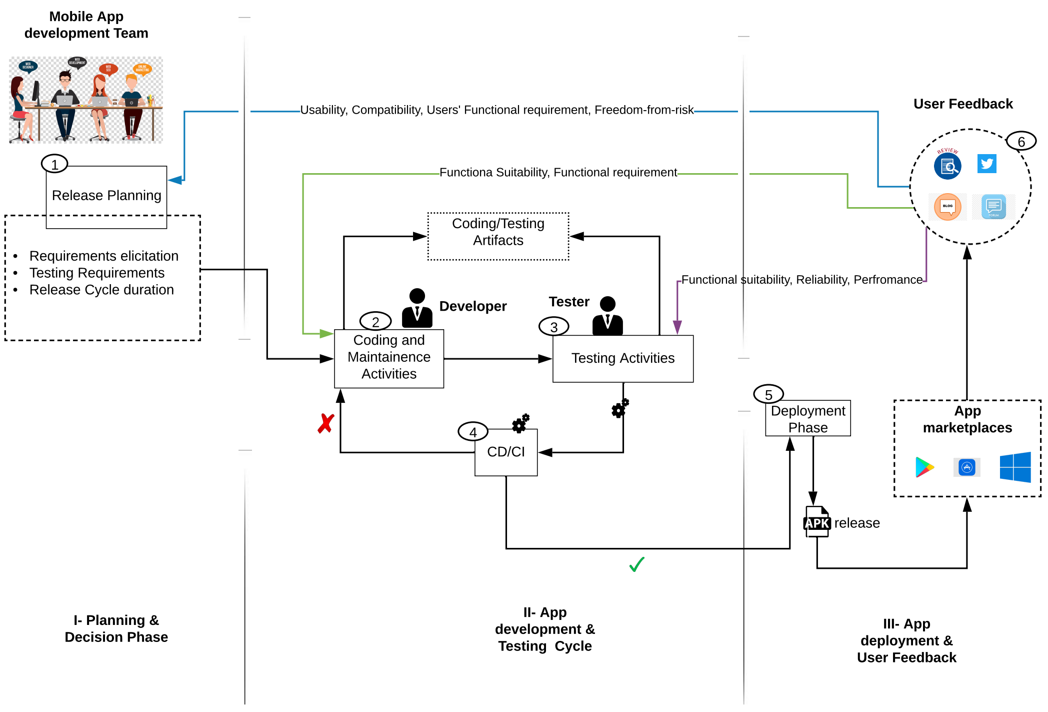

Mobile App development Team
Usability, Compatibility, Users' Functional requirement, Freedom-from-risk
User Feedback
Functiona Suitability, Functional requirement
1
Release Planning
Requirements elicitation
Testing Requirements
Release Cycle duration
Coding/Testing Artifacts
Functional suitability, Reliability, Perfromance
Developer
Tester
2
Coding and Maintainence Activities
3
Testing Activities
4
CD/CI
5
Deployment Phase
6
App marketplaces
release
I- Planning & Decision Phase
II- App development & Testing Cycle
III- App deployment & User Feedback

Figure 2.4: Overview of Mobile App Development Release Cycle

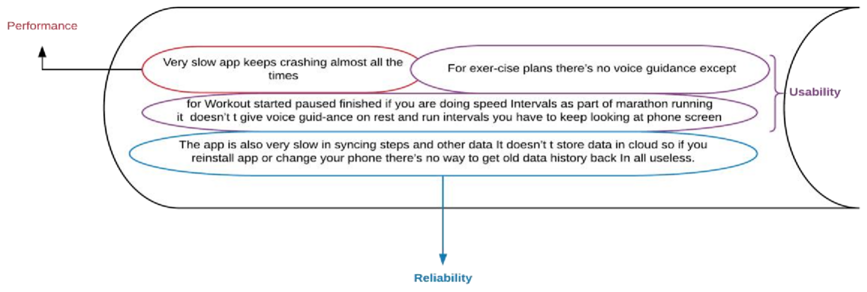
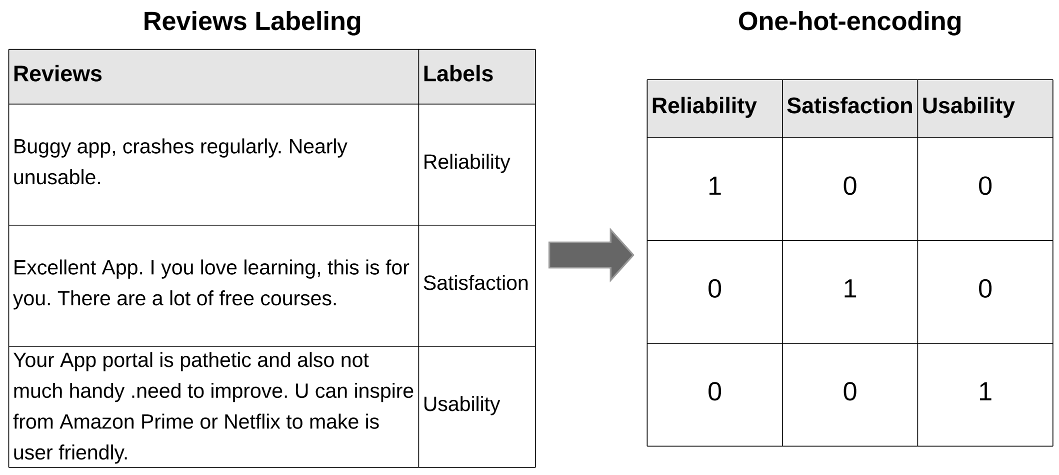

Performance
Very slow app keeps crashing almost all the times
For exer-cise plans there's no voice guidance except
for Workout started paused finished if you are doing speed Intervals as part of marathon running it doesn't t give voice guid-ance on rest and run intervals you have to keep looking at phone screen
Usability
The app is also very slow in syncing steps and other data It doesn't t store data in cloud so if you reinstall app or change your phone there's no way to get old data history back In all useless.
Reliability

Figure 2.5: Presence of Requirements in a Single Review

and one such as number one is assigned to a review that belongs to the defined label and all other labels are assigned as a number zero (see Figure 2.6). Reviews are textual data, training a model over the text requires the representation in the machine understandable format, which is a vector representation, each review x^i the review dataset represented as a vector of $x = (x_{i_{(1)}}, x_{i_{(2)}}, ..., x_{i_{(m)})}$ length m. Training examples contributes in the learning of a mapping function, using for prediction of the unseen data that is not used during training. Such as, resulting output of a training model would predict the defined set of possible labels for a



Reviews Labeling

Reviews	Labels
Buggy app, crashes regularly. Nearly unusable.	Reliability
Excellent App. I you love learning, this is for you. There are a lot of free courses.	Satisfaction
Your App portal is pathetic and also not much handy .need to improve. U can inspire from Amazon Prime or Netflix to make is user friendly.	Usability

One-hot-encoding

Reliability	Satisfaction	Usability
1	0	0
0	1	0
0	0	1

Figure 2.6: Encoding Example

review. Figure 2.5 also illustrates the occurrence of multiple labels in a single review.

2.2.1 Machine Learning for Reviews Classification

ML is a branch of AI, primarily based on the concept of building a model to automate analysis, train that model to learn various patterns from data and making decisions based on these identified patterns with less human intervention. Researchers are eager to study AI and to see the learning of machines from data, performing a specified task after learning patterns and without any explicit programming, which form the basis of ML. The iterative property of ML is significant, which enable the models to independently adapt the new data, learning from previously trained models to generate repeatable and reliable results. Therefore, studies revealed that ML is the most adaptive method for Text Analysis and Classification [40, 56, 14, 58, 57, 59]. Various approaches of ML has shown in Figure 2.7, having the capability to explore different type of datasets to gain information. In addition to the three basic ML approaches, ensemble learning and deep learning are also illustrated as part of ML. A short introduction of each approach, illustrated in the Figure 2.7, explained to get familiar with their basic purpose.

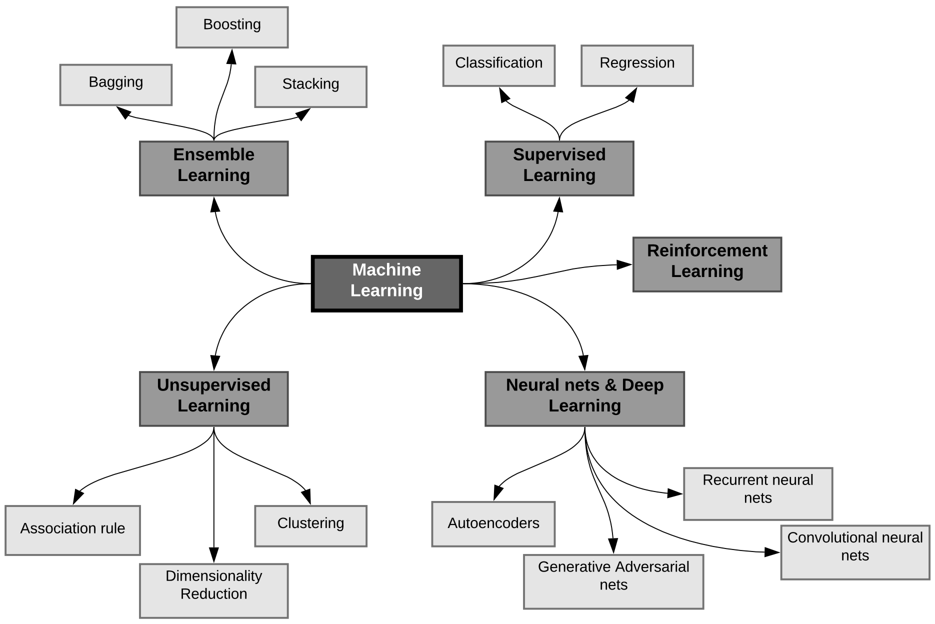


FIGURE 2.7: **Machine Learning Approaches**

Unsupervised Learning:

A category of ML, where model is well-trained to the observations with no priori output, may allow the autonomous construction of a new model rising from the data [60]. These type of ML algorithms, discover hidden patterns from the unlabelled input data, unlike SML methods where the labelled data is used to train a model. This approach of ML further segregated into three dimensions such as association, dimensionality reduction, and clustering, the most frequently used unsupervised learning example is k-means algorithm of clustering [61]. Turney et al. used unsupervised learning for review classification by analysing semantic orientation of reviews text [62].

Reinforcement Learning:

Learning from the environment, taking actions based on the learned experience is considered in reinforcement learning. Researchers have observed potential benefits of reinforcement learning that taken into consideration for classification. Such as, janasik et al. utilized inductive-learning strategies, to generalize over the similar problems [63]. Similarly, Dietterich et al. applied kernel-based approximation strategy [64], while Zhang et al. provided a RL method, to learn sentence pattern through the discovery of the structure of related tasks. They have adopted two representation models in their RL framework including HS-LSTM (Hierarchically Structured LSTM), that identify phrase structure to model hierarchical representation of sentences and ID-LSTM (Information Distilled-LSTM), that filtering the task-related words for the purpose of distilled sentence representation. [65].

Ensemble Learning:

Ensemble techniques are combination of different learning algorithms, applying a set of classifiers to classify the data, consider vote based method for prediction. Different mechanisms including bagging, boosting and stacking are utilised to implement the concept of ensemble learning [66] The motivation behind the use of ensembles is to dilute the weaknesses of various classification algorithms while emphasizing their possible strengths [67]. In the study [54], the majority voting mechanism is applied to integrate the outcome of four selected classification algorithms for App Review analysis.

Supervised Learning:

In SML, the dataset is labelled to acquire a reasonable and sensible output. Therefore, supervised learning is the most explored dimension of ML for App Review Classification. Various supervised learning approaches implemented in the literature to analyse and classify user reviews [11, 14, 17, 38, 57, 56].

SML Algorithms:

This section provides an overview of the state-of-the-art SML algorithms, implemented in this thesis for reviews classification.

Logistic Regression:

This SML algorithm is one of the basic classifiers for binary classification that model the output variables through linearly combined input features to learn from them. The equation 2.1 is used by the model to compute the weighted sum of input features $x_1, \ldots, x_n$, to predict the input feature vector $x = (x_1, \ldots, x_n)$.

$$\overline{y} = b + \sum_{j=1}^{n} w_i.x_j \tag{2.1}$$

In equation 2.1, b is the bias, and wj represents learnable weights corresponds to the input features xj. The output of the equation act as an input for a sigmoid function that gives the output value in the range of 0 and 1. The bias term and weight coefficients are learned through loss function optimization, known as cross entropy loss. The model performance can be measured using cross entropy loss, wrong predictions results in the increase of entropy loss.

Support-vector-machine:

SVM is an algorithm for regression and classification analysis, a supervised machine learning approach [68]. But mainly implemented for classification, with working principle of margin calculation. These margins are drawn with the maximum distance between classes, thus minimize classification error. An example has been shown in Figure 2.8 from [69], to illustrate the working of SVM. When the num-

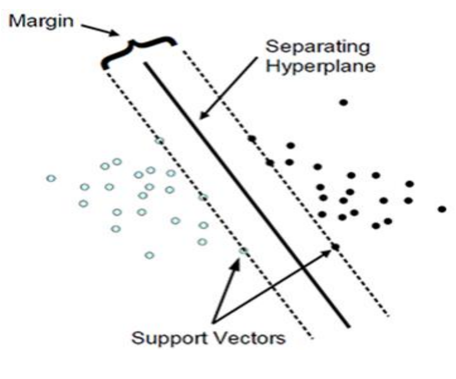


FIGURE 2.8: Working of SVM[69]

ber of classes that need to be predicted is more than two, the data is represented in multiple dimensions using multiple hyper planes. Hyper-planes are critical instances in the given training set and support vectors' location used to determine the position of dividing hyper-planes. To classify the data, SVM map the input feature vectors into an n dimensional space and determine the place of new coming data instances on the defined hyper-planes.

Naïve Bayes:

NB is based on Bayes' theorem, yet simple, linear probabilistic and efficient classifier [70]. It creates a tree like structure, based on the probability of occurring and not occurring, called Bayesian network. For text classification, NB consider the conditional independence among the presented model, assuming features as separate words from corpus of the text similar to BOW approach [71]. To classify user reviews, NB has been recognized as a competitive classifier in measure of its performance, predicting multiple categories of reviews with good accuracy rate [14, 40].

Random Forest:

RF also called Bagging algorithm, an ensemble classifier, combine bootstrap, decision tree, and voting methods. Multiple decision trees are constructed by picking up data randomly, and majority voting mechanism has been used to choose among the multiple decision trees [72]. It is a powerful ML algorithm which performs very well for categorical variables.

Deep Learning:

DL permits the learning of data representation through multiple degrees of abstraction, to computational models that comprised of several processing layers. Deep neural models proved to be effective to avoid the effort of manual feature-engineering, which is a part of traditional ML models. In the recent years, various DL models have been presented for text classification in different domains [73, 74, 75, 76, 77, 78, 79] with the achievement of encouraging results. But, their performance to classify app reviews has not yet been evaluated. Therefore, a DL model named CNN is discussed in this section which is evaluated for app reviews classification, other deep learning models or out of the scope of this thesis, hence not discussed in details.

Convolutional Neural Network:

Stacking of different layers form a CNN architecture, these layers named as embedding, convolutions, pooling and fully-connected or dense [80]. Functionality of the basic CNN is comprised of the following stacked layers The embedding-layer consider the indices of words in a sentence $w_i \in \{1, 2, 3, ..., V\}$ as input and yields a corresponding vector $X = \{v_1, v_2, ..., v_L\}$ of embedding as output, here V corresponds to the vocabulary size and D corresponds to the size of the dimensions in the embedding vector. Embedding are matrix representation $W_c \in \mathbb{R}^{D*K}$ of the input words, here L represents the sentence length including padding. Usually the pre-trained embedding vectors, such as Word2vec and Glove are used to initialize the embedding layer [76].

The 1D convolutional layer, convolves the input along with various filters of different widths. These filters serve the purpose of linguistic feature detectors, extracting n-gram model considering different degrees of granularities. Each Convolutional filter $c = \{c_1, c_2, ..., c_L\}$ maps the k number of words in the concerned field to an individual feature c. To extract new set of features $c = [c_1, c_2, \ldots, c_L]$ a window slides through the whole sentence according to the filters, using the Equation 2.2.

$$c_i = f(X_{i,i+k}.W_c + b_c) \tag{2.2}$$

In the equation above, f represents as a non-linear activation function, such as ReLU, whereas $b_c \in \mathbb{R}$ is a bias. If the filters are n_k, with the width of k than the output feature matrix C will be $C \in \mathbb{R}^{n_k * L_k}$ The pooling layer, takes the max or average value of the each convolutional filter, among the generated convolutions of features, and to yield a fixed size vector equivalent to the size of filters that is n_k. The max-pooling takes the maximum value, while average-pooling takes the mean value from each convolutional filter, to form feature convolutions. At last, the output of pooling layer act as input of the fully-connected layer, an activation function is applied to predict the output labels of the input sentences.

2.3 App Reviews Analysis

We review the existing work about automatic user review analysis done to support various activities in the Mobile Development Release Cycle. To put our research into perspective, we organized the existing literature along two directions pertinent to our research questions: (1) App Review Classification, (2) Multi-Label Review Classification.

2.3.1 Reviews Classification

Recent research have been focused on the use of text classification models to discover helpful information from app reviews. Different taxonomies are proposed in the previous studies, supporting SD activities by classification of reviews. Studies revealed that the proposed taxonomies have concentrated on supporting software evolution and maintenance activities in review classification [53, 54, 12, 40, 81]. However, various studies have considered the classification of reviews information into FR or NFRs to facilitate RE activities in the release planning phase [47, 51, 82].

Review text contains some representative features, classification models consider those features to train a model. Various methods have been proposed and applied in the literature such as BoW, POS tags, TF-IDF and many others. For example, in the study of Maalej et al. [14] and Chen et al. [13] review classification is performed using the textual features extracted by word n-gram, a method of BoW.

While, in the study of Gu et al. [83] a rich set of linguistic and lexical features have been extracted by utilizing NLP tools including parsers and taggers.

The study of Groen et al. [5] explored the existence of NFRs in user reviews of mobile apps, using the tagging method and found the main concerns expressed by the users that are reliability and usability, with the focus on fault tolerance, adaptability, interoperability and operability. In addition, authors presented a set of linguistic patterns, which capture user expressed usability concerns automatically. Evaluation of these patterns on large datasets of reviews has shown their use in identification of user quality concerns, reporting high precision whereas low recall level.

The study of Ciurumelea et al. [47] presented that analysing user reviews and codes of mobile applications support better release planning. In the proposed taxonomy, the user reviews are defined in the highly relevant mobile specific categories such as pricing, compatibility, resources, complaints and usage, helpful during software maintenance activity for the developers. Classifying reviews and recommending modification in the source code files to resolve the identified issues, a prototype is introduced, built using ML and Information Retrieval techniques. Evaluation of the proposed approach using thirty nine open source Google-Play apps, has shown that the code artifacts can be localized using this approach with 51 percent average precision and 79 percent average recall level, in addition to the organization of reviews according to the defined taxonomy with 80 percent average precision and 94 percent average recall level.

Several probabilistic methods were introduced in the study of Maalej and Nabil [14], to classify reviews into rating, feature request, bug-report and user experience. They experimented various binary and multi-class classification approaches, such as NB, Maximum Entropy and DT. Their performance is evaluated using the dataset that is sampled from Google-Play and Apple-App store, containing 4400 manually labelled reviews. Results revealed that binary classifiers such as NB, were more promising for the prediction of review type. Moreover, various factors of reviews had a little effect on the classification accuracy, these factors include; length, tense, sentiment, and star rating, along with the text pre-processing techniques including; lemmatization and stemming.

To aid release planning, Villarroel et al. [51] has introduced A Crowd Listener for releAse Planning (CLAP) that classify user reviews, using density-based clustering approach and prioritize these clusters as low or high priority, using RF algorithm. Evaluation results revealed that CLAP performed accurately and give better recommendations for release planning. The simple linguistic patterns outperformed over more complex and computationally expensive models, extract features and perform categorization without any training overhead, identified in the study by Johann et al. [34].

To mine NFRs from mobile app user reviews Lu et al. [23] performed sentence level analysis and augmented review sentences with similar words for better classification using Bag-of Word method. They handle the problem as multi-class classification and classified the reviews into four categories of NFRs and one category of functional requirement using ML classifiers, Bagging, j48 and Naïve Bayes. However, jha et al. [17] mined NFRs from mobile app store reviews, using review information and assigning multiple labels to a single reviews for classification using Binary-relevance method and Dictionary-based approach. They classified reviews under four categories (dependability, performance, usability and supportability) of NFRs, using naïve Bayes and SVM as a classification methods.

2.3.2 Multi-label Reviews Analysis

Reviews are in textual format and can be classified as a whole review, which is review level classification, or by splitting into sentences which is sentence level classification. Such as in the study of Maalej et al. [14], classification has performed at review-level. As, the users express their opinion in a single review, containing multiple types of information. Therefore, in the study of Mcllroy et al. [32] classification has performed by assigning multiple labels to a single review, identifying up to one third of the user reviews raise more than one issue type. In addition, multi-label approach is useful for stockholders, when comparing competing app issues, distributing issue over app category and detecting anomalous apps.

If a review in user reviews, is found as a unit of information, then a single review express multiple type of information. Therefore, a user review thus labelled under more than one or two categories [84, 17]. Various approaches exist to classify data having multiple labels including binary relevance, classifier chains, ensemble of classifier chains, and ensemble of binary relevance. These methods are compared by applying on different datasets with multiple labels by [85]. Liu et al. [84] compared various multi-label classification techniques on sentiment classification, using eleven state-of-the-art multi-label classification techniques, on 2 microblog datasets, and evaluated using eight evaluation metrics. Empirical evaluation and comparison of 3 sentiment dictionaries is also performed to analyse their effects for multi-label classification. The study by Pham et al. [86] used LDA (Latent Dirichlet Allocation), the hidden topic model to enrich features and ML to perform feature selection of user reviews, for multi-label review classification. Jha et al. [17] mined NFRs from mobile app store reviews, using review information and assigning multiple labels to a single reviews for classification using Binary-relevance method and Dictionary-based approach. They classified reviews under four categories (dependability, performance, usability and supportability) of NFRs, using naïve Bayes and SVM as a classification methods.

In addition to multi-label categorization problem of reviews, the feature engineering for review classification can cause false effort of feature extraction, considering unrelated information present in the reviews. To address this problem, an emerging domain of ML models has been introduced in recent years, which automatically learn features from labelled data, known as Deep Learning. Specifically, Convolutional Neural Network (CNN), a model of DL, has performed effectively for numerous text classification problems [75, 74, 76, 77, 73]. However, CNN performance for multi-label app reviews dataset has not yet known.

2.4 Model Evaluation

After training a ML classification model for reviews, evaluating a model performance on an unseen data is a critical task. To achieve this objective, several measures can be applied to evaluate the performance of the classification model on the test set. While evaluating the model, a task is given to determine the possible labels for each sample in the test set. Later, the performance is estimated, by comparing the predicted outcomes against the actual labels for an unseen data. Now, this section describes the commonly used metrics for evaluation of ML models, to assess the quality of these models.

Model Performance Measures: Construction of a confusion matrix is an easy approach to describe an ML model performance for a test set, where each row represents actual label instances whereas each column represents predicted label instances. TP (True Positives) are described as a number of correctly predicted classes of test samples that were true for both; the predicted labels and the actual labels. Similarly, TN (True Negatives) are described as a number of correctly predicted classes of test samples that were labelled negative and also predicted negative. However, the FP (False positives) are incorrectly predicted cases, where the predicted label is positive while the actual label is negative. FN (False Negatives) are test samples that predicted incorrectly as negative while the actual label is positive. Based on these predictions, some metrics are used that consider them to evaluate the model performance, includes the following. **Precision (P)** is interpreted as the total number of correct predictions (TP), dividing by the sum of

TP and FP, under specified labels.

$$P = (TP)/(TP + FP) \tag{2.3}$$

However, **recall(R)** is measured as TP over total number of instances under that category.

$$R = (TP)/(TP + FN) \tag{2.4}$$

The weighted-average of recall and precision for each category is represented as **F-measure**, computed the f1 value independent to the category as

$$F1 = 2 * (P * R)/(P + R) \tag{2.5}$$

Accuracy: is a simple metric to evaluate models, computing the percentage of correct predictions. Whereas, the imbalanced classes may lead to the biased accuracy results such that higher number of negative labels than positive ones, can mislead the accuracy by assigning that model a higher score which consistently predicts the negative label. In a real time situation, correct classification of infrequently occurring examples would be much more essential. Hence, in this scenario, use of precision and recall metrics to measure prediction quality is recommended, as both of them evaluate the two different prediction quality aspects, precision estimates the ratio of positively predicted class labels that are true while recall estimates the ratio of all positives. These metrics can be combined in a single representation, a harmonic mean, known as F1-score.

Model Validation and Selection:

The objective of ML based algorithms, is to predict unseen data using a previously trained model. It is insignificant to validate the model on the data used during training, as the model get familiar with the training data samples. Therefore, model performance should be evaluated over the unseen data, to measure the generalization capabilities. To put this into practice, the input data is split into 2 different sample sets, where the one is used for training and other is used for testing. Separating the test set from training set known as Holdout method, as the test set is held out while training the model.

A common approach for partitioning the data, is known as random sampling. At times, when the same labels as of original data, need to be observed in both training and testing, the stratification is utilized together with random sampling. SML algorithms use various hyper-parameters, specified manually, to restrict the complexity of the subsequent model. Increase in the learning model's complexity can cause an over-fitting problem, which undermine the performance of test set due to an extreme memorization of training set. Whereas, over simplified model would not have the ability to detect the underlying data patterns, which result in a situation of model under-fitting. Both, under-fitting and over-fitting would have impact on the model to test data, causing it to suffer from the performance efficiency.

Hyper-parameter tuning of the model is performed to ensure optimal performance of the model on test set. To accomplish this goal, different hyper-parameters can be configured to train several models, choosing those that yield best performance when evaluated to test data. Use of the same testing data for selection and evaluation of the model is an invalid method. Since, this would give generalized performance by an unreasonably optimistic evaluation. Consequently, the holdout approach is recommended, that split the data into three subsequent parts; train, test and validation set. At first, the model is trained using training set based on different parameter settings, then selecting the setting with best results. At last, the generalized performance is evaluated on the test data using the finalised model.

2.5 Summary

This chapter presented the overview of the literature on fundamentals of App Review Classification, Requirement Elicitation for Mobile App development and Multi-label Review Classification. These fundamentals include discussion on essential elements for app reviews analysis including review classification, text analysis approaches and multi-label review analysis. First, we discussed the concept of Requirement Engineering and the process of requirement elicitation in RE, along with the role of requirement elicitation in mobile app development. Then the automatic text analysis methods and app review analysis problem is discussed to perform reviews classification. The next chapter extends the scope of the Literature review activity by conducting and reporting the results of a Systematic Literature Review (SLR) on ML based ARAC methods. The next chapter also aggregates the findings from the current chapter as well as findings of the Chapter 3 and highlights the research gaps in the existing literature on ARAC.

Chapter 3

Mobile App Reviews Analysis and Classification

As the current research mainly emphasizes on the ML based ARAC methods, where criteria and alternatives are finite in numbers, it is imperative to analyze the extant research on this particular domain. For this purpose, an SLR was conducted, which is aimed at research gap analysis and identification of the relevant classification approaches for ARAC. The goal of this systematic review is to present a detailed and comprehensive overview of the state-of-the-art ARAC solutions based on ML approaches. In conjunction with the findings from chapter 2, this chapter also reports the existing research gaps as well as potential future directions for the future research. Section 3.1 presents a description of the SLR protocol. The section 3.2 describe a taxonomy of ML-based ARAC methods, which is used to extract data from the selected studies. Subsequently, in section 3.3 a comprehensive analysis of the selected studies and synthesis of the findings is provided. Section 3.4 provides a research gap analysis based on the findings of both chapter 2 and chapter 3 and also contains the concluding remarks. Finally, section 3.5 presents a summary of this chapter and its relevance to the subsequent chapters, chapter 3 and chapter 4.

3.1 Systematic Literature Review Protocol

An SLR protocol defines in advance how a systematic review is to be conducted. It provides a specification of the methods to be used in conducting a specific SLR. Prior construction of such protocol is indispensable to structure and guide the research and to avoid the potential biases. This section provides the description of an SLR protocol used to conduct this systematic review.

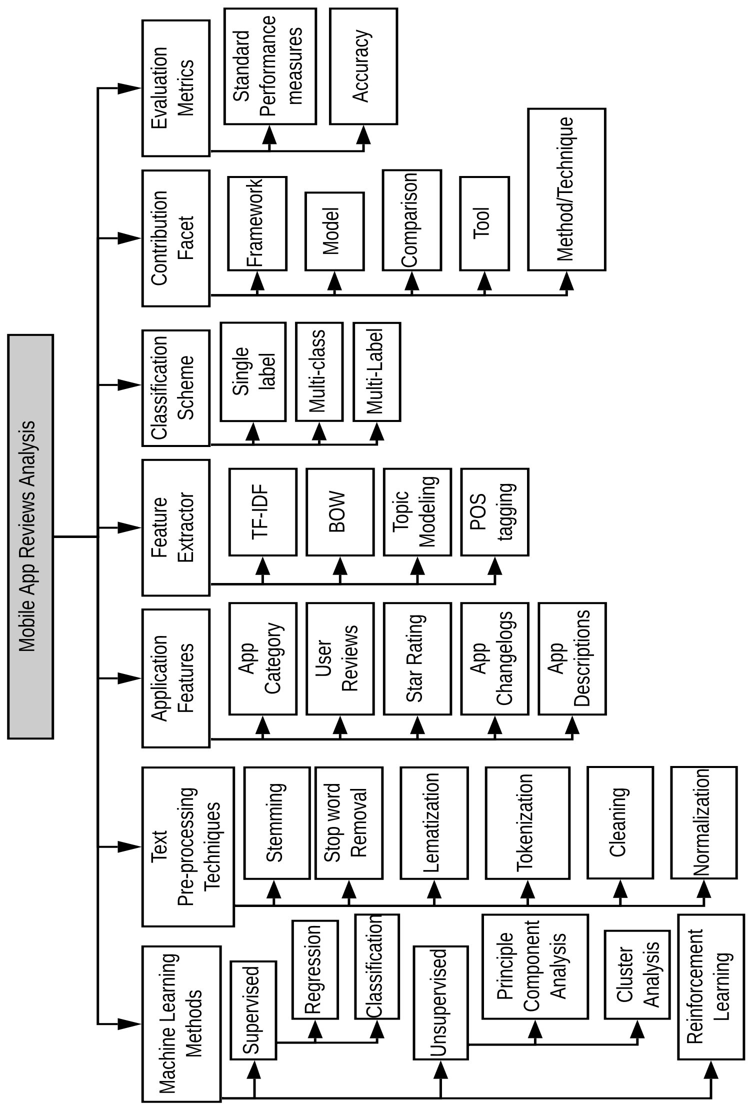


Figure 3.1: Mobile App Review Classification Taxonomy

3.1.1 Research Questions

The research questions aimed at our target to search in a specified area. The defined RQs as in Table 3.1 are established on the basis of a primary question; "What are the state-of-the-art methods to analyze mobile app reviews? The motivation for

Table 3.1: SLR Research Questions

RQ#	Research Question	Motivation
RQ1	What kind of user feedback has been mined from app reviews?	Identify the type of information collected from app reviews.
RQ2	What are the most investigated Machine learning approaches to extract information from mobile app reviews?	Categorizing various methods (i.e. supervised, unsupervised, semi-supervised algorithms) that have been applied to gather information from app reviews.
RQ3	What are the text-pre-processing techniques and methods stated for review analysis?	Identify the most widely used methods to process the raw text and effectiveness of these methods as well as their shortcomings.
RQ4	What are the app features and feature extraction methods considered to analyse mobile apps?	To enlist the mostly used application metadata (i.e. app category, rating, sentiment, textual review) in order to perform analysis. To identify the approaches used for extracting meaningful view of review text (i.e. frequent words)
RQ5	What are the most used evaluation metrics?	Investigation of all possible evaluation metrics that have used in the previous research. They can help to perform a quality evaluation for the any new proposed schemes.

each question is also described along with the research question.

3.1.2 Search Terms

To search the relevant work presented by the researchers, the key terms are the basic component for adequate search of literature. Key terms based search is divided into three parts: population, intervention, and outcome as suggested by the Kitchenham

[87]. Based on the identified search terms and using the suggestions of Kitchenham, we designed a structured query to search the databases of electronic data sources. The query to search each electronic database vary in the structure due to different search policies of digital libraries. Appendix A show the executed queries on each database and data sources utilised to perform this search.

3.1.3 Studies Selection

According to the standard guidelines of Kitchenham [87], studies are selected through a model called PRISMA. Studies are included and excluded based on some defined criteria and to further refine the outcomes, a quality assessment criteria is applied. The PRISMA model and applied criterion details are provided in the Appendix A.

3.2 Data Extraction

Selected articles need to be analysed by the researchers to extract relevant information. This information is based on the defined taxonomy (see Figure 3.1), using a data extraction form designed for the purpose of information extraction, based on the RQs (see Table 3.2).

3.3 Results and Discussion

To answer the defined research questions, taxonomy for classification of studies is considered and the results are presented accordingly. Summary of all the findings against research questions described in Table 3.3.

3.3.1 Type of Information mined from Feedback

User reviews are analysed with various intentions, depending on the type of information required to the researcher. The most common information categories to mine from user reviews, identified in the selected research studies are feature

Table 3.2: Focus of Data extraction form

Search focus	Data-item	Description
General	Identifier, Bibliography, Publication-venue type, Study objectives	Reference number given to the Article, Author, year, title, Publication Source, Aims or objectives of the study
RQ1	Feedback type mined from reviews	Requirements, categories
RQ2	Machine learning methods used Datasets	Algorithms, models and methods App stores and number of applications
RQ3	Text Processing of reviews App data considered	Pre-processing methods and approaches Descriptions, reviews, rating etc.
RQ4	App Feature extraction Methods	Techniques for automatic feature extraction
RQ5	Performance Measures	Standard performance measures (precision, recall, f-measure) Accuracy

request, bug report and sentiment. Such as the study of Chen et al. [41] presented AppReview miner to categorise reviews into two classes including 'informative and non-informative' type, while authors of two studies Panichela et al. [40], and Disorbo et al. [39] reduced the user review analysis effort by automatically categorising them into multiple classes of 'Problem Discovery, Feature Re-quest, Information Seeking, Information Giving and Other'. Similarly, other researchers like Gao et al. [52] identified issues reported in user reviews about the app, Ciurumelea et al. [47] defined a taxonomy including categories of 'Compatibility, Usage, Resources, Pricing, Protection, and Complaint' to classify reviews into multiple classes, Deocades et al. [48] performed preliminary study to classify reviews into two classes of 'functional and non-functional requirements', Liu et al. (2016) discovered hot entity discussed in user reviews and based on the hot entities performed sentiment analysis, another study also performed sentiment analysis on user reviews [36]. However, 'Problem: Consist of a bug report, complaint, and feature request, Request: Consist of a user request, Improvement: Consist of feature strength and

Table 3.3: Description of the studies included in analysis

SID	Ref.	Description
S1	[81]	App descriptions and reviews analysis to get domain knowledge
S2	[17]	Mining non-functional requirements from user reviews and categorizing into multiple classes
S3	[9]	Application quality assessment by analysing user sentiments
S4	[88]	Review classification using baseline classifiers and proving the accuracy of active learning
S5	[89]	Clustering user reviews for better labelling
S6	[90]	Categorizing reviews reporting function error and then locating that function error in code
S7	[58]	Review classification for problem discovery and feature request
S8	[50]	Classification combined with sentiment analysis to get information from reviews
S9	[82]	Considering reviews, rating and app category to analyse reviews for better release planning
S10	[91]	Identifying defects in user reviews
S11	[47]	Reviews classification and code change recommendation
S12	[23]	Augmenting user reviews with textual semantics for automatically classifying user reviews into non-functional requirements
S13	[48]	Classification of reviews into functional and non-functional categories
S14	[36]	Discovering hot entities in user reviews and then compute the sentiment of reviews based on ratings and sentiment
S15	[92]	Identifying cross platform app issues by using keyword based method
S16	[39]	Classify, cluster and summarize app reviews for recommending change
S17	[51]	Categorizing and clustering user reviews to recommend change to developers in next release
S18	[52]	Identifying issues in app by tracking reviews over the versions
S19	[8]	Extracting relevant reviews and classifying into categories to improve application
S20	[41]	Grouping and ranking informative reviews from mobile app user reviews

other' classes are considered in the study of Supryagoi et al. [50]. Furthermore, categories including 'bug-report, feature-request, performance report, security issue, excessive energy consumption report, usability improvement, non-informative, defective/not-defective, crash, battery, memory, network, privacy, spam, UI, problem discover, improvement and other' are considered to analyse reviews [49, 88, 58, 91, 92]. At last, two studies on mobile app review classification, considered FR and NFRs classes to categorise reviews, but in study of Lu et al. [23] only four categories of NFRs are considered including 'reliability, usability, portability, and performance', while study of Jha et al. [17] mined NFRs considering them as multi-label categorization problem and classified a single review under multiple available classes including 'Dependability, Performance, Usability, Supportability'.

3.3.2 Machine Learning Approaches for Mobile App Reviews

People are often prone to making mistakes while performing analysis and establishing relationship between multiple features. This makes it difficult to find solution of various problems. Therefore, machine learning can successfully be applied to solve these problems. In machine learning approach, a computer learns (without explicitly programmed) to solve a problem. There are three main dimensions of machine learning including supervised learning, unsupervised learning and Reinforcement learning. Every instance used by ML algorithms in any dataset is represented through similar features set. These features may be of binary type, categorical type or continuous type. In supervised learning, instances are given with labels while in unsupervised learning instances are unlabelled. There is an ensemble of supervised and unsupervised learning where labelled data is in small amount while unlabelled data is in large amount, it is called semi-supervised learning. While, learning from the environment, taking actions based on the learned experience is considered as reinforcement learning. Analysis of selected studies shows that supervised learning is the most widely used approach for analysing mobile app reviews. SVM and NB are the most widely applied ML algorithms to app reviews. NB is based on Bayes' theorem, yet simple, linear probabilistic and efficient classifier, while SVM is based on the working principle of margin calculation using support vectors. The authors jha et al. proposed a classification model based on SML algorithms SVM and NB, applied on 1100 user reviews, sampled from Android and IOS apps, to categorise into NFRs. The study of Lu et al. presented the use of NB algorithm, an ensemble method bagging and decision tree algorithm j48, to mine NFRs from Whatsapp and iBook user reviews, performing sentence level analysis. In the study of Panichella et al. an automatic classification system is proposed for user reviews based on a predefined taxonomy, using SML algorithms such as SVM, NB, J48 and logistic regression, to assist requirement evolution and software-maintenance. As in many real-word applications, topics revealed by Latent Dirichlet Allocation (LDA) and Aspect and Sentiment Unification Model (ASUM) are needed to be verified by experts to ensure they are semantically meaningful within the domain analysis. In the study of Chen et al., an AR-miner is presented, which collect app review labels from training and test set according to the defined rules, using EMNB, LDA and ASUM. The dataset is collected from four Google-play apps including Facebook, SwiftKey, Templerun2 and Fishtap. The study of Ciurumelea et al. presented that analysing user reviews and codes of mobile applications support better release planning. Classifying reviews and recommending modification in the source code files to resolve the identified issues, a prototype is introduced, built using ML (Gradiant boosting regression tree algorithm) and Information Retrieval techniques. Evaluation of the proposed approach using 7754 reviews gathered from thirty nine open source Google-Play apps of 17 different categories. To aid release planning, Villarroel et al. with study id S17 has introduced A Crowd Listener for releAse Planning (CLAP) that classify user reviews collected from 210 Google-play apps, prioritizing them using RF algorithm. Furthermore, the author Liu et al. with study identifier S1 used random forest, considering user reviews and app descriptions of 260 apps of Google-play to help developers, understanding domain specific knowledge. Masrury et al. in study [49] used NB algorithm to perform sentiment analysis on Google-play app reviews. In study [88]the authors Dhinakaran et al. experimented with NB, SVM and logistic regression on the reviews of 1200 Google and iOS apps. Yu et al. in study [36], map function error, identified from reviews, to code, experimenting with SVM, RF and boosted regression on 18 Google-play apps with source code availability. The study of Ekanta et al. with identifier S7, use SVM, LR and NB algorithms to classify 438 Indonesian language reviews, separated from 22,000 Google-play reviews. Suprayogi et al., in study S8, used SVM, LR, and NB to extract information from Google-play app reviews, also considering sentiments. Study S10 by Wang et al. used various SML algorithms, specifically SVM based model to perform classification on Apple-Apps. In study S16, Di Sorbo et al. used ML as sentence detector in user reviews mined from four different platforms, considering 17 different categories of apps. However, two studies S14, and S5 applied semi-supervised learning while four studies S9, S13, S15, and S18 applied un-supervised learning methods to analyse user reviews.

3.3.3 Text Pre-processing Approaches

The RQ2 investigates the techniques applied to raw text, in order to make it ready for analysis, and to remove any extra information from it. User reviews are written by application users in natural language and usually in English sentences that are easy to understand and interpret by humans. But humanly it is impossible to read and interpret large number of reviews. Therefore, machine learning came into existence to reduce the human effort for analysing and interpreting large number of user reviews. User reviews are natural language comments, which is raw text that needs to be cleaned and transformed into meaningful form. For this purpose, various text pre-processing techniques are applied to raw text. These techniques include tokenization, stemming, stop word removal, lemmatization, normalization and cleaning of text. Lengthy reviews are broken down into word tokens or sentence tokens to better perform further pre-processing operations on the text data. Normalization converts text data to standard form and scale it according to the complete words such as ‘plz’ and ‘pleease’ is transformed to its form ‘please’, while cleaning remove the punctuations, symbols and numbers from text. In addition, stop words are the most common words used in language such as ‘the’, ‘an’, ‘a’, there exists neither a universal list of stop words nor a universal tool to remove stop words. Furthermore, stemming and lemmatization both reduce morphological variation, the difference is that stemming stems the word to its root such as ‘processing’ and ‘processed’ both are stemmed to ‘process’ while lemmatization links the word to similar meaning words such as lemma for ‘caring’ is ‘care’). Text pre-processing is a more important task for supervised machine learning application than other ML techniques. There may exist some other methods of text pre-processing but we have discussed only the most widely used methods in reviewed literature. All mentioned techniques have equal importance but tokenization and stop word removal in addition to stemming and lemmatization are utilized the most. Table 3.4 shows the distribution of studies according to the techniques they utilized.

3.3.4 Application Features and Reviews Features Extraction

Application features are the additional attributes related to the application rather than its main functionality. These features are useful for commercial app developers as well as professional app developers to analyse the app market, user concerns and popular trends. In addition to app functions, various features are related to an application including: category the app belongs to, based on the underlying functions the app presents, differentiate it from others. Version of an app that represents its release history, track changes made to the app over time and significant for better release planning. App summary/description that tells about the app, functional details and the gap it fills. Rating in the form of stars (usually 1-5) is a quick way to give feedback about an app. At last, the most important app feature is user reviews that are in textual format and analysed most to get information expressed by the app users. Studies showed that user reviews are considered more often for analysis followed by rating and app descriptions. Therefore, to perform analysis on reviews text, it needs to be transformed into some machine representation format. Various approaches are presented and used in the literature to accomplish this task, including TF-IDF, BOW, CHI2, Topic Modelling, POS tagging and word embedding (word2vec). TF-IDF technique is used to calculate the importance of a particular word in a document. This importance depends upon the frequency of the word occurred in that document. BOW is a technique of creating a bag of similar words by counting their frequency and each sentence considered as a separate document. CHI2 helps to test the relationship among features to solve the problem of feature selection. Statistical method used to discover most occurring topic in a document is known as topic modelling. Various topic modelling techniques are considered useful including LDA and LSA (Latent Semantic Allocation). A word is tagged according to its grammatical structure in POS tagging, same word may have different tag in some other context. Machine cannot perform any operations on text, so that converting text to vectors is necessary which is accurately done by word2vec method. Studies on app review analysis, used various approach to transform the text into features. Literature presents various approaches utilised to perform features related operations such as feature extraction, feature selection and feature prioritization. Feature extraction is the method to transform text to features of vectors using various techniques according to the scenario. However, feature selection is a task of selecting representative features based on some criteria from extracted feature set. Whereas, feature prioritization is a recent area of study, in which extracted or selected features or optimized based on some prioritization criteria. Selection and prioritization methods of features are not analysed as part of this review, hence the scope is limited to feature extraction of mobile app reviews. Studies included in this review, focus on the feature extraction methods used for app review text such as study S1 used BoW, TF-IDF, app category and review sentiments as classification features, study S2 proposed AUR-BoW for classification in comparison with BoW, TF-IDF, and CHI2. Similarly, the results against each study are presented in the Table 3.4.

3.3.5 Evaluation Metrics

After training a ML classification model for reviews, evaluating a model performance on an unseen data is a critical task. To achieve this objective, several measures can be applied to evaluate the performance of the classification model on the test set. While evaluating the model, a task is given to determine the possible labels for each sample in the test set. Later, the performance is estimated, by comparing the predicted outcomes against the actual labels for an unseen data. Now, this section describes the commonly used metrics for evaluation of ML models, to assess the quality of these models. To measure the performance of a model, construction of a confusion matrix is an easy approach to describe a ML model performance for a test set, where each row represents actual label instances whereas each column represents predicted label instances. TP (True Positives) are described as a number of correctly predicted classes of test samples that were true for both, the predicted labels and the actual labels. Similarly, TN (True Negatives) are described as a number of correctly predicted classes of test samples that were labelled negative and predicted negative. However, the FP (False positives) are incorrectly predicted cases, where the predicted label is positive while the actual label is negative. FN (False Negatives) are test samples that predicted incorrectly as negative while the actual label is positive. Based on these predictions, some SPM are used in the literature that consider them to evaluate the model performance, including the following. Precision (P) is interpreted as the total number of correct predictions (TP), dividing by the sum of; TP and FP, under specified labels as defined in equation 2.3. However, recall (R) is measured as TP over total number of instances under that category, defined in equation 2.4 . The weighted-average of recall and precision for each category is represented as F-measure, computed the f1 value independent to the category as defined in equation 2.5 **Accuracy:** is a simple metric to evaluate models, computing the percentage of correct predictions. Whereas, the imbalanced classes may lead to the biased accuracy results such that higher number of negative labels than positive ones, can mislead the accuracy by assigning that model a higher score which consistently predicts the negative label. In a real time situation, correct classification of infrequently occurring examples would be much more essential. In addition to these metrics the authors of study S1 performed multi-label classification and used the performance measures including subset accuracy, hamming score, hamming loss as well as macro averaged precision, recall and f-measure, explained through the following example. Consider S as an instance space with n number of data samples, and $L = \{\lambda_1, \lambda_2, ..., \lambda_k\}$ as a label set with k finite categories. Let Y represent the correct labels as a vector space of $Y = y_1, y_2, ..., y_n$ for every $s_i \in S$ instance, here $y_i \in \{0, 1\}, 1 \leq i \leq n$. To perform multi-label classification, a model M, predicts $X = x_1, x_2, ..., x_n$, where $x_i \in \{0, 1\}, 1 \leq i \leq n$, represents the predicted labels of every $s_i \in S$ instance. Based on these assumptions, look at the example containing three reviews as s_1, s_2, s_3, related to the labels $\lambda_1, \lambda_2, \lambda_3$ instance. . Three prediction scenarios are shown in this example, as review s1 is predicted incorrectly, review s_2 is predicted correctly, and review s_3 is predicted partially correctly.

$$Y = \begin{array}{c|ccc} & \lambda_1 & \lambda_2 & \lambda_3 \\ s1 & 1 & 0 & 1 \\ s2 & 0 & 1 & 0 \\ s3 & 0 & 1 & 0 \end{array} \qquad X = \begin{array}{c|ccc} & \lambda_1 & \lambda_2 & \lambda_3 \\ s1 & 0 & 1 & 0 \\ s2 & 0 & 1 & 0 \\ s3 & 0 & 1 & 1 \end{array}$$

Considering the above example to measure the performance of M, we have the following measures: Subset Accuracy (SA); Known as Exact Match (EM), is total number of correctly predicted data instances divided by the whole set of classified instances.

$$\frac{1}{n}\sum_{i=1}^{n} Y_i = X_i \tag{3.1}$$

Hamming Score (HS): proportion of accurately predicted data labels over total (actual and predicted) number of class labels discovered for the instance of data. Hamming Loss (HL): Measuring averagely the number of times when a class label

$$\frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap X_i|}{|Y_i \cup X_i|}$$

was incorrectly predicted. This measurement will consider the prediction error (Wrongly Predicted class labels) as well as the missing error (where the label was not even predicted), normalised over all available class labels and all available examples in a set [93]. HL = 0 indicates that prediction is correct and no error is present at the moment. Practically, an algorithm performs better when the Hamming Loss value is smaller. Evaluation measures used by the selected studies

$$\frac{1}{n}\sum_{i=1}^{|n|}\frac{xor\,(Y_i,\ X_i)}{|l|}$$

are described in the Table 3.3 against each study id.

3.3.6 Conclusion

Mobile app review analysis has emerged as a rapidly growing research area entailing various challenges and problems. Natural language interpretation and analysis of large number of reviews automatically is more challenging than manual analysis of small number of reviews. This study was conducted to review the available literature on mobile app review analysis systematically. As machine learning and natural language processing are getting significant attention and becoming dominant technologies, more research in this area is necessary.

We have considered machine learning approaches for mobile app reviews. In this regard, a classification schema was proposed in this chapter. Text pre-processing, feature selection methods, binary or multiclass classification, app distribution platforms and evaluation metrics that measure the results of models were considered.

We have selected 20 studies related to 5 research questions. Results revealed that supervised learning methods followed by unsupervised learning have mostly been used for review analysis of mobile apps. Tokenization, stop word removal, stemming and lemmatization are widely performed in text pre-processing phase. The findings show that research work in this domain is in initial stages, and there are no mature tools and techniques to pre-process the raw text. Most of the studies focus on manual labelling of reviews to train a model but completely automating the process of review analysis is an important aspect to be researched. Moreover, the feature extraction mechanisms used to extract features from user reviews which is an extra working overhead before applying machine learning. Therefore, in future researchers must focus on these aspects to affectively interpret the user reviews and extract useful information from them while reducing the human effort.

3.4 Research Gap Analysis

- Analysis of user reviews of mobile apps has begun to attract many researchers. Reviews contain variety of information reported by the users and can be useful to draw some conclusions. Most of the researchers mined app reviews to get information related to feature request and issues reported by the users. The number of studies mined NFRs are very limited. In addition, a single review is categorised under one class only and handled as a binary class or multi class problem by most of the studies. However, only one recent study on NFRs classification, performed multi-label classification.
- SML algorithms are the most explored ML dimension to classify mobile app reviews. Some studies applied USL to identify topics discussed in reviews. However, ensemble methods are explored by one study only whereas DL models are the least used for mobile app review classification.
- The automatic extraction of application features in user reviews does not reflect the nature of the user review text. As online application reviews have unique features of the text (including, abundant information, unstructured phrases, the short length, and colloquial language) it is required to create

a unique model, particularly for app store evaluations to obtain targeted application features instead of using conventional approaches and techniques established for diverse fields and contexts.

- Along with the user expressed opinion in the form of user reviews, other app features are also considered for better categorization of reviews. In addition, rating and app descriptions could be used to understand the user satisfaction level, the relation to the posted review, and authenticity level of a review.

Table 3.4: Summary of Studies Analysed in SLR

SID	Feature Extraction Methods	Evaluation Metrics	ClassType	App Features	Preprocessing Methods	Information Type	Algorithms
S1	POS-tagging	SPM	Binary	App Reviews, Description	Parsing, Stemming	Feature Request	RF
S2	Tf-IDF	SPM, Acc.	Multi-label	Rating,App Category	Tokenization, Stemming, Stopword removal, Lemmatization	Sentiment, NFR	SVM, NB
S3	BoW, topic-modeling	SPM, Acc	Binary	Reviews	Tokenization, Stemming, Stopword removal	Sentiment	NB, LDA

Continued on next page

Table 3.5 – continued from previous page

SID	**Feature Extraction Methods**	**Evaluation Metrics**	**ClassType**	**App Features**	**Preprocessing Methods**	**Information Type**	**Algorithms**
S4	BoW	SPM	Multi-class	Reviews, Rating	Tokenization, Stemming, Stopword removal	Feature Request, Bug report, user experience	NB, LR, SVM
S5	TF, Topic- Modeling	SPM	Multi-class	Reviews	Tokenization, cleaning, Stopword removal, normalization	Feature Request, Bug report	TF-ICF
S6	POS- tagging	SPM	Binary	App Reviews	Tokenization, Normalization	Feature Request	RF, SVM
S7	Tf-IDF	SPM	multi-class	App Reviews	Tokenization ,cleaning	Feature Request, Bug report	NB, SVM, LR, DT

Continued on next page

Table 3.5 – continued from previous page

SID	Feature Extraction Methods	Evaluation Metrics	ClassType	App Features	Preprocessing Methods	Information Type	Algorithms
S8	Tf-IDF, Topic-modling	SPM	Multi-class	Reviews, Rating	Tokenization, Stemming, Stopword removal	Feature Request, Sentiment, Bug, Experience	NB, LR, SVM
S9	n-gram	SPM, Acc.	Multi-class	Reviews, Rating	Tokenization, Stemming, Stopword removal	Feature Request, bug-report, NFR	RF, NB
S10	Chi- square	SPM	Binary	App Reviews	None	Bug report	SVM
S11	TF-IDF	SPM	Multi- class	App Reviews	Stemming, stopword removal	Sentiment analysis	GBRT

Continued on next page

Table 3.5 – continued from previous page

SID	Feature Extraction Methods	Evaluation Metrics	ClassType	App Features	Preprocessing Methods	Information Type	Algorithms
S12	CHI2, TF-IDF,BOW	SPM	Multi- class	App Reviews	Stemming, Stopword removal, Lemmatization	Functional, Non-functional	NG, J48, Bagging
S13	None	Other	Binary	App Reviews	Lemmatization	Functional, Non-functional	Self-training Rasco, Rel-Rasco
S14	Topic-Modeling	SPM	Binary	App Reviews	Stop-word removal, Lemmatization	Sentiment analysis, other	LDA
S15	Embed- ding	NDCG	Binary	App Reviews	Tokenization, Stemming, Stopword removal	App issues	k-means

Continued on next page

Table 3.5 – continued from previous page

SID	**Feature Extraction Methods**	**Evaluation Metrics**	**ClassType**	**App Features**	**Preprocessing Methods**	**Information Type**	**Algorithms**
S16	Topic- modeling	SPM	Multi-class	App Reviews	Tokenization, Stemming, Stopword removal	Feature Request, Info seeking, info giving, Problem, Other	Intent classifier
S17	Tf-Idf, BoW	SPM	Multi- class	App Reviews	Stopword removal, Stemming	Feature Request, Bug report, Other	GBRT
S18	Topic Modelling	SPM	Binary	App Reviews	Stopword removal, Lemmatization	Sentiment analysis, app issues, other	LDA

Continued on next page

Table 3.5 – continued from previous page

SID	Feature Extraction Methods	Evaluation Metrics	ClassType	App Features	Preprocessing Methods	Information Type	Algorithms
S19	CHI2, TF-IDF	SPM	Multi-class	App Reviews	Stopword removal, Stemming	Feature Request, Sentiment	NB, LR, DT, SVM
S20	topic-modeling	SPM	Binary	App Reviews	Tokenization, Stemming, Stopword removal	Other	EMNB

3.5 Summary

This chapter presented a Systematic Literature Review (SLR) focusing on Machine Learning (ML) based App Reviews Analysis and Classification (ARAC) methods. First a comprehensive SLR protocol comprising of RQs, inclusion and exclusion and quality assessment criteria, data extraction strategy, data sources, search strings and study selection process, is devised and reported. Then a taxonomy is developed to comprehend the anatomy of the proposed solutions, and selected primary studies (PSs) are analyzed according to the defined RQs and taxonomy. Then results have been reported according to each RQ being investigated. This SLR not only aims at identification of the existing classification models, but also serves as the basis for the development of the proposed classification model. After discussing the findings of the SLR, classification methods used by studies are listed in a Table 3.4. This resulted in a successfully employed classification approaches for ARAC. These approaches are implemented for multi-label reviews classification in chapter 5. This chapter also highlights the research gaps in the existing literature on ARAC. On the account of identified research gaps, researcher believe that there is a need of more effective classification model for ARAC, categorising reviews into multiple labels. The next chapter presents the overall research methodology of the current research.

Chapter 4

Research Methodology

This chapter describes the research methodology and enlists the research methods that have been used for this research. The description of selected approaches and research methods used to achieve the defined objectives is provided. It also describes the overall operational model for conducting the research and tools utilized to materialize the research.

In the section 4.1, a discussion of the methodology and philosophical aspects of the research methodology is provided. This research with predominantly qualitative data mainly adopts the positivistic paradigm. The data driven methodology is adopted as the specific methodology of this research. Subsequently, the section provides discussion on the research methods adopted for this research. Finally, the mapping of the research objectives, research methods, and deliverables is provided.

4.1 Research Methodology and Design

Before discussing methodological issues, it is very important to provide clarifications of terms method and methodology. Höst and Runeson reasoned that "methodology" states to the procedures and principles of systematic, thorough or firm processes that are applied to a particular discipline of the science [94]. Collis et al. in study [95] defined the word methodology as a general approach of the research practice from the theoretic supporting processes for data collection and the analysis.

The methodology offers a preliminary point for selecting appropriate makeup of concepts, theories, definitions, and ideas of the topics. Seeing this, all sorts of research trails a distinctive methodology changing from the study to study. In contrast, the method states the particular, tools, techniques, and means for data collection and the analysis. The authors in studies [94] and [96] recommended that two-factors determine the methodology of research: nature of data, and problem under thought for the research. Equally the ideas of methodology and data are devoted and interdependent. As per Williams [97], if data collected nature is found verbal, a methodology will be classified as qualitative; however, for numerical data, the methodology will be classified as quantitative. The problem under the thought for research likewise affects the choice of research mythology. The research design defines the plan to be tracked and implemented to attain the well-defined objectives of the research and addressing the research-based questions [98]. In spirit, research design can be considered as a framework or structure which provides guidelines for the collection and the analysis of data to respond to a particular research-based opportunity or problem [99]. Meanwhile, there is no particular and finest research design, the strategic excellent research design is emphasized to arise with a method that permits problems to be responded in the best way within the given limitations such as skill, budgetary and time limitations [100]. As described by Creswell in [101], the research design should start with subject selection and a research model. The standard research texts divided the methodologies into two essential models qualitative and quantitative, for the collection of data and for analysis. This research primarily embraces the Quantitative model with principally Qualitative data. Subsequently, the section provides discussion on the research methods adopted for this research. Finally, the main tests for validation of the research are discussed and justified.

4.2 Design Science Research

This research utilizes design science paradigm [43] to guide the whole research process. Design science is primarily a problem-solving paradigm having its roots in the engineering and what the Herbert Simon (Nobel Laureate) describes as the "sciences of the artificial" [102]. Figure 4.1 depicts the general process of the Design science research framework by [43] and its core activities. Accordingly, each of the design artefacts will further discuss the following activities.

4.2.1 Identification of Problem and Motivation

This preliminary activity refers to the identification and establishment of a specific research problem along with the justifiable value of the solution to the problem. This activity seeks to motivate the researcher and audiences to strive for a solution, agree on the results, and to understand the reasoning associated with the problem [43]. In this thesis, this activity is implemented by careful and systematic investigation of the body of knowledge related to the problem.

The first phase comprises of the literature review and preliminary research processes. In current research, Literature Review (LR) is conducted from multiple perspectives including existing app review classification methods (supervised, unsupervised, and semi-supervised), text-pre-processing techniques, feature extraction methods and evaluation metrics. The first phase of the LR activity is followed by an SLR focusing on ML-based MARC- models. As the model selection lies

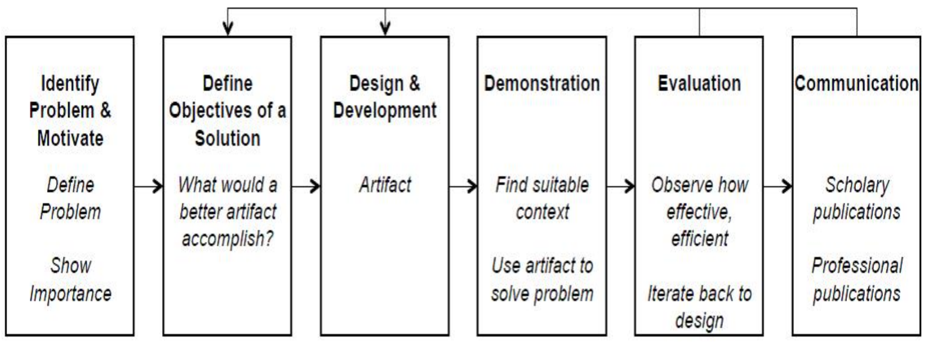

Identify Problem & Motivate
Define Problem
Show Importance
Define Objectives of a Solution
What would a better artifact accomplish?
Design & Development
Artifact
Demonstration
Find suitable context
Use artifact to solve problem
Evaluation
Observe how effective, efficient
Iterate back to design
Communication
Scholary publications
Professional publications

Figure 4.1: Peffers design-science research methodology [43]

at the heart of MARC methods, the SLR also seeks to identify the classification models reported by the existing MARC models. This phase mainly involves the planning, conducting and reporting phases of the systematic review focusing on the ML-based MARC models and evaluation factors used by these models. Then an initial list of ML-based MARC models is constructed, this serves as the basis of the Machine Learning algorithms comparison phase and the exploration of another dimension such as CNN.

Systematic Literature Review Process

A well-thought-out search plan is important to guarantee that related evaluation criteria and models might appear in the search outcomes. An SLR aims at identification, interpretation and evaluation of all existing studies associated with a particular area of interest. An SLR needs to follow an unbiased, fair, and accurate search plan. The search plan must guarantee the completeness of the search for review. In the present literature, no systematic review offers a rigorous assessment and examination of the existing research on ML-based MARC results. This research purposes to fill this gap by performing a systematic literature review using SLR process, as guided by Kitchenham [87]. This assessment process contains several steps that need to be conducted in a systematic and controlled way as demonstrated in Figure 4.2. These steps consists of the development of a systematic review protocol, conducting the review as per the well-defined protocol, examination and synthesis of the particular set of selected studies, reporting, and illustrating the outcomes. This complete analysis of the literature is followed by

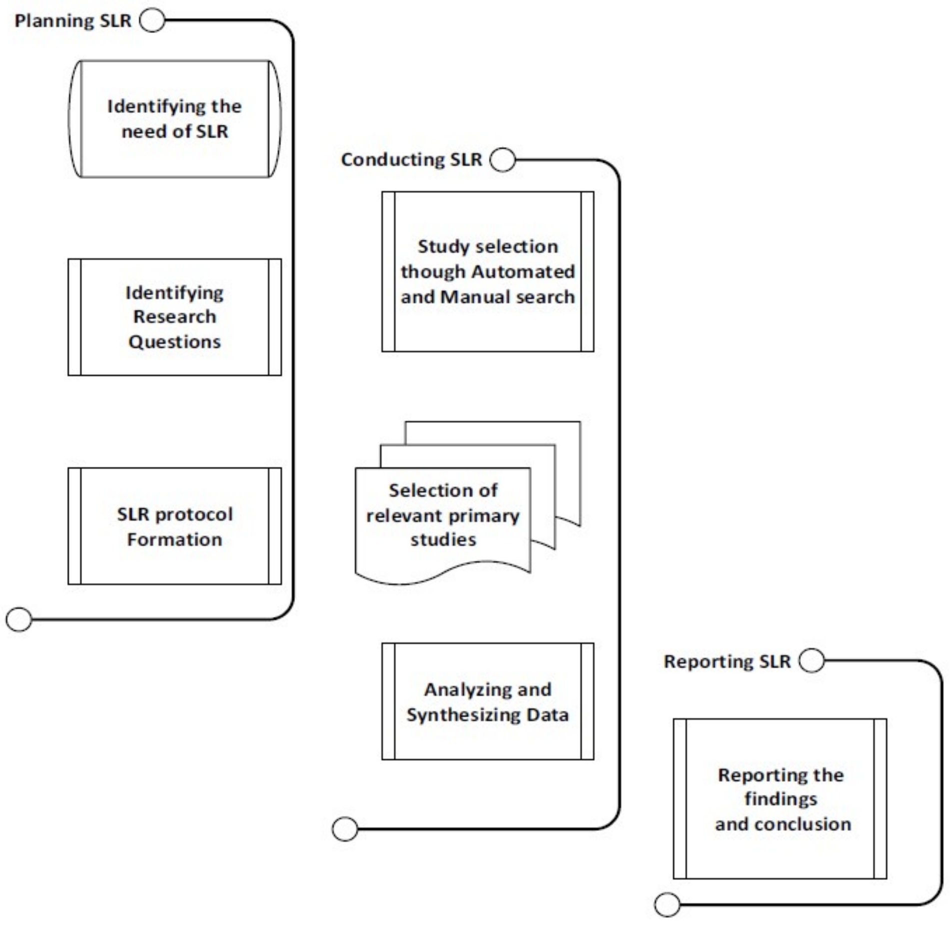

Planning SLR
Identifying the need of SLR
Identifying Research Questions
SLR protocol Formation
Conducting SLR
Study selection though Automated and Manual search
Selection of relevant primary studies
Analyzing and Synthesizing Data
Reporting SLR
Reporting the findings and conclusion

FIGURE 4.2: Systematic Process of Literature Review

identifying research gaps, formulation of the research objectives, and RQs to be responded to.

4.2.2 Defining Objectives of a Solution

This activity seeks to define specific objectives and solution requirements from the problem definition. To realize this activity, essential requirements of the solution are identified after establishing the research problem and reviewing the current state-of-the-art.

Three primary research objectives are intended to be achieved by this research. Here we provide a brief discussion on these objectives, the process of realization and research deliverable. Following objectives are intended to be achieved in this research:

- RO1: To gain insight in the state-of-the-art classification methods that need to be addressed in the context of app review analysis.
- RO2: To evaluate the performance of successfully employed ML models for multi-label review classification.
- RO3: To automatically categorise reviews into multiple labels

4.2.3 Design and Development

This activity states to the development of the artifact or solution such as methods, models, constructs. It also requires the preferred functionality of the fundamental ideas and architecture. This activity is assumed by a complete and thorough formalization of the essential ideas, algorithms and methods employed, associated with the particular solution.

As the existing studies emphases on the supervised learning models for the classification of text, principle guidelines, and essential phases of MARC model development are well-thought-out. Though there is a wide-ranging of text classification models, their implementation implies three core phases: train, test and validate. Training of a classification model involves input data, which is text data transformed into vectors that represent features of the input text. Testing of the trained model by a few hidden data instances to evaluate the learning accuracy of the model, then validation of model to predict the testing accuracy.

4.2.4 Demonstration and Evaluation

The demonstration of the effectiveness and applicability of the approach to resolve a particular problem is of equal importance. This activity pursues to deploy the proposed approach in experimentations, simulations, case studies, and other related activities to display the applicability and efficiency of the proposed approach. Typically, the demonstration of applicability and efficiency is linked with the subsequent evaluation. This thesis, apply the developed model on real world application data to its appropriateness.

The evaluation of the proposed model is essential activity that aims at observation and measurement, how well the proposed model aid the resolution of problem. This activity comprises, the objectives of the approach are compared with the real outcomes of the demo (experiment, case study, or simulation).

The proposed classification model is evaluated through a case study on real-world applications of Google play store. Supervised learning usually involves the training of models and future predictions based on these trainings.

Case studies are broadly used in the evaluation of such type of models in diverse areas as emphasized by many studies [103, 104, 32]. Therefore, the applicability of the MLRC-CNN could also be presented through a case study on real-world mobile applications.

4.3 Data Collection Process

The data is collected from the applications provided by Google play store.

Five categories are selected from a variety of different categories available on play store, then ten applications from each of five categories are picked up to extract user reviews. The most relevant 200 user reviews of each app are extracted through a scraping tool to form a dataset of ten thousands user reviews. Table 4.1 shows the categories and apps selection for reviews data extraction.

Table 4.1: List of Selected Categories and Applications

No.	Communication	Education	Health-and-fitness	Shopping	Entertainment
1	Imo	Udemy	Fitbit	Alibaba	Netflix
2	Viber	Google classroom	Google Fit	Olx Pak.	Iflix
3	True caller	Chegg study	Samsung health	Amazon	IMDB
4	Googledue	Skill share	Garmin connect	Daraz	Leisure music
5	Line	Academia.edu	Huawei health	Aliexpress	Youtube kids
6	Whatsapp	U-dictionary	Calorie counter	Souq.com	Amazon prime video
7	Messenger	Edx	Yazio	Noon shopping	Google play games
8	Cafe	Khan Academy	Myfitnesspal	Ebay online shopping	Talking tomcat2
9	Textmeup	Ted	Adidas runastic	Wish	Fake call prank
10	Coco	Course Era	Lifesum	Paytime	Yalla

4.4 Data Analysis Techniques and Methods

Since, user reviews are based on natural language text, to analyse the performance of MARC approaches the collected dataset of reviews is manually labelled according to the attributes of standard quality model of ISO25010, each review assigned a label from the label set defined in the Figure 4.3. This label set is selected from the already defined set of product quality and quality in use attributes by ISO considering the version named 25010. The quality attributes of a software product are considered as the NFRs of the product. User reviews are classified into categories of these NFRs as well as functional requirement such as a new feature request or suggestion to improve the existing features, manually by two annotators. The NFRs expand across a wide range of sub categories of ISO25010 quality model's attributes, treating these sub categories as a separate class is not a wise approach, even some of the product quality model attributes such as maintainability

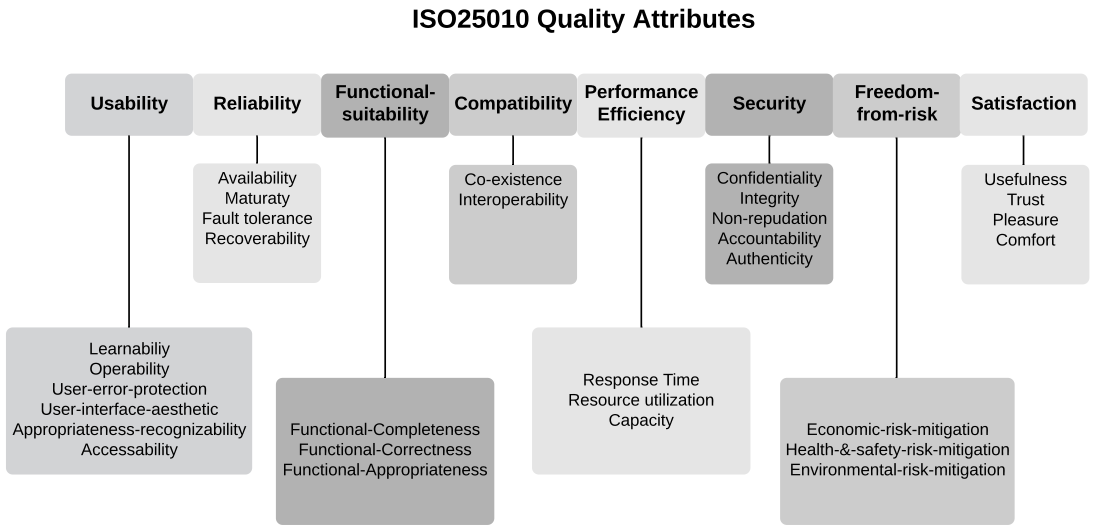


FIGURE 4.3: Selected Quality Attributes of ISO-25010 Model

and portability and some attributes of quality in use model such as effectiveness, efficiency, and context coverage are not considered as the scope of this study.

As the mobile applications developed, considering the specifications of a targeted platform such as Google play store, so the portability is not recognized as a NFRs concern for mobile apps. Similarly, maintainability is a concern after the deployment of the system, while mobile applications are published on a public platform and could only be maintained by the app developers so it is not app users concern.

In addition, 'quality in use' as the name signifies is the measure of quality by user after using the system. The reason of not considering the attribute 'context coverage' is that mobile apps are already dedicated and consist a specific set of features for that dedicated task, so the user does not need to measure either the app is fulfilling requirements of a particular context it is used in, or not. Therefore, NFRs are classified into some selected categories of quality model. These categories are defined as:

Usability:

defines a degree of use, a systems' efficiency, effectiveness and the satisfaction regarding its use. User reviews raise concerns of appropriateness recognisability, learnability, operability, user error protection, user interface aesthetics, and accessibility. As an example, we have a review "I loved this app in the past. That said I hate the new interface. It is ugly and I hate that I can't change the color theme Additionally it's too over simplified now".

Functional suitability:

include reviews that represent the issue of functional incompleteness, correctness and appropriateness. For example, "This app is for novices or idiots Multiple steps to do everything Sleep monitor is totally inaccurate."

Performance efficiency:

includes the reviews that have the issue of processing and response time, system or app's specified limits, and resource utilization. As an example we have a review, "Uses battery like crazy The app and the functions are good not a problem But the app uses battery in the background like crazy Takes between and every charge and I actually use the app less than minutes pr day The app has been denies to run in the background but it persists draining my phone daily Even when my Gearfit is not connected Please fix this Samsung."

Compatibility:

is a degree of a system or app to which it exchange information or share environment with other system or app, example includes, "Not working on android giving a msg Huawei requires a more recent version or later. While i just installed and updated it".

Reliability:

concerned about the reviews that raise issue regarding availability, recoverability, or trustworthiness of the app. We have an example of such review, "App not working properly gets disconnected with band automatically and need to connect manually every time can'tt use new faces also because asking for login with huawei I'd on OnePlus After Login still Same Issue".

Security:

this category have the reviews that are concerned about confidentiality of data, unauthorized access or modification of data, example includes, "GPS tracking feature violates privacy by keeping a record of where you go when then likely selling your data. Turn GPS off."

Freedom from risk:

this category includes the set of reviews that show any concerns of health risk, economic risk or environmental risk, as the example includes, "upgrade to premium is waste of money the sleep track record is still inaccurate waste waste waste".

Satisfaction:

the reviews in which users show their concerns regarding the level of pleasure or comfort they are achieving by using the app, as the example includes, "The app is awesome, simply love it".

At last, **Functional requirement:** contains the reviews that raise any suggestion of adding a new functionality or enhancing the existing feature, such as "Love this app it's so informative Adding food could be easier though Scanning barcodes for nutritional value and size portion would be ideal and easily accomplished because many other food tracking apps use this method Up your game and add the barcode scanning please."

As, the user reviews are in the form of raw text, so it needs to be cleaned to improve model performance as suggested by various studies [14, 40, 105].This preprocessing involves the removal of stop-words, tokenization, stemming and lemmatization. Pre-trained word embedding such as word2vec, gives better results while performing classification on text data [106, 107]. Evaluation of MLRC-CNN classifier, performed using the accuracy measure as an evaluation metric.

4.4.1 Finding a Suitable MARC Method

The selection of a particular MARC method to resolve the problem is not unimportant, and various features need to be measured in this regard. Such selection intensely depends on the nature of the standards, the number of replacements, and the context of the problem. To find the suitable MARC method to meet objectives of this research, a comparative evaluation of machine learning based classification models is conducted and then a deep learning based model is developed to classify multi-label reviews of mobile apps.

Number of Alternatives and Motivation

In the present state of the classification strategies, many more models are being presented, still, the number of replacements is not infinite and the choice of ML models for multi-label classification includes the finite number of replacements. However, in case of large number of reviews, the overall architecture of the model proposes an approach powered by multi-label classification. In this multi-label classification a single review contains more than one labels from the defined set of attributes. This helps in identification of all the requirements expressed by the user in a single review. When a review is lengthy and have multiple sentences, their categorisation is multi-labelled, which provides a finite set of models for classification.

Eventually, the classification alternatives are limited (see Figure 4.4). Thus, multi-label review classification (MLRC) is more suitable in the context of this research and effective MARC methods (supervised learning) can be considered.

Algorithms for Comparative Study

SML algorithms for App review analysis and classification are selected on the basis of the findings of the SLR. The more successfully employed algorithms to

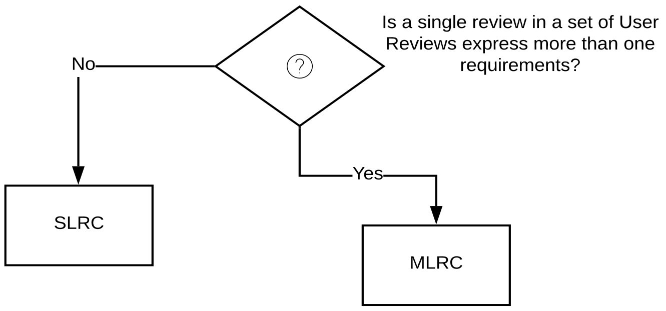

Is a single review in a set of User Reviews express more than one requirements?
?
No
Yes
SLRC
MLRC

Figure 4.4: Choice of SLRC or MLRC

classify user reviews are considered for the comparative study of MARC. Another justification of these particular algorithms is that, for classification of reviews these are the state of the art algorithms of ML such as NB, SVM, RF, and LR, and also applied successfully for other various problems.

4.4.2 Justification for Text Pre-processing Methods

A systematic review of ARAC revealed that there is a number of methods to process the user reviews, the raw text needs to be clean to improve model performance. This pre-processing involves the removal of stop-words, tokenization, stemming and lemmatization. Stop-words are most frequent words with slightly less or no meaning at all, removing them have almost no impact on the sentence but not removing them increase the number of counts of least significant words, results in inaccurate predictions. Moreover, the complete sentence is divided into token of words to identify the representative words from them. Therefore, it is worthy in terms of performance, to apply pre-processing on raw text.

4.4.3 Justification for Feature Extraction Methods

In the context of classification models, the feature extraction refers to the identification of the representative most frequent words in the data, used for model training. As identified from the systematic literature on ARAC, various feature extraction methods exist. To perform a comparative study on mobile app user reviews, two most simple and widely applied methods of feature extraction such as BoW and TF-IDF are used . However, in the context of deep learning the model use the embedding of vectors for training and processing. Words are transformed into matrixes of vectors and then these vectors are trained to generate embedding of words. There exist number of alternatives in the form of pre-trained word embedding, which are trained over millions of samples such as word2vec embedding, trained over Google news dataset of about 100 billion words. Using word embedding for deep learning model such as CNN gives better performance than the generated embedding [76].

4.4.4 Justification for Model Selection

Due to recent paradigm shift towards deep learning as discussed earlier, it is important to explore its applicability in the context of text classification, specifically for app reviews classification. CNN is the most promising algorithm for image classification and it also has adequate results in sentence classification [73, 76]. Therefore, in this research, a CNN based model for multi label reviews classification is built to explore its applicability in this dimension.

4.5 Mapping Research Question, Research Methods and Outcomes

Table 4.2 provides a mapping of the research objectives (ROs), particular research methods used to achieve these ROs and outcomes/deliverables of the each primary and secondary RO.

Table 4.2: Mapping of objectives, methods and Outcomes

No.	Objective	Research Method	Outcome
RQ1	To gain insight in the state-of-the-art classification methods that need to be addressed in the context of app reviews analysis.	Literature Review, SLR (Kitchenham's methodology)	Research gap Initial list of determinants for ARAC
RQ2	To evaluate the performance of successfully employed ML Model for multi-label reviews classification.	Comparative study of: Analyzing Android Health App Reviews to mine non-Functional Requirements	Study results signify that the machine learning models can successfully employed for classification of non-functional requirements from mobile app user reviews.
RQ3	To automatically categorise reviews into multiple labels.	Classification process of reviews	MLRC-CNN model

4.6 Summary

This chapter describes the overall research strategy and roadmap of the current research in order to achieve the defined research objectives. In this thesis, Quantitative research is predominantly used along with the contemporary methods like SLR. This chapter also enlists the specific research methods used at different stages, along with the outcomes or deliverable. Justification for certain research methods have also been discussed in this chapter. The next chapter focuses on the comparative study of state of the art machine learning algorithms for mobile app review classification.

Chapter 5

Analyzing Android Health App Reviews to Mine Non-Functional Requirements

This chapter aimed at answering to RQ2 (defined in chapter 1) on mining non-functional requirements from mobile app reviews using machine learning models and analyzing the performance of classifier using different feature extraction methods. The organization of this chapter is arranged in the five sections, where section 5.1 introduction of the study. Section 5.2 defines the research design of the study including data collection process. Section 5.3 is related to manual review classification to perform the analysis on reviews. Section 5.4 describes the automated methods of review classification to classify user reviews into multiple categories. Section 5.5 describes the threats to validity. Finally, section 5.6 concludes our findings and suggest some future work. Summary of this chapter is described in section 5.7.

5.1 Introduction

User reviews are natural language text, contain sentences which are formed by sequence of words. Review sentences are raw text, contain extra symbols and word that does not contain any information, overhead in further analysis. To remove such useless information, text pre-processing techniques are applied which includes removing stop-word, tokenizing sentences, removing punctuations and symbols of any kind [108]. The review sentences are split into word of tokens to get information from them. These words represent characteristics of review sentences, collecting them called feature extraction. To extract those features from reviews text, various methods have been applied and their extensions are proposed. However, the state of the art and most popular feature extraction methods are BoW and Tf-IDF [107, 109].

Since, classification algorithms cannot be implemented over the raw text of reviews, so that the popular feature extraction methods are used to define feature set for training a ML model. There exists a number of studies at user reviews classification using various machine learning algorithms and feature extraction methods. However, their performance still needs to be evaluated for multi-label review classification problem, to answer the two sub-questions of RQ2 defined as: RQ2.1: To evaluate the performance of simple Bag-of-Words and TF-IDF methods with ML algorithms on multi-label reviews dataset?

5.2 Research Approach

As, the research method followed in this thesis is data-driven. To answer our first research question RQ2.1, we crawled the dataset and labelled the user reviews to compare the performances of supervised classification models with the most popular feature extraction methods. Various factors could impact the model performance such as reviews language, model features, and dataset quality. Since we were interested in investigating the potential performance impact of basic feature extraction methods on supervised ML classifiers, when app features are extracted automatically from English app reviews.

5.2.1 Data Collection

Our data collection steps are following:

- Selection of apps and app category
- Selection of user reviews

Selection of apps and app category

At this data collection step we were concerned with the application category to sample our data from. We have selected the category of Health and fitness apps listed on Android app store for this comparison of classification models using distinct feature extraction methods. Therefore, we have selected 5 health applications from various top health apps of Google play store mentioned in the Table 5.1.

Table 5.1: Applications selected for comparative study

Applications	**Number of reviews**	**App category**
Fitbit	200	Health & Fitness
Huawei health	200	Health & Fitness
Garmin connect	200	Health & Fitness
Google fit	200	Health & Fitness
Samsung health	200	Health & Fitness
Total	1000	

Selection of user reviews

Reviews data is collected from the apps selected in the previous step, using a scraping tool. Two hundred top most relevant user reviews are extracted for each app selected (see Table 5.1). The 'most relevant' filter to crawl reviews returns the more informative reviews, presenting the user's concerns more specifically. This filter setting also reduce the chance of those reviews selection that are just posted to increase the app rating, such as spam and fake reviews.

5.3 Manual Classification

User reviews are classified into categories of functional and NFRs manually by two annotators, according to quality model defined in Figure 4.3. The NFRs expand across a wide range of sub categories of ISO25010 quality model's attributes, treating these sub categories as a separate class is not a wise approach, even some of the attributes are not considered as the scope of this study (i.e. maintainability, portability and some attributes of quality in use model, effectiveness, efficiency, and context coverage). Therefore, NFRs are classified into some selected categories of quality model.

5.3.1 Results and Discussion

Reviews are manually classified by two annotators, both are master students, and verified by a PhD professor, in the period of 15 days. Classification results of our manual analysis are illustrated in Figure 5.1 and Figure 5.2. Figure 5.1 shows the distribution of each label in the reviews dataset. Such as the outcome of our reviews analysis depict that out of 1000 reviews: 19 percent (281) raise satisfaction related issue, in which, either the user liked or loved the app or simply disliked it. Followed by 18 percent (274) functional-suitability requirements in which the app is not functioning accurately according to what it is designed for.

15 percent (221) of the user reviews raised compatibility related issues, usually Health-and-Fitness apps are connected to smart watches and have to sync the records. Reliability have 14 percent (206) reviews of the data while 13 percent (190) reviews are related to functional-requirements such as feature request or suggestion to update a feature. Reviews related to usability are 11 percent (156),

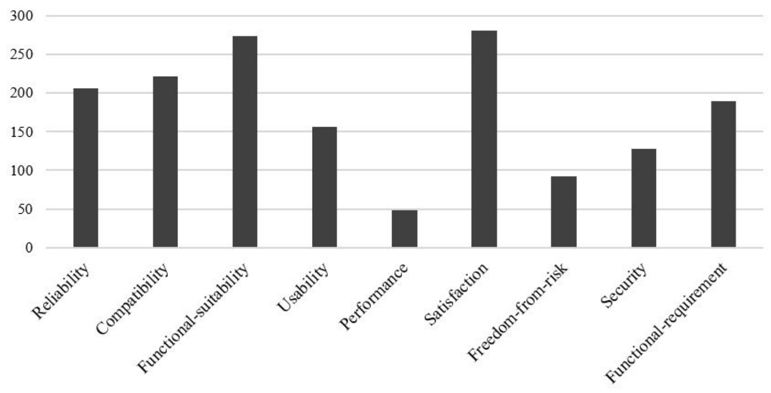


Figure 5.1: Reviews distribution over the requirement categories

to freedom-from-risk are 6 percent (92) while reviews related to performance are 3 percent (48) and the last 1 percent (12) raise security issues.

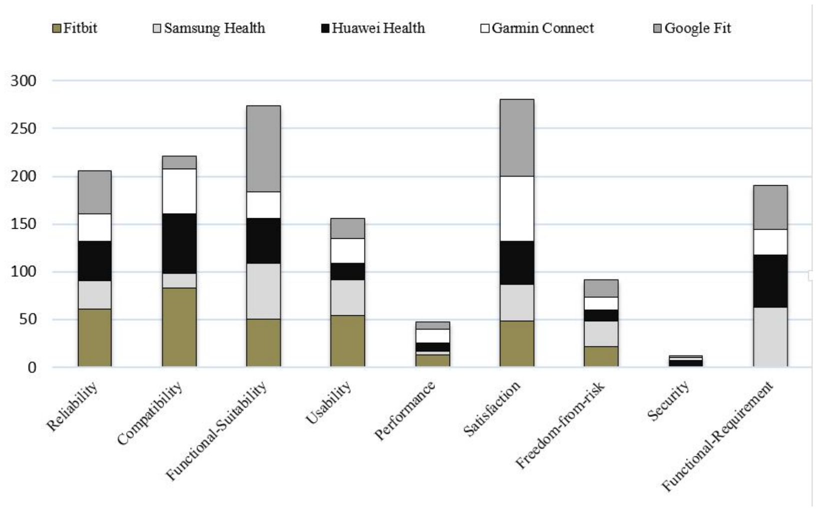


Figure 5.2: Category wise Reviews Distribution over Applications

Figure 5.2 illustrates the distribution of NFRs and functional requirements over the app category, raised in the reviews of applications. The results reveal that user reviews of Fitbit app are mostly fall under the category of compatibility, syncing is the main issue reported by users of Fitbit app. Similarly, majority of user reviews in Huawei health app are also related to compatibility. Whereas, Samsung health app suffers from functional suitability the most, where users are raising concerns about the correctness of features, in addition to new feature demands. Majority reviews of Garmin connect report the issue of satisfaction, users are getting disappointed with functional completeness and correctness of the app. Furthermore, most of the reviews of Google fit raise requirement related to app functions, as the users were not satisfied with services. As our results showed, out of selected set, large number of reviews are related to satisfaction, functional suitability and compatibility. This led to the findings, health applications usually suffer from functional accuracy and correctness which makes these apps less trusty. Finally, these results are realised by analysing Android Health Apps and results may vary when the analysis will be performed on other apps of similar category on app store reviews.

5.4 Automated Classification

First we discussed the results of manual review analysis, now we will explore the performance of classification techniques that automatically classify the reviews into the defined categories of NFRs along with a functional requirement category. Just as we described earlier that our problem is of multi-label nature, so in automated analysis we can classify each review under more than one categories. For more formal representation, we can define multi-label problem as: Let A be a label from a set of multiple labels A1, A2. . . An, let x = output space and y = input space, x, y defined as subset of label-set. For multi-label classification, a task T can be given as $T = (x_1, y_1)...(x_n, y_n) \subset x * y$ where x_1, y_1 is categorization/classification of data instances $x_1 \in x$ under a label $y_1 \in y$

5.4.1 Multi-label Classification

Previously, reviews were categorised usually into two categories of yes/no, and classified using simple approach of binary classification, where yes represented as 1 and no represented as 0. However, reviews having multiple categories belongs to multi-label classification problem and to solve this problem simple binary method would not work.

Overall, there are two main classes of multi-label methods, the one class of methods called “problem transformation” approaches which convert the multi-label problem into multiple single-label classification problems. The second class of methods adapts a specific algorithm for single-label classification problems to directly predict multiple labels. This class of methods is generally called “algorithm adaptation”. To classify reviews into multiple categories the approach of Problem Trans-

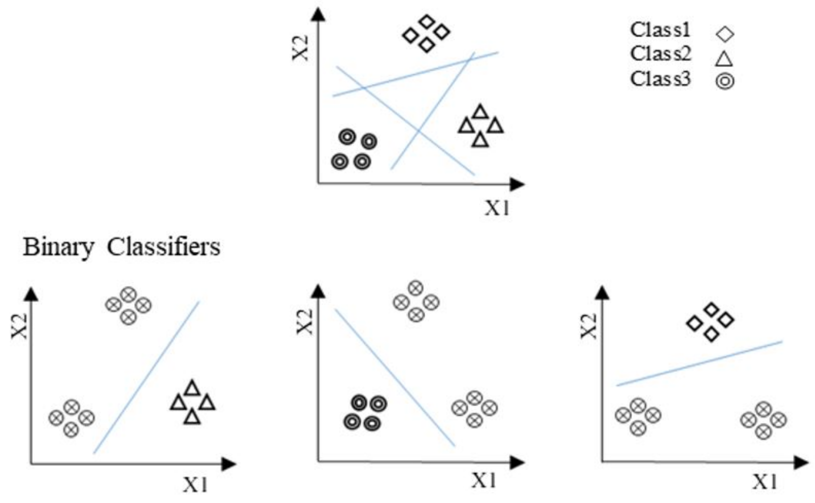


FIGURE 5.3: Illustration of the Multi-label One-vs-Rest Classifier

formation is adopted, the method of one-vs-rest classifier, in which multiple binary classifiers implemented on each category using pipelines to predict correct label for each review. In OvR or OvA labels are considered independent from each other, decomposing the problem with multiple labels into multiple binary problems. In other words, it is defined as, let a set of labels (A_1, A_2, A_3) and an input $I \in (A_1, A_2)$ OvR learns a binary model for each label, considering A_1 as correct label and the rest of the labels (A_2 in this case) as incorrect, then considering A2 as correct label and the other as incorrect [110]. Results of the classifier are obtained as union of predictions.

5.4.2 Text Pre-processing

User reviews are natural language text, contains additional information, use text reduction techniques to remove extra information, thus remove the words that might affect the predictions of a classifier. We have a strategy of stop-word-removal, to remove the common words such as 'is', 'an', 'the', carrying generalizable information and do not effect semantic meaning of reviews. Moreover, we have other text reduction techniques, lemmatization and stemming, for a term lemmatization consider the linguistic context and reduce the inflectional form of a word to its base form, for example, for word 'better' it consider both the words 'good' and 'better'. Whereas, stemming is a heuristic process that stem the word or cut it to its root word, for example, syncing and synchronize both will stem to 'sync' [111]. Therefore, both strategies are implemented to get root form of a word. Relevant user reviews are tends to be longer, splitting them into token of sentences or words is preferred to get better granularity to classify user-reviews into NFRs and functional requirements. Furthermore, to perform classification more easily and accurately it is important removing punctuations and other unusual formats.

Feature Extraction

Words in reviews text contain information that represents characteristics of reviews sentences, extracted to perform analysis, and collecting these informative words is called feature extraction. As the literature identify that various methods have been applied for feature extraction and their extensions are also proposed. However, the state of the art, simple and most popular basic feature extraction methods are BoW and Tf-IDF.

Bag-of-Words

The simplest approach to convert text into structured features is using the BoW approach. BoW simply breaks apart the words in the review text into individual word count statistics (see Figure 5.4), uses a dictionary of unique terms extracted from the corpus of user reviews. Moreover, it consider feature weights to maintain a feature matrix, storing the frequency or count of a term (TF) appeared in a document of review sentences. Later, these features are used for object classification. BoW, often called Word n-gram, is a straightforward feature extraction

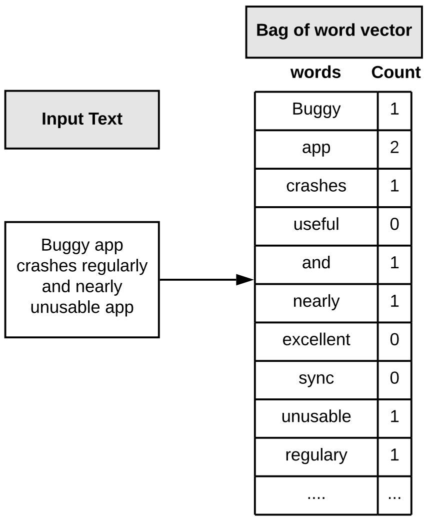


FIGURE 5.4: Bag-of-Words Representation

method [107] that returns a contiguous sequence of n-words from a given review sentence. For instance, 1 to 2 n-grams of a review sentence “plz fix this app” are ’plz’,’fix’,’this’,’app’,’plz fix’,’fix this’,’this app’. In this method, first a dictionary is built by extracting a contiguous sequence of n-words from the training corpus. Then, a feature matrix is maintained in which each row represents a review that stores the frequency of each n-gram in that review. The method doesn’t require any external linguistic tool for its usage, which makes it very appealing to practitioners. BoW features are useful when characterizing the review sentences into sentence types. For instance, the words “interface” and “look” mostly appear in review sentences belong to type usability; while the words “response”,“crash”,and “plz fix” appears in review sentences where users mention a reliability issue in an app. The study of Maalej and Nabil [14] used BoW features to classify a full review text into different categories such as feature request and bug report etc. Similarly, we used the features to classify reviews at the review level. Obviously, a full review contains more information and present more than one class label for a single review. We believe that full review sentences contain enough lexical information to classify them correctly under multiple categories.

Term Frequency - Inverse Document Frequency

TF-IDF use the same textual features (i.e., words) as BoW does, but different from BoW as it also consider the word frequency count in documents. TF-IDF technique is used to calculate the importance of a particular word in a document [57]. TF-IDF combines term frequency with inverse document frequency to get the weight of textual features, which is influenced by the frequency of the term in the user reviews of the dataset. For example, the word 'app' would be penalized because it occurs in many user reviews whereas an uncommon word like 'omission' would be given a higher weight. TF-IDF of each word is defined in an Equation 5.1:

$$TF - IDF(word_{i,j}) = f_{i,j} * log(totaluserreviews/totaluserreviewswithwordi) \tag{5.1}$$

Where $f_{i,j}$ denotes the frequency of a word i in a user review j. We use total user reviews that contain word i as denominator since some words might appear frequently in many user reviews, which means that these words contain less type information. On the other hand, IDF alone does not consider type information, and it cannot handle the situation that a word appears in many user reviews of the same type [14].

5.4.3 Classifiers

A classifier is a model trained and tested over a particular set of input features extracted from the user reviews. These models are based on ML algorithms and used for automatic classification of user reviews. Machine learning has various dimensions including: supervised learning, unsupervised learning and semi-supervised learning [71]. As the scope of this thesis is limited to supervised learning to perform classification of reviews, so the most widely used algorithms for classification are considered for this comparative study. These algorithms are discussed as follows:

Naive Bayes: NB is based on Bayes' theorem, yet simple, linear probabilistic and efficient classifier [70]. It consider conditional independence among the present model for text classification assume features as separate words from corpus of the text. To classify user reviews, NB has been recognized as a competitive classifier in measure of its performance, predicting multiple categories of reviews with good accuracy rate [8, 14].

Support-vector-machine: SVM is an algorithm for regression and classification analysis, a supervised machine learning approach [68].It works for the patterns, identified in dataset, and these patterns are linearly separable. Multiple hyper-planes are used to separate the patterns and then the optimal hyper-plane is selected. Hyper-planes are critical instances in the given training set and support vectors' location used to determine the position of dividing hyper-planes. To classify the data, SVM map the input feature vectors into an n-dimensional space and determine the place of new coming data instances on the defined hyper-planes. Empirically, SVMs has shown to be effective in the high dimension feature space and sparse instance vectors. Therefore, it has also shown to perform well in review classification [32, 17].

Random forest: Random forest also called Bagging algorithm, an ensemble classifier, combine bootstrap, decision tree, and voting methods. Multiple decision trees are constructed by picking up data randomly [72]. It is a powerful machine learning algorithm which perform very well for categorical variables.

Logistic regression: A supervised learning algorithm is popular for classification problems, based on the probability. It consider the linear combination of various features and use review-specific parameters to determine the probability of a certain review type [112]. Furthermore, it has shown to perform well when the target variables are categorical.

5.4.4 Evaluation strategies

To evaluate the underlying classification models, standard measures have been used including precision and recall along with the accuracy rate. The result of multi-label classification can either be completely accurate, partially accurate, or incorrect completely. Consequently, the standard accuracy and recall measures, normally used in the binary classification tasks, require to be complemented by additional metrics that can be used for partial accuracy. To report for such statistics, we used them in addition to the above-mentioned metrics, we calculate the macro-averaged F-Measure ($Fbeta$), Recall (R) and Precision (P) values for the model. These metrics are calculated autonomously against each classification label and averaged over all the labels [113]. We reported the results by considering each category's macro average precision, recall and f1 results, using a Geometric Mean formula to compute the average values.

$$gm = \sqrt[n]{x_1 * x_2 *, \ldots, * x_m} \tag{5.2}$$

Where, x_1, x_2 till m, are values that multiplied together, rooted with n, number of m samples.

5.4.5 Classification Setup

A classification model is trained over the dataset defined in the Table 5.1, to classify user reviews into multiple categories of requirements, through one-vs-rest classification method, using Python libraries. Sci-kit learn is a library, composed of a wide range of machine-learning algorithms for various classification problems [114].

First, split function of 'Scikit-learn' is used to split the data into the ratio of 70:30 for the purpose of training and testing in classification model. Random-number-generator is used to randomize the dataset, the parameter is set to 42. Input reviews text is pre-processed using plenty of functions, as we used: 'stopwords' for removal of stopwords, token-pattern' to tokenise the data and 'lowercase=true' to convert the data into lowercase. Furthermore, NLTK's 'wordnetlemmatizer' and 'porterstemmer' are used to lemmatize and stem the input data tokens [111]. Four ML classifiers are then modelled and trained on the dataset to predict the classification output. Performance of these models is measured against the test set, using 'Scikit-learn's' library to generate the classification-report. This process is executed for each requirement category defined in Figure 4.3. The macro average precision, recall and f1 results are calculated and presented in Table 5.2 for each requirement category. The macro average f1 score and subset accuracy of each model with word n-gram of one is averaged using geometric mean value, presented at the end of Table 5.2. The geometric mean of f1 and accuracy score for each classifier is calculated by multiplying each value and taking a root according to the number of counts for each value. The reason to consider a geometric mean rather than calculating simple average using arithmetic mean is that the GM is less affected by smaller and larger values and works better for percentages.

5.4.6 Results and Discussions

Results of macro average precision, recall and f1-measure against each category for four algorithms under one-vs-rest classification method using word n-gram of 1 has shown in Table 5.2.

This section presents classification results for RQ2.1 with uni-gram features. Along with the performance results for each category, Table 5.2 also shows the mean average results at the end of the table against each classifier and the feature strategy. The mean average performance for requirement types such as reliability, compatibility, usability, functional-suitability, performance, security, freedom-from-risk,

Table 5.2: Classification performance of the models based on unigram features for all Requirement types

Reliability								
Classification Setting	**BoW**				**TF-IDF**			
–	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.61	0.64	0.74	0.62	0.65	0.4	0.66	0.62
recall	0.59	0.67	0.52	0.64	0.51	0.5	0.51	0.63
f1-score	0.6	0.65	0.48	0.63	0.46	0.45	0.48	0.62
Compatibility								
Classification Setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.71	0.65	0.86	0.71	0.77	0.4	0.9	0.73
recall	0.7	0.71	0.56	0.73	0.56	0.5	0.51	0.77
f1-score	0.71	0.66	0.57	0.72	0.56	0.44	0.46	0.75
Functional-Suitability								
classification setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.66	0.64	0.58	0.6	0.61	0.85	0.55	0.63
recall	0.62	0.63	0.51	0.59	0.51	0.51	0.5	0.61
f1-score	0.63	0.63	0.46	0.6	0.45	0.43	0.43	0.62
Usability								
Classification Setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.62	0.61	0.42	0.57	0.42	0.42	0.42	0.6
recall	0.56	0.62	0.5	0.57	0.5	0.5	0.5	0.58
f1-score	0.57	0.61	0.46	57	0.46	0.46	0.46	0.59
Performance								
Classification Setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.47	0.53	0.47	0.67	0.47	0.47	0.47	0.58
recall	0.5	0.54	0.5	0.59	0.5	0.5	0.5	0.53
f1-score	0.49	0.53	0.49	0.62	0.49	0.49	0.49	0.53
Satisfaction								
Classification Setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.52	0.6	0.73	0.54	0.38	0.38	0.68	0.56
recall	0.51	0.62	0.55	0.54	0.5	0.5	0.52	0.57
f1-score	0.52	0.61	0.54	0.53	0.43	0.43	0.47	0.56
Freedom-from-risk								
Classification Setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.5	0.49	0.45	0.51	0.45	0.45	0.45	0.56
recall	0.49	0.48	0.5	0.51	0.5	0.5	0.5	0.53
f1-score	0.49	0.48	0.47	0.5	0.47	0.47	0.47	0.53
Security								
Classification Setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
recall	0.5	0.49	0.5	0.5	0.5	50	0.5	0.5
f1-score	0.5	0.49	0.5	0.5	0.5	0.5	0.5	0.5
Functional-Requirement								
Classification Setting	**BoW**				**TF-IDF**			
	LR	NB	RF	SVM	LR	NB	RF	SVM
precision	0.77	0.71	0.83	0.7	0.89	0.89	0.89	0.75
recall	0.68	0.73	0.55	0.67	0.51	0.51	0.53	0.68
f1-score	0.7	0.72	0.54	0.68	0.45	0.45	0.5	0.7
Average precision	0.587498	0.592388	**0.599038**	0.597613	0.549743	0.502502	0.590893	**0.609855**
Avergae Recall	0.567362	**0.603889**	0.520577	0.589159	0.509689	0.502205	0.507675	0.594976
Average f1-score	0.573249	**0.592698**	0.499738	0.590022	0.473134	0.457186	0.472856	**0.59513**

satisfaction, and functional-requirement, illustrated separately as the interesting information to improve an app is expected from these categories. The first four columns shows the results of uni-gram BoW with four ML algorithms. The last four columns present results of the TF-IDF with word unigram for four ML algorithms. At the end, the last three rows present the averaged results calculated as geometric mean for each requirement type, best results are represented as bold in Table 5.2. At last, over all requirement types on ML classification models, the NB model with features (unigram-BoW) achieved the best recall on average. Whereas, the best average score of f1 and highest precision is achieved by SVM with TF-IDF. Over all NB and SVM with BoW features both performed better than other algorithms with very slight differences in their average results. RF also achieved good precision with both BoW and TF-IDF but low f1-score. Similarly, LR after SVM and NB achieved promising results in both settings. Hence, our analysis shows that user reviews can be automatically classified under multiple categories of requirements. In general, SVM with TF-IDF followed by NB with BoW performed very well to predict labels for multiple categories. Visual representation of the results is illustrated in Figure 5.5. In this thesis, we addressed the challenge of

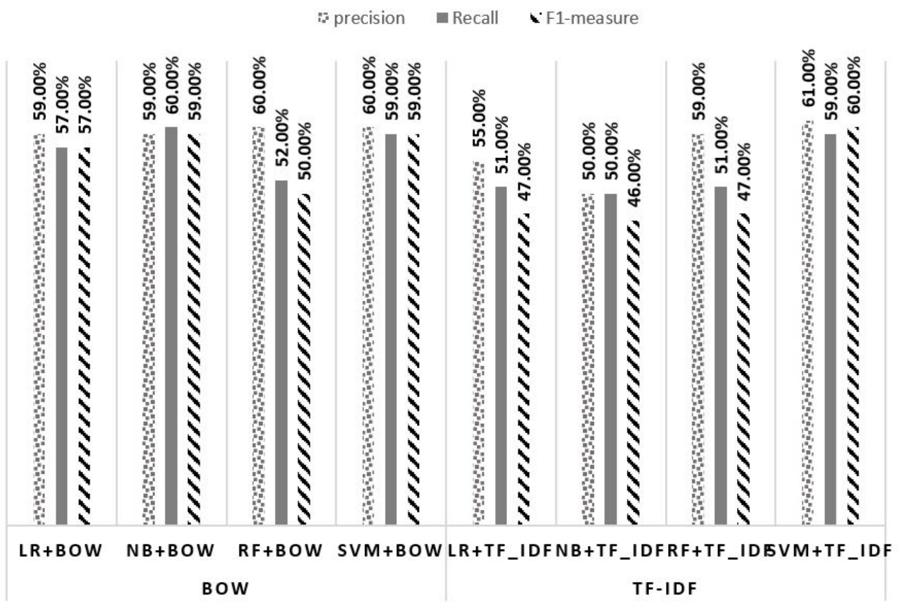


Figure 5.5: Automated Machine Learning Analysis Results

review level analysis by categorising reviews into multiple categories. Considering a full review for classification overcome the issue of context coverage. However, the features identified from the dataset impact the model performance, so that the more accurate the feature set is the more better performance we get. The feature extraction methods we used have some limitations such as BoW approach has the disadvantage of not considering the co-occurrence of words, handle each word independently, and create the vocabulary of known words only. On the other hand TF-IDF overcome some limitations of BoW approach but still lack in some characteristics such as it gives less importance to the most occurring words and not consider type information (repeated document of same type information). So, there is a need to develop a model that better handle the feature mapping problem than the available feature extractions methods.

5.5 Threats to Validity

This section defines some threats to the validity of this study, including constructive, internal and external threats, according to the guidelines of [115].

Construct validity refers to the theoretical constructs whether they are measured and described correctly. In this study, the threat related to construct validity includes whether the reviews are classified with correct labels for analysis. The general understanding of the requirement types is achieved by the definitions of the requirement categories in ISO25010 standard quality model [55]. But, this cannot assure that researchers fully understand the various NFR types the way they defined. Another validity threat is related to the assigned labels to the review sentences, either the review actually belongs to that label or not, and any conflict on the class labels were discussed and resolved by a PhD researcher.

Internal validity refers to the multiple categories of review sentences, whether the category with fewer number of reviews generate imbalanced class problem [116]. To mitigate this impact, we evaluated results independent of categories. But, we still have to further analyse the impact of imbalance class and mitigate it more accurately.

External validity focuses on the finding of the study, a few of the reviews related to the requirement category of security and performance. Thus, an empirical study that we plan to conduct with large-scale dataset from various app categories may improve the external validity threat.

5.6 Conclusion

In this study, we worked on the problem of NFRs identification and classification from mobile app user reviews. We divided our analyses into two phase, the first phase related to qualitative analysis of one thousand user reviews of five android health-and-fitness applications. The analysis is performed manually by the annotators of software engineering field, to classify reviews into eight categories of NFRs and one category of FRs.

In the next phase, we computed the performance of various classification approaches to automatically classify user reviews into various types of NFRs. The one-vs-rest classification technique is implemented with feature extraction methods such as Bag-of-words and TF-IDF. In addition, the text pre-processing methods are also considered as classification features and applied to the dataset of one thousand reviews. NB, LR, RF and SVM are evaluated to solve multi-label classification problem, implemented using a pipeline structure with one-vs-rest classification method. SVM achieved the highest F-measure of 89 percent, when implemented with bag-of-words. We have evaluated the performance independent of each class label and averaged the results.

Results of our analysis in this study can be exploited to facilitate app developers for understanding user concerns and complement the requirement elicitation for developing a new app and improving the existing health applications. This paper is the initial work in the domain of user reviews analysis, our future objective is to evaluate the results of this paper on a large-scale dataset, then take further decisions based on the outcomes and to explore other dimension such as deep learning models for classification.

5.7 Summary

In this chapter we presented the implementation of ML models for ARAC using BoW and TF-IDF approaches to extract word unigram features for training a classifier. First, it defines the approach used for data collection and preparation. Then, the reviews are analysed manually to show results of qualitative analysis. Later, an automated classification approach to classify reviews using ML models developed and implemented. Results showed the significance performance for SVM and NB algorithms. The next chapter will focus on the devising of an MLRC-CNN model.

Chapter 6

Proposed Classification Model

The purpose of this chapter is to establish the problem highlighted in Chapter 1, and to present the MLRC-CNN for MARC. This chapter is organized into three main sections. Section 6.1 presents the architecture of the proposed classification model. Subsequently, section 6.2 presents preliminary concepts and research plan for designing MLRC-CNN model. Lastly, section 6.3 presents the details on research methods and multiple phases of the proposed solution.

6.1 Architecture of the Proposed Model

This section presents a high-level architecture for the implementation of proposed classification model. This architecture mainly comprises of reviews collection, preprocessing, and Classification process (see Figure 6.1). The reviews collection phase get the reviews of selected apps from play store, using a web scraping tool named webharvy. Although, most of the time reviews are collected through the API or a scraping code implementation. However, considering a tool for crawling reviews, save extra time and effort and provide the required set of reviews. Integration of the collected reviews includes the removal of the sentence separator dot (.), to perform analysis on a whole review rather than splitting a single review into multiple sentences. Pre-processing is required for a textual data to make it clean and ready

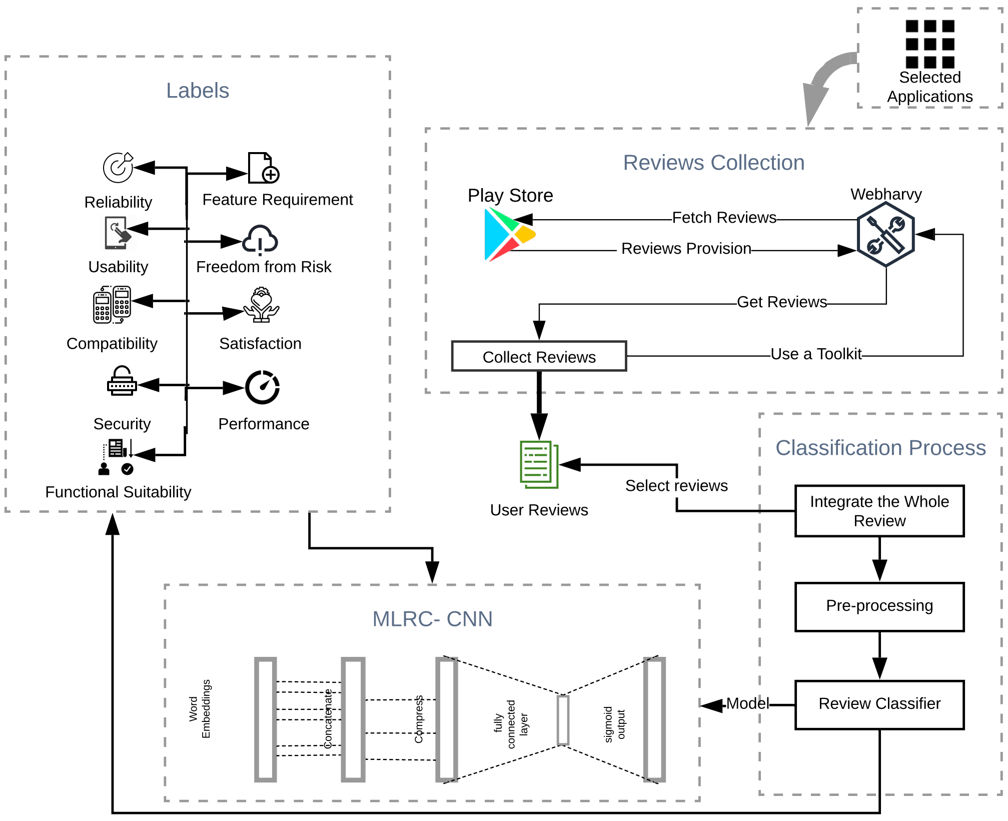


FIGURE 6.1: Architecture of the Proposed Approach

for analysis at later stages. Extra symbols and article words cause extra computation complexity, while do not have any positive impact on the model performance. Thus, the removal of punctuations and extra symbols, reduce any unproductive computational cost and prepare the data for analysis and classification. Strategies including, stop words removal, lemmatization and tokenization are used to process raw text of user reviews. Than a CNN based multi-label review classifier is implemented for classification of reviews into defined set of labels. There are three settings for a CNN classifier for text data classification, the one is static, the second is random, and the third is non-static. For static and non-static, word2vec words embedding are used as pre-trained embedding of vectors. While in random CNN settings, pre-trained embedding are not used, instead the model is trained over the randomly initialized word vectors and updated during training.

6.2 Preliminary Concepts

6.2.1 Word Embedding

Textual data needs to be converted into numerical format to perform operations on words of sentences, transforming into embedding of vectors of floating point values that will be used to initialize the CNN classification model. Generate embedding from the given dataset or use pre-trained embedding to classify user reviews into multiple categories. Pre-trained embedding, trained over Google news dataset of about 100 billion words, named as word2vec, used in the model to encode the input data into vectors, improving performance.

6.2.2 Convolutional Neural Network

Neural Network

Neural nets are a means of doing machine learning, in which a computer learns to perform some tasks by analysing training examples. Usually, the examples have been manually labelled in advance. A neural net consists of thousands or even millions of simple processing nodes that are densely interconnected. Usually, neural nets are organized into layers of nodes, and they are "feed-forward," which means data moves through them in only one direction. The first trainable neural network, the Perceptron, uses weights and bias to predict the correct label. Neural networks are widely adopted for natural language processing and text classification [117].The recent resurgence in neural networks is the deep-learning revolution. Within natural language processing, much of the work with deep learning methods has involved learning word vector representations through neural language models [107, 118] and performing composition over the learned word vectors for classification [108].

Convolution

A convolution is a mathematical function used to perform operation on two functions (f and g) that produces a third function expressing how the shape of one is modified by the other. Similarly, using filters and kernels, the convolutional layer calculates the features of input data. In text classification, several filters are learned to extract useful features from a document for the specific classification tasks based on the training dataset.

Pooling

Pooling is applied to the feature maps outputted by convolutional layer, to reduce the number of trainable parameters. There are several pooling operations including, average pooling and max pooling. The max pooling calculates the maximum value in each patch of feature map and reduce the dimensions to almost halve.

Activation Function

The basic effort of an activation function is to allow some nonlinear mappings from input to output. Although, we have various alternatives such as tanh and sigmoid, but the most widely used activation function for convolutional layer is ReLU. The reason to use ReLU is its non-linear property, it prevents the mathematical collapse between layers, due to linear combinations. However, we used the sigmoid activation function for the output layer to predict labels.

6.2.3 Classification Process

The classification process comprises of three fundamental steps of review classification: 1) pre-processing 2) classification model and 3) validation. Figure 6.2 provides a generic flow of activities to carry out these steps. During the pre-processing step, the tokenization of sentences into words and removal of stop-words is performed along with the removal of punctuations and extra symbols. After that a classifier

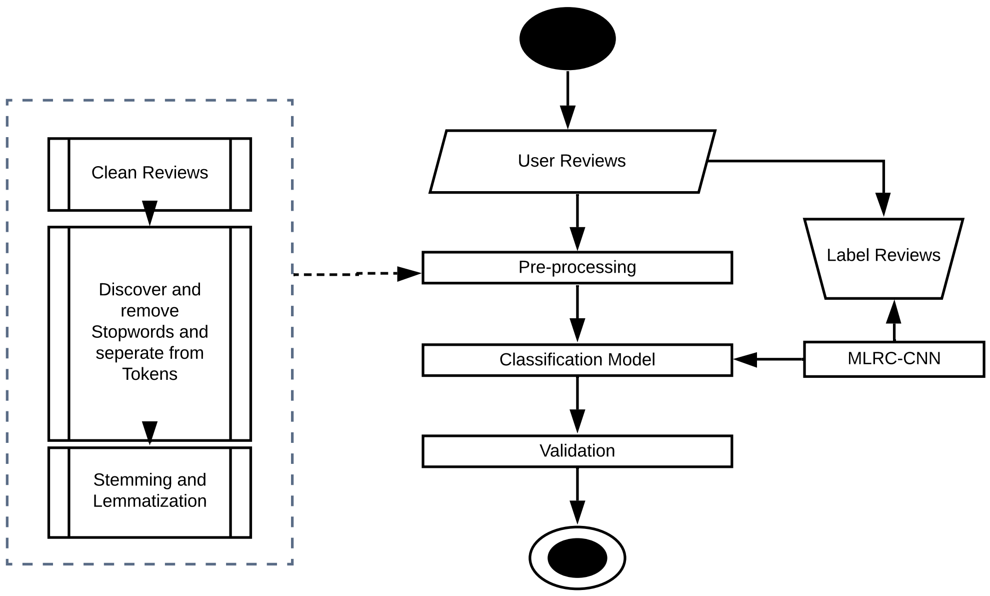


FIGURE 6.2: Generic Process Flow of Reviews Classification Approach

is trained over the input data using a classification model according to a defined set of labels. Then the trained model is tested and validated over the data not used during training.

6.3 MLRC-CNN Analysis and Design

The proposed classification model is inspired by the kim [76] CNN model for text classification, modified to solve multi-label review classification problem. The architecture of the MLRC-CNN model includes, convolutional layer, pooling layer, hidden layer, and fully connected output layer (see Figure 6.3). Convolutional layer map the features using different window sizes (filter-sizes) over the pre-trained word embedding of word2vec and capture rich sematic information. Convolutional layer, convolve filters that are used as n-gram detectors, each filter specializing in a closely-related family of n-grams. Embedding contribute as lookup table for input data, each sentence of a review is separated into token of words, indices of these words are mapped into dimension of vectors according to the look up table.

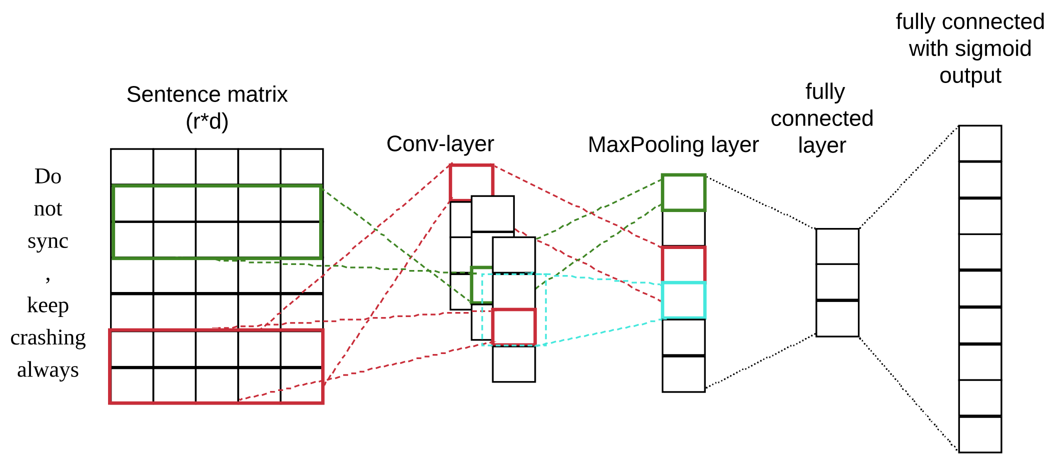


Figure 6.3: MLRC-CNN Architecture

Features are mapped by convolutional layer and then pooling operation is applied to each of these feature maps to produce low dimensional vectors. A max pooling take the maximum value from each mapped feature vector. Output of pooling layer is followed by a fully connected bottleneck layer with h hidden units and then a fully connected layer with sigmoid function is added for multi label output. The flow of activities of MLRC-CNN is illustrated in Figure 6.4.

The words that are absent in the vocabulary of pre-trained word embedding are initialized randomly. In particular, we experimented three different approaches to learn word embedding such as CNN-rand, CNN-static, and CNN-non-static. In CNN-rand, all the words embedding are initialized randomly and updated during training. However, in CNN-static, model is initialized with the pre-trained word vectors but all words are kept static and are not updated during training. At last, in CNN-non-static, the model is initialized with pre-trained word vectors and stay fixed during training. User reviews are processed and the dataset is partitioned into training, testing and validation sets. Training sample of 80 percent further partitioned into 10 percent for validation and 70 percent for training. In the first approach the CNN-rand the embedding are generated based on the input vectors from the dataset and the model is built for multi-label reviews classification, using

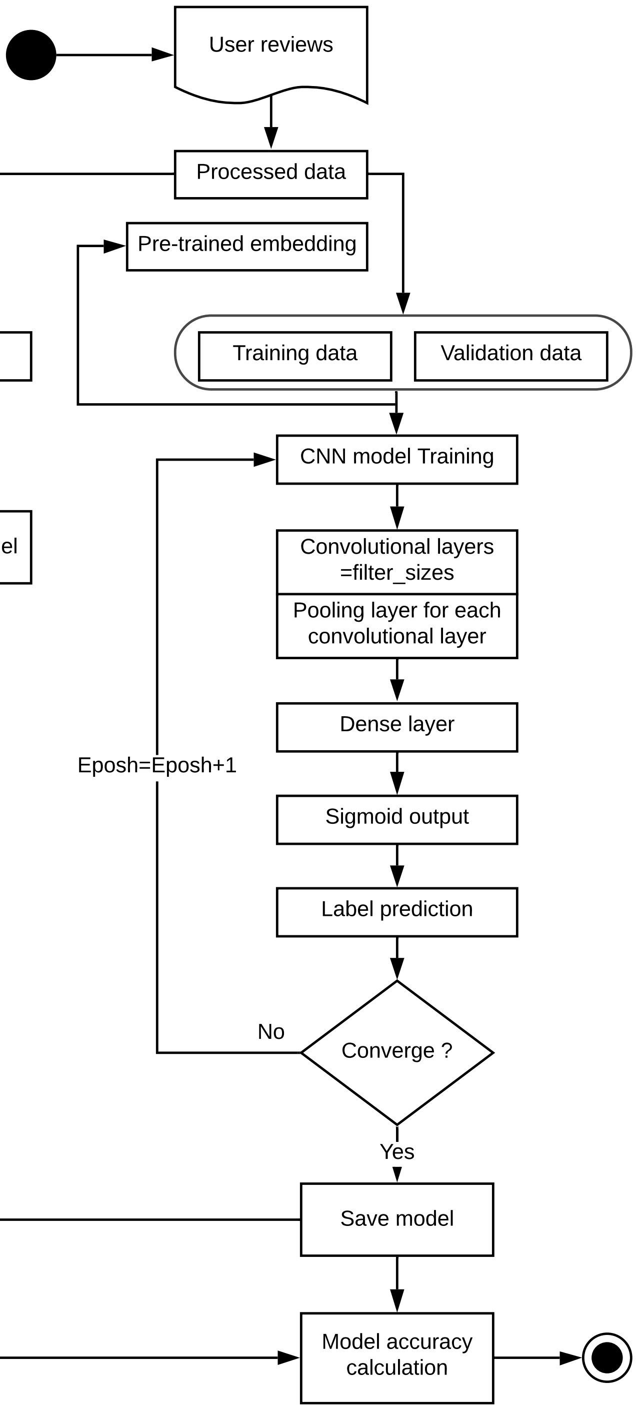


FIGURE 6.4: Activity flow of MLRC-CNN

convolutional layers that map the embedding to features. The input to the network is the embedding of vectors. To apply convolutional operations to those vectors following parameters are set. 1D convolutional layer with multiple filter sizes of 1, 2, 3, 5, and 7, kernel/window size that will have only one number which is 5, moves in one direction only and activation function named ReLu is designed. The GlobalMaxPooling operation reduced the dimensions of the output of convolutional layer. The dropout rate of 0.25 and 128 neurons in the first hidden layer while 256 neurons in the second hidden layer are applied to the output of the pooling layer. At last a sigmoid output with 9 number of units is generated. The defined model is then compiled with the loss function of binary cross entropy with adam (Adaptive Moment Estimation) optimizer and accuracy metric. Finally, the function named fit with batch size of 50 and number of epochs of 10, is used to train the model over 5,392,625 parameters. In the second approach, pre-trained word embedding named word2vec served as a lookup table for the input embedding and do not update during training. While in the third approach, the trainable parameter is set to true, so the embedding can be updated during training. However, parameter settings for the CNN layers, remains same for all three approaches.

6.3.1 Evaluation Metrics

Accuracy is the simple evaluation metric and computes the overall proportion of correctly classified instances. Accuracy is computed as a percentage of the numbers of correctly categorized occurrences under a particular label (TP) a total number of classified occurrences under the identical label (TP + FP). In addition, the val-accuracy is validation accuracy predicted on the validation dataset.

6.4 Summary

This chapter presents the proposed classification model MLRC-CNN for mobile app review classification. Firstly, it describes the fundamental concepts and principles serving as the basis of the current research. Then it provides the overall system architecture, a generalized framework and flow of activities involved in the decision making process. Subsequently, each component of the proposed system along with the detailed working process is discussed in sufficient details. The next chapter discusses the actual implementation and evaluation of the proposed approach in real-world environment.

Chapter 7

Implementation and Evaluation

In this chapter, the proposed classification model MLRC-CNN is implemented in a case of MARC to illustrate the applicability of the proposed classification model. It is significant to identify the important relevant factors that impact the model performance. MARC models are implemented to solve various classification problems. A systematic review of these models has revealed that there are significant factors that affect the model performance. Availability of alternatives to select those factors is limited, so the selection must be ideal.

First of all, the shape of the dataset impact the results, as in ML approaches ‘garbage in garbage out’, so the data set must be accurate, clean and managed according to the requirements. Therefore, we analysed the dataset first to ensure its authenticity and applicability. Following section defines the dataset attributes.

7.1 Results of Dataset Analysis

Prior to the implementation of the MLRC-CNN model, we analyse the dataset of user reviews to draw some significant results. For the purpose of this analysis user reviews are labelled into categories of NFRs manually according to the quality model defined in Figure 4.3 and FR, by two annotators. The NFRs expand across a wide range of sub categories of ISO25010 quality model's attributes, treating these sub categories as a separate class is not a wise approach, even some of the attributes are not considered as the scope of this study (i.e. maintainability, portability and some attributes of quality in use model, effectiveness, efficiency, and context coverage). Therefore, NFRs are classified into selected categories of 'product quality model' and 'quality in use' model. The dataset contains ten thousands instances and two attributes, as illustrated in the table. Total number

Table 7.1: Dataset Description

Dataset	No. of instances	No. of attributes	No. of target attributes	Missing values?	Pre-processing required?
Entertainment apps	2000	1	9	Yes	yes
Health & Fitness apps	2000	1	9	None	Yes
Shopping apps	2000	1	9	None	Yes
Education apps	2000	1	9	Yes	Yes
Communication apps	2000	1	9	Yes	Yes

of instances in the whole dataset are ten thousands collected from 50 different apps of 5 different categories, where each category have 2000 instances from 10 different applications. Instances are simply the user reviews that are crawled and analysed in this research. Now, the number of dependent attribute in the dataset, used to perform analysis is one, the attribute of 'Reviews'. Whereas, the number of independent or targeted attributes are nine, including reliability, compatibility, usability, functional-suitability, performance, security, freedom-from-risk, satisfaction, and functional-requirement. Furthermore, missing values are those reviews that are missed when crawled from the play store, filled manually by searching the app reviews from Google-play store. User reviews are raw text and sometimes informal language, using extra symbols or punctuation, and without considering any sentence order is used by the users to post a review about an app. To prepare this raw text for analysis, some pre-processing has been required including, the removal of punctuation and symbols. After preparation of the data, results of analysis are reported here including the distribution of the occurrence of each requirement type over the number of reviews for each category. At first, the results of the distribution of labels or requirement types for the category of Shopping apps has been illustrated in Figure 7.1. As the Figure 7.1 shows that in the large

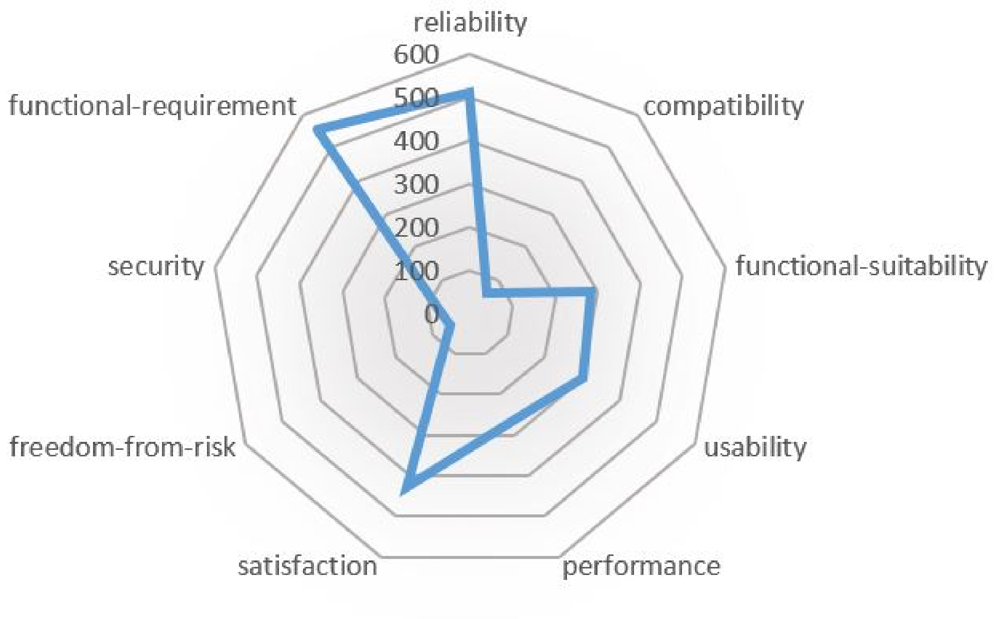


Figure 7.1: Reviews Distribution over Class-labels for Shopping Apps

number of reviews, the users suggested a new feature or requested to enhance the existing feature with the total count of 552, followed by reliability with the count of 510. Either the app is frustrated or loved by the users, 429 reviews reported their concern related to the satisfaction. The interaction of user with the app, either achieving good level of ease-to-use or not is reported in 301 user reviews, labeled as usability. While the app causing error or not performing its functionality properly is a functional-suitability concern with 285 reviews under that label. When the error is related to the response time an app taking to perform a function or utilizing extra resources such as battery or internet, then its performance needs to be considered as 271 reviews reported this concern. In shopping apps, users provide their credentials to buy products and they are concerned about the confidentiality and integrity of the data they are providing, in 76 reviews users showed security concern. If the app is not syncing data with other devices such as smart watch properly, then it is a compatibility concern reported in 63 reviews. At last, the least number of reviews (54) are occurred under the label of Freedom-from-risk.

Now, the results of the distribution of labels or requirement types for the category of Communication apps has been illustrated in the Figure 7.2. Either the app

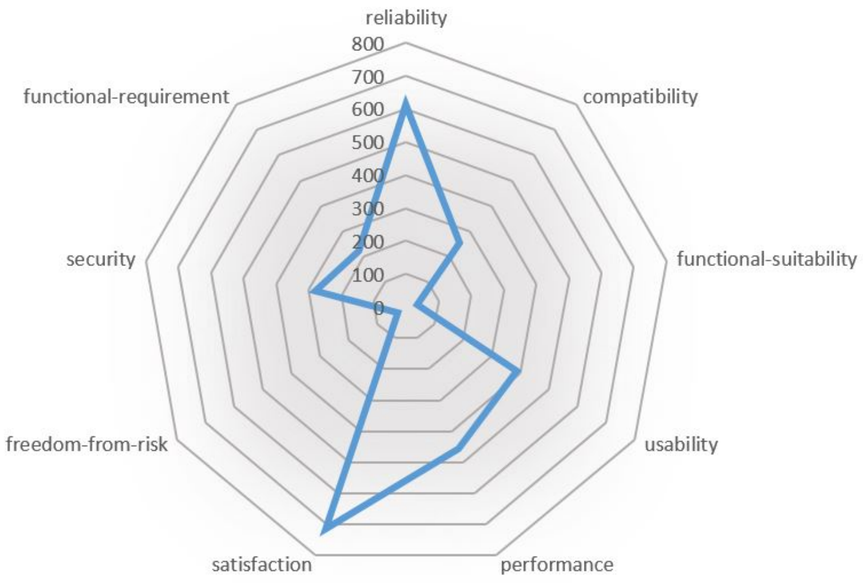


FIGURE 7.2: Reviews Distribution over Class-labels for Communication Apps

is frustrated or loved by the users, 714 reviews reported their concern related to the satisfaction, the most reported user experience, followed by the reliability with the count of 614. When the error is related to the response time an app taking to perform a function or utilizing extra resources such as battery or internet, then its performance needs to be considered as 457 reviews reported this concern. The interaction of user with the app, either achieving good level of ease-to-use or not is reported in 388 user reviews, labeled as usability. In communication apps, users provide their credentials such as phone number, email or contact access to communicate using the app and they are concerned about the confidentiality and integrity of the data they are providing, in 280 reviews users showed security concerns. If the app is not syncing data with other devices such as smart watch properly, then it is a compatibility concern reported in 253 reviews. The users suggested a new feature, or requested to enhance the existing feature reported in the total number of 223 reviews. While the app causing error or not performing its functionality properly is a functional-suitability concern with 35 reviews under that label. At last, the least number of reviews (31) are occurred under the label of Freedom-from-risk.

The results of the distribution of labels or requirement types for the category of Education apps has been illustrated in Figure 7.3. As the Figure 7.3 shows, the

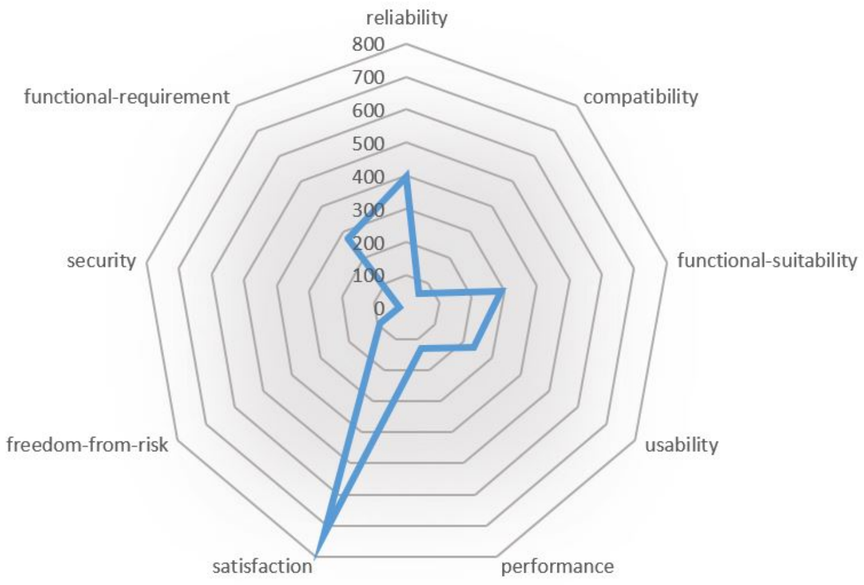


FIGURE 7.3: Reviews Distribution over Class-labels for Education Apps

large number of reviews belong to the category of satisfaction (729) followed by reliability (397) and the functional suitability (289). The users suggested a new feature, or requested to enhance the existing feature reported in the total number of 275 reviews. The interaction of user with the app, either achieving good level of ease-to-use or not is reported in 236 user reviews, labeled as usability. When the error is related to the response time an app taking to perform a function or utilizing extra resources such as battery or internet, then its performance needs to be considered as 129 reviews reported this concern followed by freedom-from-risk (92). In the end, compatibility (58) and security (22) are least reported concerns in the user reviews of educational apps.

Furthermore, the results of the distribution of labels or requirement types for the category of Health-and-Fitness apps has been illustrated through Figure 7.4 Health-

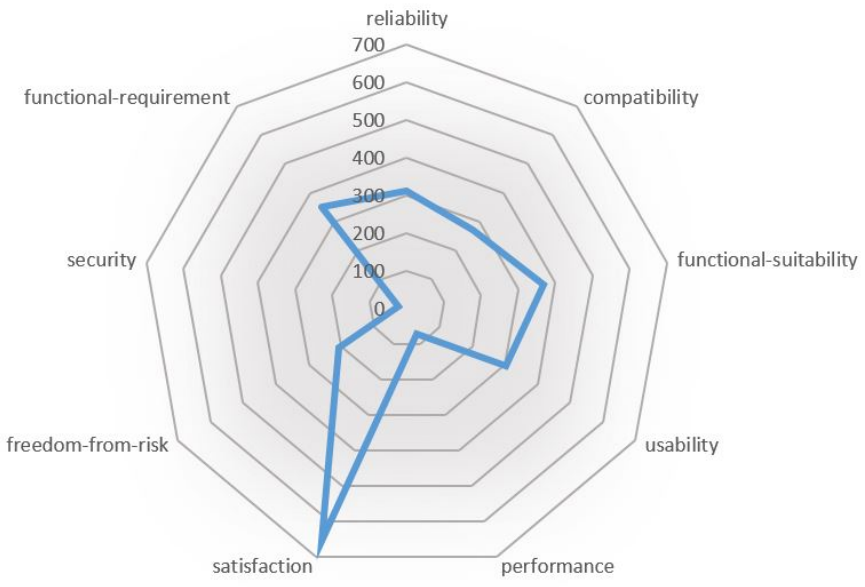


FIGURE 7.4: Reviews Distribution over Class-labels for Health-and-Fitness Apps

and-Fitness apps are lack in multiple features that reported most in the user reviews of these apps including, the satisfaction (656) user achieved using the app, the functional correctness and completeness (369) of the app, addition and improvement of the features of the apps (351) and compatibility (270) with other health monitoring devices such as smart watch after the reliability (310). The interaction of user with the app, either achieving good level of ease-to-use or not is reported in 302 user reviews, labeled as usability. Furthermore, freedom-from-risk

(207) is the most reported concern by the users than performance (73). The least number of requirement types occurred in reviews are related to security (23) is the ultimate concern of health app users. In addition, the results of the distribution of labels or requirement types for the category of Entertainment apps has been illustrated through Figure 7.5 Users are love or hate the app, express their opinion

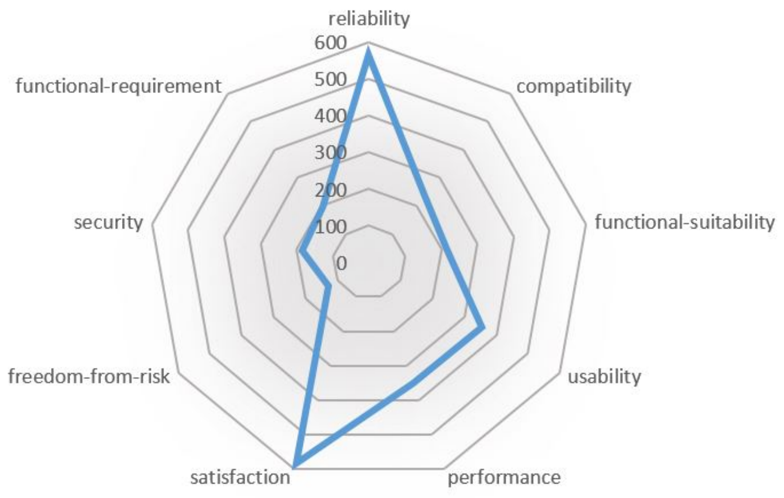


FIGURE 7.5: Reviews Distribution over Class-labels for Entertainment Apps

about the app in 581 user reviews. Then the reliability related opinions about apps are reported by users in 566 reviews, followed by the usability related 353 reviews and performance related 352 reviews. The apps are not supporting the devices such as headphones, experience reported by users in 237 reviews. Functional error, or the incomplete functionality experiences by users mentioned in 215 user reviews. At last, the functional requirements to add new feature or enhance the existing feature is reported by 197 users, followed by security with 186 reviews and freedom-from-risk with 127 reviews.

At last, the distribution of the occurrence of each requirement type over the total number of reviews for all five categories has been illustrated in the Figure 7.6. As the Figure 7.6 shows, the large number of reviews belong to the category of satisfaction followed by reliability and the functional requirement (feature request). After

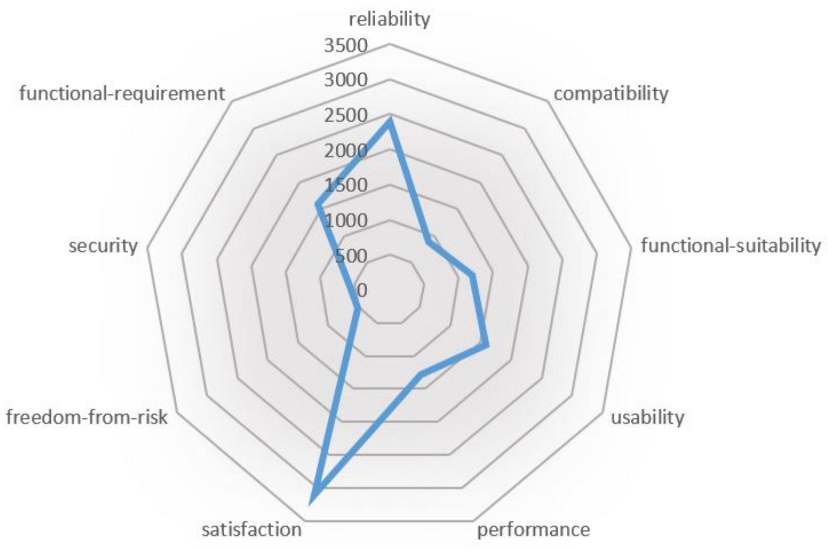


Figure 7.6: Reviews Distribution over Class-labels for All Five App Categories

them, the usability is the most occurring requirement type then the users reported functional error the most in apps as requirement type of functional suitability. App is not syncing data with other devices or apps is a compatibility concern reported most after functional-suitability by users in the reviews of apps. The least number of requirement types occurred in reviews are freedom-from-risk and security. These results are based on the manually labelled reviews and uncertainty in the human judgments could occur at any stage, therefore, results may vary when evaluated by other researchers.

7.2 Results of the MLRC-CNN classifier

After analysing the dataset, reviews text is cleaned using pre-processing methods, a set of relevant best approaches reported in the literature. On the basis of the most widely used text processing methods, dataset of user reviews is processed before applying the classification model. This is followed by the feature extraction phase, which utilize the word embedding to map the important features of user

Table 7.2: Initial Results of MLRC-CNN

Epochs	**Rand**		**Static**		**Non-static**	
	Accuracy	Val_accuracy	Accuracy	Val_accuracy	Accuracy	Val_accuracy
1	0.8287	0.8528	0.8305	0.8535	0.8379	0.5476
2	0.8562	0.8535	0.8550	0.8563	0.8530	0.8601
3	0.8622	0.8528	0.8589	0.8438	0.8591	0.8545
4	0.8833	0.8583	0.8610	0.8497	0.8628	0.8535
5	0.9103	0.8604	0.8675	0.8528	0.8703	0.8611
6	0.9505	0.8639	0.8760	0.8549	0.8837	0.8569
7	0.9714	0.8646	0.8884	0.8524	0.8908	0.8524
8	0.9801	0.8652	0.8966	0.8531	0.9041	0.8573
9	0.9941	0.8697	0.9110	0.8572	0.9227	0.8582
10	0.9906	0.8693	0.9170	0.8573	0.9319	0.8590
Average	0.886096	0.861028	0.875814	0.853091	0.883079	0.819459

reviews. The dataset of user reviews is collected from total fifty apps of five different categories. Total ten thousands reviews are labelled manually. The labelled data is processed and relative feature maps are generated. To classify the reviews, a CNN model is designed and trained over the reviews dataset. 1D convolutional layer with five filters has been implemented to map the features set. Max pooling operation applied to reduce the dimensions of that feature set, output of pooling layer act as an input to the fully connected layer. The proposed approach defined in chapter 6 in detail, here we discuss the results after implementing the MLRC-CNN in python using keras library. In this section, results of MLRC-CNN classification are presented in detail. Initially the MLRC-CNN model is trained over the data samples of 1440 instances of reviews, validated on 160 samples and tested on 400 samples for all three settings. Results of ten iterations for MLRC-CNN are averaged over all settings using the Geometric-Mean formula. The outcome shows that type of rand-CNN output the highest training and validation accuracy whereas, non-static-CNN have the lowest validation accuracy and static-CNN achieved the second highest validation accuracy. Detailed results with all three settings, static, non-static and rand are shown in Table 7.2. Now perform the MLRC-CNN over the whole dataset of ten thousand reviews till 3 epochs due to device limitations. The model is trained over the data samples of 7200 instances, validated on 800 samples and tested on 2000 samples for all three settings. Results of three iterations for MLRC-CNN are averaged over all settings, using the Geometric-Mean formula. The outcome shows that type of rand-CNN output the highest training and validation accuracy whereas, non-static CNN have the lowest validation accuracy and static-CNN achieved the second highest validation accuracy. Detailed results with all three approaches of CNN-static, CNN-non-static and CNN-rand are shown in Table 7.3.

Table 7.3: MLRC-CNN Results over the Dataset of Ten Thousands Reviews

Epochs	**Rand**		**Static**		**Non-static**	
	Accuracy	Val_accuracy	Accuracy	Val_accuracy	Accuracy	Val_accuracy
1	0.85	0.8567	0.8544	0.8550	0.8530	0.8650
2	0.8607	0.8632	0.8552	0.8550	0.8581	0.8570
3	0.8797	0.8563	0.8555	0.8530	0.8684	0.8554
Average	0.863379581	0.858727519	0.858727519	0.858727519	0.858727519	0.858727519

The prediction accuracy on test set for all three types is equal to one, when tested over the unseen data samples.

7.3 Summary

This chapter implemented the proposed model for app reviews classification named as MLRC- CNN (Multi Label Review Classifier Convolutional Neural Network). At first, the results of dataset analysis are reported, where the number of reviews per category are identified and illustrated. Furthermore, MLRC-CNN model is trained, validated and tested against the three settings of CNN model, with the dataset of two thousand user reviews. Later, the results are analysed for MLRC-CNN model with ten thousand data instances and several epochs equal to three. Evaluation metrics, accuracy is used and results are reported by averaging them over epochs using GM average formula. MLRC-CNN with random and static settings returned promising results.

Chapter 8

Conclusion

In this chapter, the proposed research work is summarized, the approaches and methods used to achieve the defined objectives, contributions, threats to validity and the future suggestions of the proposed model

This thesis aims at automatic extraction of NFRs from mobile app reviews to facilitate the requirement elicitation process. Existing MARC approaches have serious limitations in terms of the types of requirements extracted and the feature sets used to classify these requirements. Most of the literature studies have extracted basic and abstract level of requirements, while neglecting several important factors. In addition, a combination of product quality and quality in use model is rarely seen in the existing body of knowledge on MARC. Considering the users' expectations, diversity of users, and accessibility of the mobile applications, it is imperative to consider the set of multiple types of requirements.

The state-of-the art exposes an immediate demand of more comprehensive set of requirements to be extracted from user reviews, while incorporating the uncertainty and vagueness at multiple levels. In this research, we have developed an MLRC model based on the CNN algorithm using three different approaches. Several recent studies have emphasized that to cope with the uncertainty of requirements and to classify them automatically, a model must be developed for multi-label reviews classification problem.

For automated multi-label requirement extraction, this research employs, Machine Learning based classification models such as SVM, NB, LR, and RF, and a deep-learning-based classification model such as CNN. These models are implemented in python with different settings and results are reported separately for both of them with varying dataset sizes. An automated requirement extraction model MLRC-CNN is implemented in Python programming environments and a case study on fifty real world application user reviews is performed to demonstrate the applicability of the proposed classification model. To classify user reviews into multiple categories of NFRs and FRs, three objectives were defined and fulfilled over the course of this research.

8.1 Achieving Research Objectives

This section provides a discussion on the research methodology, findings and deliverable of the three defined objectives. The overall research process can be broadly seen as dataset extraction, preparation, extraction of features, implementation and evaluation of the ML classification models and the proposed deep learning based classification model. Three primary research objectives are intended to be achieved by this research. Here we provide a brief discussion on these objectives, the process of realization and research deliverable.

RQ1: To gain insight in the state-of-the-art classification methods that need to be addressed in the context of app review analysis

To achieve the RO1, comprehensive and multi-faceted literature review was conducted encompassing mobile app reviews analysis and classification methods, test pre-processing and feature extraction approaches to facilitate classification process, widely considered requirement types and ML methods. Subsequently, a Systematic Literature Review (SLR) was conducted particularly focusing on classification models based on ML. This SLR was intended to address the significant research gap of no systematic study on a wide range of machine learning based solutions for mobile app reviews analysis and classification.

A systematic literature review (SLR) aims at identifying, evaluating and interpreting all available research related to a specific field of interest. We aim to fill this gap by conducting a systematic literature review using SLR guidelines by Kitchenham [87]. Five Databases were considered as the primary data sources for the relevant studies. After analyzing the results from 20 primary studies on ARAC, some critical issues were highlighted in the existing solutions. This SLR intended to highlight research gap and issues that needed to be addressed.

The findings of this SLR reveal, an increasing research towards ML based Classification methods. SVM, NB and RF models have largely been used to aid classification of mobile app reviews. Tokenization, stemming and stop-word removal were found to be the most widely used pre-processing methods for app reviews. However very few studies have actually classified the reviews into multiple categories, most of the studies have only extracted sentiments or performed multi-class classification. Alternatively, studies classifying reviews under multiple labels, have largely neglected the requirement categorization according to any standard model such as ISO25010. We aim to extract the requirements defined by ISO25010 model to facilitate the requirement elicitation process. **RO2: To evaluate the performance of successfully employed ML models for multi-label review classification.**

The first step to evaluate the performance of classification methods and models, is to identify the most significantly applied methods. These methods are identified through SLR, utilized for ARAC are considered. This generalization is necessary as the diverse and significantly large number of classification methods have been reported in the existing literature. Then a criteria of majority voting is used to select the most widely and successfully implemented classification methods for mobile app reviews analysis and classification. In this research, the four ML based classification models that are reported by most of the studies with significant results in various classification settings are implemented with the feature extraction methods including BoWs and TF-IDF. The selected ML models are implemented and evaluated over the dataset of one thousand reviews, extracted from five Android Health Fitness apps, and labelled manually into multiple categories of NFRs. The performance of ML models with BoW feature extraction method and with TF-IDF is reported separately for each category and then the results are averaged. Implemented models are evaluated using standard performance measures, considering the macro average scores for each category of requirements, results are averaged across each category using geometric mean.

RO3: To automatically categorise reviews into multiple labels

Identification of the research gap from literature signify the development of a deep learning based classification model to automatically classify user reviews into multiple categories. A CNN classifier is implemented for multi-label reviews, named as MLRC-CNN. Three approaches of CNN with same architecture are implemented over the data set of two thousand user reviews, results are reported for ten epochs. Furthermore, the dataset of ten thousands reviews is used to evaluate the MLRC-CNN model performance, results are reported for three epochs. At last, the proposed model is evaluated using the accuracy metrics and averaged the results using geometric mean.

8.2 Contribution and Significance

The proposed classification model for mobile app reviews has developed to support both future researchers and app developers. The app review analysis and classification is a real-world and significant problem. Extraction of requirements from reviews according to the standard quality model, helps developers in the elicitation process. Users express their opinion about an app in user reviews, review level analysis express multiple requirements in a single review. This demands an automated model to classify reviews into multiple categories. The proposed model can be effectively used for classification of reviews, considering their context. The researchers may also explore other efficient MARC methods for the Multi label classification. Character level implementation of embedding for CNN can also be explored for multi-label review classification.

8.3 Limitations and Future Work

This section briefly describes the limitations of the current research work and future implications of the present study. In the context of limitations of the ongoing work, the proposed classification model focuses explicitly on non-functional requirements and functional requirements such as feature request with the assumption that all reviews sentiments are according to the context of reported requirement type. Although, MLRC-CNN effectively handles the multi-label classification of reviews. However, the number of epochs are minimal and can be increased to achieve better performance. The implementation of the proposed classification model used the pre-trained word embedding of Google news. However, the other available embedding can also be considered. Furthermore, the context specific embedding can be generated and used for app user reviews analysis. In future we will consider these aspects. This research will be further extended towards more empirical case studies on other sources of user opinion, such as social media reviews and tweets. The proposed classification model is evaluated using accuracy metrics. However, the other performance measure can also be considered to evaluate the model performance.

Bibliography

[1] Ani Nenkova and Kathleen McKeown. "A Survey of Text Summarization Techniques". In: *Mining Text Data*. Ed. by Charu C. Aggarwal and ChengXiang Zhai. Boston, MA: Springer US, 2012, pp. 43–76. ISBN: 978-1-4614-3222-7 978-1-4614-3223-4. DOI: `10.1007/978-1-4614-3223-4_3`. URL: `http://link.springer.com/10.1007/978-1-4614-3223-4_3` (visited on 05/17/2020).

[2] Thanasis Petsas et al. "Rise of the planet of the apps: a systematic study of the mobile app ecosystem". In: *Proceedings of the 2013 conference on Internet measurement conference - IMC '13*. the 2013 conference. Barcelona, Spain: ACM Press, 2013, pp. 277–290. ISBN: 978-1-4503-1953-9. DOI: `10.1145/2504730.2504749`. URL: `http://dl.acm.org/citation.cfm?doid=2504730.2504749` (visited on 04/02/2020).

[3] J. Clement. *Ratings of apps on Google Play as of September 2019*. Mar. 19, 2020. URL: `https://www.statista.com/statistics/266217/customer-ratings-of-android-applications/`.

[4] Dennis Pagano and Walid Maalej. "User feedback in the appstore: An empirical study". In: *2013 21st IEEE International Requirements Engineering Conference (RE)*. 2013 IEEE 21st International Requirements Engineering Conference (RE). Rio de Janeiro-RJ, Brazil: IEEE, July 2013, pp. 125–134. ISBN: 978-1-4673-5765-4. DOI: `10.1109/RE.2013.6636712`. URL: `http://ieeexplore.ieee.org/document/6636712/` (visited on 12/09/2019).

[5] Eduard C. Groen et al. “Users — The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews”. In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017 IEEE 25th International Requirements Engineering Conference (RE). Lisbon, Portugal: IEEE, Sept. 2017, pp. 80–89. ISBN: 978-1-5386-3191-1. DOI: 10.1109/RE.2017.73. URL: http://ieeexplore.ieee.org/document/8048893/ (visited on 04/02/2020).

[6] Muneera Bano, Didar Zowghi, and Matthew Kearney. “Feature Based Sentiment Analysis for Evaluating the Mobile Pedagogical Affordances of Apps”. In: *Tomorrow’s Learning: Involving Everyone. Learning with and about Technologies and Computing*. Ed. by Arthur Tatnall and Mary Webb. IFIP Advances in Information and Communication Technology. Springer International Publishing, 2017, pp. 281–291. ISBN: 978-3-319-74310-3.

[7] 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. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2016.11.027. URL: http://www.sciencedirect.com/science/article/pii/S0164121216302291.

[8] Sebastiano Panichella et al. “ARdoc: App Reviews Development Oriented Classifier”. In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. FSE 2016. event-place: Seattle, WA, USA. New York, NY, USA: ACM, 2016, pp. 1023–1027. ISBN: 978-1-4503-4218-6. DOI: 10.1145/2950290.2983938. URL: http://doi.acm.org/10.1145/2950290.2983938.

[9] William Martin et al. “A Survey of App Store Analysis for Software Engineering”. In: *IIEEE Trans. Software Eng.* 43.9 (Sept. 1, 2017), pp. 817–847. ISSN: 0098-5589, 1939-3520, 2326-3881. DOI: 10.1109/TSE.2016.2630689. URL: https://ieeexplore.ieee.org/document/7765038/ (visited on 04/02/2020).

[10] Maya Daneva Chong Wang. “A systematic mapping study on crowdsourced requirements engineering using user feedback”. In: (July 15, 2019). DOI: 10.1002/smr.2199. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2199?mi=q8l17g&af=R&ConceptID=68&PubType=journal&content=articlesChapters&dateRange=%5B20080101+TO+*%7D&field1=AllField&target=default&text1=%28%28+%22mobile+app*%22+OR+%22online+app*%22+OR+%22smartphone+app*%22+OR+%22android+app*%22+OR+%22ios+app*%22+OR+%22apple+app+store%22%29+AND+%28review+OR+comment*+OR+feedback+OR+opinion%29+AND+%28analy*+OR+mining+%29%29.

[11] Hui Yang and Peng Liang. “Identification and Classification of Requirements from App User Reviews.” In: *SEKE*. 2015, pp. 7–12.

[12] N. M. Rizk, A. Ebada, and E. S. Nasr. “Investigating mobile applications’ requirements evolution through sentiment analysis of users’ reviews”. In: *2015 11th International Computer Engineering Conference (ICENCO)*. 2015 11th International Computer Engineering Conference (ICENCO). Dec. 29, 2015, pp. 123–130. DOI: 10.1109/ICENCO.2015.7416336.

[13] Runyu Chen, Qili Wang, and Wei Xu. “Mining User Requirements to Facilitate Mobile App Quality Upgrades with Big Data”. In: *Electronic Commerce Research and Applications* (2019), p. 100889. ISSN: 1567-4223. DOI: https://doi.org/10.1016/j.elerap.2019.100889. URL: http://www.sciencedirect.com/science/article/pii/S1567422319300663.

[14] Walid Maalej et al. “On the automatic classification of app reviews”. In: *Requirements Eng* 21.3 (Sept. 1, 2016), pp. 311–331. ISSN: 1432-010X. DOI: 10.1007/s00766-016-0251-9. URL: https://doi.org/10.1007/s00766-016-0251-9 (visited on 09/27/2019).

[15] Maleknaz Nayebi, Henry Cho, and Guenther Ruhe. “App store mining is not enough for app improvement”. In: *Empir Software Eng* 23.5 (Oct. 1, 2018),

pp. 2764–2794. ISSN: 1573-7616. DOI: 10.1007/s10664-018-9601-1. URL: https://doi.org/10.1007/s10664-018-9601-1 (visited on 09/27/2019).

[16] S. Muñoz et al. "A Cognitive Agent for Mining Bugs Reports, Feature Suggestions and Sentiment in a Mobile Application Store". In: *2018 4th International Conference on Big Data Innovations and Applications (Innovate-Data).* 2018 4th International Conference on Big Data Innovations and Applications (Innovate-Data). Aug. 6, 2018, pp. 17–24. DOI: 10.1109/Innovate-Data.2018.00010.

[17] Nishant Jha and Anas Mahmoud. "Mining non-functional requirements from App store reviews". In: *Empir Software Eng* (June 7, 2019). ISSN: 1573-7616. DOI: 10.1007/s10664-019-09716-7. URL: https://doi.org/10.1007/s10664-019-09716-7 (visited on 09/27/2019).

[18] A. Ahmad et al. "Toward Empirically Investigating Non-Functional Requirements of iOS Developers on Stack Overflow". In: *IEEE Access* 7 (2019), pp. 61145–61169. DOI: 10.1109/ACCESS.2019.2914429.

[19] Wen Zhang et al. "An empirical study on classification of non-functional requirements". In: *The twenty-third international conference on software engineering and knowledge engineering (SEKE 2011).* 2011, pp. 190–195.

[20] David Griol et al. "An Approach to Sentiment Analysis for Mobile Speech Applications". In: *Future and Emergent Trends in Language Technology.* Ed. by José F. Quesada, Francisco-Jesús Martín Mateos, and Teresa Lopez-Soto. Lecture Notes in Computer Science. Springer International Publishing, 2016, pp. 96–107. ISBN: 978-3-319-33500-1.

[21] Phillipe Rodrigues et al. "Beyond the Stars: Towards a Novel Sentiment Rating to Evaluate Applications in Web Stores of Mobile Apps". In: *Proceedings of the 26th International Conference on World Wide Web Companion.* WWW '17 Companion. event-place: Perth, Australia. Republic and Canton

of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2017, pp. 109–117. ISBN: 978-1-4503-4914-7. DOI: 10.1145/3041021.3054139. URL: https://doi.org/10.1145/3041021.3054139.

[22] Bin Fu et al. "Why People Hate Your App: Making Sense of User Feedback in a Mobile App Store". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '13. event-place: Chicago, Illinois, USA. New York, NY, USA: ACM, 2013, pp. 1276–1284. ISBN: 978-1-4503-2174-7. DOI: 10.1145/2487575.2488202. URL: http://doi.acm.org/10.1145/2487575.2488202.

[23] Mengmeng Lu and Peng Liang. "Automatic Classification of Non-Functional Requirements from Augmented App User Reviews". In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering.* EASE'17. event-place: Karlskrona, Sweden. New York, NY, USA: ACM, 2017, pp. 344–353. ISBN: 978-1-4503-4804-1. DOI: 10.1145/3084226.3084241. URL: http://doi.acm.org/10.1145/3084226.3084241.

[24] Anas Mahmoud and Grant Williams. "Detecting, classifying, and tracing non-functional software requirements". In: *Requirements Eng* 21.3 (Sept. 2016), pp. 357–381. ISSN: 0947-3602, 1432-010X. DOI: 10.1007/s00766-016-0252-8. URL: http://link.springer.com/10.1007/s00766-016-0252-8 (visited on 04/02/2020).

[25] Lawrence Chung and Julio Cesar Sampaio do Prado Leite. "On Non-Functional Requirements in Software Engineering". In: *Conceptual Modeling: Foundations and Applications.* Ed. by Alexander T. Borgida et al. Vol. 5600. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 363–379. ISBN: 978-3-642-02462-7 978-3-642-02463-4. DOI: 10.1007/978-3-642-02463-4_19. URL: http://link.springer.com/10.1007/978-3-642-02463-4_19 (visited on 04/02/2020).

[26] Harleen K. Flora and Dr. Swati V. Chande. "A Review and Analysis on Mobile Application Development Processes using Agile Methodologies". In:

IJORCS 3.4 (July 5, 2013), pp. 8–18. ISSN: 22498257, 22498265. DOI: 10.7815/ijorcs.34.2013.068. URL: http://www.ijorcs.org/manuscript/id/68/doi:10.7815/ijorcs.34.2013.068/harleen-k-flora/a-review-and-analysis-on-mobile-application-development-processes-using-agile-methodologies (visited on 05/17/2020).

[27] Javed Iqbal et al. "Requirements engineering issues causing software development outsourcing failure". In: *PLoS ONE* 15.4 (Apr. 9, 2020). Ed. by Baogui Xin, e0229785. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0229785. URL: https://dx.plos.org/10.1371/journal.pone.0229785 (visited on 08/19/2020).

[28] "IEEE Standard Glossary of Software Engineering Terminology". In: *IEEE Std 610.12-1990* (Dec. 1990), pp. 1–84. DOI: 10.1109/IEEESTD.1990.101064.

[29] Klaus Pohl and Chris Rupp. *Requirements engineering fundamentals: a study guide for the Certified Professional for Requirements Engineering exam ; foundation level, IREB compliant*. 1. ed. Rockynook computing. OCLC: 750710786. Santa Barbara, Calif: Rocky Nook, 2011. 163 pp. ISBN: 978-1-933952-81-9.

[30] Anthony I. Wasserman. "Software engineering issues for mobile application development". In: *Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10*. the FSE/SDP workshop. Santa Fe, New Mexico, USA: ACM Press, 2010, p. 397. ISBN: 978-1-4503-0427-6. DOI: 10.1145/1882362.1882443. URL: http://portal.acm.org/citation.cfm?doid=1882362.1882443 (visited on 05/17/2020).

[31] Thomas D. LaToza and Andre van der Hoek. "Crowdsourcing in Software Engineering: Models, Motivations, and Challenges". In: *IEEE Softw.* 33.1 (Jan. 2016), pp. 74–80. ISSN: 0740-7459. DOI: 10.1109/MS.2016.12. URL: http://ieeexplore.ieee.org/document/7367992/ (visited on 05/17/2020).

[32] Stuart McIlroy et al. "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews". In: *Empir Software Eng* 21.3 (June 1, 2016), pp. 1067–1106. ISSN: 1573-7616. DOI: 10.1007/s10664-015-9375-7. URL: https://doi.org/10.1007/s10664-015-9375-7 (visited on 09/27/2019).

[33] Walid Maalej, Maleknaz Nayebi, and Guenther Ruhe. "Data-driven Requirements Engineering: An Update". In: *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*. ICSE-SEIP '10. event-place: Montreal, Quebec, Canada. Piscataway, NJ, USA: IEEE Press, 2019, pp. 289–290. DOI: 10.1109/ICSE-SEIP.2019.00041. URL: https://doi.org/10.1109/ICSE-SEIP.2019.00041.

[34] Timo Johann et al. "SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017 IEEE 25th International Requirements Engineering Conference (RE). Lisbon, Portugal: IEEE, Sept. 2017, pp. 21–30. ISBN: 978-1-5386-3191-1. DOI: 10.1109/RE.2017.71. URL: http://ieeexplore.ieee.org/document/8048887/ (visited on 12/09/2019).

[35] C. Iacob and R. Harrison. "Retrieving and analyzing mobile apps feature requests from online reviews". In: *2013 10th Working Conference on Mining Software Repositories (MSR)*. 2013 10th Working Conference on Mining Software Repositories (MSR). May 18, 2013, pp. 41–44. DOI: 10.1109/MSR.2013.6624001.

[36] Y. Liu et al. "Stratify Mobile App Reviews: E-LDA Model Based on Hot "Entity" Discovery". In: *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. 2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS). Dec. 28, 2016, pp. 581–588. DOI: 10.1109/SITIS.2016.97.

[37] A. Di Sorbo et al. “SURF: Summarizer of User Reviews Feedback”. In: *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). May 20, 2017, pp. 55–58. DOI: 10.1109/ICSE-C.2017.5.

[38] Faiz Ali Shah, Kairit Sirts, and Dietmar Pfahl. “Is the SAFE Approach Too Simple for App Feature Extraction? A Replication Study”. In: *Requirements Engineering: Foundation for Software Quality*. Ed. by Eric Knauss and Michael Goedicke. Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 21–36. ISBN: 978-3-030-15538-4.

[39] Andrea Di Sorbo et al. “What would users change in my app? summarizing app reviews for recommending software changes”. In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*. the 2016 24th ACM SIGSOFT International Symposium. Seattle, WA, USA: ACM Press, 2016, pp. 499–510. ISBN: 978-1-4503-4218-6. DOI: 10.1145/2950290.2950299. URL: http://dl.acm.org/citation.cfm?doid=2950290.2950299 (visited on 12/09/2019).

[40] S. Panichella et al. “How can i improve my app? Classifying user reviews for software maintenance and evolution”. In: *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME). Oct. 29, 2015, pp. 281–290. DOI: 10.1109/ICSM.2015.7332474.

[41] Ning Chen et al. “AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace”. In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. event-place: Hyderabad, India. New York, NY, USA: ACM, 2014, pp. 767–778. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568263. URL: http://doi.acm.org/10.1145/2568225.2568263.

[42] Emitza Guzman and Walid Maalej. “How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews”. In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. 2014 IEEE 22nd International Requirements Engineering Conference (RE). Karlskrona, Sweden: IEEE, Aug. 2014, pp. 153–162. ISBN: 978-1-4799-3033-3 978-1-4799-3031-9. DOI: 10.1109/RE.2014.6912257. URL: http://ieeexplore.ieee.org/document/6912257/ (visited on 12/09/2019).

[43] Ken Peffers et al. “A Design Science Research Methodology for Information Systems Research”. In: *Journal of Management Information Systems* 24.3 (Dec. 2007), pp. 45–77. ISSN: 0742-1222, 1557-928X. DOI: 10.2753/MIS0742-1222240302. URL: https://www.tandfonline.com/doi/full/10.2753/MIS0742-1222240302 (visited on 05/18/2020).

[44] A.M. Hickey and A.M. Davis. “Elicitation technique selection: how do experts do it?” In: *Journal of Lightwave Technology*. 11th IEEE International Requirements Engineering Conference. Monterey Bay, CA, USA: IEEE Comput. Soc, 2003, pp. 169–178. ISBN: 978-0-7695-1980-7. DOI: 10.1109/ICRE.2003.1232748. URL: http://ieeexplore.ieee.org/document/1232748/ (visited on 05/18/2020).

[45] Wei Jiang et al. “For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Vincent S. Tseng et al. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 584–595. ISBN: 978-3-319-06605-9.

[46] C. Gao et al. “AR-Tracker: Track the Dynamics of Mobile Apps via User Review Mining”. In: *2015 IEEE Symposium on Service-Oriented System Engineering*. 2015 IEEE Symposium on Service-Oriented System Engineering. Apr. 30, 2015, pp. 284–290. DOI: 10.1109/SOSE.2015.13.

[47] A. 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)*. 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). Feb. 20, 2017, pp. 91–102. DOI: `10.1109/SANER.2017.7884612`.

[48] Roger Deocadez, Rachel Harrison, and Daniel Rodriguez. “Preliminary Study on Applying Semi-Supervised Learning to App Store Analysis”. In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. EASE’17. event-place: Karlskrona, Sweden. New York, NY, USA: ACM, 2017, pp. 320–323. ISBN: 978-1-4503-4804-1. DOI: `10.1145/3084226.3084285`. URL: `http://doi.acm.org/10.1145/3084226.3084285`.

[49] R. A. Masrury, Fannisa, and A. Alamsyah. “Analyzing Tourism Mobile Applications Perceived Quality using Sentiment Analysis and Topic Modeling”. In: *2019 7th International Conference on Information and Communication Technology (ICoICT)*. 2019 7th International Conference on Information and Communication Technology (ICoICT). July 24, 2019, pp. 1–6. DOI: `10.1109/ICoICT.2019.8835255`.

[50] E. Suprayogi, I. Budi, and R. Mahendra. “Information Extraction for Mobile Application User Review”. In: *2018 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. 2018 International Conference on Advanced Computer Science and Information Systems (ICACSIS). Oct. 27, 2018, pp. 343–348. DOI: `10.1109/ICACSIS.2018.8618164`.

[51] Lorenzo Villarroel et al. “Release Planning of Mobile Apps Based on User Reviews”. In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE ’16. event-place: Austin, Texas. New York, NY, USA: ACM, 2016, pp. 14–24. ISBN: 978-1-4503-3900-1. DOI: `10.1145/2884781.2884818`. URL: `http://doi.acm.org/10.1145/2884781.2884818`.

[52] C. Gao et al. “PAID: Prioritizing app issues for developers by tracking user reviews over versions”. In: *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*. 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE). Nov. 2, 2015, pp. 35–45. DOI: 10.1109/ISSRE.2015.7381797.

[53] Laura V. Galvis Carreno and Kristina Winbladh. “Analysis of user comments: An approach for software requirements evolution”. In: *2013 35th International Conference on Software Engineering (ICSE)*. 2013 35th International Conference on Software Engineering (ICSE). San Francisco, CA, USA: IEEE, May 2013, pp. 582–591. ISBN: 978-1-4673-3076-3 978-1-4673-3073-2. DOI: 10.1109/ICSE.2013.6606604. URL: http://ieeexplore.ieee.org/document/6606604/ (visited on 12/09/2019).

[54] E. Guzman, M. El-Haliby, and B. Bruegge. “Ensemble Methods for App Review Classification: An Approach for Software Evolution (N)”. In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). Nov. 9, 2015, pp. 771–776. DOI: 10.1109/ASE.2015.88.

[55] ISO/IEC. *ISO/IEC 25010: 2011 Systems and software engineering–Systems and software Quality Requirements and Evaluation (SQuaRE)–System and software quality models*. CH: ISO Geneva, 2011.

[56] Zijad Kurtanović and Walid Maalej. *Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning*. Pages: 495. Sept. 1, 2017. 490 pp. DOI: 10.1109/RE.2017.82.

[57] Han-Xiao Shi and Xiao-Jun Li. “A sentiment analysis model for hotel reviews based on supervised learning”. In: *2011 International Conference on Machine Learning and Cybernetics*. 2011 International Conference on Machine Learning and Cybernetics (ICMLC). Guilin, China: IEEE, July 2011, pp. 950–954. ISBN: 978-1-4577-0305-8. DOI: 10.1109/ICMLC.2011.6016866.

URL: http://ieeexplore.ieee.org/document/6016866/ (visited on 05/30/2020).

[58] Y. Ekanata and I. Budi. “Mobile application review classification for the Indonesian language using machine learning approach”. In: *2018 4th International Conference on Computer and Technology Applications (ICCTA)*. 2018 4th International Conference on Computer and Technology Applications (ICCTA). May 3, 2018, pp. 117–121. DOI: 10.1109/CATA.2018.8398667.

[59] Andrew Ng. *Machine learning yearning.* 2017. URL: URL:http://www.mlyearning.org/(96).

[60] H.B. Barlow. “Unsupervised Learning”. In: *Neural Computation* 1.3 (Sept. 1989), pp. 295–311. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1989.1.3.295. URL: http://www.mitpressjournals.org/doi/10.1162/neco.1989.1.3.295 (visited on 05/15/2020).

[61] Munir Ahmad et al. “Machine Learning Techniques for Sentiment Analysis: A Review”. In: *INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING* 8 (Apr. 30, 2017), pp. 2045–7057.

[62] Peter D. Turney. “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews”. In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL ’02*. the 40th Annual Meeting. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2001, p. 417. DOI: 10.3115/1073083.1073153. URL: http://portal.acm.org/citation.cfm?doid=1073083.1073153 (visited on 05/15/2020).

[63] Nina Janasik, Timo Honkela, and Henrik Bruun. “Text Mining in Qualitative Research: Application of an Unsupervised Learning Method”. In: *Organizational Research Methods* 12.3 (July 2009), pp. 436–460. ISSN: 1094-4281, 1552-7425. DOI: 10.1177/1094428108317202. URL: http://journals.sagepub.com/doi/10.1177/1094428108317202 (visited on 05/18/2020).

[64] Thomas G. Dietterich and Xin Wang. “Support Vectors for Reinforcement Learning”. In: *Machine Learning: ECML 2001.* Ed. by Luc De Raedt and Peter Flach. Red. by G. Goos, J. Hartmanis, and J. van Leeuwen. Vol. 2167. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 600–600. ISBN: 978-3-540-42536-6 978-3-540-44795-5. DOI: `10.1007/3-540-44795-4_51`. URL: `http://link.springer.com/10.1007/3-540-44795-4_51` (visited on 05/18/2020).

[65] Tianyang Zhang, Minlie Huang, and Li Zhao. “Learning Structured Representation for Text Classification via Reinforcement Learning”. In: *AAAI Conference on Artificial Intelligence; Thirty-Second AAAI Conference on Artificial Intelligence* (2018). URL: `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16537/16174` (visited on 01/01/2018).

[66] Thomas G. Dietterich. “Ensemble Methods in Machine Learning”. In: *Multiple Classifier Systems.* Red. by Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen. Vol. 1857. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15. ISBN: 978-3-540-67704-8 978-3-540-45014-6. DOI: `10.1007/3-540-45014-9_1`. URL: `http://link.springer.com/10.1007/3-540-45014-9_1` (visited on 05/15/2020).

[67] Donghai Guan et al. “A Review of Ensemble Learning Based Feature Selection”. In: *IETE Technical Review* 31.3 (May 4, 2014), pp. 190–198. ISSN: 0256-4602, 0974-5971. DOI: `10.1080/02564602.2014.906859`. URL: `http://www.tandfonline.com/doi/abs/10.1080/02564602.2014.906859` (visited on 05/18/2020).

[68] Christopher JC Burges. “A tutorial on support vector machines for pattern recognition”. In: *Data mining and knowledge discovery* 2.2 (1998). Publisher: Springer, pp. 121–167.

[69] David Meyer and Technische Universität Wien. *Support Vector Machines. The Interface to libsvm in package e1071. Online-Documentation of the package e1071 for &quot;R*. 2001.

[70] David D Lewis. "Naive (Bayes) at forty: The independence assumption in information retrieval". In: *European conference on machine learning*. Springer, 1998, pp. 4–15.

[71] Aurangzeb Khan et al. "A review of machine learning algorithms for text-documents classification". In: *Journal of advances in information technology* 1.1 (2010). Publisher: Academy Publisher, PO Box 40 Oulu 90571 Finland, pp. 4–20.

[72] Andy Liaw, Matthew Wiener, and others. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[73] Cícero dos Santos and Maíra Gatti. "Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts". In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, Aug. 2014, pp. 69–78. URL: `https://www.aclweb.org/anthology/C14-1008`.

[74] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level Convolutional Networks for Text Classification". In: *arXiv:1509.01626 [cs]* (Apr. 3, 2016). arXiv: `1509.01626`. URL: `http://arxiv.org/abs/1509.01626` (visited on 05/30/2020).

[75] Alexis Conneau et al. "Very Deep Convolutional Networks for Text Classification". In: *arXiv:1606.01781 [cs]* (Jan. 27, 2017). arXiv: `1606.01781`. URL: `http://arxiv.org/abs/1606.01781` (visited on 05/27/2020).

[76] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar:

Association for Computational Linguistics, 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. URL: http://aclweb.org/anthology/D14-1181 (visited on 05/17/2020).

[77] Rie Johnson and Tong Zhang. "Effective Use of Word Order for Text Categorization with Convolutional Neural Networks". In: *arXiv:1412.1058 [cs, stat]* (Mar. 26, 2015). arXiv: 1412.1058. URL: http://arxiv.org/abs/1412.1058 (visited on 05/30/2020).

[78] Gang Liu and Jiabao Guo. "Bidirectional LSTM with attention mechanism and convolutional layer for text classification". In: *Neurocomputing* 337 (Apr. 2019), pp. 325–338. ISSN: 09252312. DOI: 10.1016/j.neucom.2019.01.078. URL: https://linkinghub.elsevier.com/retrieve/pii/S0925231219301067 (visited on 05/18/2020).

[79] Yuebing Zhang et al. "Three-way enhanced convolutional neural networks for sentence-level sentiment classification". In: *Information Sciences* 477 (Mar. 2019), pp. 55–64. ISSN: 00200255. DOI: 10.1016/j.ins.2018.10.030. URL: https://linkinghub.elsevier.com/retrieve/pii/S002002551830848X (visited on 05/18/2020).

[80] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. "Understanding Convolutional Neural Networks for Text Classification". In: *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 56–65. DOI: 10.18653/v1/W18-5408. URL: http://aclweb.org/anthology/W18-5408 (visited on 05/31/2020).

[81] Yuzhou Liu et al. "App store mining for iterative domain analysis: Combine app descriptions with user reviews". In: *Softw: Pract Exper* 49.6 (June 2019), pp. 1013–1040. ISSN: 0038-0644, 1097-024X. DOI: 10.1002/spe.2693. URL:

https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2693 (visited on 01/15/2020).

[82] S. Scalabrino et al. “Listening to the Crowd for the Release Planning of Mobile Apps”. In: *IEEE Transactions on Software Engineering* 45.1 (Jan. 1, 2019), pp. 68–86. DOI: 10.1109/TSE.2017.2759112.

[83] Xiaodong Gu and Sunghun Kim. “"What Parts of Your Apps are Loved by Users?" (T)”. In: *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). Lincoln, NE, USA: IEEE, Nov. 2015, pp. 760–770. ISBN: 978-1-5090-0025-8. DOI: 10.1109/ASE.2015.57. URL: http://ieeexplore.ieee.org/document/7372064/ (visited on 12/09/2019).

[84] Shuhua Monica Liu and Jiun-Hung Chen. “A multi-label classification based approach for sentiment classification”. In: *Expert Systems with Applications* 42.3 (Feb. 2015), pp. 1083–1093. ISSN: 09574174. DOI: 10.1016/j.eswa.2014.08.036. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417414005181 (visited on 05/19/2020).

[85] Jesse Read et al. “Classifier chains for multi-label classification”. In: *Mach Learn* 85.3 (Dec. 2011), pp. 333–359. ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-011-5256-5. URL: http://link.springer.com/10.1007/s10994-011-5256-5 (visited on 05/19/2020).

[86] Thi-Ngan Pham et al. “Hidden Topic Models for Multi-label Review Classification: An Experimental Study”. In: *Computational Collective Intelligence. Technologies and Applications*. Ed. by Costin Bădică, Ngoc Thanh Nguyen, and Marius Brezovan. Red. by David Hutchison et al. Vol. 8083. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 603–611. ISBN: 978-3-642-40494-8 978-3-642-40495-5. DOI: 10.1007/978-3-642-40495-5_60. URL: http://link.springer.com/10.1007/978-3-642-40495-5_60 (visited on 05/19/2020).

[87] Barbara Kitchenham et al. “Systematic literature reviews in software engineering – A systematic literature review”. In: *Information and Software Technology* 51.1 (Jan. 2009), pp. 7–15. ISSN: 09505849. DOI: 10.1016/j.infsof.2008.09.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S0950584908001390 (visited on 01/15/2020).

[88] V. T. Dhinakaran et al. “App Review Analysis Via Active Learning: Reducing Supervision Effort without Compromising Classification Accuracy”. In: *2018 IEEE 26th International Requirements Engineering Conference (RE)*. 2018 IEEE 26th International Requirements Engineering Conference (RE). Aug. 20, 2018, pp. 170–181. DOI: 10.1109/RE.2018.00026.

[89] A. Puspaningrum, D. Siahaan, and C. Fatichah. “Mobile App Review Labeling Using LDA Similarity and Term Frequency-Inverse Cluster Frequency (TF-ICF)”. In: *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*. 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE). July 24, 2018, pp. 365–370. DOI: 10.1109/ICITEED.2018.8534785.

[90] L. Yu et al. “Localizing Function Errors in Mobile Apps with User Reviews”. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). June 25, 2018, pp. 418–429. DOI: 10.1109/DSN.2018.00051.

[91] Y. Wang, H. Wang, and H. Fang. “Extracting User-Reported Mobile Application Defects from Online Reviews”. In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. 2017 IEEE International Conference on Data Mining Workshops (ICDMW). Nov. 18, 2017, pp. 422–429. DOI: 10.1109/ICDMW.2017.61.

[92] Y. Man et al. “Experience Report: Understanding Cross-Platform App Issues from User Reviews”. In: *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*. 2016 IEEE 27th International

Symposium on Software Reliability Engineering (ISSRE). Oct. 23, 2016, pp. 138–149. DOI: `10.1109/ISSRE.2016.27`.

[93] *Metrics and scoring: quantifying the quality of predictions.* URL: `https://scikit-learn.org/stable/modules/model_evaluation.html`.

[94] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empir Software Eng* 14.2 (Apr. 2009), pp. 131–164. ISSN: 1382-3256, 1573-7616. DOI: `10.1007/s10664-008-9102-8`. URL: `http://link.springer.com/10.1007/s10664-008-9102-8` (visited on 05/28/2020).

[95] Jill Collis and Roger Hussey. *Business research: A practical guide for undergraduate and postgraduate students.* Macmillan International Higher Education, 2013.

[96] Paul D. Leedy, Jeanne Ellis Ormrod, and Laura Ruth Johnson. *Practical research: planning and design.* Twelfth edition. NY, NY: Pearson, 2019. 441 pp. ISBN: 978-0-13-477565-4.

[97] Carrie Williams. "Research Methods". In: *JBER* 5.3 (Feb. 7, 2011). ISSN: 2157-8893, 1542-4448. DOI: `10.19030/jber.v5i3.2532`. URL: `https://clutejournals.com/index.php/JBER/article/view/2532` (visited on 05/30/2020).

[98] Carl D. McDaniel and Roger H. Gates. *Contemporary marketing research.* 4. ed. OCLC: 245664238. Cincinnati: South-Western College Publ, 1999. ISBN: 978-0-538-88507-2 978-0-324-00603-2 978-0-324-00602-5.

[99] Emma Bell and Alan Bryman. "The Ethics of Management Research: An Exploratory Content Analysis". In: *Br J Management* 18.1 (Mar. 2007), pp. 63–77. ISSN: 1045-3172, 1467-8551. DOI: `10.1111/j.1467-8551.2006.00487.x`. URL: `http://doi.wiley.com/10.1111/j.1467-8551.2006.00487.x` (visited on 05/30/2020).

[100] Pervez N. Ghauri and Kjell Grønhaug. *Research methods in business studies: a practical guide.* 2nd ed. Harlow, England ; New York: Financial Times Prentice Hall, 2002. 212 pp. ISBN: 978-0-273-65110-9.

[101] John W. Creswell. *Research design: qualitative, quantitative, and mixed methods approaches.* 4th ed. Thousand Oaks: SAGE Publications, 2014. 273 pp. ISBN: 978-1-4522-2609-5 978-1-4522-2610-1.

[102] Herbert A. Simon. *The sciences of the artificial.* URL: `https://monoskop.org/images/9/9c/Simon_Herbert_A_The_Sciences_of_the_Artificial_3rd_ed.pdf`.

[103] Suhaib Mujahid et al. "Examining User Complaints of Wearable Apps: A Case Study on Android Wear". In: *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems.* MOBILESoft '17. event-place: Buenos Aires, Argentina. Piscataway, NJ, USA: IEEE Press, 2017, pp. 96–99. ISBN: 978-1-5386-2669-6. DOI: `10.1109/MOBILESoft.2017.25`. URL: `https://doi.org/10.1109/MOBILESoft.2017.25`.

[104] Y. Lai, F. Huang, and P. Chiou. "Analysis of user feedback in the mobile app store using text mining: A case study of Google Fit". In: *2017 IEEE 8th International Conference on Awareness Science and Technology (iCAST).* 2017 IEEE 8th International Conference on Awareness Science and Technology (iCAST). Nov. 8, 2017, pp. 50–54. DOI: `10.1109/ICAwST.2017.8256508`.

[105] Cuiyun Gao et al. "INFAR: Insight Extraction from App Reviews". In: *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.* ESEC/FSE 2018. event-place: Lake Buena Vista, FL, USA. New York, NY, USA: ACM, 2018, pp. 904–907. ISBN: 978-1-4503-5573-5. DOI: `10.1145/3236024.3264595`. URL: `http://doi.acm.org/10.1145/3236024.3264595`.

[106] Jenq-Haur Wang et al. "An LSTM Approach to Short Text Sentiment Classification with Word Embeddings". In: *Proceedings of the 30th Conference on Computational Linguistics and Speech Processing (ROCLING 2018)*. Hsinchu, Taiwan: The Association for Computational Linguistics and Chinese Language Processing (ACLCLP), Oct. 2018, pp. 214–223. URL: `https://www.aclweb.org/anthology/O18-1021`.

[107] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]* (Sept. 6, 2013). arXiv: `1301.3781`. URL: `http://arxiv.org/abs/1301.3781` (visited on 06/01/2020).

[108] Ronan Collobert et al. "Natural Language Processing (almost) from Scratch". In: *arXiv:1103.0398 [cs]* (Mar. 2, 2011). arXiv: `1103.0398`. URL: `http://arxiv.org/abs/1103.0398` (visited on 05/27/2020).

[109] Karen Sparck Jones. "A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL". In: *Journal of Documentation* 28.1 (Jan. 1972), pp. 11–21. ISSN: 0022-0418. DOI: `10.1108/eb026526`. URL: `https://www.emerald.com/insight/content/doi/10.1108/eb026526/full/html` (visited on 06/01/2020).

[110] Grigorios Tsoumakas et al. "Correlation-based pruning of stacked binary relevance models for multi-label learning". In: *Proceedings of the 1st international workshop on learning from multi-label data*. 2009, pp. 101–116.

[111] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[112] David G Kleinbaum et al. *Logistic regression*. Springer, 2002.

[113] Shantanu Godbole and Sunita Sarawagi. "Discriminative methods for multi-labeled classification". In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2004, pp. 22–30.

[114] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of machine learning research* 12 (Oct 2011), pp. 2825–2830.

[115] Forrest Shull, Janice Singer, and Dag IK Sjøberg. *Guide to advanced empirical software engineering.* Springer, 2007.

[116] Aida Ali et al. "Classification with class imbalance problem: a review". In: *Int. J. Advance Soft Compu. Appl* 7.3 (2015), pp. 176–204.

[117] Yoav Goldberg. "A Primer on Neural Network Models for Natural Language Processing". In: *jair* 57 (Nov. 20, 2016), pp. 345–420. ISSN: 1076-9757. DOI: `10.1613/jair.4992`. URL: `https://jair.org/index.php/jair/article/view/11030` (visited on 05/27/2020).

[118] Yoshua Bengio et al. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (null Mar. 2003). Publisher: JMLR.org, pp. 1137–1155. ISSN: 1532-4435.

Appendix A

SLR Protocol

A.1 Electronic Databases

The electronic data sources mentioned in table used for the systematic literature review. These data sources are popular and the most used data sources to publish good and quality research. Google Scholar data source used for manual keyword-based searching.

ED#	**Database**	**URL**
ED1	ACM	http://dl.acm.org/
ED2	IEEE Xplore	http://ieeexplore.ieee.org/
ED3	Science Direct	http://sciencedirect.com/
ED4	Springer Link	http://link.springer.com/
ED5	Wiley	http://onlinelibrary.wiley.com/
ED6	Google Scholar	https://scholar.google.com.pk/

A.2 Searched Strings over each database

To search the relevant work done by the research community, the key terms are the main ingredient for adequate research of literature. Search key terms divided into three parts: population, intervention, and outcome as suggested by the Kitchenham

[87]. Based on the identified search terms and using the suggestions of Kitchenham, we designed a structured query to search the databases. The query for each data base vary in the structure due to different search policies of digital libraries. The table shows the executed queries on each database.

Database name	**Searched String**
IEEE Explore	(("mobile app*" OR "online app*" OR "smartphone app*") AND (review OR comment* OR feedback OR opinion) AND (analy* OR mining))
ACM	(("mobile app*" OR "online app*" OR "smartphone app*" OR "android app*" OR "ios app*" OR "apple app store") AND (review OR comment* OR feedback OR opinion) AND (analy* OR mining))
Science Direct	(("mobile app*" OR "online app*" OR "smartphone app*" OR "android app*" OR "ios app*" OR "apple app store") AND (review OR comment* OR feedback OR opinion) AND (analy* OR mining))
Springer Link	(("mobile app*" OR "online app*" OR "smartphone app*") AND (review OR comment* OR feedback OR opinion) AND (analy* OR mining)) Searched within search: 'mobile app user review feedback comment analysis mining'
Wiley	"(("mobile app*" OR "online app*" OR "smartphone app*" OR "android app*" OR "ios app*" OR "apple app store") AND (review OR comment* OR feedback OR opinion) AND (analy* OR mining))" anywhere

A.3 Inclusion and Exclusion Criteria

These criteria used to find potentially related studies from data sources to response research questions in our SLR (systematic literature review).

A.4 Quality Assessment Criteria

From a rigorous review and screening, 20 papers were selected by the joint application of inclusion/exclusion and QA criteria on the full text of 60 research articles. This ensured additional confidence in the selected articles.

IC#	**Description**
IC1	Articles that are peer-reviewed
IC2	Abstract of study explicitly mentioned review analysis in the context of mobile applications
IC3	Novelty and total relevance will be considered in case of no validation
IC4	Include studies from 2008 to 2019
IC5	Inclusion of available Full-text articles
EC#	**Description**
EC1	Studies other than the English language
EC2	Articles with no validation of proposed techniques
EC3	Articles providing unclear results and findings
EC4	Articles with no full-text availability after all possible sources
EC5	Articles less than 5 pages
EC6	Review analysis of social media or web sites
EC7	All secondary studies

QC#	**Description**
QC1	Objectives and context are clearly specified.
QC2	Are the Research design and conclusion support the objectives of the study?
QC3	Is there a clear statement of Limitations and Contributions?
QC4	Is there any validation of the proposed method/ approach?

A.5 Prisma Flow

Studies selected through the PRISMA model of selection study, standard guidelines of SLR by Kitchenham are followed. The IC and EC were applied in the 2nd and 3rd step of the study selection process. In the 2nd phase, abstract and conclusion considered to apply inclusion/exclusion criteria.

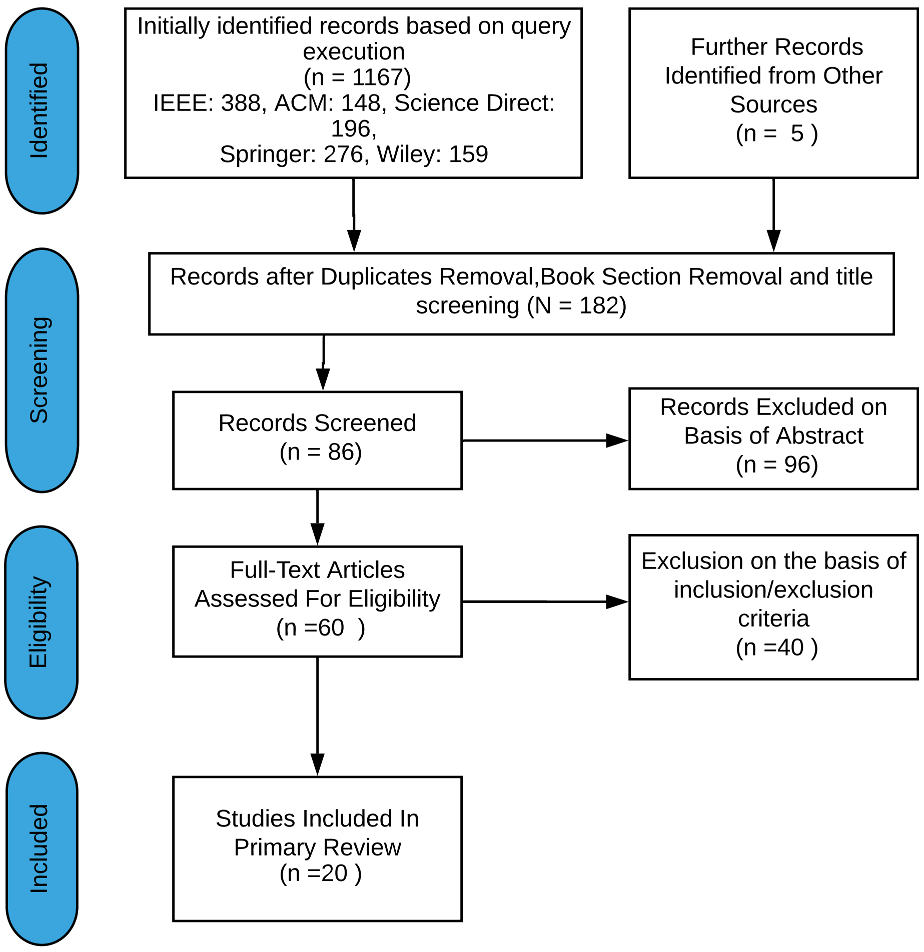

Identified
Initially identified records based on query execution
(n = 1167)
IEEE: 388, ACM: 148, Science Direct: 196,
Springer: 276, Wiley: 159
Further Records Identified from Other Sources
(n = 5)
Screening
Records after Duplicates Removal,Book Section Removal and title screening (N = 182)
Records Screened
(n = 86)
Records Excluded on Basis of Abstract
(n = 96)
Eligibility
Full-Text Articles Assessed For Eligibility
(n =60)
Exclusion on the basis of inclusion/exclusion criteria
(n =40)
Included
Studies Included In Primary Review
(n =20)

Figure A.1: Prisma Flow