An Automated System for Identification of Useful User Reviews for Mobile Application Development

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

IN THE FACULTY OF SCIENCE AND ENGINEERING

July 2021

By

Mohammadali Tavakoli

Department of Computer Science

Contents

Word Count: 39,914

List of Figures

List of Tables

Abstract

In recent years, mobile app reviews are known to provide a rich source of user feedback which is of great value for software evolution. However, the volume of such user reviews is huge, particularly for famous applications and large companies offering several applications. Addressing this issue, several automatic approaches are proposed recently for identifying useful reviews. The applied criteria for measuring the review usefulness in these approaches are originated from the few existing exploratory studies, wherein the usefulness of a review is interpreted as inclusion of requirement engineering related topics. Such interpretations of usefulness, however, is based on authors' understanding of usefulness rather than developers' requirements. Ignoring developers' viewpoint, the authors defined some usefulness metrics based on their own observations, and developed extraction approaches accordingly. Thus, expecting interesting results from such approaches for developers dealing with thousands of reviews daily is awkward. To bridge this gap in this study, related studies across several domains analysing human generated feedback, such as reviews, tweets, requirement notes, bug reports, and application testing reports, were perused to define a set of factors for accurately measuring the usefulness of user reviews. The usefulness factors were, then, validated in a focus group discussion session by experienced mobile app developers. Next, the task of extracting each of the approved factors was automated applying Deep Learning and Natural Language Processing (NLP) techniques. Finally, the models designed for extracting each factor were integrated to form a final system for automatically extracting useful reviews. Testing on different review datasets, the novel system achieved high accuracy (i.e., Aspects: 87%, Feature Requests: 72%, Issues: 67%, User Actions: 73%, and System Actions: 81%) and outperformed state-of-the-art extraction techniques. Moreover, unlike the state-of-the-art, the proposed system is completely aligned with developers' viewpoint as it emphasises on developers' approved factors for measuring the usefulness.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Submitted in 2021

Copyright

Dedication

This thesis is dedicated to Imam Hasan ibn Ali (peace be upon him) for his generosity and favours.

Acknowledgements

I express my heartfelt gratitude to my compassionate wife, Atefeh Heydari, for her countless sacrifices and encourages over these years. She sacrificed her own wishes and provided me with encouragement and support to complete this thesis successfully. She has also been doing her PhD which is admirable. My special thanks to her again

In addition, I would like to thank my courageous son Amirabbas for his lovely smiles and for tolerating my absence while I was studying. Also, my resplendent daughter, Ellarose, welcome your birth with love.

My deepest gratitude goes to my supervisors, Dr. Liping Zhao and Prof. Goran Nenadic, for all their unwavering support and tremendous guidance to my PhD. Their guidance, experience, patience, and clarity have been invaluable.

In addition, I express my appreciation to The University of Manchester and School of Computer Science for supporting my research over three years granting me a President's Doctoral Scholar Award and a School of Computer Science Kilburn Award.

Finally, I would like to thank the examiners of my oral examination, Dr. Sandra Sampaio and Dr. Hoshang Kolivand, for their constructive comments helping me to improve the presentation of this research.

Publication List

- Mohammadali Tavakoli, Liping Zhao, Atefeh Heydari, Goran Nenadic, "Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools," *Expert Systems with Applications,* 2018.

Blank Page

Chapter 1 Introduction

1.1 Introduction

Mobile application reviews reflect complaints, expectations, and requirements of users which is of great importance to developers. They can assist developers in diagnosing and updating the target app effectively. In this thesis, a novel system is proposed to automatically measure the usefulness of a given app review with respect to the developers' viewpoint. To measure the usefulness, five novel usefulness factors are proposed to be extracted from review text. The system, then, labels the review into useful or useless based on the extracted factors.

Section 1.2 discusses the problem background. The research problem to be addressed in this thesis is stated in Section 1.3. Section 1.4 explains the research methodology used in this study. Finally, Section 1.5 outlines and introduces chapters included in the thesis.

1.2 Problem Background

In order to develop effective and efficient software systems, incorporating users' requirements and needs is significant [1-3]. User involvement in development of software systems leads to not only high-quality software, but also user satisfaction [4, 5], and maximising the amount of application sales [6].

Addressing user expectations in development process, software development teams frequently involve users in various development stages [7] by gathering early feedback and expectations for prototyping the system [8], and interfering users in design [9] and implementation [10, 11] phases. Therefore, early stages of software

development have been targeted by majority of previous research in extraction of user needs [12]. However, recent studies have pointed out that involvement of user feedback and requirements in post deployment phases is also crucial for maintaining a software system up to date [13], improving upcoming releases [14], and obtaining user satisfaction [15].

In recent years, mobile app reviews are known to provide a rich source of user feedback. Web-based software application distribution platforms have become increasingly popular among the Internet users. According to statista.com, the number of mobile app downloads from Google Play and App Store has increased from 17 billion in 2013 to 218 billion in 2020. These platforms also allow users to share their opinions on their downloaded apps. From the user perspective, such opinions can influence their decision on purchasing or choice of a particular app [16, 17].

From the app provider perspective, positive reviews can attract more customers and bring financial gains [17]. Similarly, negative reviews often cause sales loss [18, 19]. Instructive feedback written by app users usually contains valuable information, such as reporting bugs and seeking causes, offering solutions, complaining about cost or performance, requesting features to be added to the application, explaining their personal experiences with different parts of the application, and appreciating developers, to name a few. Application vendors and developers are enthusiastic about mining and analyzing such contents [20]. Interviewing 73 developers, Palomba et al. [21] found that over 75% of developers carefully monitor user reviews to improve their applications. Addressing these requirements is also of paramount importance. Carefully responding to user needs discussed in reviews in a reasonable period of time is known as a key factor for any successful mobile application [22].

However, there are several restrictions for developers effortlessly reaping benefits of user feedback. Firstly, the quality of reviews varies significantly which is due to the openness of online review sharing websites for everyone [23]. Many people generate reviews for asking question from or answering to other users, advertising other applications, or repeating their assigned star rating in words.

Besides, some of the applications require a certain level of expertise causing ordinary users encountering difficulties using them. Reviews generated by these users are not usually interesting for developers. Secondly, reliability of reviews could not be guaranteed. Stakeholders sometimes hire spammers to generate not only glamorized positive reviews on their applications, but also harmful negative reviews on competitors' [24, 25]. Finally, huge amount of generated reviews and the unstructured nature of its textual content demand a lot of effort and cost to be manually analysed by development teams.

1.3 Problem Statement

In recent few years, researchers have developed several techniques, ranging from sentiment analysis [26-28] and spam detection [23, 29-31] to more general mining techniques [32-34] for automating the task of processing user reviews to reduce the developers' effort spent in collecting and understanding informative user feedback from app reviews. The proposed review mining approaches automatically discern relevant reviews from irrelevant ones have been systematically analysed before setting out to address the research aim of this project.

The main shortcoming of these studies is interpretation of the usefulness. Many of the authors applied their own opinion for defining usefulness metrics [35, 36], while others hired improper persons, such as undergraduates studying software engineering, to annotate reviews, and defined usefulness metrics inspiring from the annotation output [37, 38]. Such behaviours have resulted in disregarding the viewpoint of developers in identifying useful reviews. Therefore, the output of such research projects might not be in line with what developers trying to achieve. After analysing the state-of-the-art, the lack of research on properly measuring usefulness of app reviews with respect to developers' viewpoint became apparent.

1.4 Aims and Objectives of the Research

After conducting a systematic review of the existing literature to discover existing research efforts and proposed approaches in analysing mobile application reviews, research gaps and limitations were identified. Accordingly, the following aim was defined.

The aim of the research presented in this thesis is to bridge the abovementioned gap found in current approaches by conceptualising the usefulness and proposing Deep Learning and NLP based approaches for effectively measuring the usefulness of mobile app reviews for software development purposes. To achieve this, the key research questions are defined:

RQ1. How to effectively measure the usefulness of app reviews? What usefulness factors could be used?

RQ2. How to validate the proposed factors for measuring the usefulness of app reviews?

RQ3. How to automatically extract the usefulness factors from the text of app reviews?

RQ4. How to automatically detect useful reviews using the usefulness factors?

RQ5. How to evaluate the proposed automatic approaches?

With the aim of systematically addressing these research questions, the objectives of this project are defined as follow:

Obj1. To conceptualise and formulate the usefulness attribute for app reviews. This is to quantifiably measure the usefulness of app reviews using a set of usefulness factors.

To fulfil this objective, a list of candidates for usefulness factors for measuring the usefulness of app reviews was defined inspiring from in-depth analysis of related work on several domains and the dataset of app reviews.

Obj2. To validate the effectiveness of usefulness factors using experienced mobile application developers. This validation will help not only to ensure the inclusion of developers' viewpoint in measuring the usefulness, but also to approve the effectiveness of such factors in measuring the usefulness.

To achieve this, the list of candidates of usefulness factors was evaluated in a Focus Group Discussion (FGD) composed of six experienced mobile app developers. The output of the FGD was a set of five usefulness factors. Proposing five effective

and approved factors for measuring the usefulness of app reviews is one of the contributions of this thesis.

Obj3. To propose Deep Learning and NLP based approaches for extracting each usefulness factor from reviews. This is not only to check the applicability of the factors in real world, but also to facilitate development of the final system.

Neural networks, NLP techniques, and combination of them was used to automatically extract each usefulness factor from the text of a given review. Another contribution of this thesis is proposing these extraction approaches that achieved higher accuracy compared to state-of-the-art. Besides, extraction of User Action and System Action are automated for the first time in this domain.

Obj4. To integrate extraction approaches as a unique system for detecting useful reviews.

Successfully proposing automatic extraction approaches for each of the usefulness factors, the models built for extracting the factors were placed in a pipeline to form the whole system. The system gets a review as input, extracts usefulness factors, and applying a decision tree-based approach estimates the usefulness of the review. Proposing a fully automated system for measuring the usefulness of app reviews completely aligned with developers' viewpoint is another contribution of this thesis.

Obj5. To evaluate the integrated system and individual extraction approaches.

To measure the effectiveness of the usefulness factors individually and as a whole system, datasets of app reviews manually coded by app developers as a part of this project were used. The achieved Precision, Recall, and F-Score reported in Chapter 7 shows how the proposed approach outperforms the state-of-the-art in this domain.

1.5 Thesis Context

To provide a clear description of this study, a context diagram is illustrated in Figure 1.1 representing the activities of this project. The diagram is designed based on Design Science Research Methodology [42] which consists of the following steps.

1. Identifying the problem,

2. Providing evidence for the existence of this problem,
3. Making a testable hypothesis about how to solve the problem and/or formulating research questions,
4. Designing and developing a solution,
5. Demonstrating and evaluating the solution, and
6. Communicating the results to research community.

The diagram consists of three main phases (i.e., Literature review, Proposing the Approach, and Validation) to properly fulfil the objectives defined in previous section and meet the aim of this PhD project.

There are three tasks to be addressed in the Literature Review phase: First, to identify the research problem and provide evidence for the existence of this problem. The research problem stated in Section 1.3 was identified by identifying and perusing related work in the area of app review analysis and requirements engineering followed by a technical discussion with scholars and professionals in the domain. Providing evidence for existence of such problem, discussed in Chapter 2, is another part of this task. This task was the cornerstone of this research forming the overall aim and initiating the research questions.

Second, studies related to user feedback analysis are collected and reviewed for extracting any factor used to measure the usefulness of reviews. In a broader scale, studies reporting on analysing user requirements for software development are also analysed. This task is to fulfil the first objective for conceptualising the usefulness of app reviews.

Finally, existing datasets are collected and assessed in this phase for the purpose of validating any further implemented approach. This task is designed to address the last objective about evaluating the approaches.

The Approach phase contributes to objectives 1, 3, and 4. In this phase, the usefulness of app reviews are to be measured (Obj1) defining a set of usefulness factors to be used as metrics for judging the usefulness. These factors are then discussed and validated in the FGD of well-skilled app developers and requirements engineers (Obj2) as a part of Validation phase.

Moreover, the task of automatically extracting the usefulness factors using machine learning and NLP techniques is one of the tasks defined in the Approach phase to fulfil Obj3. This is also a step designed, according to step 4 of the abovementioned methodology, to provide solution for the research problem.

The last phase of this study, Validation, is responsible for validating any approach developed in the former phases. The tasks assigned in this phase are designed to address objectives 2 and 5.

The first major task in this phase to address objective 2 is conducting a FGD with professional mobile application developers to validate significance of usefulness factors defined in phase 1. The output of this FGD is a set of professionally approved usefulness factors for measuring the usefulness of app reviews.

The second task to fulfil objective 5 is manual coding of a set of real mobile application reviews to validate the accuracy of the proposed approaches from the previous phase. The approaches are designed to automatically extract the usefulness factors that are manually labelled in the ground truth dataset. Two datasets of app reviews obtained from related work were manually annotated with the usefulness factors by experienced app developers to form the ground truth datasets of this project. The datasets are then used to train and test the models. So, the accuracy of the approaches is measured by comparing their outputs against the ground truth dataset labels.

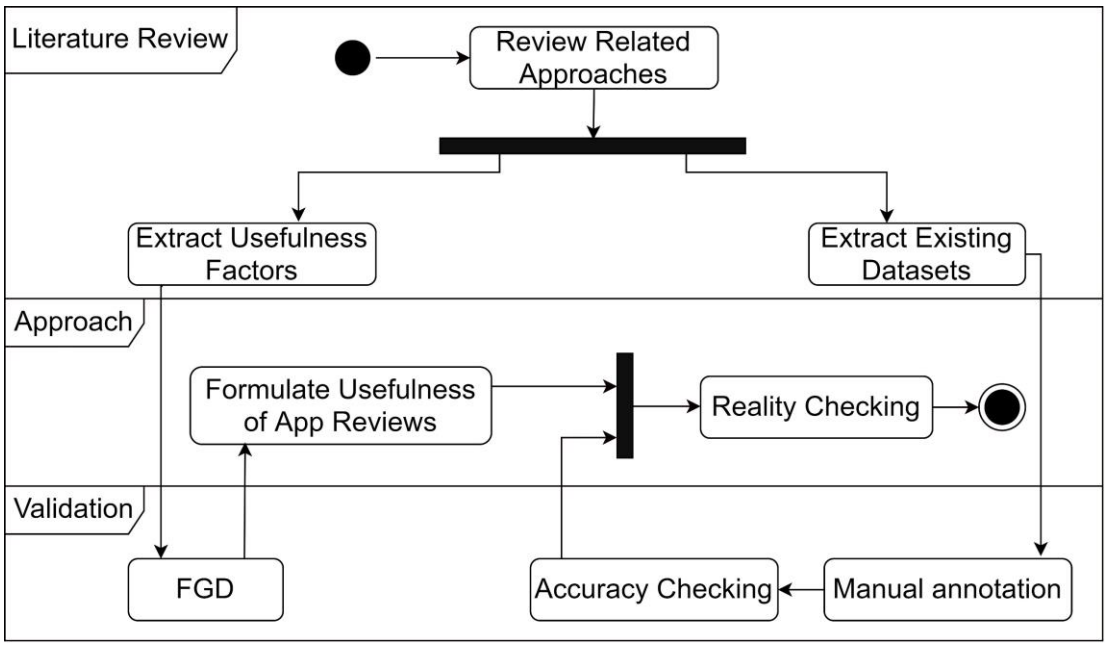

Figure 1.1: The Activity diagram of this study

1.6 Thesis Outline

This thesis is organised into six chapters as follow:

- Chapter 2. Literature Review: A systematic literature review of current approaches analysing application reviews for software evolution is presented. The research gaps are identified and demonstrated to be alleviated with a novel approach.

- Chapter 3. Identification of Factors Relevant to the Usefulness of App Reviews: To address the identified gap in Chapter 2, any possible metrics to be used for measuring the usefulness of application reviews from developers' viewpoint were gathered from related approaches in variety of domains, analysed carefully, and reported in this chapter.
- Chapter 4. Validation of the Usefulness Factors: The list of usefulness factors was discussed in a Focus Group Discussion (FGD) session with senior mobile application developers for validation and amendments. The results are reported in this chapter.

- Chapter 5. Modelling the Usefulness of Application Reviews: This chapter describes an app review and available data and metadata associated to it for analysis. The partial process of reviewing an application is also conceptualised in this chapter.
- Chapter 6. The Proposed Approach: This chapter discusses the proposed approaches for automatically extracting each usefulness factor and integrating them in a pipeline to form a final system. In this chapter, the architecture of the proposed approach is illustrated, data pre-processing steps are explained, and the strategy for measuring the usefulness is discussed.
- Chapter 7. Results and Discussion: Apart from explaining the datasets used for this experiment, in this chapter, results obtained from testing each of the proposed approaches for extracting usefulness factors are discussed. The final system is also validated, and the results are presented in this chapter. The chapter also covers a detailed discussion on the obtained results.
- Chapter 8. Research Limitations and Challenges: Challenges and limitations encountered in different phases of the project are reported in this chapter.
- Chapter 9. Conclusion: This chapter provides a summary of the thesis emphasising on addressing research questions. It also discusses the future work and possible research directions.

1.7 Conclusion

To provide readers with a general understanding and an overview of the research reported in this thesis, the following key concepts and elements of the project are discussed in this chapter.

After providing a general introduction to the field of research, the problem investigated in this project is discussed. Aim, research questions, and objectives are elaborated to reflect contributions and novelties of this project. Finally, the structure of the thesis is outlined after discussing the context of the research.

Chapter 2 Related Work

This chapter is based on a paper published in The Journal of Expert Systems With Application [39] in 2018. This survey highlights and compares the areas of research that have been explored thus far, drawing out new directions future research could take to address open problems and challenges.

2.1 Introduction

Extraction of useful information for software evolution from mobile app reviews is crucial because of the importance of user post-release feedback on the success of the application [40]. However, not all the user reviews are relevant and useful for app development [41]. In order to unleash the value of app reviews for app development, intelligent mining tools that can help discern relevant reviews from irrelevant ones are required. In recent years, a variety of such techniques have been proposed, ranging from sentiment analysis [26], spam detection [23, 29], to more general mining techniques [32]. Yet, there is a lack of systematic understanding of these techniques in the context of mining mobile app reviews and their support tools.

One of the related survey studies assessing App Store mining techniques by Nayebi and Abran [42] provides an analysis of general data-mining techniques for spam detection, opinion mining, review evaluation, and feature extraction. The second survey by Martin et al. [43] is concerned with App Store analysis, such as API analysis, feature analysis, app review analysis, and so on. So, investigation of the utilised techniques and proposed tools for mining software development related information from app reviews was not covered in these survey studies.

In this chapter, specific mining techniques and tools for processing app reviews are analysed and reported, their shortcomings and gaps are identified, and the future research directions are highlighted. There are several reasons why this literature review is conducted. First, it provides a background on processing techniques used for analysing app reviews. So, it represents research efforts in this area. Second, the research problem was formulated after analysing existing

approaches and identifying their gaps and shortcomings. Third, it provides a systematic support for the novelty and contribution of this project. Finally, it has several other advantages for readers, such as summarizing challenges, limitations, and future research direction in this area.

A process of analysing related works is also reported in Chapter 3, but the aim of that analysis is to identify potential usefulness factors to be used for measuring app review usefulness.

We found that the first related paper was published in 2011. So, the starting point of the study was set to 2011. The most recent selected study was published in 2017 at the time of writing the systematic literature review paper. Recent papers are also analysed and added to the set of selected papers to set the end point of the review to 2020.

Over 55 existing approaches dealing with analysing mobile application reviews have been investigated. Following an analysis of these approaches it was established that supervised machine learning-based approaches make up the majority of the current state-of-the-art.

2.2 Survey Methodology

The methodology for conducting our survey is systematic literature review proposed by Kitchenham et al. [44]. To avoid confusion, we have used the term "survey", instead of "review", to refer to this paper, as the subject matter of the paper is app reviews. Based on the systematic literature review (SLR) guidelines provided by Kichenham et al. (Barbara Kitchenham, et al. [44], in what follows, we describe the steps in our survey process.

2.3 Research Questions

As stated in Section 1, the main goal of this chapter is to survey the state of the art in the development of mobile app review mining techniques and tools. Specifically, the survey will cover the primary studies that report the development of mobile app review mining techniques and tools. In addition, this paper will also find out what specific app development topics the reported tools are used to discover, as this

finding will help us evaluate the maturity of the current mobile app mining tools. In line with these goals, we have formulated the following research questions and will use them to drive our survey:

RQ1: What types of analysis on mobile app reviews and what techniques have been reported in the literature?

RQ2: What software tools have been developed to support these techniques?

2.4 Identifying Relevant Primary Studies

The definition of a robust strategy for performing an SLR is essential as it enables researchers to efficiently retrieve the majority of relevant studies. In this section, we discuss our search strategy in detail.

We developed our search string by selecting keywords from the studies that we had already reviewed in the domain. Then, we identified and applied alternatives and synonyms for each term and linked them all by the use of AND/OR Boolean expressions to cover more search results. In order to perform a widespread search, formulating a comprehensive query is indispensable. Thus, we optimized and refined our preliminary search string during multiple iterations as mulling over the revealed results and skimming retrieved relevant studies helped us to manipulate our search string effectively with more appropriate keywords. We excluded keywords whose inclusion was not advantageous or replaced them with more apt ones. Our search terms are presented in Table 2.1 and the finalized search term is as follow:

'(mobile OR android OR IOS) AND (app OR application) AND (feedback OR review OR opinion OR comment) AND (bug report OR feature request OR complain OR requirement OR issue OR expectation) AND (analysis OR process OR mining OR extract OR discover) AND (developer OR development team OR requirements team OR software vendor) OR (requirement engineering OR RE OR requirements elicitation OR requirements analysis)'

After constructing the search string, we used it to search the following databases: ScienceDirect, IEEEXplore, ACM, GoogleScholar, Scopus, and

SpringerLink. The search returned a total of 59,066 results. We found that the first related paper was published in 2011 and the last one in 2020. Table 2.2 summarizes the search results.

Table 2.2: Types and Components of The Search Strings

Type	Search Term
Domain	(mobile OR android OR IOS)
Content	(app OR application)
Review	(feedback OR review OR opinion OR comment)
Review Type	(bug report OR feature request OR complain OR requirement OR issue OR expectation)
Development	(developer OR development team OR requirements team OR software vendor)
Requirement Engineering	(requirement engineering OR RE OR requirements elicitation OR requirements analysis)
Technique	(analysis OR process OR mining OR extract OR discover)

Table 2.1: Search Results (2011 – 2020)

Database	Search Results
Scopus	10,135
Science Direct	5,118
ACM	4,128
SpringerLink	23,069
IEEEXplore	4,133
Google Scholar	12,483
Total	59,066

2.5 Selecting Relevant Primary Studies

Based on the search results, we have used the following inclusion and exclusion criteria to select the relevant primary studies. The selection phases are shown in Figure 2.1.

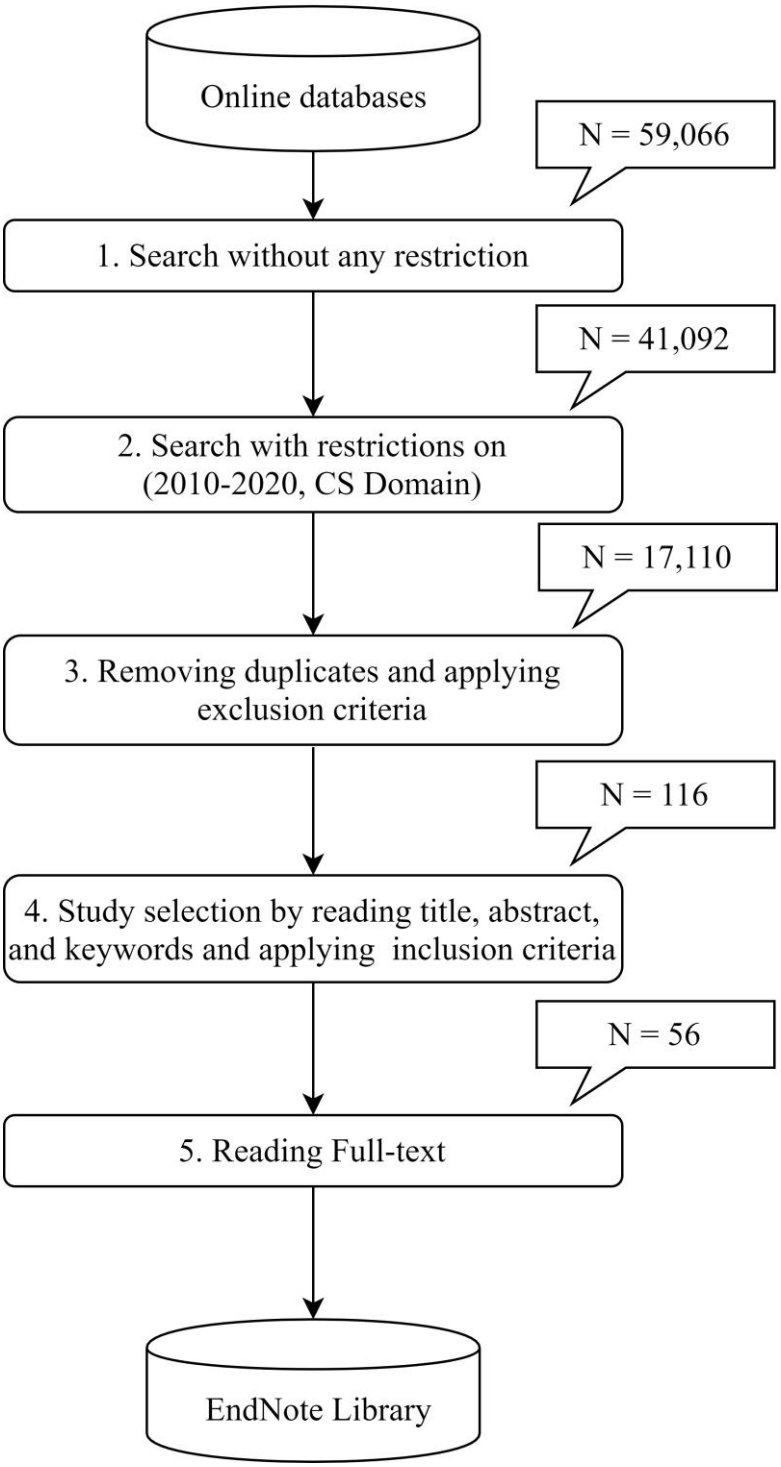

Figure 2.1: Phases in the selection process. N denotes the number of papers.

Inclusion criteria:

I1. Studies reporting app reviews related to application development were included.

I2. In addition to I1, studies reporting detailed empirical research methods, such as case studies, surveys, experiments, and ethnographical studies, were included.

I3. If more than one paper reports the same study, only the latest or fullest paper was included.

Exclusion criteria:

E1. White and gray literature (i.e. research outputs that are not peer reviewed, such as reports, online documents, and working papers) was excluded.

E2. Abstract papers with no full-text available were excluded.

E3. Short papers with less than four pages were excluded.

These inclusion and exclusion criteria were applied in the following three steps:

1. E1, E2 and E3 were applied in turn to the search results to exclude irrelevant studies.
2. I1 and I2 were applied to each remaining study to include the studies that met these criteria.
3. I3 was applied to duplicate studies to include the fullest studies.

At the end of Step 3, a total of 56 primary studies were selected as relevant to our survey, the full text of which was imported into an Endnote library for data extraction. The references of these studies are listed in Appendix I.

Figure 2.2 shows the distribution of the selected 56 studies published from 2011 to 2020. The maximum number of papers was published in 2015, whereas 2011 only had one paper. It is worth mentioning that the small number of the papers found in 2020 is due to the search period. Of the 56 selected papers, 39 of the selected papers are conference papers and 18 of them are journal articles. Appendix II summarizes the number of the papers published in each channel.

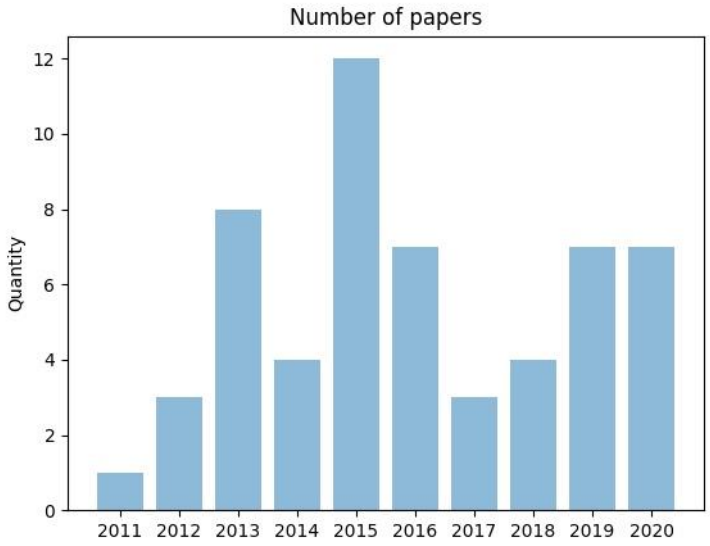

Figure 2.2: Distribution of the selected 56 primary studies from 2011 to 2020

2.6 Extracting and Synthesizing Data

In this step, the required data were extracted from each of the 56 primary studies. A predefined data extraction form (see Table 2.3) was used to record the data for each study. Two types of data were extracted: the data required for answering the research questions and the data required for displaying the bibliographic information of the study. The extracted data were stored in an Excel file for further process and analysis.

The extracted data were synthesized using the constant comparison method (CCM) [45]. The steps involved in data synthesis were:

1. Comparison within a single paper: To summarize the core of the paper and to understand, any categories, difficulties, and highlights.
2. Comparison between papers within the same category that is papers, which used same techniques or had same aims: To conceptualize the subject and to produce a category of studies.
3. Comparison of papers from different groups: To identify the effectiveness and efficiency of each category of techniques in solving the overall issue.

Table 2.3: Data Extraction Form

Data Item	**Description**
Paper ID	The unique ID assigned to each paper
Year	In which year was the study published?
Author(s)	The author(s) of the paper
Title	The title of the paper
Venue	Publication venue of the study
Techniques	Mining techniques used in the study
Tools	Support tools for extracting software development information
Topics	Software development topics discovered in the study

2.7 Survey Results

This section reports on the analysis of review results and answers our research questions.

2.7.1 Mobile App Review Mining Approaches and Techniques (RQ1)

RQ1: What types of analysis on mobile app reviews and what techniques have been reported in the literature?

The aim of this research question is to identify different types of analysis performed on user reviews. The main goal of analysing and processing mobile application reviews is to help developers in identifying useful reviews for application improvements and evolution. There are several purposes researchers have analysed user reviews for, such as categorizing user reviews in different groups based on the content, discovering what topics are discussed in reviews, opinion mining, finding relations between different review component, and so on. They used several techniques to perform these tasks that are discussed in this section. Apart from finding gaps to be addressed in this project, answering this research question provides a comprehensive technical background for factors extraction components of the proposed approach.

2.7.1.1 Discovering What Topics Are Discussed in Reviews

To better understand the content of mobile application reviews, selected studies in this group have analysed them to discover what topics are discussed in the text of reviews. In this section, each of the studies are explained in detail.

Ha and Wagner [46] (S6) manually analysed 556 reviews from 59 different applications of Google Play to find what users are talking about. They identified 18 categories of topics discussed in the reviews after multiple irritations. To understanding what recurring issues users like to report in their reviews, Iacob, et al. [47] (S7) manually analysed 3279 Google play reviews. The authors defined a coding scheme to capture the recurring issues. Two coders annotated 125 randomly selected reviews resulting in identification of 9 classes of codes, namely positive, negative, comparative, price related, request for requirements, issue reporting, usability, customer support, and versioning. Then they divided each review into significant snippets of text to assign them an associate refined code. Their approach is vague, as they have not described how the whole 3279 reviews had been processed.

In an exploratory study, to demonstrate how user reviews are useful for developers, Pagano and Maalej [48] (S12) investigated how and when users provide feedback, how to identify and classify topics in reviews, and how to integrate user feedback into requirements and software engineering infrastructures (more focus was on the impact of reviews on rating and on community of users). The authors applied a descriptive statistic to investigate usage of feedback. They, then, manually analysed a random sample of reviews (528 reviews) to explore and assign topics to each review. They grouped their 17 observed topics into four themes, namely community, rating, requirements, and user experience.

Another group of studies focused on analysing negative reviews to extract topics related to application problems discussed in them. In preliminary contribution in the field, Khalid [49] (S8) manually analysed 6,390 one-star and two-star reviews for 20 iOS apps in order to aid developers by listing the most frequent complaints. They discovered 12 types of issue and complaint about iOS apps in user feedback and repeated their approach in [50] (S15). In 2015, they extend their

approach by proposing a review system enabling users to see more visualized ratings, reply to comments, sort reviews by like/dislike, and categorize the reviews [51] (S19). Another study investigating topics related to application problems is conducted by Fu, et al. [52] (S5). As a part of their research, the authors applied topic modelling techniques, Latent Dirichlet Algorithm (LDA), to discover specific problems of each app behind the users' complaints. To identifying global trends in the market, they analysed top complaints of each app and found similar complaints and the most critical aspects of apps. LDA was also used in [53] (S49) to discover topics discussed in reviews and to, consequently, identify issues mentioned in reviews.

Galvis Carreno and Winbladh [54] (S10) used topic modelling and IE techniques to discover the topics from reviews that can be used to change and/or create new requirements for a future release of software. They used Aspect and Sentiment Unification Model (ASUM) for extraction of topics. The authors focused only on requirement changes, while other kinds of valuable information for developers are neglected in their approach. Moreover, the authors manually classified reviews to build their gold standard dataset and acknowledged that this way of procuring training data is error prone as the expertise of authors in the domain is under question.

Apart from the studies discussed in this section, there are other topics discovered in pre-processing and data preparation phases of other experiments. As the main goal of them for analysing reviews was other than discovering discussed topics, selected papers reporting such experiments are discussed in further sections. However, topics discovered by any of the selected papers analysed in this chapter are grouped and listed in Appendix III.

2.7.1.2 Categorization of User Reviews

Selected papers fallen in this group have proposed methods and approaches for categorising mobile application reviews into two or more different groups mainly for assisting developers in more straightforwardly finding useful reviews for application development purposes. Although NLP techniques are scarcely used for this task, majority of the selected papers have applied machine learning techniques.

In the earliest selected paper analysing content of mobile application reviews found in this SLR, Platzer [55] (S1) used motivational models to find usage motives that are addressed in the text of user reviews. To classify reviews based on these motives, they applied Support Vector Machine (SVM) and Naïve Bayes (NB) algorithms.

Oh, et al. [56] (S9) believed that uninformative reviews should be filtered out to help developers overcoming the overload of feedback. To achieve this, they defined three categories for reviews (i.e., functional bug, functional demand, and non-functional request) and manually classified 2800 reviews into the categories to train their filtering model. Identification of categories was done by app developers who analysed the review contents manually, though details of the process, such as number of experts, number of analysed reviews, and selection criteria is not provided. Then, they extracted keywords of each review using Latent Dirichlet Allocation (LDA) and heuristic methods and applied SVM algorithm to classify reviews into informative and uninformative. The authors argued that the performance of LDA in identifying keywords is weak due to shortness of reviews in length.

Chen, et al. [57] (S13) developed a computational framework for App Review Mining, namely AR-Miner which filters useless reviews and uses topic modelling techniques to group informative reviews based on the topics discussed in them, and a review ranking scheme to prioritize them with respect to the developers' needs. AR-Miner uses Expectation Maximization for Naive Bayes (EMNB), a semi-supervised algorithm in machine learning to classify reviews to informative and uninformative. In next phase, it uses LDA for grouping, and creates a ranked group of reviews based on their rating, fluctuation in time of reviews, and volume of reviews reporting a similar issue/request. The authors compared the performance of two algorithms in topic modelling, i.e., Latent Dirichlet Allocation (LDA) and Aspect and Sentiment Unification Model (ASUM) for grouping informative reviews based on their contents.

In classifying user feedback into informative and non-informative, they did not justify the superiority of Expectation Maximization for Naive Bayes (EMNB) to other

existing algorithms. Testing the performance of other classifiers and comparing the outcomes was a good way for justification of superiority of their approach. Moreover, there are several important aspects neglected in their approach. Firstly, sentiment of the review relates to its content and could be used to leverage the performance of the tool [58]. Secondly, although they employed time stamp, text, and rating of a review to do their task, important features such as title of review and meta-data features are neglected. Finally, they removed stop-words, stemmed, and applied other pre-processing techniques on raw crawled reviews. State-of-the-art approaches [59] show that these pre-processing tasks increase the chance of losing helpful content. Additionally, they discriminated informative and uninformative reviews based on their own understanding of "informative" and reviewing few numbers of online forums, while to identify what really is important for mobile app developers to extract from user reviews, their needs and requirements should be studied comprehensively.

Maalej and Nabil [60] (S21) designed and applied different probabilistic techniques and heuristics to automatically classifying reviews into four basic types: Bug reports, feature requests, User experiences, and rating. They generated a list of keywords by string matching, bag of words, sentiment scores, NLP pre-processed text, review rating and length, to be used for classification task. Then, they applied Naive Bayes, Decision Trees, and MaxEnt to compare the performance of binary to multiclass classifiers in classification of user feedback into the predefined basic types. The authors extended their approach in [61] (S29) by adding bigram and its combinations to utilized classification techniques, and by improving pre-processing phases and classification scripts. They argued that by the use of meta-data combined with text classification and natural language pre-processing of the text, the classification precision rises significantly.

Guzman, et al. [62] (S18) relied on categories found in [48] to form their taxonomy with 7 categories relevant for software evolution including bug report, feature strength, feature shortcoming, user request, praise, complaint, and usage. The authors then used the taxonomy to investigate the performance of various machine learning techniques (i.e., Naive Bayes, Support Vector Machines (SVMs), Logistic Regression and Neural Networks) in classification of reviews. The set of

features they used includes number of upper/lower case characters, length, positive/negative sentiment, and rating. However, many other features could be found in user feedback to be used for classification purposes in order to enhance the throughput of the model [63]. Similarly, NN was used in [64] (S46) to classify reviews into General opinion, Functional feature, and Out of domain. To help developers find useful reviews faster, they also identified review sentiments using NLTK and found key phrases using RAKE-NLTK which is a domain-independent keyword extraction algorithm.

Panichella, et al. [65] (S26) argued that among the 17 topics identified by Pagano and Maalej [48], only 8 of them were relevant to software maintenance and evolution tasks . They proposed a method to identify constructive feedback for software maintenance and evolution tasks. The authors hypothesized that understanding the intention in a review has an important role in extracting useful information for developers. And to understand the intention of a review, they used sentences structure, sentiment of a review, and text features contained in a review. Thus, they formed a taxonomy (i.e., Information Giving, Information Seeking, Feature Request, and Problem Discovery) by manually reviewing a number of reviews. Then they extracted a set of features by the use of NLP, text analysis, and sentiment analysis techniques to train a classifier and finally classified reviews according to the taxonomy. They compared performance of different machine learning techniques, namely, the standard probabilistic naive Bayes classifier, Logistic Regression, Support Vector Machines, J48, and the alternating decision tree (ADTree) and reported that J48 performed well. The authors identified and grouped review sentences into 6 categories (i.e., feature request, opinion asking, problem discovery, solution proposal, information giving, and information seeking). They compared these categories with topics identified by Pagano and Maalej and found that all 17 topics match with 4 out of their 6 identified categories.

Panichella, et al. [66] (S32) improved the approach in S26 by proposing ARdoc (App Reviews Development Oriented Classifier), a tool that automatically classifies useful sentences in user reviews according to a taxonomy designed in S26 to model developers' information needs when performing software maintenance and evolution tasks. After dividing the review into sentences, ARdoc extract lexicon,

grammatical structure, and sentiment of each sentence to be used by a machine-learning algorithm (J48) for classification purposes.

Buchan et al. [67] (S47) also used classification techniques (i.e. SVM, Naïve Bayes, and Logistic Regression) for classifying reviews into feature requests and non-feature requests. They used n-gram and review sentiment as classification attributes. Their results show that the performance of SVM in this task was superior comparing to other classifiers.

The SURF (Summarizer of User Reviews Feedback) was proposed in [68] (S27), which is a tool for categorization and summarization of reviews. It splits a review into sentences and performs the summarization task in three phases. Firstly, it employs the approach proposed in their previous study S26 and their identified sentence categories to classify sentences into user intentions and assigns one of the intentions to each review sentence. Secondly, it employs sets of keywords to build an NLP classifier and automatically assign one or more concept (topic) to each sentence in a review. Specifically, the authors manually analysed each sentence in 438 reviews selected as a training set to discover topics discussed by reviewers resulting in identification of 12 topics. They manually assigned keywords to each sentence and for each topic, created a finite set of keywords and enriched it with WordNet synonyms. Finally, for summarization purposes, they relied on their observations and assigned a score to each sentence for each observation. The tool categorizes sentences according to their topics and intention categories and generates summaries as structured HTML. However, their approach suffers from lack of a comprehensive research on what really is needed by app developers as these points are observed only by authors and a software developer. Moreover, they assigned different relevance scores to each category of intentions to score the first observation without studying the impact of each score.

To overcome the problem of processing colloquial terminologies used by users in their informal language which causes complex classification models and overfitting problems, Jha and Anas [69] (S33) proposed a FrameNet tagging based approach to classify reviews based on the notion of semantic role labelling (SRL). The aim of using SRL is to obtain a higher level of abstraction from sentences. SLR

classifies the words used in a sentence into semantic classes describing an event along with its participants. In their classification task using Naive Bayes (NB) and Support Vector Machines (SVM), the authors used frames generated from each review, rather than each word. However, their target classes are limited to bug reports and feature requests indicating that other types of valuable information are ignored in their approach. The authors released their classifier as a tool in [70] (S34) and conducted a very similar experiment in [71] (S43). Further in 2019, they used a multi-label classifier to extract non-functional software requirements from user reviews [72] (S44). They compared the performance of Naïve Bayes and SVM in classifying reviews into the categories of non-functional requirements defined by Kurtanovic and Maalej [73] (i.e. dependability, reliability, performance, and supportability). As classification attributes, they used bag of word, category of the target application, and sentiment of the review.

In another study targeting only one-star and two-star reviews, McIlroy, et al. [59] (S31) studied the extent of multi-labelled user reviews (reviews raising more than one issue type) and proposed an approach to automatically labelling multi-labelled user reviews. They defined 13 types of issues and labelled a number of reviews manually to form their gold standard dataset. For labelling task, they transformed the problem of multi labelling into single labelling and used a classifier for each label and combined their results. They used several different classifiers e.g., support vector machines (SVM), decision tree (J48) and Naive Bayes (NB) as well as several different multi-labelling approaches e.g., Binary Relevance (BR) [it does not leverage the correlations between labels], Classifier Chains [74] [it does leverage the correlations between labels], and Pruned Sets with threshold extension (PSt). They defined a threshold to assign each label to a review. Finally, they used 10-fold cross-validation to evaluate results.

In pre-processing phase, they removed numbers and special characters, but not stop words, expanded abbreviations, filtered words occurring less than three times in dataset, stemmed words, and removed reviews consisting of three words or less. However, observations exhibit that reviews with less than three words report bugs and issues as well (e.g.: “poor camera”, “save button sucks”, and “can’t upload picture”). Moreover, they used (TF-IDF) as a mean to increase the weight of

words that occur frequently in a single user review and to decrease the weight of words that occur frequently in many user reviews. Although it helps to devalue ordinary words, this way of weighting words might demote issues repeated and discussed between several users.

To compare the accuracy of various classifiers in categorization of user reviews into four software maintenance tasks (i.e. feature request, information giving, information seeking, and bug reports), Al-Hawari et al., [75] (S35) trained associative classification (AC) algorithms using the KEEL software [76] and the RapidMiner studio [77, 78]. The authors used two different datasets taken from [61] and [65] to test the accuracy and argued that AC algorithms have a better average performance than J48, KNN, RF, SVM, and NB.

Classification algorithms have also been used in [79] (S36) to classify reviews of healthcare-domain applications into bug reports, feature requests, sentimental, security, application performance, and user interface. 7500 reviews from ten different health-related mobile applications were manually annotated and used for training and testing purposes. As the classification attributes, they extracted TF-IDF and n-gram features and used them with Naive Bayes, Multinomial Naive Bayes, Random Forest, and Support Vector Machines (SVM). They have reported that Multinominal NB has outperformed other classifiers.

Performance of traditional machine learning techniques and deep learning techniques are compared in [80] (S52) where the authors employed both approaches to classify reviews into problem reports, inquiries, and irrelevant. Their obtained results show that the accuracy of neural networks used is more or less similar to the traditional classifiers using several text related extracted features.

Manual analysis is one of the main techniques used in abovementioned approaches to train classifiers. These classification models, however, suffer from incomprehensiveness as they are classifying reviews into a limited number of classes. Moreover, these approaches are very domain specific. This is because of the fact that by changing the target application, aspects and features will be changed as well. So, major updates in the proposed model would be required.

Iacob and Harrison [81] (S11) studied identification of proportion of feature requests among users' feedback. They developed a prototype to automatically mine online reviews of mobile apps and retrieve feature requests. It crawls reviews by the use of a data crawler, extract feature requests by 237 predefined linguistic rules, summarize extracted feature requests by ranking them based on their frequency and length, and visualize the results. The authors crawled 3279 reviews from Google play to manually read them and formulate the linguistic rules to identify feature requests. To identify common topics across the feature requests, they then applied the LDA model.

To facilitate the use of crowdsourcing in manually classifying reviews for software improvements, Martijn et al. [82] (S41) proposed a list of micro-tasks and guidelines for crowd workers to categorise useful reviews into the taxonomy defined in [83]. The guideline has three steps. First, crowd workers need to identify useful reviews from useless ones. Second, they do the same thing for each sentence in a review and find if there is any content helpful for developers. Finally, they classify reviews into the predefined categories. the question here is that how a crowd worker with almost no experience of mobile application development should extinguish between useful and useless reviews at the first step? Several rules and constraints should have been defined for crowd workers to train them how to identify useful reviews.

To study the enhancement requests of Android OS by users, Lee, et al. [84] (S20) applied NLP techniques (i.e. n-gram and PMI analyses) to identify the most pressing requests, though their focus was on android OS which reviews vary from the ones generated for applications. Their mere focus on text and ignoring the available meta-data of feedback is an obvious drawback of their approach.

Moghaddam [85] (S22) defined some lexical-POS patterns (8 for improvements and 5 for defects) to classify implicit reviews containing improvement/defects. Then she used these reviews as positive cases to train her distance learning classifier. After using SVM to extract sentences containing defects/improvements, the author applied LDA to cluster similar sentences and score the importance of founded topics. However, her effort on finding patterns to

capture defects/improvements was not sufficient as in her approach, problem, issue, and bug report is defined as defects and improvements include modification, upgrade, enhancement request, but only 13 patterns are defined to extract them among user reviews. Thus, many forms of explanations used by users to report defects/improvements are most probably missed by her approach. These patterns are used to label reviews and it causes inaccuracy in the results. Moreover, the proposed approach provides developers with more categorized sets of reviews, but they still need to expend too much effort and sources to explore these sets for the definition of defects/improvements which is too general in terms of categories fallen in these two groups.

Yang and Liang [86] (S25) proposed an approach to extract user reviews from AppStore, extract requirements information from them, and classify them into functional and non-functional (NFR) requirements. In their approach, requirement engineers manually identify and classify a certain number of user reviews as NFRs or FRs. Then, TF-IDF technique extracts keywords from these reviews to be used for automatic classification of reviews by the use of predefined regular expressions. However, human judgement is required to check and select the keywords which makes their approach labour-intensive. Their corpus consists of only 1000 reviews which is not appropriate to obtain a stable evaluation. Moreover, the authors annotated the data and extracted keywords by themselves. And they defined the regular expressions too which was, most probably, based on their observations over the data. Thus, performance of their approach severely depends on their preferences and evaluation accuracy is under question. Table 2.4 summarizes the categories and classification techniques used in the selected papers discussed in this section.

Table 2.4: Categories and Techniques Used to Classify Reviews

Categories	Techniques Applied	Used in Study
Informative/ Non-Informative	SVM, LDA	S9
	NB	S13
Feature requests/ Non-Feature request	SVM, NB, LR	S47
	Linguistic rules	S11
	PMI analysis	S20
Information Giving/ Information Seeking/ Feature Request/ Problem Discovery	NB, LR, SVM, J48, DT	S26, S32, S27
	AC	S35
Bug Reports/ Feature Requests	SVM, NB	S33, S34, S43, S44
Problem Reports/ Inquiries/ Irrelevant	SVM, NB, RF, DT, NN	S52
General Opinion/ Functional Feature/ Out of Domain	NN	S46
Issue Types	SVM, NB, J48, BR, PSt	S31
Bug Reports/ Feature Requests/ User Experiences/ Rating	NB, DT, MaxEnt	S21, S29
Bug Reports/ Feature Requests/ Sentimental/ Security/ Application Performance/ User Interface	SVM, NB, RF	S36
Bug Reports/ Feature Strength/ Feature Shortcoming/ User Request/ Prise/ Complaint/ Other	SVM, NB, NN, LR	S18
Functional and Non-Functional Requirements	Regex	S25
Feature/ Stability/ Quality/ Performance/ None	Manual	S41
User's Motivations	Motivational Models	S1
Improvements/ Defects	POS Patterns	S22

2.7.1.3 Opinion Mining

In this section, selected papers focusing on extracting user opinions about application/product aspects are discussed. Although extracting user opinions, opinion mining, is not the target of this thesis, analysing these studies provides a wide technical background for the task of identifying aspects in this project.

To help developers decide which feature needs to be refined in next application release, Zhang et al. [87] (S51) proposed an approach to extract application aspects and estimate rating of each aspect given by all reviewers of a

certain application. They applied topic modelling techniques, LDA algorithm, to extract application aspects. Then they used a linear regression model to calculate the rating for each aspect. Finally, they prioritize aspect refinement tasks based on aggregated ratings obtained for each aspect.

Guzman and Maalej [58] (S14) extracted features from the text of reviews by the use of collocation finding algorithm. The algorithm finds collection of words frequently occurring often (e.g., "battery life" and "screen resolution"). Then they applied SentiStrength, an automated sentiment analysis techniques [88] designed to tackle with short and low quality texts, to extract opinions and sentiments associated with each feature. SentiStrength divides the input text into sentences and assigns a positive value along with a negative value to each sentence. Finally, they grouped related features and aggregated their sentiments using topic modelling techniques [89]. Their aim was to automatically identify application features and their associated sentiments mentioned in user reviews. Guzman, et al. [62] (S18) extended their previous approach [58] and studied the identification of conflicting opinions. They developed DIVERSE, to identify diverse user opinions concerning different applications. The framework groups reviews by mentioned features and sentiment. Similar to their previous approach, the authors used collocation finding algorithm to extract features from user reviews and a lexical sentiment analysis tool to find opinions and experiences concerning the features. Then, they used a greedy algorithm to retrieve a set of diverse feature-sentiments.

There are several issues affecting the accuracy of their method. Firstly, a feature in their approach is a collection of two words occurred in more than three reviews including different orders, synonyms, and misspells. Limiting the feature to be consisted of two keywords prevents the approach to comprehensively cover various types of features mentioned in user feedback. Secondly, according to definition of a feature, their approach ignores features appearing in less than three reviews. These features might be very important for development team. finally, the lexical sentiment analysis scores each sentence in the review. Then it assigns this score to the feature mentioned in the sentence. However, in many cases, the overall sentiment of a review is negative, but the user is admiring a feature in that sentence. In the other word, negative words may be used in the favour of a feature [90].

Vu, et al. [91] (S24) developed MARK (Mining and Analysing Reviews by Keywords), a semi-automated framework assisting developers in searching reviews for certain features. It uses keyword-based approaches to search for relevant reviews. Several NLP techniques are used for keywords extraction, ranking keywords, and categorizing them based on their semantic similarity. MARK uses the standard Vector Space Model to query its review database and fetch results with respect to the keyword set.

Gu and Kim [92] (S16) applied sentiment patterns to parse review sentences and elicit app features. Their proposed tool (SURMiner), firstly, splits each review into sentences and applies Max Entropy to classify reviews into five categories, namely aspect evaluation, bug reports, feature requests, praise and others. Then, it identifies aspects and their associated opinions from sentences fallen in the category of aspect evaluation by designing a pattern-based parsing method. To design the method, the authors manually analysed 2000 sentences from reviews labelled as aspect evaluation and identified 26 templates. The tool, then, analyses the sentiment of each sentence and assigns a rating to that. It, finally, mines frequent items for all aspect words and clusters aspect-opinion pairs with common frequent items to summarize the output. Their focus is merely on aspect evaluation sentences in reviews. Thus, majority of topics such as feature requests and bug reports are ignored in their study. The main aim of their study was to extract application aspects and their associated opinions by designing a pattern-based parsing method. However, definition of patterns for unclear aspects is error prone as there is uncertainty in the aspects discussed in the user reviews. Besides, from app to app, aspects differ significantly which demands dramatic updates in predefined patterns.

In an opinion mining study, Haroon et al. [93] (S38) have also investigated the extraction of application features using syntactic patterns between nouns and sentiment words. In a pilot study, they manually extracted application features from real world reviews and observed that noun and noun phrases do contain 98% of application features. So, they defined some syntactic rules to capture certain relations between the nouns and sentiment words to identify application features. However, their approach misses all features appear in the form of verbs or implicit ones. Besides, one of the major issues with opinion mining-based feature extraction

techniques proposed so far is that they all miss lots of features mentioned in reviews without any sentiment or opinion mentioned for them. The following review, for example, does not contain any sentiment or opinion word, but includes 'menu button' which is an application feature.

"When I press the menu button on the top left side of the screen, the app shows only 3 items."

Syntactic rules was initially used in [94] (S53) to identify product features and facilitate aspect-based opinion mining task. To find aspect-opinion pairs, the authors used a lexicon of opinion words [95] and enriched it with WordNet [96] synonyms and acronyms and Semantics extracted from SenticNet [97]. Then, they defined five syntactic rules to extract aspects appearing in noun or noun phrase associating with the opinion words. They experimented employment of neural networks in identifying aspects, but applied same rules to the output of the algorithm as the accuracy was not sufficient [98] (S54). Further in 2017, Rana and Cheah [99, 100] (S55) defined an algorithm to apply syntactic rules and extract aspect-opinion pairs. The algorithm which is a modified version of the one proposed in their previous study [101] (S56) finds noun and noun phrases which are associated with opinions in a sentence. However, aspects might not be associated with any opinion. Besides, detection of aspects is based on frequency of repeating aspect candidates in reviews, but observations show that there are aspects not repeated in several reviews.

To identify useful reviews for software evolution, Guo et al. [102] (S45) extracted aspects from reviews using bootstrapping method [103] which is a semi-supervised learning method taking a set of application aspects as seeds, searching for sentences containing the seeds in dataset of reviews, generating extraction patterns for seeds, and discovering new aspects mentioned in reviews using the patterns. In the second step, they applied semantic analysis tools (i.e., SnowNLP) to discover semantic of reviews and considered reviews containing neutral or negative semantics along with an aspect as effective reviews. A simple example for demonstrating ineffectiveness of their approach is the following useless review containing an aspect and having a neutral sentiment:

"I use the navigation feature to travel overseas"

The main reason why most of these approaches are not fully applicable in the aspect extraction phase of this project is that detection of aspect in them is contingent upon identification of opinion words. This is because most of these papers have studied the feedback of users on certain aspects of the target product or application. The aim of conducting these experiments was to help product sellers, manufacturers, and developers in understanding satisfactory level of customers/users about different aspects of products or applications. However, this study focuses on identifying user requirements and discovering application aspects is one of the requisites of such a broad aim. Moreover, empirically evaluating several feature extraction approaches on real world user reviews, Dabrowski et al. [36] observed that these approaches achieve lower effectiveness than reported originally which puts a question mark on the accuracy of the reported approaches. Initial investigations performed in this study confirms this observation as well. Table 2.5 summarizes selected papers in this section with respect to the techniques used in them.

Table 2.5: Selected Papers Extracting Opinion-Aspect Pairs

Purpose	**Techniques Applied**	**Used in Study**
Extracting Opinion- Aspect Pairs	Collocation Finding Algorithm	S14, S18
	Bootstrapping Method	S45
	LDA	S51
	Syntactic Patterns	S38, S55, S56
	Sentiment Patterns	S16
	Syntactic Patterns +NN	S53, S54
Searching reviews about a certain aspect	Keywords, semantic similarity, VSM	S24

2.7.1.4 Relation Extraction Between Review Components

In this section, another group of selected papers that have been investigated the relationships between different components of a review and the target application are discussed.

The research attention in this group of studies is mainly on the impact of users' given star-rating on number of downloads and amount of sale.

In order to understand why users dislike apps, Fu, et al. [52] (S5) developed a system, named WisCom, analysing reviews in three different levels. Firstly, at the review level, they discovered reviews with inconsistent ratings by applying a regularized regression model. This was to understand individual reviews and the words used in them. Secondly, at the app level, they applied topic modelling (LDA) techniques to discover specific problems of each app behind the users' complaints. Finally, in entire app market level, they analysed top 10 complaints of each app to find similar complaints and the most critical aspects of apps leading to identifying global trends in the market. Their focus was only on negative reviews as they hypothesized that those could be directly used to enhance the quality of applications. However, researchers observed that feature requests that are important feedback for developers mostly appeared in positive reviews [47, 81].

AR-MINER [34] was used further by Palomba, et al. [74] (S23) to investigate how addressing user feedback by developers influences the overall rank of the app. Their proposed tool, named CRISTAL (Crowdsourcing RevIews to SupporT App evoLution), collects reviews posted for previous release of a certain application and tracks its rating. Then it extracts informative reviews using the AR-MINER and checks whether the comments are applied in next release. Finally, it checks the effect of them on rating after the last release.

Manually analysing reviews of BlackBerry apps, Harman, et al. [104] (S2) confirmed that the rating given by user has a significant impact on number of downloads. To better understand what users communicate in their reviews, Hoon, et al. [105] (S3) analysed 8.7 million reviews from 17,330 apps and categorized keywords appearing frequently in each star rating as they hypothesized that it can inform and focus development efforts. The authors determined the distribution of word and character counts per star rating respectively applying a regular expression to extract words from the review entities, and monitored which star rating the appearance of the extracted keywords pertains to. However, they did not apply appropriate pre-processing techniques (e.g., stemming, removing of stop

words, and spell checking) to normalize input reviews. Moreover, their focus was only on single words resulting in missing multi-word expressions.

Their dataset was also used in another approach [106] (S4) to discover possible relations between rating of a review and its content. They argued that reviews with lower ratings include more useful feedback, and that the depth of feedback in certain categories is significantly higher than for other. In both of their approaches, the authors did not discuss details and settings of the analysis. Obviously, beside manual analysis and observations they have used some automatic processing techniques that are not mentioned in the studies.

Noei et al. [107] (S48) have also manually analysed mobile application reviews to investigate the role of application categories and key topics in star-rating improvements. They discovered 10 categories for applications, such as business, health and fitness, media and video, travel and local, photography, etc. To assign topics to each user review they applied topic modelling technique, LDA [89] in particular, and discovered 23 topics such as comparing versions, bug reports, feature requests, laying audio and video, web browsing, etc. Finally, they applied PMVD approach [108, 109] to identify key topics in each category and to identify contribution of each topic in star-rating. Proportional Marginal Variance Decomposition (PMVD) applies weighted averages with data-dependent weights to average over orderings.

Table 2.6 summarizes selected papers in this section with respect to the techniques used in them.

Table 2.6: Selected Papers Studying Relations Between Review Components

Purpose	Techniques Applied	Used in Study
Identifying Relations Between Rating and Review Content	Regression Models	S5
	Manual	S4
Identifying Relations Between Rating and Number of downloads	Manual	S2
Identifying Relations Between Rating and Distribution of Keywords	Manual, Regex	S3
	PMVD approach	S48

2.7.2 Support Tools for Mining Mobile App Reviews (RQ2)

RQ2: What software tools have been developed to support these techniques?

Thirteen support tools were found in this SLR. A summary of which is presented in Table 2.7. In what follows, we provide an overview of these tools.

MARA. This tool analyses user feedback in several steps. First, web sources of reviews are crawled and parsed. Second, the tool uses 273 syntactical rules to mine the review content for feature requests expressed by users. An example of such a rule is 'Adding <request> would be <POSITIVE-ADJECTIVE>' (e.g., 'Adding an exit button would be great'). Feature requests are then summarized according to a set of predefined rules that rank the extracted requests based on their frequency and length. To identify topics that can be associated with these requests, Latent Dirichlet Allocation (LDA) model is used. Finally, during the feature requests visualization phase, the results of the summarization are displayed to the user.

WisCom. This tool analyses user feedback at three levels of detail, involving discovering inconsistencies in reviews, identifying reasons why users like or dislike a given app, and identifying general user preferences and concerns over different types of apps. Firstly, a regularized regression model is used for discovering inconsistencies and detect ratings that do not match the actual text of the feedback. Secondly, feedback comments of individual apps are aggregated and LDA algorithm is applied to discover why users dislike these apps. The algorithm was trained using

words that receive negative weight in the regression model. Finally, to identify outstanding complaints in each category of apps, most common complaints from negative feedback comments of each app identified in the previous step are aggregated on categories, summarized and displayed to the user.

AR-Miner. This tool analyses user feedback comments. It filters, aggregates, prioritizes, and visualizes informative (information that can directly help developers improve their apps) reviews. Non-informative (noises and irrelevant text) reviews are filtered out applying a pre-trained classifier (i.e., Expectation Maximization for Naive Bayes). The remaining informative reviews are then put into several groups using topic modelling techniques (i.e., LDA and Aspect and Sentiment Unification Model) and prioritized by application of a ranking model. Finally, the ranking results are visualized in a radar chart to help app developers spot the key feedback users have.

CRISTAL. This tool is used for tracing informative (providing any insight into specific problems experienced or features demanded by users) reviews onto source code changes, and for monitoring how these changes impact user satisfaction as measured by follow-up ratings. AR-Miner is used first to discard non-informative reviews. A set of heuristics are used, then, extract issues and commits driven by each review. IR techniques are used then, to identify possible links between each review and the issues/commits. The set of links retrieved for each informative review is stored in a database grouping together all links related to a certain release. This information is exploited by the monitoring component, which creates reports for managers/developers and shows stats on the reviews that have been implemented.

DIVERSE. This is a feature and sentiment centric retrieval tool for generating diverse samples of user reviews that are representative of the different opinions and experiences mentioned in the whole set of reviews. When a developer queries the reviews that mention a certain app feature, the tool will retrieve reviews, which represent the diverse user opinions concerning the app features. The tool applies the collocation finding algorithm to extract the app features mentioned in the reviews, uses lexical sentiment analysis in order to excerpt the sentiments associated to the extracted features, uses a greedy algorithm to retrieve a set of

diverse reviews in terms of the mentioned features, and group reviews whose content and sentiment are similar.

SURMiner. This tool summarizes users' sentiments and opinions on certain software aspects. By the use of Max Entropy algorithm, the tool classifies each sentence in user reviews into five categories (i.e., aspect evaluation, praises, feature requests, bug reports, and others) and filter only aspect evaluation sentences for extraction of aspects and corresponding opinions and sentiments. The tool uses a pattern-based parsing method to extract aspects and sentiments. The method analyses the syntax and semantics of review sentences. The resulting aspect-opinion-sentiment triplets are then clustered by mining frequent opinionated words for all aspect and clustering aspect-opinion pairs with common frequent words. Finally, the results are visualized on graphs.

MARK. This tool is a semi-automated review analysis framework that takes relevant keywords from developers as input and then retrieves a list of reviews that match the keywords for further analysis. The tool has a keyword extraction component which extracts a set of keywords from raw reviews. These keywords could be clustered based on Word2Vec and using K-mean algorithm or expanded based on based on the vector-based similarity of the keywords. Once the analyst specifies a set of keywords (via clustering or expanding), the tool will query its database and return relevant the most relevant results. MARK employs the popular tf-idf (term frequency - inverse document frequency), a standard term weighting scheme to compute the element values for those vectors, and the Vector Space Model for this task.

SURF. This tool summarizes user reviews to assist developers in managing huge amount of user reviews. The tool relies on a conceptual model for capturing user needs useful for developers performing maintenance and evolution tasks. Then it uses summarisation techniques for summarizing thousands of reviews and generating an interactive agenda of recommended software changes. The tool is equipped with a dictionary of topics and uses the NLP classifier to automatically assign a sentence in a review to one or more topics. To suggest the specific kinds of maintenance tasks developers have to accomplish, it also classify intentions in a

review using intent classifier [65]. Based on a certain scoring mechanism, the tool then generates the summaries as structured HTML.

ARDoc. This tool automatically classifies useful feedback contained in app reviews that are deemed to be important for performing software maintenance and evolution tasks. The tool divides review text into sentences and extracts from each of these sentences three kinds of features: Firstly, the lexicon (words) feature is extracted through the TA Classifier which exploits the functionalities provided by the Apache Lucene API to extract a set of meaningful terms that are weighted using the TF (term frequency). Second, the structure feature (i.e., grammatical frame of the sentence) is extracted through the NLP Classifier. Using NLP heuristics and 246 predefined recurrent syntactical patterns, the NLP Classifier automatically detects the occurrences of specific keywords in precise grammatical roles and/or specific grammatical structures. Finally, the sentiment is extracted through the SA Classifier using the sentiment annotator provided by the Stanford CoreNLP. In the last step the ML Classifier uses the NLP, TA and SA information extracted in the previous phase of the approach to classify app reviews according to a predefined taxonomy by exploiting the J48 algorithm.

Casper: is an automatic method for extracting events from user reviews on mobile applications proposed by Hui and Munidar, [110] (S37). The tool extracts events from reviews and classify them into user stories and problems. Then it represents pairs of user stories-problems to developers. Casper finds verbs in a sentence using POS-tagging technique and find the subtree rooted on this verb in the dependency-based parse tree as a candidate of an event. The output events are then classified into user story and problem. To build the model, the authors used Universal Sentence Encoder (USE) [111] to convert the event phrases into vectors of real numbers and adopted SVM to classify the sentence vectors.

IDEA: In order to identify emerging app issues from user reviews, Gao et al. [112] (S39) proposed a framework named Identify Emerging App issues (IDEA) which takes reviews of various versions of the target application and applies AOLDA (Adaptively Online Latent Dirichlet Allocation) which is the LDA algorithm adopted to work online to identify topics distributions with respect to the application version

from reviews. Then it finds emerging topics using anomaly detection methods. Then it considers emerging topics as the identified application issues.

A year after, the authors found that IDEA does not produce stable results even with same input data and has a long running time. So, they designed another tool called DIVER, (i.e. iDentifying emerging app Issues Via usER feedback) [113] (S40) to more efficiently identify emerging application issues. Similar to IDEA, the tool takes reviews as input and extracts word collocations using ECLAT (Equivalence Class Transformation) [114], a depth-first search algorithm for pattern mining. Then it identifies emerging word collocations to achieve emerging issues by extracting and comparing proportion of reviews containing the word collocations for each version at each time.

SRR-Miner: is an automatic tool proposed by Tao et al. [115] (S42) to summarize security issues and users' sentiments in the reviews. SRR-Miner applies a keyword-based approach to identify security related review sentences. Then it uses six very basic POS-based semantic patterns to capture misbehaviour-aspect-opinion triples as candidates for security issue reports.

RISING: Review Integration via claSsification, clusterIng, and linkiNG is an automated approach proposed by Zhou et al. [116] (S50) to categorise user reviews into fine-grained groups concerning similar user requirements via classification, clustering, and linking of user reviews. The approach uses ARDOC (explained as another tool in this section) to classify reviews into information giving, information seeking, bug reports, and feature requests. Then it focuses only on the two latest groups for further fine-grained clustering of feature requests and bug reports using a customised version of traditional K-mean algorithm [117].

Table 2.7: Support tools for mining mobile app reviews

Tool Name	Description	Underlying Technique	Unit of Analysis	Study ID
MARA	Identifying and summarizing feature requests using syntactical rules	Syntactical rules, LDA	Sentence	S11
WisCom	Identifying inconsistent feedback using a regression model and complaints using LDA	Regression models, LDA	Multi-level	S5
AR-miner	Filtering out useless reviews using topic modelling, group and prioritizing informative reviews	Naive Bayes, LDA	Sentence	S13
CRISTAL	Detecting traceability links between reviews and source code	AR-MINER IR	Document	S23
DIVERSE	Detecting conflicting opinions based on aspects, answering developer's query by grouping reviews by their mentioned features and sentiment.	Collocation finding, Lexical sentiment analysis, Greedy algorithm	Document	S17
SUR-Miner	Classifying and summarizing reviews using a pattern-based parser	Sentiment patterns, Max Entropy	Sentence	S16
MARK	Assisting developers in searching opinions about aspects	keywords, NLP, VSM	Document	S24
SURF	Classifying and summarizing reviews using topics and intentions classifiers	NLP classifier, WordNet	Sentence	S27
ARdoc	Classifying reviews using a taxonomy.	NLP, J48 Sentiment analysis	Sentence	S32
Casper	Extracting events and problems from reviews using POS patterns	POS, SVM, USE	Sentence	S37
IDEA	Identifying emerging app issues by tracking topics distribution	AOLDA	Document	S39
DIVER	Identifying emerging app issues by tracking word collocations	ECALT	Document	S40
SRR-Miner	Summarizing security issues and users' sentiments using a keyword-based approach	Keywords, POS	Sentence	S42
RISING	Classifying and clustering reviews using ARDOC and K-mean	ARDOC, K-mean	Sentence	S50

2.8 Open Issues

This section reports the challenges and open problems in the area along with some future research directions for researchers. There are several issues making useful information extraction from user feedback a challenging task. Majority of these issues arise from the nature of the feedback to be mined, interpretations of researchers, and technical environments. While many of these problems have been identified and investigated, no single extraction technique is capable of addressing all of the challenges. The selected studies had focused on a subset of these issues to solve. Thus, a future direction for the research in this area could be to designing models capable to solve the following issues as much as possible.

1) Huge Volume of User Feedback: The volume of user-generated feedback is huge. Extracting insights from massive number of reviews is a labour-intensive and challenging task for researchers. Quantity of reviews generated for an application usually exceeds a human capacity to manually analyse them and extract useful information. It makes difficulties in identifying recurring trends and issues across reviews. Moreover, processing such an amount of data is time consuming. In case of application updates and revealing new versions, the time commitment involved in extracting required changes for next release is crucial [54]. Thus, more novel approaches for effectively and efficiently processing massive amounts of feedback by taking relations between them into account are on demand.

2) Unstructured Data: Unstructured nature of user feedback is one of the main challenges in automatically processing them. App users often write their reviews in unconventional manners making automate interpretation as difficult as possible [81]. They tend to express their reviews using informal language, which often includes colloquial terminologies. Moreover, they regularly neglect grammar and punctuation rules and use eccentric syntactic entities, and ironic and sarcasm sentences in their reviews [48]. One key challenge to app developers is dealing with such unstructured short pieces of text. To overcome these problems, technical text mining and NLP approaches should be integrated into proposed models and accompany them in order to deal with these types of user generated text.

3) Annotating Data: Data annotation is required in supervised approaches for training the machine and for evaluation purposes. It is a time and cost consuming task usually accompanied with mistakes and errors, though it is necessary to obtain better results in supervised and semi supervised approaches. As it is discussed in Section 3.1, in majority of cases, studies have provided different approaches demanding certain types of annotation resulting in impossibility of creation of a gold standard dataset to be used for evaluation of various approaches. Apart from difficulties of manual annotation of data, accuracy of the annotated data is under question as instead of domain experts, article authors or non-experts are employed to do it in majority of cases. Thus, more efforts and novel approaches are required to alleviate the problem of data annotation.

4) Defective Use of Data Pre-processing: With the use of some certain pre-processing tasks, researchers try to prepare the data to be used as the input of their method. Stop-word removal techniques are an example of these tasks. Various proposed methods include these tasks to perform more efficiently in terms of time and computation. However, a group of approaches have argued that using some of the pre-processing tasks causes missing valuable information in user feedback [69]. Therefore, more comprehensive, and analytical research is required to investigate the impact of applying each pre-processing task on performance and accuracy of the techniques.

5) Data Interpretation: While various approaches and techniques for extraction of actionable information from user feedback have been proposed by researchers, their interpretation of actionable information is, to a certain extent, different from developers'. Previous research [118] has tried to discover what developers are highly interested in, and what the designers' viewpoint is when mulling the reviews over [119, 120]. Although some of these factors (i.e. functionality, and app features) are used by Guzman, et al. [62], and informativity of reviews from developers' perspective was investigated by reading some relevant forum discussions by Chen, et al. [57], many proposed approaches have made the data analysis from the authors' point of view. Inconsistency between researchers' and developers' interpretation results in development of inefficient methods and

techniques. This argument is backed up by the discussions provided after analysing each selected paper in section 2.7.1.1 and 2.7.1.2.

More precise study of what exactly needs to be extracted from user reviews is required to make proposed approaches more efficient.

2.9 Validity Threats to This SLR

The primary threats to validity of results from this SLR are concerned with comprehensiveness and coverage of the relevant studies. Firstly, the search only covers publications that were published before the end of October 2020. We conducted the review in 2020. We, therefore, used 2020 as an upper bound on the search term. Other relevant studies might have been published since November 2020 that we have not included in this review.

A further search-related limitation of the review is that we might have missed some papers in identification of relevant studies. The completeness of the search is dependent upon the search criteria used and the scope of the search, and is also influenced by the limitations of the search engines used [121]. We used a set of well-known references to validate the search string and made necessary amendments before undertaking the review. However, four papers were identified by snowballing that were indexed by the digital libraries but were not found with the search terms used in the review.

Aiming to cope with construct validity which is related to generalization of the result to the concept or theory behind the study execution [122], we defined and applied several synonyms for main constructs in this SLR: ‘‘requirements engineering’’, ‘‘feedback analysis’’, and ‘‘software evolution’’.

The other validity threat could be related to selection, analysis and synthesis of the extracted data being biased by the interpretation of the researcher. Inclusion/exclusion of studies has passed through accurate selection, on-going internal discussion and crosschecking between the authors of the SLR. We tried to mitigate the thread by conducting the selection process iteratively. Furthermore, collecting data by two extractors who are PhD candidates in the field was also helpful to minimize any risk of researchers’ own bias.

If the identified literature is not externally valid, neither is the synthesis of its content [123]. To alleviate this threat, we formed our search process after multiple trial searches and compromise of the authors. Unqualified papers were excluded as well by the application of our exclusion criteria.

2.10 Conclusion

This chapter reports our research effort aimed at systematically reviewing and analysing application feedback processing practices toward assisting developers with extraction of actionable information and insights. The review was conducted by the guidelines provided in [44]. In total, 56 relevant studies were identified and analysed. Firstly, this thesis categorized and discussed studies based on types of analysis and underlying techniques. Then, available supporting tools for feedback mining were investigated. Finally, challenges and open problems in feedback mining, which require further research were discussed. The findings from this review provide several implications for researchers, requirements engineers and tool developers to gain a better understanding of the available studies and tools and their suitability for different contexts resulting in significant improvements in development of intelligent app review mining techniques and tools.

Chapter 3 Identification of Factors Relevant to the Usefulness of App Reviews

3.1 Introduction

According to the literature review reported in previous chapter, a set of effective metrics for identification of useful reviews based on developers' viewpoints has not been proposed yet. In order to facilitate the process of overcoming such a large problem and to effectively measure the usefulness of user reviews, a comprehensive set of usefulness factors are required. This chapter discusses the process and strategy used to achieve these factors. Definition and examples are also provided for any discovered factor.

3.2 Research Framework

It was discussed in Chapter 1 that one of the objectives of this project is to conceptualise the usefulness of app reviews with respect to the developers' viewpoint. To achieve this, a set of usefulness factors were required to effectively measure the usefulness of a review. Accordingly, the idea of performing the research reported in this chapter is to analyse related work for identifying any potential usefulness factor and for discovering any concept helping in definition of a usefulness factor.

To define a comprehensive set of usefulness factors, related studies are carefully analysed to discover what characteristics of the user generated content (e.g., forum post, review, comment, email, tweet, etc.) have been reported as helpful for facilitating the discovery of useful information for application developers, product designers, etc. Unlike the literature review reported in Chapter 2, the scope of this analysis covers several domains of research, such as RE reports mining, app testing reports analysis, customer services emails analysis, etc.

Performing the analysis, a list of potential usefulness factors are identified as usefulness metrics enabling properly detecting useful reviews. In this chapter, the

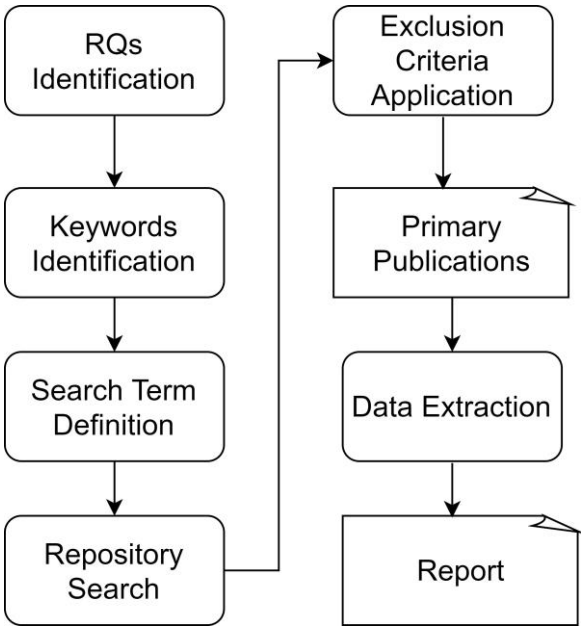

Figure 3.1. The Conceptual Framework of this study

systematic process of identifying them, definition of each factor, and examples are explained in detail.

3.3 Survey Methodology

This section introduces the methodology used to conduct this exploratory study. Figure 3.1 illustrates the process of the literature review. Research questions are defined at the first step. Then, a set of keywords is identified to narrow down the scope and limit the number of results to only related works. Using the keywords, search terms are formulated and applied on an online repository search process to harvest related publications. Irrelevant retrieved papers are then excluded from this study defining and applying some exclusion criteria. Finally, the selected primary publications are analysed in three phases (i.e., Title reading, Abstract reading, and Full-text reading) to achieve a fine-grained selection of closely related works and to perform the review process.

3.3.1 Defining Research Questions

The main objective of this literature survey is to find existing factors used in related studies for measuring the usefulness of user reviews for application development

purposes. To achieve this objective, the following research questions are formulated.

RQ.1 What are the factors for measuring usefulness of user reviews for software development?

To answer this research question, we searched three different areas of research working on analysis of user feedback (i.e., analysing reviews on mobile applications, analysing customer opinions on online products, and analysing post-deployment customer feedback, reports, logs on software systems). This search was to find and analyse related studies in order to understand how the researchers have measured usefulness and what factors they have identified to do so. It is noteworthy to point out that studying the usefulness of a review from other readers' viewpoint has been reported in several papers as well [124, 125]. However, our aim is to assess it from developers and requirement engineers' perspective.

RQ.2 How the importance of each factor has been validated?

To report how the importance of each factor is validated in a study, we analysed the study to see how the authors have justified the importance of the factor. At the end of each sub-section discussing a factor, we have reported the validation process of the studies fallen in the sub-section.

3.3.2 Searching Online Repositories

User reviews, usefulness, and requirement elicitation compose the main points of the focus of this study. Various terms and keywords might have been used in existing literature to refer to these concepts. Therefore, we defined an extended set of keywords for each concept represented in Table 3.1. To build the final search term, conjunction of the groups was used, while disjunction of keywords had built each group.

The search term was validated to fetch more relevant papers and minimise the remaining search results using a "quasi-gold standard" [126], whereby, five related works were manually identified. These papers were obtained during the analysis of selected studies in Chapter 2. The search term was then refined repeatedly to include these papers when fetching minimum possible number of papers.

To collect related papers from online repositories, the search term was applied on three well-known online repositories (i.e., ScienceDirect, IEEE Xplorer, ACM Digital Library). Some of the inclusion and exclusion criteria (e.g., Language, year, venue) where applicable using the automatic search option provided by the repositories. So, we applied those criteria at the searching step. However, the advanced search options of these repositories are reported to reveal huge number of irrelevant papers [127, 128] impelling us to define and apply manual scripts, wherever possible, for searching the repository. Finally, we achieved a collection of 893 papers.

Reading title, abstract, and conclusion of the collected papers along with the application of exclusion criteria explained in further section, we discarded 862 reviews. Snowballing was the last step in the process of selecting relevant papers wherein references of the selected papers were searched to identify any possible

Table 3.1: Groups and keywords in the search term

Concepts	Keywords
User	User, End user, Customer, Client
Review	Review, Opinion, Comment, Post, Feedback
Usefulness	Useful, Helpful
Target product	Mobile app, Mobile application, Application, Software, Product
Requirement elicitation	Requirement elicitation, Development, Evolution, Design, Preferences, improvement

appropriate publication related to this study. The criteria for analysing the papers for snowballing were similar to selecting papers in this chapter. Besides, during the process, more attention was given to identify related papers as, unlike search engines, human analysis skills were involved.

Performing the snowballing, 2 papers were added to the collection of our selected studies. Hence, we started our analysis with 26 papers. A summary of our search process is provided in Table 3.2.

3.3.3 Exclusion Criteria

In order to focus on a consistent set of primary studies and reduce the eventual effort in further in-depth analysis, we decided to approve a paper for further analysis only if it does not satisfy any of the exclusion criteria. The following exclusion criteria are, therefore, defined and applied on the selected papers:

- Papers written in languages other than English
- Papers published before 2008 or after 2019. We used 2008 as starting date as the research on reviewing online products was started at this year.
- Short papers and tutorials for lack of sufficient information for our study
- Duplicated papers as some authors publish extended versions of their works. We have only considered the last version found as relevant.
- Papers focusing on the helpfulness of reviews from customers' point of view
- Papers analysing reviews for identifying users' behaviours

Table 3.2: Number of selected papers during the search process

Step	Count
Online repositories search	893
After reading titles	112
After reading abstracts and conclusion	31
After skimming and scanning full text	24
After snowballing	26

Table 3.3: List of the Usefulness Factors

No.	Factors	Explanation	Reference
1	Application aspect	Where the issue/ requested feature is/supposed to happened in the application? (e.g., function, quality, component, etc.)	[68, 129] [65] [130] [58] [62] [120] [131] [119, 132] [133, 134] [132, 135-137] [138-140] [141, 142] [48] [143]
2	Feature request	What feature is needed to be added to the application?	[65] [48] [37] [69, 144] [70] [60, 145] [61] [57, 146] [54]
3	Issue report	What is the problem in the application?	[37] [60] [61] [49] [51] [73] [147, 148]
4	User action	How the user has produced or faced the issue? / How the user will work with the desired requested feature.	[133] [138] [141]
5	System action	What the system does when an issue occurs? / What the system should do with adding the desired requested feature?	[138, 141]
6	Expected action	What the system should do if no issue occurs (i.e., normal situation)?	[138, 141]
7	Device information	What user device and model are and what OS and OS version is installed on user device?	[138, 141]
8	Pros and cons	Both pros and cons of the application is mentioned in a review	[120] [131] [119]
9	User expertise	Number of reviews posted by a user	[120] [149] [150] [151]
10	User rating	Number of rating stars given by the user	[49] [51] [59] [106]
11	Length of the review	Number of words and sentences in the review	[149] [150] [152]
12	Readability	Number of grammar and spelling mistakes in the review	[138, 141] [150]

3.4 The Usefulness Factors

To answer the main research question of this chapter, "What are the existing factors for measuring usefulness of user reviews for software development?", selected

papers are analysed to figure out how the authors measure the usefulness. The analysis, then, results in a list of factors represented in Table 3.3. These factors are defined and explained with examples in the following sub-sections.

According to the table, some of the factors might have received few research attentions making them sound insignificant at the first glance. However, as the objective at this step of the project was to perform a systematic method to discover any possible usefulness factor, this section discusses whatever is reported to form a comprehensive collection. Moreover, some of the factors receiving minor citations are proposed in important research projects involving many experts in the domain confirming the importance of the factor. Besides, at this step, this thesis does not argue the significance of these factors, but reporting available ones discovered through the literature review. Further sections report how these factors are to be validated and filtered with the use of the experts' judgement.

Research question 2 asks "How the importance of each factor has been validated?". To answer this research question, after defining and explaining what each factor is in the following sub-sections, a summary of the approach is provided to show how this factor helps or is considered to be helpful in identifying useful requirements from user feedback in variety of domains. Each sub-section ends with a discussion on validation process of the selected papers in the sub-section.

3.4.1 Application Aspect

Application aspect is defined as "a prominent or distinctive user-visible aspect, quality or characteristic of a software system" [153]. This factor refers to the mentions of any aspect (e.g., component, function, process, etc.) of the application in a review. It can be any description of specific app functionality visible to the user (e.g., "viewing pdf"), a specific screen of the app (e.g., "log in screen"), a general quality of the app (e.g., "load time", "size of storage", or "price"), as well as specific technical characteristics (e.g., "encryption technology") [58].

Tianjun et al. [143] reviewed product design science literature to extract concepts related to product design and manually processed a sample of 265 review sentences to study whether reviewers expressed their requirements on these

concepts. Then, they defined some linguistic rules to capture these requirements from reviews to be used for analysing user preferences during the design phase. Although these concepts are defined as means of identifying user preferences, one of them is product feature (Application Aspect).

During the development of SURF (Summarizer of User Reviews Feedback), an automatic tool for summarizing mobile application reviews, Di Sorbo et al. [68] manually analysed each sentence in 438 reviews selected as training set to discover topics in reviews. One of the observations they made through the annotation was that "developers need reasonably useful sentences discussing a specific aspect of an app with respect to other review sentences."

In an exploratory study, to demonstrate how user reviews are useful for developers, Pagano and Maalej [48] investigated how to identify and classify topics in reviews. They applied a descriptive statistic to investigate usage of feedback. They, then, manually analysed a random sample of 528 reviews to explore and assign topics to each review. Application Aspect was one of the 17 observed topics in the reviews.

Panichella et al. [65] proposed a taxonomy to classify app reviews into categories relevant to software maintenance and evolution. To understand developers' viewpoint in analysing user feedback, they extracted 300 emails from the development mailing lists of Qt and Ubuntu projects and tried to categorize sentences learning from previous studies on this domain [154, 155]. Among identified categories, the one matching categories identified in [48] was 'Information giving' which is defined as follow: "sentences that inform or update users or developers about an aspect related to the app". Their approach was improved in [66] proposing ARdoc (App Reviews Development Oriented Classifier), a tool that automatically classifies useful sentences in user reviews using the same taxonomy.

To assess the quality of product reviews for summarization purposes, Liu et al. [130] defined a set of specifications for judging the quality. They proposed a classification-based approach to detect low-quality reviews. For training the classifier and evaluating the model, they collected 23,141 reviews on 946 digital

cameras from Amazon website. According to their inspirations from the data they defined four categories of reviews (i.e., best, good, fair, and bad). They defined *best* review as follow:

"A *best* review must be a rather complete and detailed comment on a product. It presents *several aspects of a product* and provides *convincing opinions with enough evidence*."

Extending their previous approach [58], Guzman and Bruegge [62] developed DIVERSE, to identify diverse user opinions concerning different applications. The tool groups reviews by their mentioned aspects and sentiment. Collecting and manually analysing 2800 reviews, the authors argued that "app store reviews include information that is useful to analysts and app designers, such as user requirements, bug reports, and documentation of user experiences with specific app aspects."

Jin et al. [119, 131] extracted several categories of features from review content (i.e. linguistic features, product aspects, features based on information quality and features using information theory) to be used for automatically predicting usefulness of product reviews. The authors relied on the responds to their two questionnaires to consider product aspects as a feature. Some subjects replied to the question 'Why you have chosen this review as useful?' as "this review mentions many product aspects", while some argued that "many reviews shared the aspects he\she likes and dislikes". The impact of this specific feature is not reported in their examinations.

Qi et al, [120] used similar features plus meta data of reviews and combined conjoint analysis with the traditional KANO method to measure the helpfulness of reviews for product designers. The authors confirmed that features proposed in [131] and [119] combined with features gained from meta data were effective on identifying helpful reviews. Effectiveness of these features in extracting customer requirements from mass online product reviews was assessed and confirmed by them in [135]. They applied SVM-based model to their conjoint analysis model and reported the impact of number of product aspects in a review as: (Estimate:0.15439, Std.error:0.01279).

Ko et al. [133] analysed the titles of problem reports from several online bug tracking systems to investigate what roles do the words with various parts of speech play in identifying software problems. They collected 187,851 reports title from Eclipse, Apache, Linux, and OpenOffice and manually classified the titles' words and phrases, generating descriptive categories. On a sample of 1000 titles, they reported that *App components, User action* (steps to reproduce), and *Data type* are frequently repeated.

A group of studies in this sub-section [48, 58, 62, 65, 68, 120, 130, 143] have selected the Application Aspect as an important factor for identifying useful reviews based on their observations over the collected data, inspirations gained from analysing the data, or relying on related work. The importance of this factor was validated in [119, 131] by hiring six product design students for annotating useful customer reviews for product design purposes and surveying them with two questionnaires about usefulness factors. Ko et al. [133] found that this factor is one of the frequently repeated phrases in bug tracking systems indicating the importance of this factor. The findings of analysing the results of a vast survey of developers, in [138, 141] revealed the importance of this factor from developers' perspective (discussed in Section 3.3.4).

3.4.2 Feature Request

This factor indicates the request of any feature, component, or functionality to be added to the target application. For example, broadening the width of the display is a feature request in "Very annoying to not be able to see an entire file name. Put some flexibility in the width of the display".

To address his/her requirements, sometimes a user suggests workarounds and ideas that occasionally deliver motivations and ideas for new features [156, 157].

Feature request was another category identified in [65] and [48] and is defined as "sentences expressing ideas, suggestions or needs for improving or enhancing the app or its functionalities."

Al-Subaihin et al. [37] studied the app store as a phenomenon from the developers' perspective. They analysed records of interviewing 10 app development team members and 103 completed surveys of app developers, regarding their interactions with app stores. Analysing questionnaires, they found that 66% of participants have considered reviews containing feature request as important.

Jha and Mahmoud, [69, 70] proposed a FrameNet tagging based approach to classify reviews based on the notion of semantic role labelling (SRL) into *bug reports* and *feature requests*. In their classification task using Naive Bayes (NB) and Support Vector Machines (SVM), the authors used frames generated from each review, rather than each word. "Our system MARC extracts and classifies user reviews into fine-grained software maintenance requests, including *bug reports* and *user requirements*", they said which indicates the importance of these two characteristics of app reviews.

Galvis and Winbladh, [54] used topic modelling and IE techniques to discover the topics from reviews that can be used to change and/or create new requirements for a future release of software. The authors manually classified reviews to build their gold standard dataset and observed that Feature Request is one of the frequently repeated topics in reviews.

Maalej et al. [60, 61] proposed a method for classifying reviews into four basic types: *Bug reports*, *Feature requests*, *User experiences*, and *Rating*. They defined these types according to previous studies [48, 54] and for the importance of these types of reviews for software evolution tasks.

Studying some relevant online forums to identify what kinds of information do real app developers consider as constructive, Chen et al. [57] found that *issue reports* and *feature requests* contain important information that app developers are looking to identify.

Validation of the importance of Feature Request as an important factor for identifying useful reviews was based on authors interpretation of usefulness in [69] [70] [54], while Al-Subaihin et al. [37] validated it by interviewing developers and

analysing questionnaires filled by them. Referring to forum discussions of experts was the reason of defining this factor in [57] and authors in [60, 61] relied on related work for proofing the importance of this factor.

3.4.3 Issue Report

This factor will be captured when a review is complaining about an issue. In this study, an issue consists of bug, crash, freeze, force close, error, lack of a feature, security issue, usability issue, and any other type of problem that might happen for a mobile application. For example, the following review is reporting an issue about uploading videos: "I can upload pics very quickly, but uploading videos is very slow".

The survey and interviews conducted by Al-Subaihin et al. [37] indicate that 28% of participants receive bug reports from Google play store, 26% from Apple store, and 14% from other app stores. Moreover, 70% of them said that bug reports are a very important review type. Bug report was also one of the important types of reviews defined by Maalej et al. [60, 61].

Khalid [49, 51] manually analysed 6390 one-star and two-star reviews for 20 iOS apps in order to aid developers by listing the most frequent complaints. They discovered 12 types of issue and complaint about iOS apps in user feedback focusing on the user rating given to the app.

To explore how accurately they can mine rationale concepts from the reviews, Kurtanović and Maalej [73, 147] applied classification algorithms Naive Bayes, Support Vector Machine, Logistic Regression, Decision Tree on a set of annotated data. Seven software developers analysed 32,414 reviews. Through a grounded theory approach and peer content analysis, they investigated how users argue and justify their decisions. In their content analysis task (open coding), codes related to software evolution were grouped into concepts. They assigned code for *Issue* concept to a sentence, if it reports a *concrete issue* or problem with the software. The code *alternative feature* was assigned when the user mentions a feature of another software, an improved version of a feature, and a missing or *requested feature*.

According to the discussion in previous sub-section, the authors in [60, 61] relied on related work for justifying and validating the importance of Feature

Request in identification of useful reviews. Inspirations and ideas gained from manually analysing reviews was the justification for the importance of this factor in [49, 51], while interviewing developers was conducted for this aim in [37, 73, 147].

3.4.4 User Action

User action refers to the action that the user has taken before encountering the issue. In particular, user should have taken one or several actions raising the issue. This factor helps developers to figure out the steps to reproduce the issue. The following reviews comprise examples of user action: "every time I open it, it forces to shut down", and "current file view is lost when I multitask between windows."

Bettenburg et al. [138, 141] conducted a survey among 872 developers from APACHE, ECLIPSE, and MOZILLA projects to find out characteristics of a good bug report. They asked the developers to complete the survey on important information in bug reports and the problems they faced with them. There were two main questions for developers in the survey. For the first question; "Which items have developers previously used when fixing bugs?", they provided 16 items selected based on Eli Goldberg's bug writing guidelines [158] for developers to choose. For the second question: "Which problems have developers encountered when fixing bugs?", they listed 21 problems for developers to choose. The authors analysed a total of 156 received responses and reported that: "the most widely used items across projects are *steps to reproduce (user action)*, *observed behaviour (system action)* and *expected behaviour (expected action)*". *OS version*, *components (Application Aspects)*, and *spell and grammar errors* were considerably voted by developers as well.

User action (steps to reproduce) was one of the frequently repeated title words in software problem reports analysed by Ko et al. [133].

As it is discussed in Section 3.3.1, Bettenburg et al. [138, 141] validated the importance of this factor by surveying developers, while Ko et al. [133] found that this factor is one of the frequently repeated phrases in bug tracking systems.

3.4.5 System Action

This factor is captured when the review is reporting an issue and explaining what the system does when the issue occurs. Identification of this factors helps the developers to diagnose the issue easier. The following reviews comprise examples of system action: “After the last update, it crashes every time I open the manual upload”, and “after the last update the app has a constant and annoying sidebar that steals screen real estate.” Analysing the questionnaire results, Bettenburg et al. [138, 141] reported system action as a most widely used feature. Please refer to section 3.3.1 for validation of the importance of this factor.

3.4.6 Expected Action

This factor explains what action is expected from the system. Particularly, when an issue occurs, the system action will be to reflect to the issue (e.g., freezing, showing an error message, crashing, etc.). when the user reports what was expected from the app to do instead of performing the system action is named expected action. The review “Why is the option gone to hide the menu and dedicate that screen real estate to full screen viewing of content?” is an example of reporting app expected normal behaviour. Expected action is reported as an important factor in discovering useful application reviews [138, 141]. Please refer to sections 3.3.1 and 3.3.4 for validation of the importance of this factor.

3.4.7 Device Information

This factor stands for any information reported in the review helping developers to identify user device. To meet this factor, three types of information are expected to be seen in the review. First, what device the user owns (e.g., iPhone, iPod, tablet, etc.). Second, what model is the device (e.g., S5, 6S plus, Note II, etc.). Finally, what Operating System is installed on the device (e.g., iOS7, Android 11, etc.). These types of information are reported as important factors in discovering useful application reviews [138, 141]. Please refer to sections 3.3.1 and 3.3.4 for validation of the importance of this factor.

3.4.8 Pros and Cons

This factor wants the review to mention both pros and cons of the target application together.

Jin et al., argued that "A review tends to be regarded as a helpful one once this review mentions both the pros and cons of a product [131]. To build a model for extracting helpful reviews, Jin et al. [119, 131] randomly selected 1,000 reviews on eight mobile phone brands from Amazon and annotated them hiring six full-time final year undergraduate students as product designers who scored review helpfulness from -2 to 2. Interviewing the annotators in [119], two subjects explicitly stressed that the appearance of ''*both pros and cons*'' is an important factor for helpfulness evaluation. Building on results reported in [119, 131], Qi et al, [120] verified the helpfulness of combining pros and cons with meta-data factors for identification of useful reviews (Section 3.3.1).

In [119, 131], the validation of the importance of this factor was interviewing product design students. This validation was cited in [120].

3.4.9 User Expertise

Value of this factor shows how expert is the reviewer of a written review. The importance of this factor is based on this argument; the more expert user, will generate more useful review for software development purposes [149]. Considering limitations of publicly available datasets, user expertise is defined as total number of reviews generated by a reviewer [151].

Studying forum instructions and developers opinions, and surveying developers Heydari et al [149, 151] defined several quality metrics and developed a voting model based retrieval system to retrieve more relevant threads in technical forums for a given user query. They enhanced the retrieval process by leveraging the quality metrics. One of these quality metrics was user expertise defined as number of reviews posted by a reviewer. Their results show that using reviewer's metrics, including user expertise enhances the quality of retrieved threads by about 4%. The results of merely using this factor are not reported. However, the Normalised Discounted Cumulative Gain (NDCG) @30 improved from 0.318 to

0.345 using a combination of user related factors. NDCG is an evaluation tool to validate the effectiveness of search engines.

For identifying useful reviews, *reviewer's expertise,* one of the features obtained from review meta data, was considered as an important factor in [120] (section 3.3.1) in which two metrics are defined for this factor; Number of reviews by a reviewer which is the volume of reviews posted by a reviewer, and The grade of reviewer which indicates the reviewer's activeness in the website. They applied SVM-based model to their conjoint analysis model and reported the impact of 'The grade of reviewer in a review' as: (Estimate:0.00091, Std.error:0.0002).

With the aim of predicting the helpfulness of product reviews, Lee and Choeh [150] proposed a tool to predict the level of review usefulness using the determinants of product data, the review characteristics, and the textual characteristics of reviews. They collected and analysed 1834 product reviews from Amazon.com and adopted A neural network with a three-layer architecture consisting of input, hidden, and output layers for detection of usefulness. In a table, they listed the strengths of features as a result of running the experiment. Although *product type* (product name), *number of spelling mistakes*, and *reviewer's expertise* was in the list, greater values were for *product rating, review extremity* and *length of review*.

Heydari et al [149, 151] defined user expertise as an important factor based on their analysis of technical forum guidelines, interviewing reviewers and engineers using these forums. This factor was validated in [120] by citing related studies. In [150], the factors were defined based on authors' discretion.

3.4.10 User Rating

Rating is the number of stars given to an application in a review. It has been argued that more negative reviews (i.e. receiving less than 3 stars) contain more useful information for application development [58] [62] [133] [135]. Researchers have argued that *negative comments* should be considered more helpful than positive ones because they deviate from the accepted norm of staying positive [159, 160].

Focusing on the rating of reviews, Khalid et al. [49, 51] manually analysed 6390 one-star and two-star reviews to aid developers by listing the most frequent complaints. They discovered 12 types of issue and complaint about iOS apps in user feedback focusing on the *user rating* given to the applications.

In another study targeting only one-star and two-star reviews which highlights the role of *user rating*, McIlroy et al. [59] studied the extent of reviews raising more than one issue type and proposed an approach to automatically labelling user reviews to help developers better understand users' concerns. They defined 13 types of issues and labelled a number of reviews manually to form their gold standard dataset.

User rating was also highlighted in [106]. To better understand the reason of leaving positive or negative review by a user, the authors analysed 8.7 million reviews from 17,330 apps and categorized keywords appearing frequently in each star rating as they hypothesized that it could inform and focus development efforts. The authors determined the distribution of word and character counts per star rating respectively applying a regular expression to extract words from the review entities, and monitored which star rating the appearance of the extracted keywords pertains to.

Definition of this factor in [49, 51, 59, 106] was based on authors' inspirations gained from analysing a sample of reviews.

3.4.11 Length of the Review

This factor reflects the number of words and sentences in a review. Findings from different studies reveal that longer reviews contain more useful information for developers [60, 61, 149, 150, 161, 162]. Salehan et al. [152] applied binomial regression with logit transformation to examine the effectiveness of the features in predicting usefulness of product reviews. They collected and analysed 2616 online reviews of 20 different products from Amazon.com to evaluate their model. Their results revealed that *review length* and *review sentiment* are significant predictors of helpfulness.

Validation of the importance of this factor in [152] was their insights gained during the analysis of reviews, while the authors in [149], relied on their studies of forum guidelines and interviewing software engineers. Authors' experiences were the source of definition of this factor in [150].

3.4.12 Readability

Readability of a review depends on number of grammar and spelling mistakes found in the review. Related studies have argued that reviews with less grammar and mistakes are more useful for software development purposes [138, 141, 150].

Lee and Choeh [150] used number of spelling mistakes as one of the classification attributes in their neural network model to detect useful product reviews believing that reviews written with more selling mistakes contain less useful information for product designers. Number of spell and grammar errors was one of the factors chosen by developers surveyed by Bettenburg et al. [138, 141] while they were reflecting their problems and issues in the process of reading bug reports and fixing the bugs.

3.5 Discussion

As it could be observed in previous section, variety of approaches are proposed so far to extract useful information from user feedback to develop and improve software products. However, in majority of these studies, metrics and factors to measure the usefulness are defined based on variety of considerations such as authors' interpretation of the usefulness, consulting a limited number of experts, and etc. Each of the studies has their own arguments for relying on a factor which might be reasonable. This could be a probable reason for using different combinations of factors and metrics for measuring usefulness in different studies and reporting the accurate ones for extracting useful reviews.

Another noteworthy point in the literature review section is that all of the references mentioned in Table 3.3 for an attribute might not be reported in the sub-section discussing the attribute for the following reasons: (1) the study is reported in another sub-section as its main focus was on another factor, yet has used this

factor as well, (2) the study has just reasoned about the importance of the factor without experimenting with it. Studies of the latter group are cited as readers might want to refer to these studies to better rationalise and justify their usage of a factor.

Apart from above-mentioned discussion, as the extraction of user requirements from user feedback has gotten considerable research attention, the rest of our discussion section will discuss the usability of the outcome of the research reported in this chapter as well as challenges that the community should still attempt to address. This section becomes to an end with discussing threats to validity.

3.5.1 Usability of the Research Output

Our observation and analysis on the related works revealed that although variety of approaches has been proposed to extract useful information from user reviews for software application development, limited research attention is given to appropriately measuring the usefulness from developers' viewpoint. Yet, the availability of reliable sets of factors for properly measuring the usefulness is necessary not only to limit the efforts in subsequent works but also to encourage true progress beyond the state-of-the-art.

3.5.2 Open Issues and Challenges

According to the lessons learnt from this study, various challenges facing the future research in this domain and future research directions are suggested in this section.

3.5.2.1 Comprehensiveness of the Factors

The factors we came up with in this chapter are defined to facilitate and enable measuring the usefulness of user reviews. However, more important factors might be undiscovered yet. A systematic in-depth research examining real world data (i.e., user reviews) employing the expertise of RE experts and app developers might result in discovering more important factors.

Besides, analysing other areas of research might help in identifying other important factors. For example, Heydari et al. [149] has studied the quality metrics in identifying quality enhanced answers to the given queries from software forum

discussions. The impact of such quality metrics could be examined in the scope of this study as well.

3.5.2.2 Validation of the Factors

As we explained earlier, many of the factors discussed in this study are defined based on the authors' interpretation of the usefulness which might be far from what real developers and requirement engineers believe. Therefore, similar to the previous challenge, a future research direction could be studying developers' and requirement engineers' behaviour in dealing with user incoming reviews to approve how effective the existing factors are in identification of useful reviews.

3.5.2.3 Purification of the Reviews

Although identification and use of proper factors leads to more accurately filtering reviews for requirements engineering purposes, the destructive role of spamming activities in online opinion sharing platforms should not be neglected. Spam reviews could significantly mislead not only automatic extraction tools, but also human annotators [23, 163]. Variety of approaches are proposed so far to detect spam reviews [16]. However, this area of research still needs lots of attention [63]. Therefore, another future research direction could be assessing the impact of spamming activities and spam reviews in utilization of user reviews for software evolution and application development purposes.

3.5.2.4 Automatically extracting factors

Although variety of approaches are proposed to extract some of the factors discussed in this study, such as Application Aspect, Rating, and Readability of a review, there are important factors, such as User Action and System Action, that have not been analysed for automatic extraction yet. Configuring, manipulating, and adjusting current NLP and ML techniques in a way to be able to capture such factors from the context of a review will be extremely helpful for further developments and evaluation purposes.

3.5.2.5 Robust tools for identifying useful reviews

Obviously, when each of the current approaches has focused on a limited set of unapproved or partially approved factors, existing automated systems not only do not cover all available factors, but also are based on a set of factors which might not convey the viewpoint of developers. Thus, another open challenge is developing robust tools integrating existing techniques and developing new techniques for extracting each factor and calculate, accordingly, the usefulness of a given review based on developers' viewpoint.

3.5.3 Validity Threats

The first and foremost threat to validity of the research reported in this chapter is about the search process. The main risks in the search process are using an incomplete keywords list and limitations of utilised search engines. The risk of using incomplete keywords list was mitigated by performing the "quasi-gold standard" task (Section II.B). To minimize the risk implied by using a search engine with specific limitations, we used three different search engines (Section 3.2.2).

Another threat to the validity of this study is the validity of discovered factors. In this chapter, the factors defined and used in the related work are summarised in a list. However, the impact of these factors in identification of useful reviews from developers' viewpoint is still under question. To mitigate this issue, I tried to explain arguments and justifications of the selected studies about each factor to give a better understanding regarding the impact of the factor to the reader.

3.6 Conclusion

The main objective of this survey study was to summarize and synthesize the existing studies related to analysis of user feedback in order to identify which factors have been used so far for measuring the usefulness of user generated text from the viewpoint of a developer, designer, service provider, or requirements engineer. To fulfil this objective and answer the defined research questions, a survey of related works was conducted. After defining a series of systematic steps for selecting and

assessing the quality of the related papers, we comprehensively analysed them to answer the research questions.

Apart from outcomes explained in previous sections, findings from this study reveals that a set of factors is identified to be used for measuring the usefulness of user reviews from developers' viewpoint. However, validity of these factors is not yet approved for the following reasons: (1) The validation methods used in related papers were based on authors' interpretation of usefulness rather that developers' one, (2) The related work has just introduced the factor without validating it, (3) The factor is obtained from studies focusing on other fields of research than mobile application development. Therefore, using them for implementing detection tools is subject to a rigorous validation process.

Chapter 4 Validation of the Usefulness Factors

4.1 Introduction

The aim of this chapter is to use the skills and experiences of real developers dealing with huge volume of user feedback daily to validate and qualify the role of each factor discovered in previous chapter. Coming up with a set of qualified and confirmed factors results in implementation of extraction tools reflecting developers' viewpoint in detection of useful reviews for software evolution purposes.

To evaluate the effectiveness of each factor and to obtain the correct viewpoint of application developers, A focus group discussion (FGD) with senior mobile application developers were conducted asking them to quantify the impact of each factor and to propose new factors. Details on conducting the focus group and obtained results from analysing the outcomes of the FGD are discussed in this chapter.

4.2 Focus Groups

Focus groups research techniques are originated from social science for exploring and better understanding how people think about a specific topic [164, 165]. Focus group discussion involves assembling a group of peers and free-flowingly discussing the topic. Depending on the topic, the participants should be selected based on their specific characteristics. For example, they might be experts and engineers to be selected for a technical topic, patients suffering from a particular illness to be selected for a medical topic, or teachers, teaching assistants, and mentors to be selected for an educational topic. Asking several questions, participants will be able to give their opinions and views. Then, any raised issues will be discussed in the group. A researcher controls the discussion and asks the main questions. The

process of designing, conducting and reporting the FGD in this project is illustrated in Figure 4.1.

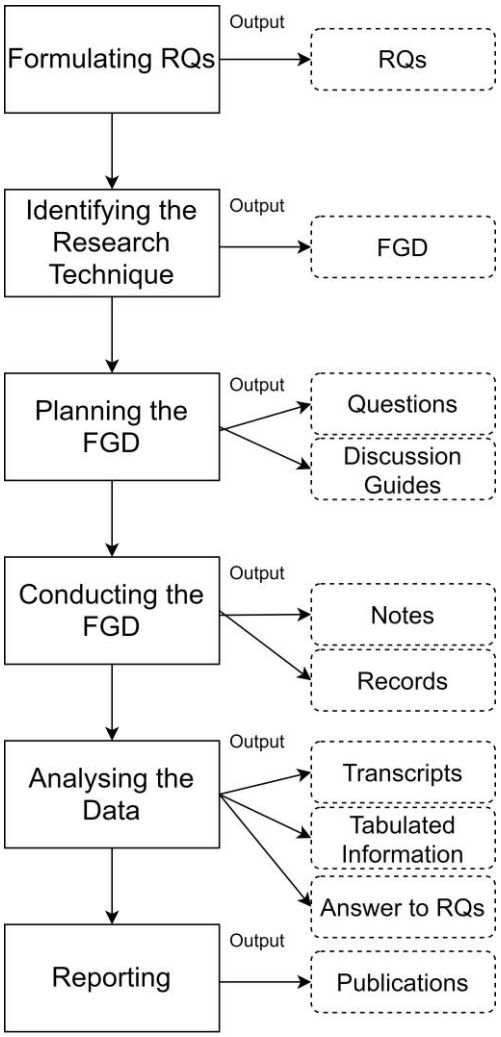

Figure 4.1. The process of conducting the FGD

4.2.1 Selecting Focus Group Discussion

In this research, Focus Group Discussion (FGD) was chosen as an evaluation method to validate the identified factors. The key means of generating data in a FGD is interaction of participants [166] which results in gathering valuable insights and data [167, 168]. Views and experiences of different participants will be purified during the discussion and will contribute to build various aspects of a quality amplified idea. The aim of my task was not only to explore common customs among the community of mobile application developers about usefulness of application reviews, but also to synthesize and discuss various views of different developers

about each factor to have a comprehensive assessment and quantitation of the effectiveness of each factor in identification of useful reviews for software evolution purposes. Therefore, the FGD technique was selected among other methods.

Alternative approaches would have been to use interview or questionnaire. Interview was not chosen as the preferred approach because the aim of this study was not to probing developers' individual experiences, but on a more collective view of what developers' considerations are in identifying useful reviews. Thus, to judge a factor or propose a new one, active discussion and interaction of experts was required.

Another alternative method would have been conducting a survey among developers, but this was not chosen as well. Apart from the reasons mentioned above for not conducting interviews, not many developers filled the designed online questionnaire[1] to generate a reasonably sized sample that would give the study enough power to yield useful results. Moreover, to fully understand each factor and being able to correctly score them, target developers should have been exposed to examples, interactive discussion, and extra explanation.

4.2.2 Aim and Research Questions for the FGD

The aim of conducting FGD was to validate the discovered factors and to investigate the importance of each factor on identification of useful reviews as well as probing for emerging factors not covered in my literature survey. To achieve this aim, the following research questions were defined to design the FGD accordingly.

RQ1. To what extent developers pay attention to app reviews as a source of user requirements?

---

1 https://docs.google.com/forms/d/e/1FAIpQLScvSU5VbjEkhM0I0Y_WGo6jXVK2nGrNeNl48_VccWLhmUOb5w/viewform?usp=sf_link

This research question is defined to discover the importance of user reviews for developers in terms of extracting user requirements. Particularly, the aim of this research question is to understand to what extent the developers consider user reviews as a source of user requirement to improve and update their existing applications or to get insights and ideas for developing new applications.

RQ2. What factors are critical for identifying useful app reviews from developers' viewpoint?

Answering this research question, a list of factors for a mobile application review to be considered as useful for software evolution will be generated. Although performing the literature review, a list of factors was already generated, considering this research question in the FGD will result in obtaining probable emerging new factors.

4.2.3 Sampling Strategy and Characteristics of Participants

The target group is defined as a sufficient number of mobile application developers with at least five years of relevant experience. To recruit the participants, the following constraints are defined based on the aim and situation of the project. To be considered as a candidate for taking place in the focus group, each of the developers needs to (1) have, individually or as a member of a team, more than ten mobile applications successfully developed, (2) have received more than 1000 reviews via local or international online mobile application sharing platforms, (3) have more than 4 years of mobile application development experience. No gender balance or age was considered in recruiting the participants as the issue to be discussed was only based on technical experiences of developers. So, representative (i.e., random) method was not considered as the sampling strategy, rather, purposive sampling was used to cover important group experiences about mobile application review analysis. Purposive sampling refers to designing the sample in a way that includes important segments of the population, or experiences and expertise required by the research [169, 170].

As attending the FGD sessions in two different locations (i.e., Iran and the UK) was possible for the moderator, the focus of searching process for finding

developers was logistically limited to these two countries. Searching websites serving as platforms for hiring expert freelancers, online Android and iOS communities, and online mobile application sharing platforms such as Google play store and App store, a list of 500 developers in the UK and 200 developers in Tehran meeting the above-mentioned criteria as candidates was generated to design the sample.

The developers were then contacted by e-mail to check their availability and willingness for participation in a focus group. The email was briefly explaining the research project and outlining the focus group content. From the UK list, only one developer was willing to attend. However, 15 individuals and companies replied from Tehran asking for introductory meetings to discuss the project and the need for FGD, explain their constraints, and assess the sampling requirements. Finally, 6 developers were selected as the target group for the FGD in Tehran.

The guidelines designed to be followed in the FGD, the type of questions, and expertise level of participants

4.2.4 Composition of Focus Group

The information of participants is given in Table 4.1. In this thesis, each participant will be referred to using an ID represented in Table 4.1. Among the developers shortlisted to take place in the FGD, there were two head of companies with thousands successful developed applications worldwide. D1 is the owner of MTeam.co developing mobile applications for variety of businesses and end users in Persian, Arabic, and English languages. Besides developing applications, her company provides consulting services for founding businesses based on mobile applications.

D2 is the founder of Saffaran.co, an international mobile application developing company. After analysing application developers' requirements for several years, his company has proposed a mobile application development SDK. Software Development Kit (SDK) is a tool facilitating the process of developing a software or mobile application.

D3 is the head of mobile application development department in Delta.ir and

Table 4.1: Information of Participants

ID	Skills	Year of Experience	Role	Organization	Size of the Org.
D1	Android/iOS development, RE, Project management	12	Head of Company	MTeam.co	89
D2	Android/iOS development, RE, Project management	10	Head of Company	Saffran.co	83
D3	Android/iOS development, RE, Project management	9	Head of App development Dept	Delta.ir	103
D4	Android/iOS development, RE,	7	Head of App development Dept	Farzan.co	75
D5	Android/iOS development, RE,	8	Mobile app developer	Freelancer	NA
D6	Android/iOS development, RE,	7	Mobile app developer	Freelancer	NA

is the founder of TaMart start up, an online local grocery with rapid deliveries. D4 to D6 are mobile application developers working in various companies and doing freelance mobile application projects.

4.2.5 Number and Size of Focus Group

Different factors such as scale and aim of the project, circumstances and characteristics of participants mediates the number and size of groups required for a project [165].

Pertaining to the size of the group, the number of participants in this project, six developers, is reasonable. This is because the type of questions is straight forward, technical, and generally quantitative. So, unlike FGDs discussing human behaviours, personal opinions and people thoughts, several individuals with different beliefs are not required. Moreover, based on the requirements of the candidates in this study, and considering the top management level of their positions, finding more well-skilled developers who accept to collaborate in research projects was almost impossible.

The initial plan was to do only one FGD because the aim of the research was not too broad demanding various kinds of participants' opinions and points of view as a qualitative research project probing complex issues varying from one person to another such as human behaviours. My aim was to discuss the usefulness factors and validate them based on participants' technical expertise.

Moreover, the participants of such a focus group should be well-skilled mobile application developers with certain characteristics making them rare among the society. They usually receive high salaries and are extremely busy making it complicated to gather them in a focus group. Considering the circumstances of the research, my initial focus was to conduct the FGD in one session.

However, the circumstances of D1 and D2 forced me to conduct the second session of the FGD. They are working at top management level and had very tight timetable. On the other hand, they had an incredible level of experience in dealing with user feedback of various types urging me to include them in the FGD. So, I managed to conduct the FGD in two sessions. D1 and D2 had to meet regularly as they have been cooperating on several projects. Therefore, gathering them in one of their companies was not a big problem. Number of participants attended in the first session was two, while four developers took place in the second.

4.2.6 Planning the Focus Group

The instruction provided by Krueger and Casey [171] was followed for development of discussion guide. I drafted an initial list of questions for the focus group and circulated it to my supervisory team. Their comments were further applied in the

revised version of the questions and the guide was developed to be followed in the focus group.

After getting informed by participants about their interest in taking part in FGD, the time and venue of each session was fixed. I agreed on the time of the first session with D1 and D2 on a phone conversation, while for the second, several time slots were proposed by participants (D3 to D6) and all agreed on one in an email conversation. To set the venue for the first session, suggestion of D1 and D2 was to use their companies. Thus, the session took place in a room at the Saffran.co normally used for executive meetings. The second session took place in one of the discussion rooms allocated for business meetings in a café. A café at the centre of the city was suggested by me and agreed by participants to facilitate their commuting and access to refreshments.

Participants were contacted by email a week before conducting the focus group to get remind and to discuss any requirements and services they may need during the meeting. A day before the focus group, I called each participant to remind him/her and to get his/her confirmation for attending the focus group. Both of the sessions were conducted in August 2019. The first one lasted approximately 110 minutes with two participants, while the second lasted 140 minutes with four.

4.2.7 Content and Conduct of Focus Group Discussion

Following the relevant steps for conducting the FGD (Recording, Introduction, Pausing and probing, Managing discussants, Concluding) [172], I started the discussion with welcoming the participants followed by asking each of them to introduce him/her self and to give a brief summary of his/her background and experience in mobile application development and involvement with user feedback in form of reviews. Next, a brief summary of the operation of focus groups were explained as some of the participants were not familiar with FGD techniques.

Then, I had a short presentation on my project, particularly, explaining the role of the FGD in my research. Focus of the presentation was on the impact of mobile application reviews on software evolution and formulating the usefulness of them. A sample of the usefulness factors and examples of real user reviews were given to

clarify to the participants how the approach aims to formulate the usefulness. As the facilitator and moderator of the focus group, I posed the questions designed in the guide to discussants and managed the discussion.

4.2.8 Data Analysis

The analysis discussed in this section is a synthesis of the two conducted sessions. The first phase of analysis begins during the FGD [173]. Participants' responses and contributions should be clearly understood, and any ambiguity should be addressed as they might convey important information for further analysis phases. At this phase, I tried to listen carefully to the discussions and asked for more explanation for any unclear statement. Pausing and probing techniques were used to gather more in-depth information about the interesting topics.

I audio recorded the focus group discussions. Following Jenny Cameron's guidelines [174] on FGDs, the recordings were translated into English and abridgedly transcribed (i.e. only key points and section of the discussion were transcribed).

The strategy used next to proceed was inspired from Brown and Ward [175] as the responses did not diverge widely from the questions. Their suggested methodology for analysing and reporting the data gathered in focus groups is to draw a table, one column for each question or theme and listing key points and quotes for each question. The themes and important quotes will then be used in reporting the results.

Analysis of the responds to the first research question: 'How important user reviews are for you to improve and update your applications?' reveals that user reviews are viewed by respondents as an important source of user needs, though they are too noisy consisting irrelevant contents. All the participants agreed on a fact stating that although a large percentage of user reviews are irrelevant, there are important useful reviews persuading us to analyse the whole bunch to discover them.

As the following responses suggest, there was a strong feeling amongst the participants that user reviews are very important for software evolution. D1

mentioned that “when we develop apps for a business (BtoBtoC) we do not care about user reviews because the business ordering the app is responsible for its quality, rather, we only make changes in the app based on either reasonable or unreasonable requirements of the intermediate business. However, when we produce an app for public (BtoC), we carefully track the reviews and apply useful ones to keep our product user friendly and bug free.”

Moreover, D3 stated that “publishing a mobile app online, the best mean to get feedback from actual users is analysing reviews. We hold designers’ and developers’ point of view, not end users of the application interacting it every day. So, when we test the app instead of the end user, we cannot completely understand their issues and comprehensively cover their requirements.” “there are plenty types of issue emerging when the app goes under load of hundreds or thousands of clients. So, we might miss some of these issues in application testing phase before releasing the application. It is very likely to see these issues reported in user reviews.” D5 pointed out to support D3’s argument about usefulness of user feedback.

What eventuates is the importance of user reviews for development teams and the necessity of analysing them by requirements engineers to better understand user requirements and to identify issues missed out during the testing step of mobile application development lifecycle.

Research question 2 probes and discusses developers’ experiences in investigation and analysis of user reviews. The aim of this research question is to discuss any characteristics and factors developers take into account while processing user reviews to filter out useful reviews.

After asking participants to manually classify five reviews to useful or useless based on their experience and to address research question 2, participants were confronted with this question “What characteristics and factors do you consider when distinguishing between useful and useless reviews?”. The method for analysing this part of the discussion was a standard content analysis approach proposed by Elo and Helvi, [176] to identify and cluster factors proposed by the participants.

Participants initially remarked that the skill of identifying useful reviews demands years of experience for a developer or requirement engineer, whereby, he/she can decide to filter out useful reviews for extraction of requirements.

D2 explained that "We always ask well-skilled developers or requirement engineers to do this task. We do not ask them to filter out useful reviews according to a given checklist, but we accompany them double checking their work few times to see how well they can detect useful reviews."

However, more in-dept discussion on this matter between participants revealed that some of them have already defined some characteristics and factors for useful reviews, while the others are involuntarily following some factors. All participants jointly exposed that the most important reviews are the ones reporting an issue.

D6 mentioned that "The most important types of information I look for in the reviews is bug reports or technical issues that the user has encountered with. If I find enough information to diagnose the problem in the application, I will consider the review as useful." D5 supported this argument saying "Yes, exactly, but the reported problem needs to be clearly explained. I mean, reasons for dissatisfaction of the user must be mentioned in the review. Otherwise, we cannot use the review. For example, we receive many reviews mentioning that the app crashes and users cannot use the app. However, we have tested the app on several devices successfully. Moreover, it happens only for a few percent of our clients indicating that the rest are using the application flawlessly. Thus, the cause of crashing might not be originating from the application, but other factors such as user phone, storage limitation of user device, version of the OS used, etc. when we do not have these details, we cannot address the issue."

These arguments reveal that one of the factors of a useful review is mentioning an issue with a certain amount of complementary information helping developers to diagnose the application.

Apart from problems reported in the reviews, requesting new components and functionalities was interesting for developers. D4 stated that "not only bugs and

problems reported in the review are what we are looking for when analysing user reviews, but also any information that gives us some idea to improve the application are on demand."

Supporting D4's argument, D3 said "The application designer has a limited knowledge and ability to cover all reasonable user requirements. This is because we must guess most of the requirements for applications that we want them to be publicly available on application sharing platforms. So, many times users ask for some functionality or quality to make the application more attractive for them. In these cases, we do not discard these reviews, but put them in backlogs with low priority".

It was discovered that prioritising the factors, developers are more concerned about addressing users' reported issues to preserve the user satisfaction, application rank, and popularity. Then, they will take inspiring ideas and suggestions into consideration.

A list of factors extracted from the literature was exposed to the participants afterwards. The participants were asked to discuss the impact and necessity of each factor in identification of useful reviews to answer the last research question. The importance of issue report, feature request, user action and system action were confirmed by all the participants after a short discussion as they all believed that without these factors, a review cannot be useful at all.

However, they had a negative opinion about expected behaviour because it reflects the normal behaviour of the application. D1 argued that "This factor reports what I already know about my application. it does not bring any new information for me. So, I do not care if it is mentioned in a review or not." The only positive vote was for D6 who stated that "when diagnosing a reported issue, this factor help us to, at least, understand that the user knows what the normal performance of the faulty aspect is. When the user has misunderstood the normal performance, reports an issue, and explains the expected action of a process, for example, there might not be an issue with the process, but a wrong expectation of the user is the issue."

Application aspect was considered as a necessary factor as it helps developers to find which part of the application is the subject of the review. D5 mentioned that "if a review is an issue report, we must know which part of the application is involved to diagnose and fix it. Similarly, if the review is a feature request, we must know what aspect or feature is required to be added to what part of the application."

Discussing the impact of device information eventuates that this factor helps to figure out the cause of the reported issue only in very particular types of issue. D2 stated that "when we receive a review complaining about a problem, if there really is a problem on the application side, we usually diagnose the application and address the problem without requiring to know about user's device, because our audiences are public using variety of devices demanding our applications to be responsively designed to support majority of available devices. However, if the cause of the problem is on the user side (e.g., old user device, full memory, incorrect usage, etc.), we usually ignore the review."

D1 argued then that "in all of these years that we have been processing user reviews, we rarely needed to know these details to know what the reported issue technically caused by. As far as I can remember, in all of these cases, there was nothing wrong with the application that we can fix, but the client's side was the cause of the issue."

Beside these arguments, D5 mentioned that "When we develop an application, we provide documentations and specifications for the customers mentioning the versions of the operating system that are supported by the application. So, if the user uses unsupported version, it is obvious that some of the features will not work properly."

Impact of the review length was discussed next in the focus group. The overall opinion of the participants was that although longer reviews have the potential to include more useful information, but lots of short reviews point out exactly what developers need to know. Moreover, the experience of developers shows that when they release applications in domains such as tourism, leisure, beauty and skin care, etc., they will receive lots of lengthy reviews explaining users' experiences and how

they enjoyed the application which reading them is nothing but waste of time for developers.

Mentioning both pros and cons together in a review was known as an ineffective factor. D6 highlighted that "lots of the useful reviews we receive daily are completely negative without mentioning any pro about the application or vice versa. So, why should we care about this factor?" D4 added "In case of negative reviews, that are usually very important, the user is indignant with an aspect of the application. Thus, that is meaningless to expect him/her to write about positive points of the application."

A similar decision was made for readability factor as well. According to the participants' discussions, carefully and correctly writing it does not guarantee the quality of the review. D3 stated that "people are from different levels of education. However, these differences do not affect their ability of reporting helpful feedback."

Asking participants about the impact of the user expertise, D5 said "We do not have access to reviewer's profile on application sharing platforms. So, we cannot properly judge his/her expertise." Discussions revealed that relying merely on number of reviews generated by a reviewer is irrelevant to the quality of review. D3 said "there are lots of spammers generating tens of crap reviews, while normally regular users use the application and generate only one review explaining issues, requirements, or their experience with the application."

Impact of data type was considered as minimal. Developers believed that data type helps, but only if the aspect of the application being criticized deals with input/output or data. For example, complaining about the size of menu items does not involve any data type".

The last discussed factor in the list was user rating. Participants had variety of opinions and analysis about this factor and its impact on identifying useful reviews. Summarizing the discussion reveals that the polarity of sentiment in a review might help in predicting what type of review it is. 5-star reviews are usually written for appreciation and promoting the application. 3-4 stars generally show that the user is happy with the application but needs some modifications or extra functionality.

1-2 stars reflect user complaints and issues with the application. However, developers had been encountered with spam activities generating burst negative reviews posted for reducing the application overall rank. These reviews, obviously, consist of useless words.

After discussing each factor in the list, the participants were asked to score the importance of each factor from 1 to 10. Averaging the scores for each factor, Figure 4.2 represents the level of importance of each factor. This scoring helped in defining a threshold and focusing only on factors achieving more scores. Next section details this process.

Inspired by discussing the list of factors elicited from the related studies, participants came up with some ideas about new factors based on their experiences. Therefore, apart from the factors discussed in answering research question 2, the following factors were elicited from discussing existing factors.

One of the suggested factors was number of helpful feedbacks. When a reviewer posts a review in an online application sharing platform, other users who read this review can give it a positive or negative vote. Summation of these votes form the number of helpful feedbacks for a review. However, discussing this factor

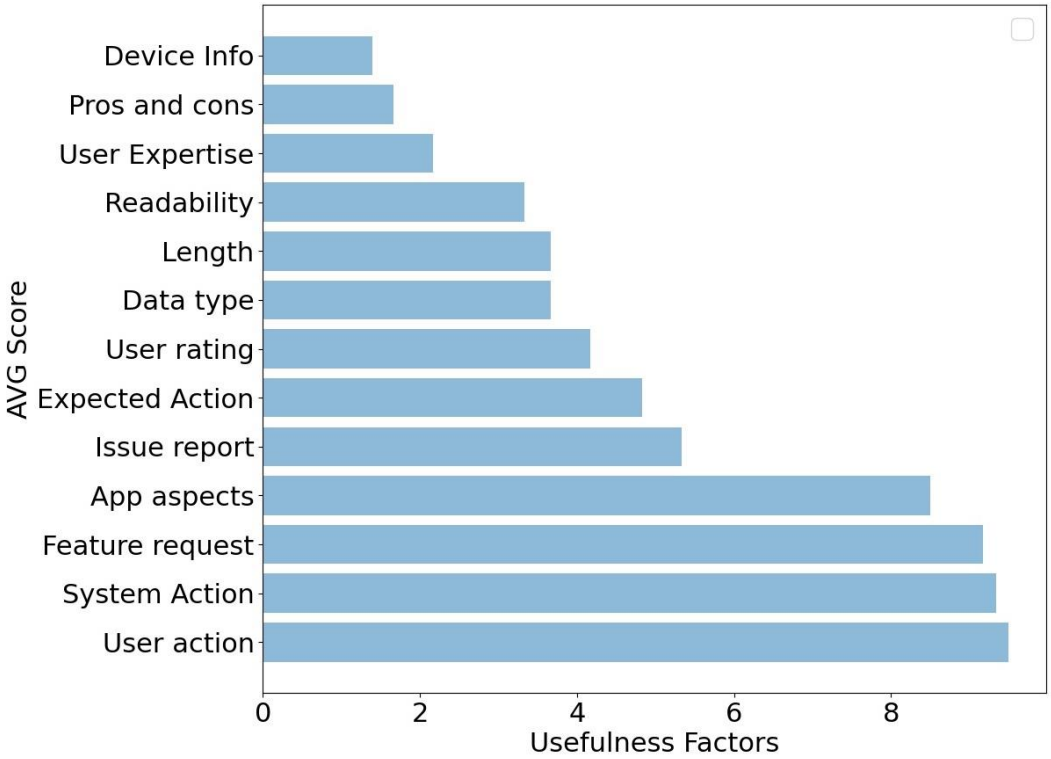

Figure 4.2: Ranking of the usefulness factors according to experts' scorings

revealed that these votes are given by other reviewers. Thus, they do not cover developers' viewpoint for voting the review.

Another suggested factor is reviewer's age. D3 believed that if a developer knows the reviewer's age, he/she can judge the usefulness of the review more effectively. An example was given by him "When an application developed for nursing purposes (e.g., monitoring patients after a heart surgery), is reviewed by a 14-year-old user, this review will not obviously be considered as useful."

D6 suggested reviewer's location as a helpful factor. He argued that "Usually, people of each country, or district have a certain lifestyle, culture, and taste. People in some countries use variety of warm colours in their cloths, while some others prefer mono colour. Therefore, if we study these preferences and have the reviewer's location in hand, we can classify reviews more effectively."

4.2.9 Conclusion

After analysing the outcomes of the focus groups, a proper decision on what factors significantly affect the detection of useful reviews was made. The impact of

application aspect, feature request, issue report, user action, and system action were justified and confirmed throughout the focus groups. Almost all of the participants agreed that without these factors, it is impossible to detect useful reviews. Most of these factors were suggested by participants before representing the list of the factors elicited from the related studies which is another evidence of the great impact of these factors.

Participants denied the impact of expected action, pros and cons, readability on detection of useful reviews as they believed that these factors are irrelevant to the usefulness concept. However, about the importance of data type, user rating, and length of the review, they argued that these factors might be helpful in some cases, but they cannot be considered as criteria for measuring the usefulness. Data type was refuted because it might not make any sense in many situations. Furthermore, it has been reported that user rating has not been matched with the sentiment of the content in a review [16, 63, 177].

To refute the length of the reviews, example of short, but useful reviews were given by participants. This is noteworthy that although the importance of some factors is refused by participants, they have given scores to these factors. The scores are not too high yet indicating the importance of the factors. Participants justified this behaviour by explaining that these factors could be used as complementary factors meaning that meeting them in a review will incentivise developers to find useful information, but they are not useful themselves. It means that missing these features, they still can find useful reviews.

Reviewer's age and location were supposed as helpful factors in identification of useful reviews by participants. However, accessing these types of information on online mobile application sharing platforms is impossible due to data protection policy restrictions. Similarly, number of helpful feedbacks was unsuccessful due to not reflecting developers' viewpoint.

Chapter 5 Modelling the Usefulness of App Reviews

5.1 Introduction

With the analysis of the outputs achieved from the literature review and the focus group discussion, this thesis proposes a novel approach for conceptualizing the usefulness of mobile application reviews for software development purposes. To provide a background for the proposed approach, this chapter describes review components and conceptualises the reviewing process.

Section 5.2 introduces a mobile application review and its components. The process of reviewing an application is conceptualised in Section 5.3 showing what a user encounters while using the application and how reflects issues and feature requests in his/her review. Finally, Section 5.4 summarises and concludes the contents of this chapter.

5.2 Components of a Review

A mobile application review posted on one of the existing online mobile application sharing platforms such as App Store and Google Play Store, consists of several components. The most important part of a review attracting other reviewers, potential customers, and developers is the review body, also called content or text in different contexts. The message that the reviewer intends to deliver to the reader is to be located in this part. A review also has a title briefly explaining what the review is about.

Apart from its main components, a review comes with several metadata fields that could be used for variety of review analysis purposes, subject to availability. These metadata fields are as follow:

- Target application: The mobile application on application sharing websites which the review is posted for
- Temporal information: Date and time of posting the review

- Reviewer: The Id, account name, or profile name of the reviewer shown on his/her posted reviews
- Reviewer's thumbnail: Profile picture of the reviewer
- Rating: Number of star-ratings given to the application by reviewer which could be selected from 1 to 5 stars provided
- Helpful: Number of helpful positive or negative votes the review has received from readers

An example of data and metadata items of a review posted for an application provided on Apple App Store is presented in Figure 5.1. 'Gholam Abbas' is the reviewer name. Reviewer's thumbnail is not provided. 2 stars is given by him to the application. The review is posted on 3 April 2018. 2 readers have found this review helpful. These are the metadata fields that could be crawled along with the review content to perform related analysis tasks.

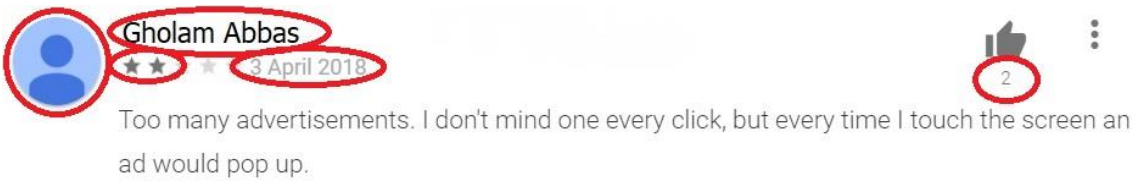

Figure 5.1: Available data and metadata for a review on Google Play Store

The target application offered on online application sharing platforms also conveys several metadata fields. These fields can also be used for processing reviews for certain reasons. Depending on the design of the online application sharing platform providing the reviewing facility, the availability of the following metadata items varies from website to website;

- Application Rank: The average of given star-ratings
- Application description: a short statement introducing the application and its main functionalities
- Developer: Name of the individual or company developing the application
- Developer contact: contact information of the developer (i.e., email, website, and phone number)
- Price: Price of the application in the market

- Size: Size of the application on the phone memory
- Version: current version of the application
- Number of instals: number of persons downloaded and installed the application
- iOS version: Version of the OS required for running the application
- Update: Date of the most recent update of the application

All of the abovementioned data and metadata items are publicly available on the websites and could be used for business and research purposes with respect to the GDPR and host countries data protection regulations. Figure 5.2 illustrates available data and metadata of an application published on Apple App Store.

Another group of metadata are privately kept by the application sharing websites and are typically not available for research purposes. These metadata items are, but not limited to;

- Location of the reviewer
- Duration of writing and posting a review
- Click stream of reviewer
- Device of the reviewer
- Number of downloads by location
- Number of reviews posted by a reviewer

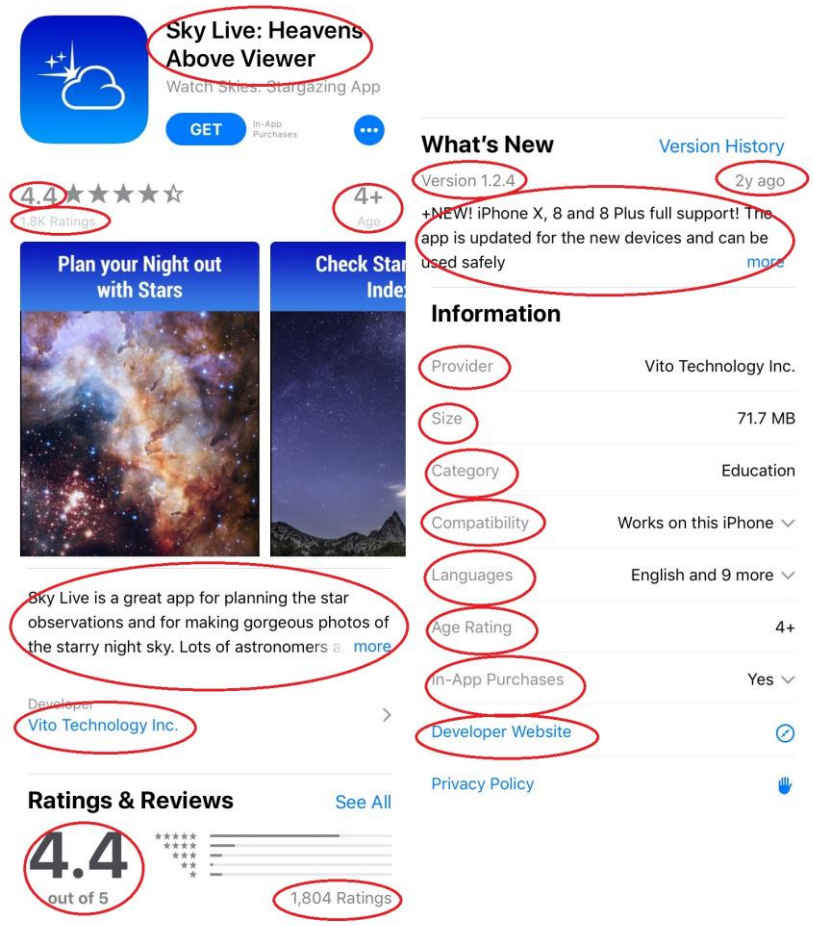

Figure 5.2: Available data and metadata for an application on App Store

Availability of these metadata items is key to researchers' ability to analyse reviews for variety of purposes such as review spam detection, sales forecasting, application localising and customising, and market demand analysis to name but a few.

In this project, the focus is merely on the content of each review as the problem identified to be addressed by the proposed approach is a proper interpretation of usefulness of reviews for application development purposes. The main objective of this study is to use the outputs and findings of the researchers studying application review mining and developers extracting requirements from user reviews to define a systematic process for accurately benchmarking and measuring the usefulness of reviews. Therefore, this project focuses on review content as the only available data item with the ability of conveying user requirements.

5.3 Conceptualising the Application Reviewing Process

The ontological conceptual model of the application reviewing lifecycle illustrated in Figure 5.3 helps to understand the significance of the usefulness factors in appropriately distinguishing between useful and useless reviews with respect to the developers' viewpoint. The provided partial application reviewing lifecycle demonstrates how a user encounters an issue while using the application and how report it properly via the reviewing option provided in application sharing websites.

In this process, a user downloads and starts using the application which includes several components and functionalities. The user takes some actions as well to use the facilities and services provided by the application. When an issue is observed by the user, an aspect of the application is involved. Reporting the system action and user action along with the involving aspect in a review helps the developers in straightforwardly diagnosing the application and addressing the issue.

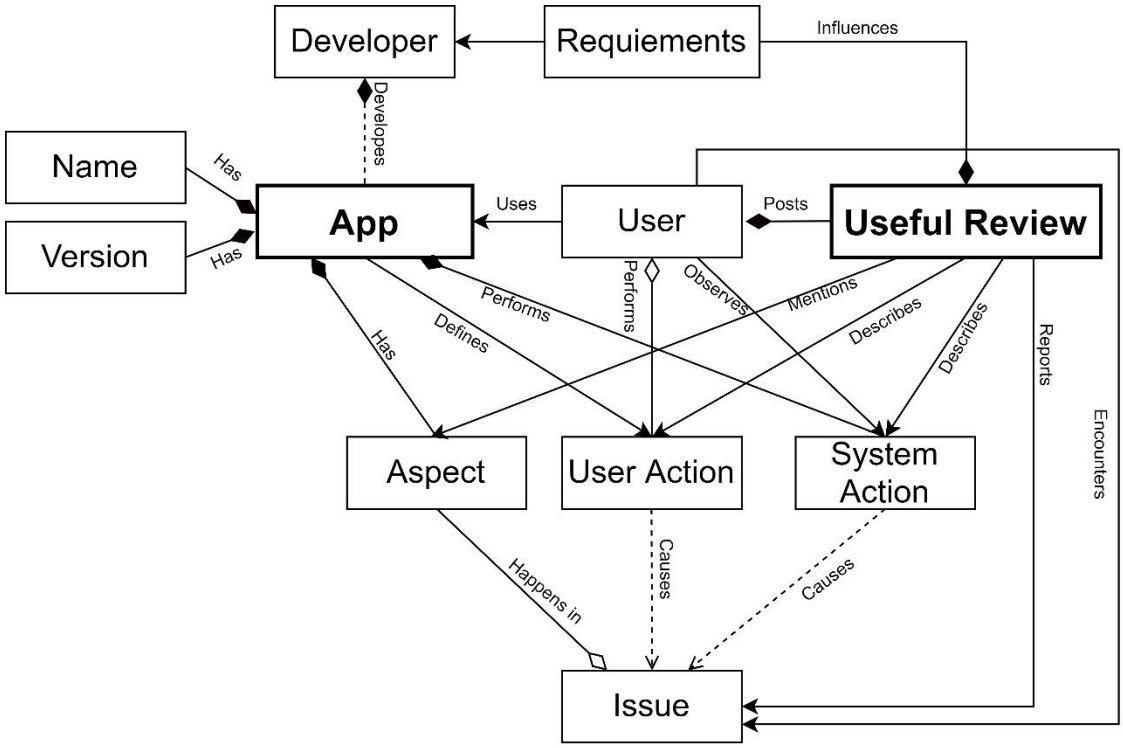

Figure 5.3: Ontological Conceptual Model of Application and Review (Partial)

Similarly, when the user of the application is requesting or suggesting a new feature, reporting the target aspect of the application is crucial to help developers understand user requirement properly. System action and user action related to the requested feature are also important. Figure 5.4 shows a review reporting an issue which is considered as useful according to the proposed approach as it explains what the issue is, which aspect of the application is involved, what is the user scenario encountering the issue, and what is the system action.

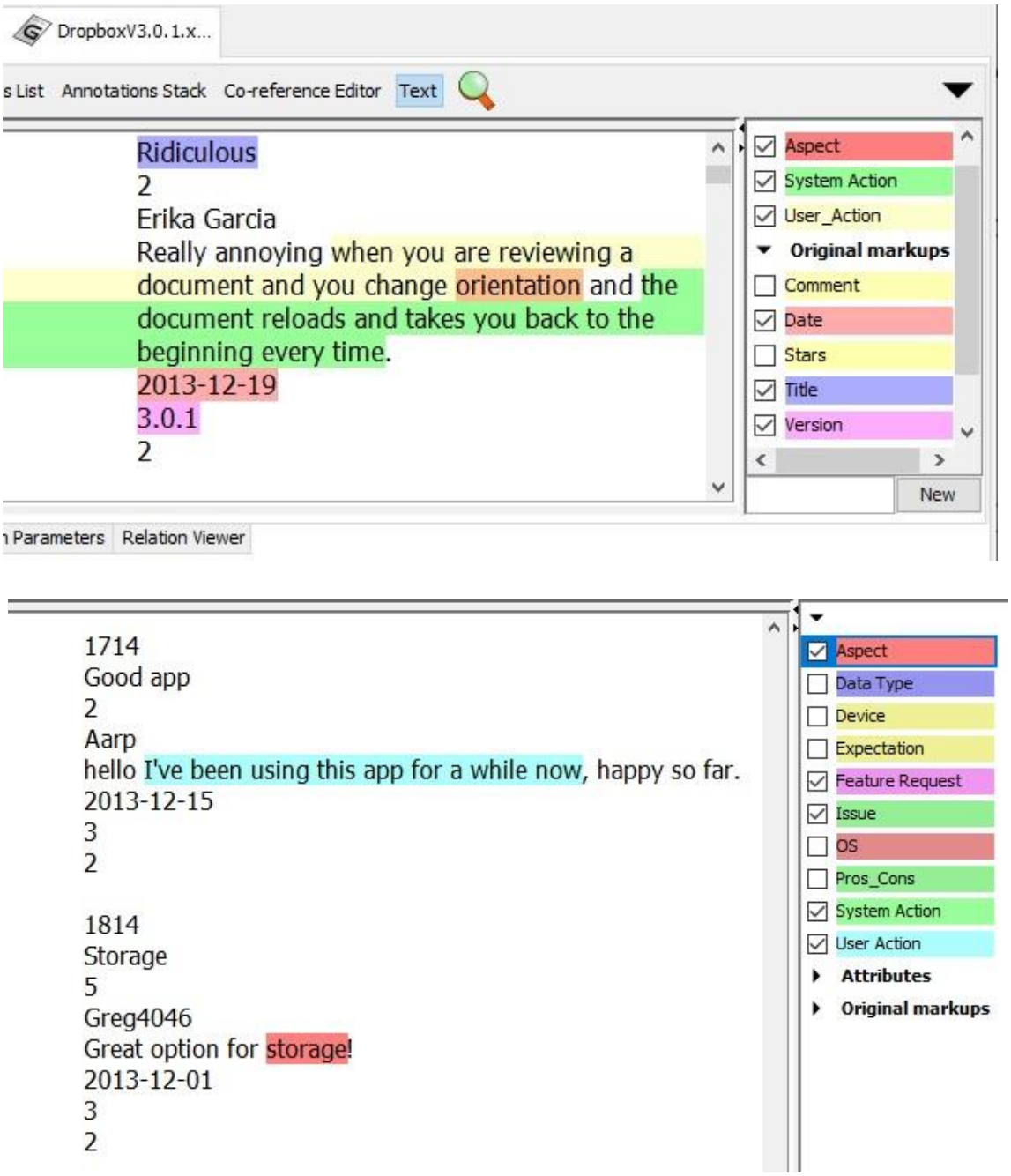

Figure 5.4: Top: A useful review containing issue, Aspect, User action, and System action. Down: two examples of useless reviews

5.4 Conclusion

This chapter discusses app review components and reviewing process to provide an overall understanding required for introducing the proposed approach in next chapter. In this chapter, different review components and metadata available on online opinion sharing platforms that can be used for research activities and the ones used in this experiment are discussed. The process of using the discovered usefulness factors for identifying useful mobile app reviews for software evolution is modelled and conceptualised. Finally, an example of an app review identified as useful applying the usefulness factors is provided.

Chapter 6 The Proposed Approach

6.1 Introduction

The proposed approach discussed in this chapter composed of an extraction technique for each usefulness factor discussed in Chapters 3 and 4. These extraction techniques are then integrated in a pipeline to form a framework taking user reviews as input and identify to what extent the review is useful for app developers to improve the application.

Section 6.2 depicts the architecture of the proposed approach. Parsing input reviews is explained in Section 6.3. Section 6.4 details step by step of the process of extracting usefulness factors. The section details techniques and approaches used for extracting each factor in five subsections. Measuring the usefulness of the target review by analysing extracted factors is described in Section 6.5. Finally, Section 6.6 summarises and concludes the contents of this chapter.

6.2 Architecture of the Proposed Approach

Assuming a user has posted a review for a target application, the process of measuring its usefulness for software development, the proposed approach, is briefly described in this section. The framework consists of three key components (i.e., Review Parsing, Extracting Factors, and Measuring Usefulness) discussed in following subsections.

The framework takes the review as input and labels it with the degree of usefulness for software evolution after analysing its content across several modules and checking the existence of the predefined usefulness factors.

In the first module, the approach applies a specialised parser to pre-process the review. The pre-processed review is then fed into the Extracting module. The function of the Extracting module is twofold. First, it transforms the text of the review into vectors of real values and learns word embeddings. Second, it applies a

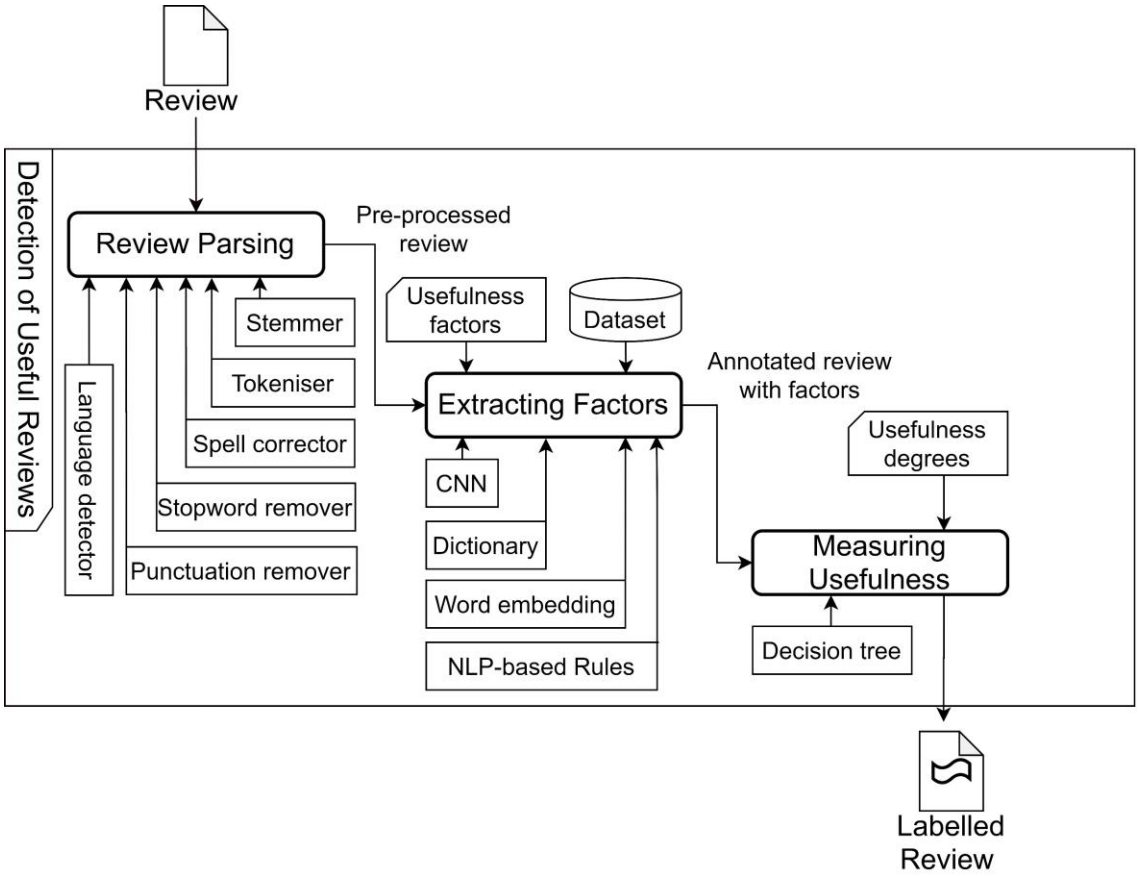

Figure 6.1: Architecture of the proposed approach

classification trained model for each usefulness factor to identify which factors are observed in the review. Finally, based on observed usefulness factors, a decision tree determines to what extent the given review is useful for software development purposes in Measuring Usefulness module.

Figure 6.1 presents a high-level view of the architecture of the Proposed Approach. It has three main modules discussed in the following sections.

6.3 Review Parser

A review in our dataset consists of several elements such as name of the application, rating, date of posting the review, etc. explained in Section 5.2. However, these elements might not be helpful with the task of measuring the usefulness of a review. Moreover, the focus of this study is to identify usefulness using the content of a review, a piece of user generated text (i.e., natural language) which needs to be prepared properly before proceeding to the next module.

When the text of a review is fed into the system, several pre-processing tasks are applied on the input review in this module to properly prepare the review content based on the requirements of the NLP techniques and classification models to be applied in Extracting module.

Language Detector discards non-English reviews as the scope of this study is to focus on English reviews only. This task is performed using langdetect library in python.

Short Review Remover removes any review with less than four words. Annotators' observations over the sample data reveal that reviews of 3 words or fewer generally introduce little information and describes personal sentiment rather than describing application related user requirements. "hate this update", "highly recommended", and "must have app" are the examples of short useless reviews in the dataset used in this study. This task is done by a simple algorithm splitting the review content into tokens (words) using The Natural Language Toolkit (NLTK) [178] word tokenizer tool and counting the output tokens.

Spell corrector checks the text for misspelled English words and corrects them. This task is performed using A Python library named 'pattern.en'.

Punctuation Remover removes all punctuations (i.e.!"#$%&'()*+,-./:;<=>?@[\]^_`|~) used in the text of the given review. This task is performed using a simple algorithm identifying and deleting each of the punctuations from the text. Applying this technique removes emojis from the reviews too.

Stop-word remover strips stop-words (e.g., a, an, the, of, at, by, for, to, etc.) from the text of the given review. This task is performed using NLTK stop-word removal technique comparing each word with the list of stop-words provided by NLTK and filtering out the stop-words.

Tokenizer splits the text of the given review into simple tokens such as numbers and words of different types. Similar to the short review removal task, NLTK word tokenizer is used to perform this task.

Lemmatizer alters each word in the text of the given review with its lemma form. Preserving the readability of a word, this technique normalises gerund endings (i.e. -ing), plural form of words and other grammatical details. This task is performed using NLTK lemmatizer tool.

Performing all these pre-processing tasks on the input review text provides a prepared piece of text to be used in the next modules. Each of the pre-processing tasks described above are selected as a part of the approach through two steps. First, the task was suggested by other researchers conducting experiments using similar techniques. Second, different combinations of the suggested pre-processing and text cleaning tasks are tested in this project and the one resulting in the highest accuracy of the approach is selected.

6.4 Extracting Usefulness Factors

To analyse a review for investigating existence of each usefulness factor, several Machine learning models are built, and NLP techniques are adopted in this study. This module explains how the pre-processed review coming from previous module is fed into a set of classifiers and NLP approaches to be labelled with the usefulness factors it holds. The five usefulness factors discussed in previous chapters (i.e., Issue, Feature Request, Aspect, System Action, and User Action) are considered in this step as target labels to be associated with the review text.

The adopted technology for extracting Issues, Feature Requests, User Actions, and System Actions is a CNN model built with input reviews represented as fixed length vectors applying word embedding techniques.

There are several reasons why, among several existing methods, CNN classifier is adopted to solve such a problem. First, as it is discussed in Chapter 2, related works investigating performance of different machine learning techniques on classifying user reviews have reported the privilege of neural networks [62, 64, 80, 179]. Second, studies comparing the performance of CNN on several natural language processing (NLP) tasks, such as POS tagging, NER, and SRL, with the state-of-the-art methods have reported the significant improvements in the results [180, 181]. Third, better generalization capabilities are available when using CNNs

integrated with pre-trained word embeddings [182]. Finally, unlike other classifiers, CNNs detect ordering of words in the input text as proper classification attributes [182, 183].

To identify reviews containing Aspects, however, several syntactic rules are defined. As applying syntactic rules captures certain predefined patterns in the data, its precision is higher than other approaches which is the key point in identifying Aspects. Deep learning models are far less accurate than syntactic patterns for this task as the input of the CNN should be single words, and the algorithm cannot extract features from the context of the target word.

The following subsections describe proposed approaches for extracting each of the usefulness factors in detail.

6.4.1 Convolutional Neural Network (CNN)

The convolution operation used in CNN involves cross-channel summation of the element-by-element multiplication. This operation consists of a convolutional filter (kernel) for each input channel which is randomly initialized and the CNN tunes and adopts its parameters to achieve the classification task. The output of the feature convolution is a feature map. The size of feature map would be same as the input matrix by adding a certain number of zeros to each dimension of the input matrix. This is because the computation of the summation in the convolution operation is over a sliding window on the input tensor. The convolution operation transforms a multi-dimensional input into a one-dimensional output matrix. Maximum pooling layer is used in this process to reduce the dimensions of a matrix. Sliding the convolutional filter over the input matrix, it captures the element with highest value and discard the rest elements fallen in the filter to generate an output matrix of reduced dimensions. Thus, convolutions in a convolutional model are passed through several pooling layers to reduce the dimensions and then will be fully connected together to finalize the classification or prediction task.

Reperesenting text as fixed length vectors applying word embedding techniques is a proper input for the CNN in this module to perform the text classification task. The word-level embeddings with fixed dimensional vectors are

fed into the convolution model. The outputs of the convolutional and pooling layers are, then, flattened. Next, applying some fully connected layers, the classification output will be generated. The example model of classifying a sentence using CNN [129, 184] is represented in Figure 6.2.

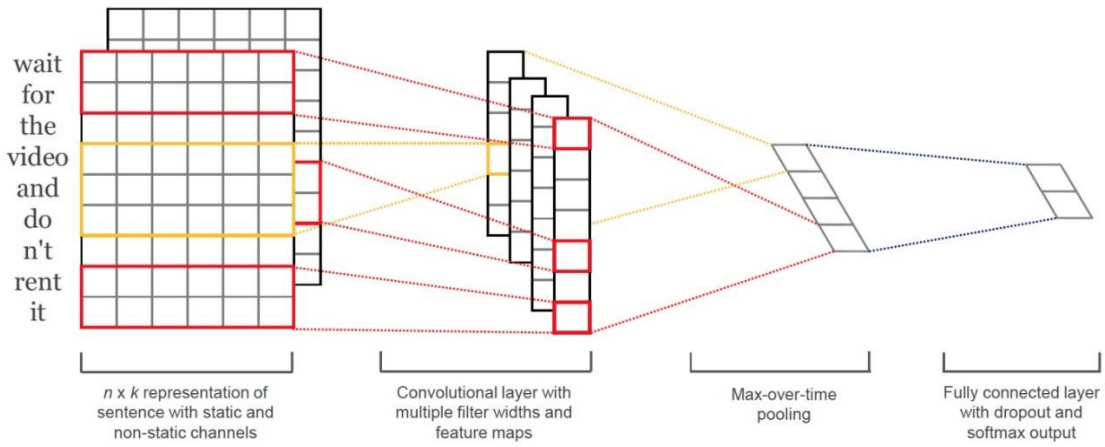

Figure 6.2: CNN model for sentence classification [184]

6.4.2 Word Embedding

The concept of word embedding could be traced back to 1956 when Zelling Harris [185] introduced the "distributional hypothesis". This theory argued that there is a relationship between the context of words and their meanings. In other words, two words that are used in a similar context have similar meanings.

The aim of this distributed word representation is to map semantic meaning of the words into their geometric spaces. It represents words as real-valued vectors in a predefined vector space where words with similar semantic meanings also have a similar representation. For example, in a well-trained word embedding set of vectors the words "orange" and "apple" are very close to each other in the space while they are far from "pencil". Similarly, the words "pen", "pencil", and "paper" might cluster in another corner.

In traditional one-hot vector representation, each word in a fixed size dictionary is represented as a binary vector with values all set to zero except the index of the word in the dictionary. For example, if a dictionary with 10,000 words

is transformed to one-hot vectors, it will have 10,000 binary vectors each has 10,000 values. In this dictionary, if the index of the word Apple is 350, the binary vector for this word would be all zero values except for the 350^{th} value which is marked with a 1. However, in word embedding, each word is associated with a point in a vector space and the technique learns from many different contexts of words how they are semantically related to each other.

Representing words as dense and low-dimensional vectors significantly improves the performance of majority of neural network tools as their accuracy using high-dimensional, sparse vector representation is reported to be low [186]. Another outstanding advantage of word embedding comparing to one-hot vector representation is that the number of features is much smaller than the size of the vocabulary [187].

Word2vec is one of the effective techniques to learn word embeddings. It transforms words in each text samples of the dataset into vectors applying a two-layer neural network. Two main algorithms used by word2vec for training are skip-gram and continuous bag of words (CBOW) [188]. The later uses context to predict target word, while the former uses a word to predict the target context (Figure 6.3).

Although it can be understood from definition of the algorithms that the performance of skip-gram is better as it can consider multiple meanings for a word such as bank (e.g., riverbank, food bank, and financial bank), accuracy of each of these training algorithms depends on the circumstances of the dataset and the machine learning task.

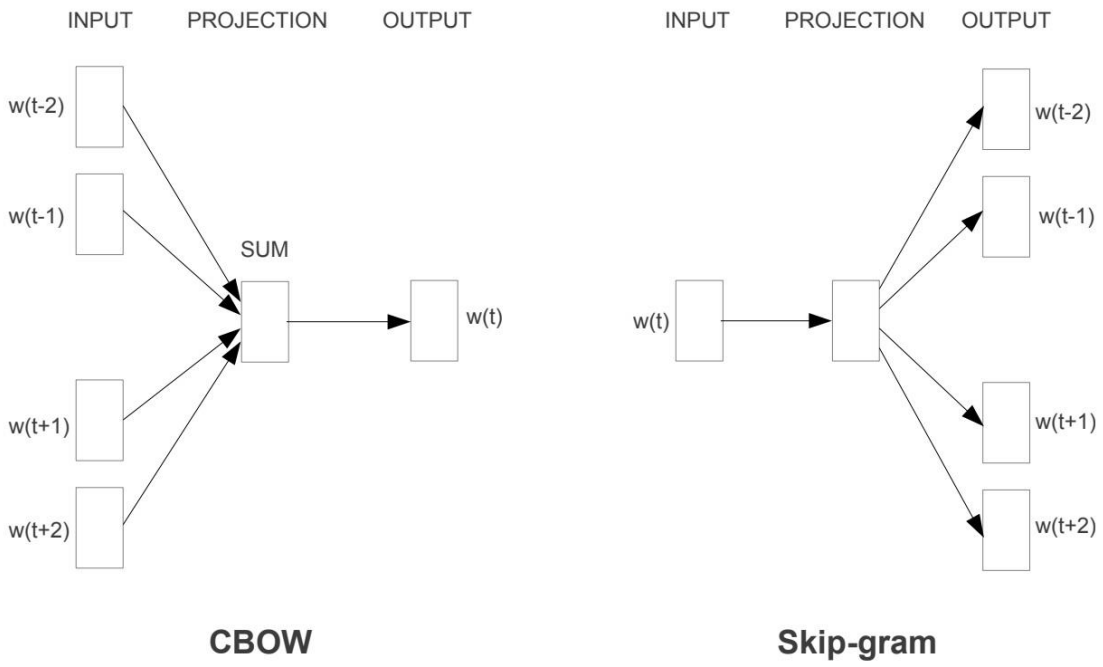

Figure 6.3: CBOW architecture (predicting word from context) and Skip-gram architecture (predicting context from word) [188]

6.4.3 Extraction of Issues

In this task, a pre-processed review text is fed into the model as input whereby it will be labelled as issue if the text conveys any issue reported. The way for distinguishing between reviews reporting an issue with other types of reviews in this task is adopting a Convolutional Neural Network (CNN) classifier with input words represented as word embeddings.

As both datasets include reviews manually tagged with Issue Report labels, a combination of dataset one and dataset two are used for training and testing the model built for this task. Each of the datasets were split into 80% training and 20% testing parts.

6.4.3.1 Vocabulary Building

Creating a vocabulary of known words is of great importance when word embedding model is to be used. Using all existing words in the dataset, very large

representation for documents will be generated. Thus, defining vocabulary of known words is an effective way to optimise the performance of the system.

To create the vocabulary all reviews from training set are cleaned, tokenised into words, and occurrences of each one in the dataset was computed. Then, words with frequency less than two were filtered out to avoid negligible words. The final vocabulary was saved into a text file to be used further for filtering input data prior to encoding them for modelling. This task was performed designing a simple algorithm in Python.

6.4.3.2 Training Embedding Layer vs. Using Pre-trained Embeddings

There are two techniques for creating embeddings to be used with the model. First, training embeddings using existing train data. Second, using off the shelf embeddings trained on huge volumes of human generated text. In this study, both techniques are applied to compare their impact on the accuracy of the proposed model.

In the first model configuration, the word embedding is designed to be trained while the classifier is training. To this end, an embedding layer is added to the model using the Keras deep learning library in Python. The embeddings can also be trained in advance and saved into a file to be loaded and added to the model later. The word2vec algorithm is used in such cases to train embeddings in a standalone manner. Prior to adding the embedding layer, the created vocabulary is loaded to filter out negligible words from reviews.

To meet the requirements of the Keras embedding layer, each review is encoded as a sequence of integer values indicating a word each. The preliminary values represented as vectors are random but becoming meaningful while training the algorithm. Training the model using all training reviews from the truth set maps all words from the vocabulary to unique integer values which are used for encoding input reviews.

Alternatively, pre-trained embeddings on huge volume of text data are available to be used with neural network models. It is reported in the literature that using pre-trained embeddings for text processing tasks offers better performance

[179, 189]. Google has created a 300-dimension word2vec embeddings trained on Google News using more than 100 billion words which is also used with the model built in this task.

6.4.3.3 CNN Model Building

Continuing configuration of the model, Keras requires input documents of a same length for optimising the efficiency of the model. Therefore, a maximum length size for reviews was defined and used for zero-padding training reviews to the maximum length size.

Class labels are required to be defined in the Keras to fit the supervised neural network model to classifying reviews. As the class labels need to be defined as integer values, reviews containing Issue Report are labelled by 1 and are otherwise tagged by 0. The test dataset is also encoded and padded for validating the model.

An embedding layer is the first element added to the model. For training embedding while training the model, the embedding layer was set to vocabulary size, 300-dimensional word representation, and maximum length of the reviews. For the Google News pre-trained embedding, the embedding layer was set to the setting of the pre-trained embedding.

Next layer, one-dimensional convolutional layer, was preliminarily configured with a rectified linear (relu) activation function which is easy to adopt and has greatly improved the performance of feedforward networks comparing to sigmoid hidden units [190]. The kernel size was set to 5 and 200 filters were used for processing words.

To consolidate the output of the convolutional layer, a max pooling layer with pool size set to 3 was then added to the model. To reduce the dimensions of the output of the max pooling layer, a flatten layer is also added to the model.

Finally, a sigmoid activation function is used in the output layer to generate 1 for reviews containing Issue Reports and 0 otherwise. A summary of the defined neural network model is illustrated in Figure 6.4. The summary shows that the input of the embedding layer is documents of 233 words length. Each word is encoded into a vector with 300 elements.

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 233, 300)	789900
conv1d_1 (Conv1D)	(None, 229, 200)	300200
max_pooling1d_1 (MaxPooling1	(None, 76, 200)	0
flatten_1 (Flatten)	(None, 15200)	0
dense_1 (Dense)	(None, 10)	152010
dense_2 (Dense)	(None, 1)	11

Figure 6.4: Summary of the model configuration

To compile the model, which is a binary classifier, a binary cross entropy function was selected along with adam optimiser. Training reviews were fitted then on the model and number of epochs was set to 15 indicating that the model iterates 15 times over the training reviews. The optimal configuration defined is based on successful empirical results on binary text classification problems [191, 192] and several trial and errors.

After configuring and building the model, test reviews are fed into the model for evaluation purposes. Accuracy measures used in this project are Precision, Recall and F-measure. More detail on evaluation of the model is provided in Chapter 7.

6.4.4 Extraction of Feature Requests, User Actions, and System Actions

To classify reviews into reviews containing Feature Request and reviews with no Feature Request, similar to Issue Reports, a CNN model using word embeddings is designed. Same configuration and setup were applied. The only difference was in input data that the reviews from dataset one and dataset two containing Feature Requests and Other reviews were fed into the classifier for training and testing purposes.

Pertaining to User Action and System Action, to the best of our knowledge, this is the first time these two factors are to be identified automatically. Thus, one of the contributions of this project is proposing an approach for automatically identifying them. Due to the novelty of the approach, no ground truth dataset was found for this task resulting in using only dataset one that is specifically annotated for this project to train and test two binary classifiers designed for this task.

Again, a binary CNN model with word embedding was designed for each of the usefulness factors (i.e., User Action and System Action). Unlike classifiers identifying Issue Reports and Feature Requests taking a whole review as a data sample, reviews from dataset one were split into sentences and the model was designed to perform the classification in sentence level which is due to the nature of these factors. As a short text, a review is designed to have a topic and few sentences expounding the topic. Issues and Feature Requests are also generic topics requiring to be explained in several sentences. However, User Action and System Action are small parts of an Issue or Feature Request with no more than one sentence to express.

6.4.5 Extraction of Aspects

Conducting a CNN model with word embedding is not efficient for identifying Aspects. The effectiveness of this approach is due to inclusion of several words in the input text. The neural network extracts several features from semantic relationships between words in a sentence enabling the classifier to perform accurately. However, the Aspects usually appear in a single word and using word embeddings is less efficient for such situations. On the other hand, the existence of an Aspect in a review is critical for identifying useful reviews for its key position in

the decision tree discussed in the following section. Therefore, an accurate approach for identifying Aspects is required to be designed.

A group of proposed approaches detecting Aspects have focused on opinionated sentences. These approaches use an opinion word in a sentence and analyse its relationship with other words to find Aspect candidates. However, as it is discussed in Chapter 2, many aspects are mentioned in reviews without any opinion expressed. Moreover, many of the recent studies have used topic modelling for detection of Aspects which is not an efficient approach for short texts [58, 87][41]. Other draw backs of topic modelling-based approaches are discussed in Chapter 2. As the performance of syntactic rule-based approaches is reported significantly optimal in capturing certain patterns from text [193-196], a set of syntactic rules are defined in this project to identify Aspects.

NLP-based rules are usually defined over constituency and dependency parsing, and POS tagging results. The extraction rules defined in this project leverage dependency parsing and POS tagging to identify mentions of Aspect in a review. It is noteworthy mentioning that the aim of this task is to identify reviews mentioning an Aspect not specifically detecting which word or word phrase is the Aspect.

6.4.5.1 Part of Speech (POS) Tagging

Part of Speech tagging refers to the task of reading a piece of text and labelling the part of speech for each word. An example of POS tags is shown in Figure 6.5 where a sample sentence is tagged with abbreviations of part of speeches. Each tag has a certain meaning. PRP stands for preposition, VB is a verb, NN is noun, and so on. The diagram is drawn using Stanford CoreNLP online text processing tool. A list of POS tags is provided in Appendix IIII.

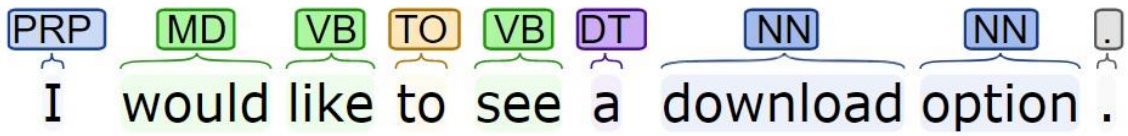

Figure 6.5: An example of POS tags

StanfordNLP, a toolkit for natural language processing tasks used in this project, uses a Java implementation of the log-linear part-of-speech taggers proposed in [197, 198]. Stanza [199], a Python NLP package offered by StanfordNLP is used to perform the POS tagging task. An example of the output of Stanza POS tagger is provided in Figure 6.6.

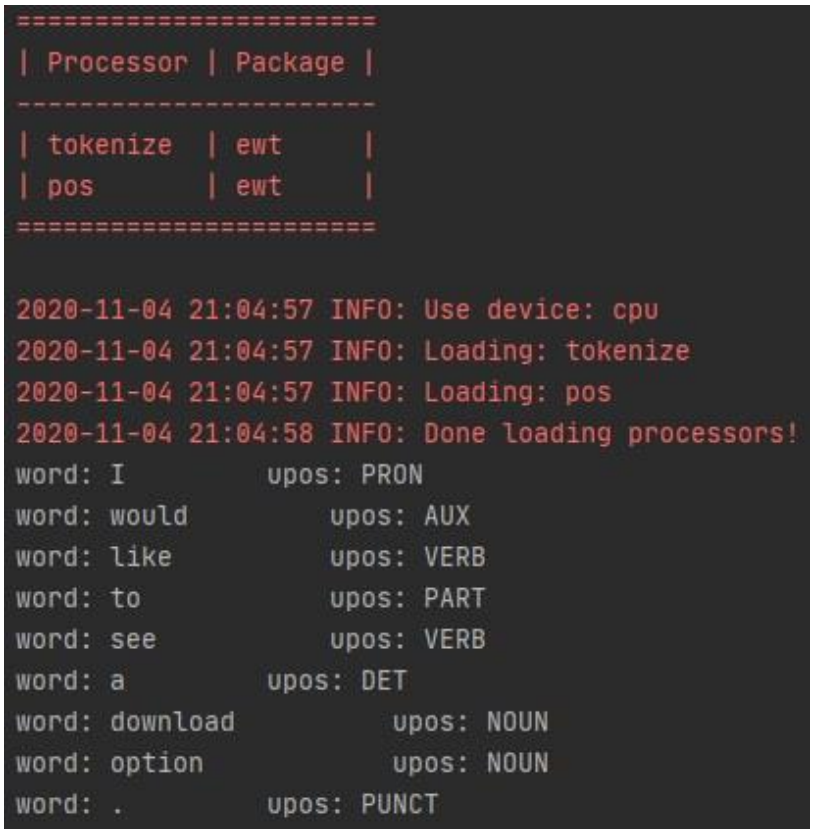

Figure 6.6: An example of Stanza POS tagger output

6.4.5.2 Dependency Parsing

Dependency parsing is the task of clarifying syntactic structure of a sentence using words and grammatical relationships among them. A dependency tree consists of a root node which is the head of other nodes (words) and labelled directed arcs from head nodes to dependent ones. In a dependency relation $n \rightarrow m$, the head of the relation is n and the dependent is m. The labels show syntactic dependencies between nodes, such as direct object, adjective modifier, and clausal subject, to name a few. A complete list of universal dependencies [200] is provided in [201]. Figure 6.7 shows a dependency tree drawn for a sample sentence, "I would like to see a

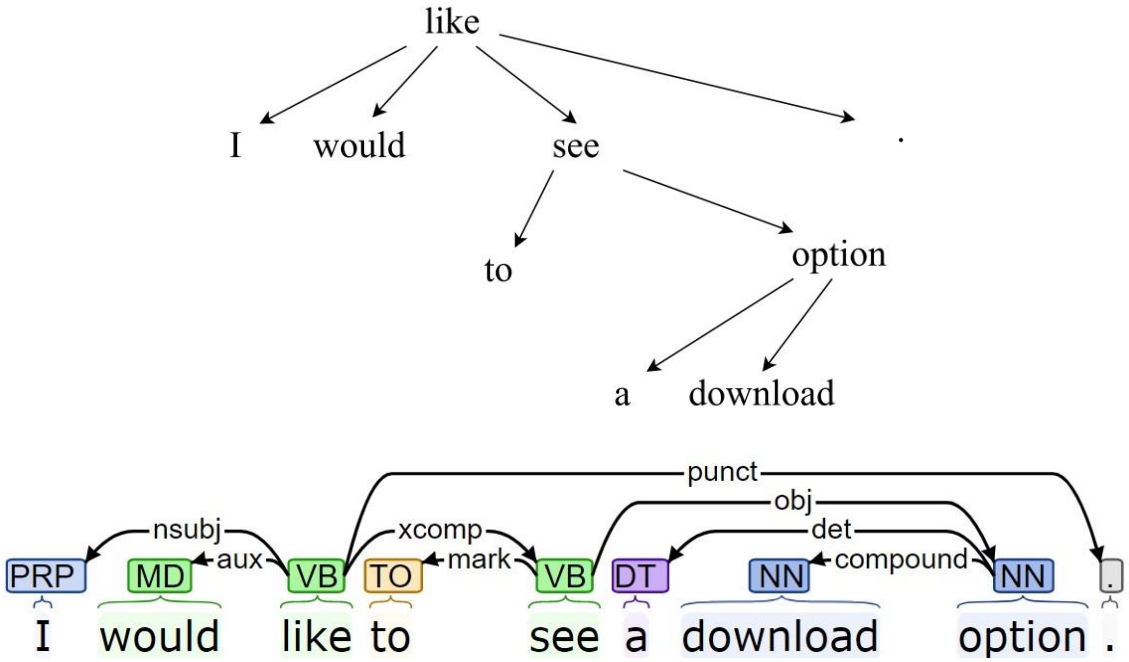

Figure 6.7: An example of dependency tree and its relationships

download option.", and its dependency relationships. The diagram is drawn using Stanford CoreNLP online text processing tool.

Recent improvements in NLP has approved the significant enhancement of novel NLP techniques such as dependency parsing [202] which is advanced enough now to be used for defining extraction rules. For this task, Stanford parser is used from Stanza package in Python as it had the best performance comparing to other tools [203, 204].

6.4.5.3 Rules Extracting

The rules proposed in related approaches [94, 98, 99] have been investigated in terms of capability in identifying application aspects. The investigation revealed that this thesis cannot merely rely on them for identifying aspects for the following reasons:

Firstly, empirically evaluating several feature extraction approaches on real world user reviews has revealed that these approaches achieve lower effectiveness than reported originally [36].

Secondly, detection of aspect in related work is contingent upon identification of opinion words. This is because most of these papers aimed at understanding users'/customers' opinion about an aspect. However, many aspects are mentioned

in reviews without any opinion word associated with. For example, in the following review, ‘message’ is an aspect with no opinion word in the sentence.

“Every time that I open the app, a message pops up asking me to review the app.”

Thirdly, there is negligence in definition of rules. Preliminary analysis of the rules in this project revealed several contradictions among them. Some of the examples are explained in the following sentences. Rule 1 and rule 2.1 in [98] are same, but all sub-rules of rule 2 are applicable only if there is no auxiliary verb. Rule1 needs the modifier of the verb to be seen in an opinion lexicon, but rule 2.1 does not. Rule 2.2 and 2.3 in [94] are same. However, rule 2.2 needs the noun to be out of opinion lexicon, but rule 2.3 needs the noun to be in the lexicon. Rule 2.3 in the same study comes with an auxiliary verb, while all the sub-rules of rule 2 are applicable when there is no auxiliary verb in the sentence. Rule 3 in [98] considers a noun which is in copula relation with an auxiliary verb as an aspect, but the noun is not in relation with the verb in provided example (i.e. the camera is nice). These are just few examples to show how unreliable are existing rules.

Finally, some types of sentences including important aspects are not detected by the existing rules. For example, the following review contains an aspect which is “edit section”, but the rules defined by Rana et al.,[99, 100] cannot capture this aspect.

“It would add great value to have an edit section”

Accordingly, useful rules are taken from related works, improved, and adopted for the aim of this project. More rules are also defined based on observations and analysis performed in this project to identify Aspects effectively.

Investigating the appearance of aspects in product reviews, researchers have argued that 98% of the aspects are noun phrases [93, 100, 205]. Results of several studies show that aspects appear only in the form of nouns in reviews [206, 207]. Besides, manual analysis of reviews in a study conducted by Malik et al. [93] has revealed that nouns, verbs, and adjectives are key part of speeches in identifying application aspects. Therefore, to improve the precision, the focus of this thesis is

on noun phrases for identification of aspects. The following rules are defined and applied in this project for identifying reviews containing an Aspect.

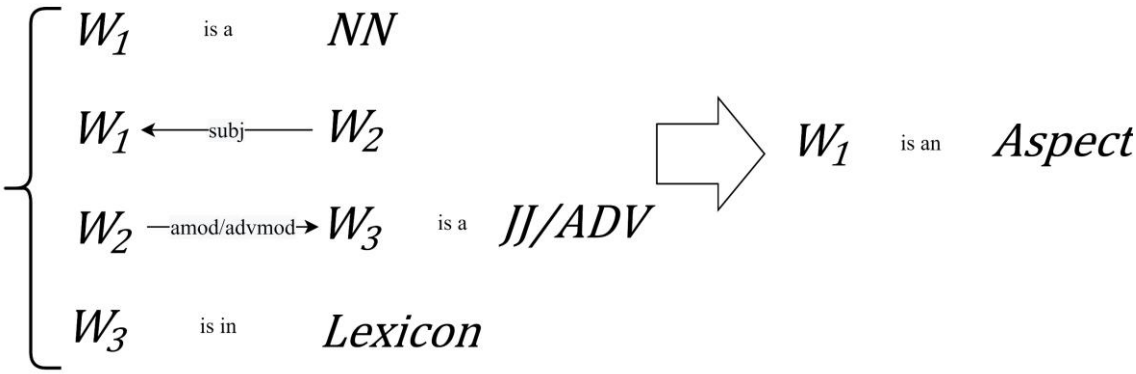

Figure 6.8: Logical representation for Rule1

R1. If a word W_1 has a noun POS tag and is dependent of the node W_2 with a subject relationship, and W_2 has an adjective or adverb modifier which is in the opinion lexicon, then W_1 is an Aspect. The opinion lexicon[2] is a collection of English negative and positive opinion words and sentiments gathered by Hu and Liu, [208] and then compiled over many years by the authors. Figure 6.8 illustrates this rule in a logical formula where W is a word, and NN, JJ and ADV stand for noun, adjective, and adverb, respectively.

The noun 'background' in the following review, for example, is subject for the verb 'drains' which is modified by the adverb 'quickly'. 'quickly' exists in the opinion lexicon. So, 'background' is captured as an Aspect. Figure 6.9 illustrates the dependency tree for the review used in this example.

"the white background of the app drains the battery quickly."

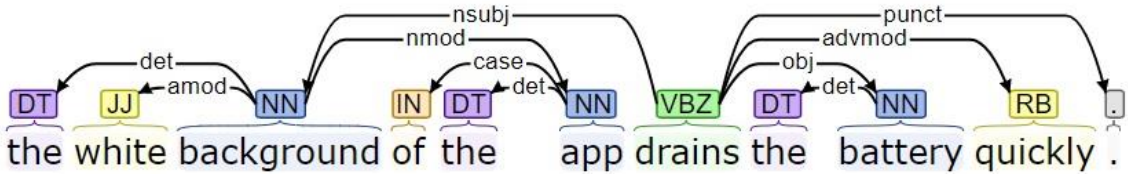

Figure 6.9: Dependency tree for Rule1_example

[2] https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

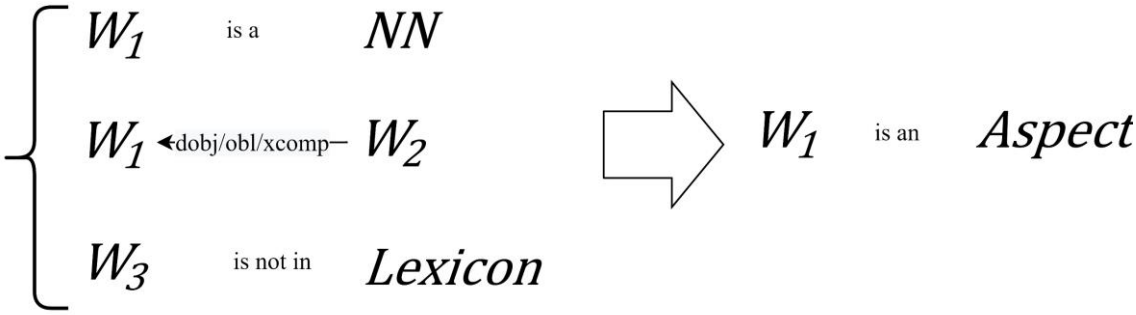

Figure 6.10: Logical representation for Rule2

R2. If a word W_1 has a noun part of speech and is dependent of the node W_2 with a direct object, oblique nominal, or open clausal complement (xcomp) relationship, and W_1 is not in the opinion lexicon, then W_1 is an Aspect. The logical formula for this rule is shown in Figure 6.10 where *dobj*, *obl* and *xcomp* stand for direct object, oblique object, and open clausal complement relationships, respectively.

The noun 'option' in the following review, for example, is in open clausal complement relationship with the verb 'needs' and is not in the opinion lexicon. So, 'option' is an Aspect. Figure 6.11 illustrates the dependency tree for the review used in this example.

"There needs to be a bulk download option."

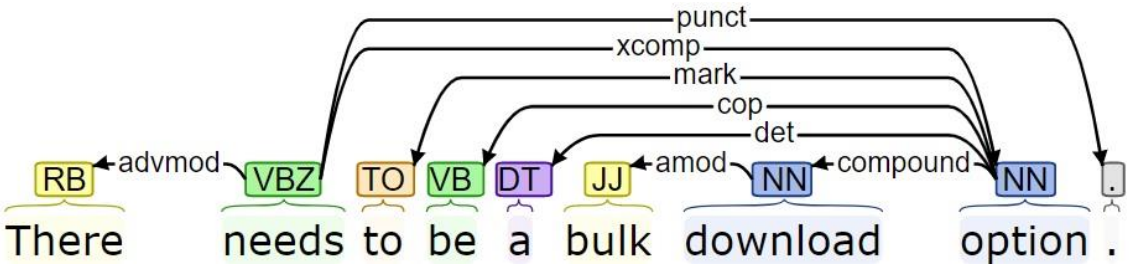

Figure 6.11: Dependency tree for Rule2_example

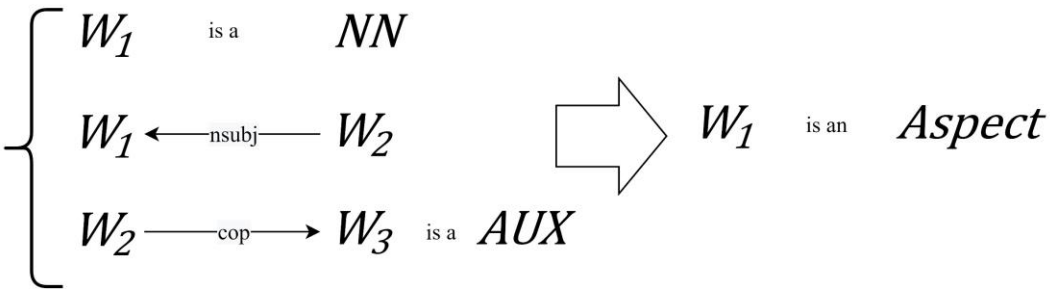

Figure 6.12: Logical representation for Rule3

R3. If a word W_1 has a noun POS tag and is dependent of the node W_2 with a noun subject relationship, and W_2 is in copular relationship with an auxiliary verb W_3, then W_1 is an Aspect. Figure 6.12 illustrates this rule in a logical formula where AUX is an auxiliary verb.

The noun 'option' in the following review, for example, is subject for the word 'nice' which is in copula relationship with the auxiliary verb 'is'. So, 'option' is an Aspect. Figure 6.13 illustrates the dependency tree for the review used in this example.

"The new search option is nice."

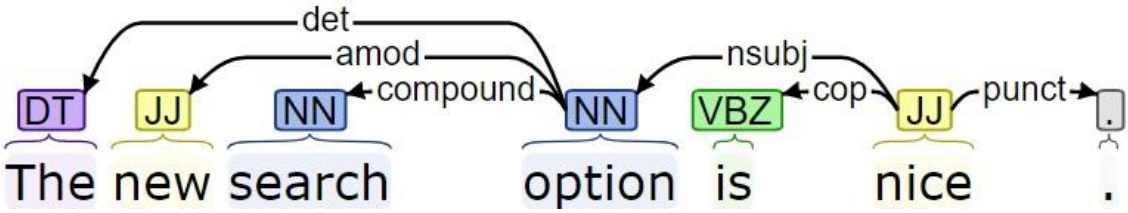

Figure 6.13: Dependency tree for Rule3_example

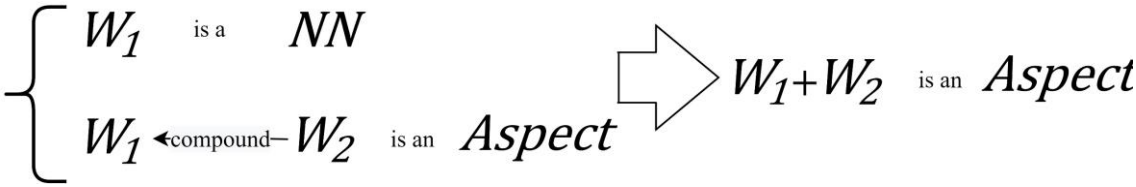

Figure 6.14: Logical representation for Rule4

R4. If a word W_1 has a noun part of speech and is dependent of the node W_2, which is detected as an Aspect by other rules, with a compound relationship, then W_2 is concatenated to W_1 as a part of the Aspect. For example, the word 'search' in

Figure 6.13 is in compound relationship with ‘option’ which has been detected as an Aspect by R3. So, ‘search’ will be concatenated to ‘option’ to form the new Aspect, ‘search option’. Figure 6.14 illustrates this rule in a logical formula.

6.5 Measuring the Usefulness

applying a decision tree approach in this module, reviews labelled with usefulness factors are assessed for measuring their degree of usefulness for software development. This study does not investigate the usefulness as a binary concept. In the proposed extraction model represented in Figure 6.15, four classes of usefulness (i.e., Fully useful, Insufficiently useful, Less useful, Useless) are designed to classify reviews toward usefulness. Assessing the given review using the proposed model, we can decide to what extent the review is useful. A Fully useful review is what developers are looking for and count as important, while Useless reviews are to be discarded. Insufficiently useful and Less useful reviews are partially important for developers.

This strategy facilitates developers and companies to focus on a certain degree of usefulness in analysing their reviews. For example, a large company with huge volume of reviews may only wish to filter Fully useful reviews, while in small scales, a developer managing a few apps and only receiving tens of reviews daily may prefer to get rid of Useless reviews and include Less useful reviews in his/her reading list as well.

Moreover, the rationale behind this classification is that with a binary classification (i.e., Useful and Useless), there is a probability of missing reviews classified as Useless, but very close to the border of the Useful class. These reviews usually contain partially useful information which is ignored in a binary

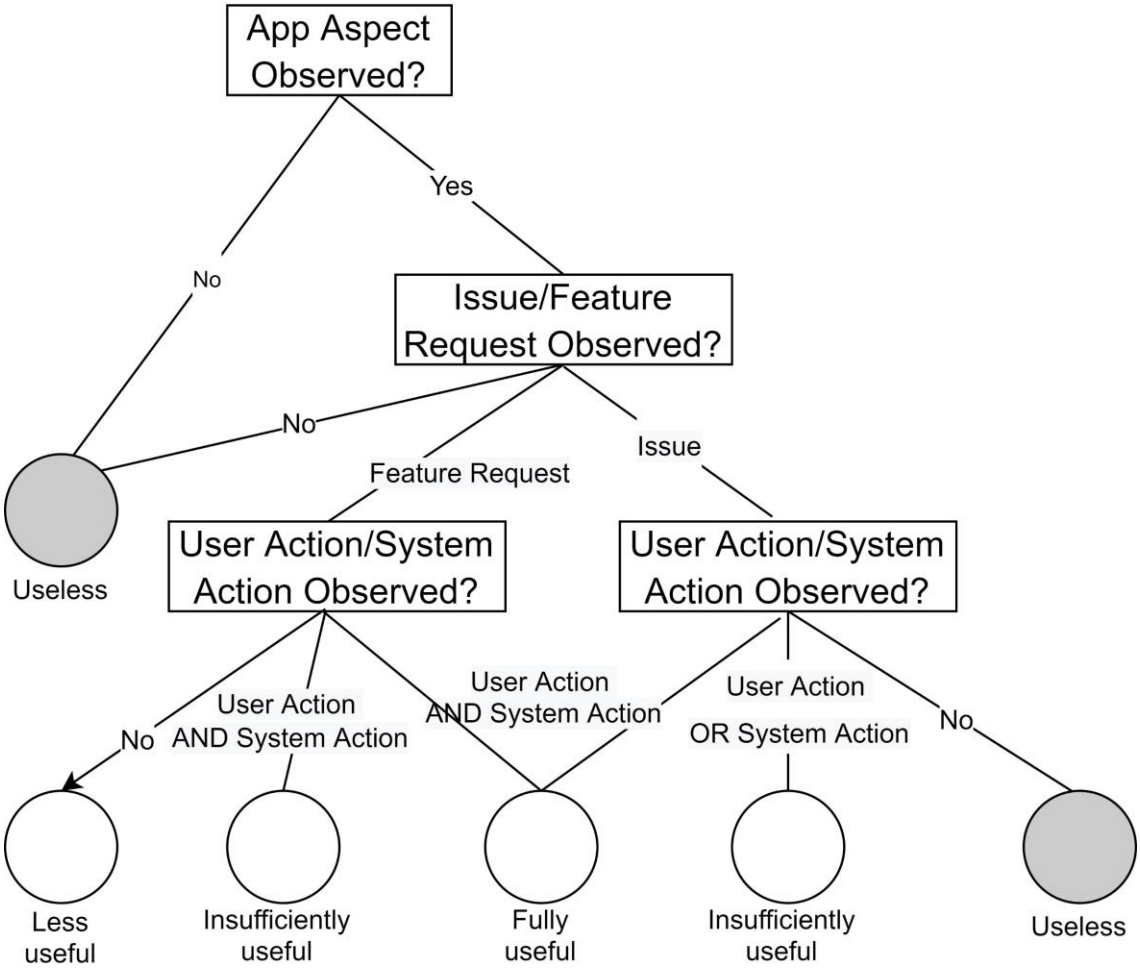

Figure 6.15: The customized Decision Tree for measuring the usefulness

classification. Thus, defining intermediate classes helps to purifying the samples fallen in each target class by defining the degree of membership.

The decision tree approach (Figure 6.15) is designed at this step to perform the classification task in an organized manner. The very first branching point in the tree is the Aspect, "Is there any application Aspect mentioned in the review?", because in terms of the purity, it gains the lowest possible Entropy which equals to 0. Entropy is a technique to measure the uncertainty of a class in a subset of examples. It gives a number between 0 (i.e., 100% certain) and 1 (i.e., 100% uncertain). It measures the purity of split in a decision tree as shown in the following equation:

$$H(s) = P_{(+)}\, Log_2(P_+) - \; P_{(-)}\, Log_2\big(P_{(-)}\big) \qquad 6.1$$

Where, $H(s)$ is the Entropy of the s which is a subset of the annotated reviews. $P_{(+)}$ is percentage of the true class and $P_{(-)}$ is percentage of the false class in s. In this research, true class for each factor is number of useful reviews in the sample including the target factor, while false class refers to number of useful reviews missing the target factor.

The review will be classified as Useless if not referring to any aspect of the target application ('No' answer at the root node). Otherwise, the type of the review is assessed at the second level decision node where the outcomes are Issue, Feature request, or Other. Falling in the Other category, the review will be classified as Useless, while Issues and Feature requests are to be assessed for inclusion of User action and System action. Observing both factors in the review makes it Fully useful, but one of them causes the review to be classified as Insufficiently useful. For feature requests, if the review is free from User Action and System Action, it will be fallen in Less useful class. However, such a review will be classified as Useless in Issue report type.

The decision tree implemented in this study, is a modified and customised version which is optimised based on requirements of this project. Therefore, calculation of the Entropy combined with the insights gained during the data annotation and previous tasks were used to construct the tree based on the selected factors.

6.6 Conclusion

This chapter presented an overview of the proposed approach in this project to identify the usefulness of a given user review for software development purposes. The approach has three main modules, which are Review Parsing, Factors Extracting, and Usefulness Measuring. The modules are explained in this chapter and their technical details are discussed in depth in the subsequent chapter.

Chapter 7 Results and Discussion

7.1 Introduction

In this chapter, results of the experiments conducted in this study are reported, and performance of the proposed approach is evaluated. Another purpose of this chapter is to interpret and discuss the results.

Section 7.2 describes the datasets used in this study for not only training and testing the supervised approaches for extracting usefulness factors, but also validating syntactic rule-based approaches. Performance results of the proposed approach are discussed in section 7.3 wherein two levels of evaluation are introduced. The section reports on the accuracy of each usefulness factor extraction technique as a standalone model, and the integrated system composed of these techniques. Finally, section 7.4 compares the results with baseline studies.

7.2 Data Collection

The first dataset used in this project, named dataset one, is a collection of mobile application reviews posted by real users which is crawled by Guzman and Maalej [58] in February 2014. The dataset includes thousands of reviews on 8 well known mobile applications from App Store and Google Play. A subset of 1000 reviews on 4 applications was selected as the corpus of this study for the following reasons. Firstly, the reviews were collected from various sources. So, some original meta data coming with the context of the review were missing for some reviews. Second, initial observations revealed that some applications might have been target of massive spam attacks as there were lots of reviews not delivering a sensible meaning (e.g., “very good”, “love it”, “Excellent!”, etc.). Third, in some versions of some applications, a critical issue was observed reflected in a huge number of reviews causing bias in the data. Finally, circumstances and scope of this project demand such a subset.

Each review in the dataset consists of 7 original meta-data items coming with the context of the review (i.e., ID, App ID, App Version, Title, Reviewer, Star rate, and Date), though the circumstances and objectives of this project demands to use review content only. The data is publicly available by the researchers in the form of SQL commands to create and populate relevant tables. Thus, SQL tables were populated with data samples to obtain the complete dataset used in [58]. Running simple SQL queries, the final subsample of the data was obtained which is called dataset one in this thesis. Transforming the data into .xml files, they were fed into the General Architecture for Text Engineering (GATE) [209] for further annotation and analysis.

Considering that the main approach for extracting the usefulness factors is deep learning, a set of manually annotated data samples are required. So, for training and testing the proposed classifiers, a peer manual analysis of the 1,000 sampled reviews was conducted to create the truth set. The data annotation task was to manually identify and label occurrences of each usefulness factor (i.e., Aspects, Feature Requests, Issues, System Actions, and User Actions). The author of this thesis and a PhD candidate with more than six years of mobile application development experience have performed this task. A coding guideline was provided to gain a higher Inter Annotator Agreement (IAA) [210] and maximise the quality of the output [211]. The guideline included explanations and clarifications on what each usefulness factor is, how to deal with uncertainties, and review examples for each factor.

To handle disagreement between annotators they were asked to separately label 100 reviews in a preliminary annotation task. Then, in a meeting comparing and justifying judgements, they gleaned insights from discussions and prepared to start the main data annotation task. The coding guideline was updated accordingly and provided to the annotators. After coding all the reviews in the dataset, in a meeting discussing disagreements with coders, the unresolved ones were discarded from dataset one and new reviews where replaced. Therefore, each of the reviews in dataset one is jointly labelled by the two coders.

The second dataset used in this study, called dataset two, is collected by Maalej and Nabil, [60] in 2014 which has reviews classified into four categories including bug report, feature request, user experience, and rating. The crawled dataset contains 1,246,057 reviews collected from App Store and Google Play Store. The authors randomly sampled 4,400 reviews for manual analysis. They hired computer science master students to perform the annotation. Each review was classified by two coders into the categories. As identification of user experience and rating classes are out of the scope of this project, the reviews fallen in these two classes are discarded from dataset two. The final dataset contains 3,691 reviews classified into feature request, bug report, and other.

The reason for using this dataset as the second dataset for this study is to enrich the number of data samples for Issues and Feature Requests resulting in the improvement of the proposed classifier for extracting these factors.

The original downloaded dataset was provided in JavaScript Object Notation (JSON) format which was converted into Comma-Separated Values (CSV) to be combined with dataset one for training and testing of feature request and issue report classifiers.

7.3 Performance Results

Evaluation of the proposed approach is performed in two levels, component level and system level. In component level, the process of extracting each usefulness factor is validated as a standalone model, and the accuracy of that certain task is measured. Putting these distinct models in a pipeline, an integrated system getting reviews as input and labelling them with the pre-defined degree of usefulness is achieved. System level evaluation measures the accuracy of the system. Before discussing each of these validation levels in this section, the metrics used for measuring the accuracy of the techniques are introduced.

7.3.1 Evaluation Metrics

The evaluation metrics used in this study are the precision, recall and F-Measure. The procedure for calculating the precision and recall is to run each of the proposed

models to classify reviews to positive (i.e., when the review includes the target usefulness factor) and negative (i.e., when the review does not include the target usefulness factor). The outcome is then compared to the ground-truth dataset built for this study implying the evaluation metrics subsequently described.

Calculating the recall, number of reviews captured as positive by the proposed model which also are judged as positive by human experts is divided by the number of all reviews judged as positive by human experts.

$$Recall = \frac{|\{Relevant\ items\ from\ truth\ set\} \cap \{Items\ detected\ by\ the\ algorithm\}|}{|\{Relevant\ items\ from\ truth\ set\}|} \quad 7.1$$

Likewise, Precision is computed dividing the number of reviews identified as positive by the algorithm which also are judged as positive by human experts by the number of all detected reviews by the algorithm.

$$\text{Precision} = \frac{|\{Relevant\ items\ from\ truth\ set\} \cap \{Items\ detected\ by\ the\ algorithm\}|}{|\{Items\ detected\ by\ the\ algorithm\}|} \quad 7.2$$

Finally, F-Measure is used to combine the Precision and Recall. The F-Score could be calculated using the subsequent formula.

$$F\text{-}Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad 7.3$$

7.3.2 Component Level Evaluation

Five usefulness factors are defined in this study to measure the usefulness of a given user review (Chapters 3 and 4). To facilitate the task of extracting each of them from review text, an NLP based approach is proposed. In this sub-section, proposed approach for automatically identifying each usefulness factor is validated and the corresponding results are reported.

7.3.2.1 Extraction of Issues

The proposed approach for identifying reviews containing an Issue (Chapter 6) is validated using a ground truth dataset (Section 7.2). Evaluating the automatic approach for extracting this usefulness factor, both dataset one and dataset two were used to train and

test the model. Precision, Recall, and F-Measure were also expected from the classifier validation setting.

The domain specific embeddings trained in this experiment and the pre-trained embeddings from Google were used to represent the input words to the neural network (Section 6.4.3.2) to compare the results obtained using each of the embeddings with the neural network model. After successfully building the models, the following results were obtained (Table 7.1).

Table 7.1: Accuracy of the neural network used for identifying Issues in reviews

	Precision	Recall	F-Measure
CNN + Training embeddings	0.48	0.66	0.55
CNN + Google Pre-Trained embeddings	0.60	0.76	0.67

As the results show, performance of the model is significantly better when the input words are represented as embeddings using the pre-trained Google embeddings comparing to the embeddings trained in this study.

Word embedding learns semantic relations between words in the given training sentences to build the word vectors causing semantically relevant words to be clustered in a same group or very close to each other on the vector space. Decision of the utilised CNN model is contingent upon these features.

Thus, at the first glance, embeddings trained using application reviews are expected to have more positive impact on the performance of the classifier rather than Google embeddings trained with news data. This is a reasonable expectation as the wording and writing style of all the data samples are for reviewing apps resulting in more relevant relationships between words. However, the key for Google embeddings success is for the huge volume of the input training data. Billions

of sentences written by human used to train the embedding covering variety of individuals' phrasing patterns and discovering too many relationships between used words. In this experiment, few thousands of reviews are used to train the domain specific embeddings.

7.3.2.2 Extraction of Feature Requests

Similar to Issue reports detection, dataset one and dataset two were jointly used for training and testing the CNN model designed for identifying reviews requesting a new feature for the application. This is because both datasets include reviews manually tagged with the 'Feature Request' label. The model was configured to compute Precision, Recall, and F-Measure as validation metrics. The two different embeddings were also used to represent the input words to the neural network and to compare the accuracy. Training and testing the models using each of the embeddings, the following results obtained (Table 7.2).

Table 7.2: Accuracy of the neural network used for identifying Feature Requests in reviews

	Precision	Recall	F-Measure
CNN + Training embeddings	0.55	0.72	0.62
CNN + Google Pre-Trained embeddings	0.67	0.78	0.72

Again, the results show that the accuracy of the model when is fed with Google pre-trained embeddings is significantly better. The reason for such behaviour is explained in previous section.

In general, comparing the results with the ones obtained from evaluating the Issue detection model, one can see a more accurate model which is due to the nature of the factors. Issues have a large application domain so that variety of problems, bugs, shortcomings, and user complaints are all branched off from this concept. Thus, many different expressions and wordings are used by individuals to write the

reviews to be identified as Issue reports by our neural network. On the other hand, this variety is far less broad when users are requesting a feature causing the use of not only more similar wordings and expressions, but also less dispersed grammatic, semantic, and even syntactic relations in review text. This fact influences the performance of the neural network models as word embeddings used in these models capture similarities between words and their semantics. Therefore, more similar wordings used in target class data samples leads to higher accuracy for the model.

7.3.2.3 Extraction of User Action

Referring to the discussion in Chapter 5, no manually coded dataset was available to be used as truth set for training and testing the model designed for this task as this is the first research attempt for automatically identifying reviews reporting a User Action. So, the only ground truth dataset used for this task is dataset one manually coded as a part of this experiment. Results of evaluating the proposed model using reviews from dataset one is shown in Table 7.3.

Table 7.3: Accuracy of the neural network used for identifying User Action in reviews

	Precision	Recall	F-Measure
CNN + Training embeddings	0.77	0.68	0.72
CNN + Google Pre-Trained embeddings	0.81	0.67	0.73

The results reveal that the model can more precisely extract User Action sentences from reviews. It is worth noting to mention that this classification is performed at sentence level. Particularly, the model identifies sentences explaining User Action from the text of a review rather than labelling a whole review as User Action or Other. This fact could considerably impact the accuracy of the proposed model.

7.3.2.4 Extraction of System Action

The model constructed for identifying System Actions has similar configurations and specifications to the model discussed in previous section for extracting User Actions. The proposed model for this task is also a novel approach. Thus, the only available dataset for training and testing the classifier was dataset one manually annotated in this research. Results of validating the proposed model is represented in Table 7.4.

Table 7.4: Accuracy of the neural network used for identifying System Action in reviews

	Precision	Recall	F-Measure
CNN + Training embeddings	0.79	0.74	0.76
CNN + Google Pre-Trained embeddings	0.82	0.80	0.81

7.3.2.5 Extraction of Aspects

The proposed approach for extracting Aspects discussed in previous chapter is validated using a ground truth dataset discussed in Section 7.2. dataset two does not include any human judgment regarding this usefulness factor. So, dataset one is used to validate the proposed approach for this task. Unlike other usefulness factors, Aspects are identified defining a set of semantic rules. The accuracy of these rules is presented in Table 7.5.

Table 7.5: Accuracy of the semantic rules used for identifying Aspects in reviews text

	Precision	Recall	F-Measure
Semantic rules	0.85	0.90	0.87

The results show that although the set of semantic rules is not completely accurate in detecting Aspect mentions in reviews, employing it can reasonably accurately perform the task.

Pattern-based approaches are known for their good accuracy in detecting true positives. However, as they are designed to capture a certain pattern from the text, these techniques always suffer from high volume of false negatives. To alleviate this issue, the defined rules are supported with a taxonomy of marker words which are the words frequently observed in a sentence containing an Aspect. So, if an Aspect candidate is captured by the semantic rules, but there is no marker in the sentence, the word will not be considered as an Aspect.

Another reason for such a good accuracy achieved is that the aim of this approach is to label sentences including an Aspect mention rather than capturing exact Aspect word/phrase. Thus, false positives fallen in a sentence that already has a true positive will be effectless.

7.3.2.6 Further Discussion on Results of Factor Extractors

Comparing the results of the neural network models shows that the accuracy of the classifiers in sentence level is significantly better than review level. This finding could be justified and interpreted in two ways. First, as it has been discussed earlier, the word embeddings are contingent upon the similarity of the words used in the review. Observing the outcome of the classifier used for identifying Issues, a considerable number of false positives (i.e., reviews predicted as Issue which are not really an Issue) were feature Requests and vice versa. The reviews reporting an issue, and the ones requesting a feature, both contain sentences explaining system and user behaviour (i.e., System Action, and User Action) making the reviews more similar in terms of utilization of words and semantic relations among them. This could be one of the convincing reasons for such behaviour of the classifiers.

Although its impact should be minimal, the second possible reason for such gap in the accuracy of the two groups of neural network models could be the use of dataset two for training and testing the first group of classifiers with lower accuracy. Dataset two is crawled and annotated by another group of researchers for distinct

aims and objectives. Thus, it is inevitable that slightly different interpretations had been used for manually identifying Issues and Feature Requests among reviews. Moreover, domain of the applications selected in their dataset, existence of outliers, spam reviews, burst review patterns for a critical issue or for the lack of an essential feature are other possible treats to the validity of dataset two.

7.3.3 System Level Evaluation

Accuracy of the approaches for identifying each usefulness factor is reported in previous sections. As components of a final system for detecting the degree of usefulness of a given review, these five approaches are integrated in a pipeline and formed the final system. Testing on dataset one which contains reviews labelled with all five usefulness factors, accuracy of the final system is measured and reported in Table 7.6. According to the decision tree represented and discussed in section 6.5, the final decision for any given reviews would be one of the four possible degree of usefulness. The following conditions might happen when processing a review.

1. A fully useful review contains Aspect AND (Issue OR Feature Request) AND (User Action AND System Action).
2. An Insufficiently Useful review contains Aspect AND (Issue OR Feature Request) AND (User Action OR System Action).
3. A Less Useful review contains (Aspect AND Feature Request) AND (User Action OR System Action).
4. A Useless review does not match with any of the above conditions.

So, the accuracy of the integrated system to process the reviews according to the decision tree for the possible outputs is represented in Table 7.6.

Table 7.6: Accuracy of the integrated system designed for identifying the degree of usefulness of a given review

	Precision	Recall	F-Measure
Fully Useful Reviews	0.5	0.57	0.53
Insufficiently Useful Reviews	0.56	0.63	0.59
Less Useful Reviews	0.49	0.58	0.53
Useless Reviews	0.61	0.69	0.65

The accuracy achieved at this level shows that integrating the approaches for extracting each usefulness factor to form the final system reduces the accuracy. This accuracy reduction could be due to the stochastic nature of neural networks. Stochastic is a mathematical term explaining the randomness and some uncertainty the outcome of neural networks is involved with. This algorithmic behaviour is a fundamental concept in machine learning assisting justification and analysis of algorithms output. Training and testing the convolutional neural network models several times in this study, slightly different results were obtained. To report consistent results, each model was evaluated 10 times and the average of the reported accuracy is computed and reported in this chapter. So, the model saved and loaded to be used in the final system might have been of a lower accuracy than what is reported in Sections 7.3.2.1 to 7.3.2.4.

Another reason for dropping accuracy when the models are used to predict a large amount of data could be the limited variety of samples of the target class in training phase. Individuals use to express their views using their own language [41, 212]. Their creative brains, therefore, generate diverse types of expression for a same concept. Training the model with huge volume of data samples will obviously help the algorithm to learn more variety of such expressions which alleviates the accuracy problem.

Finally, putting individual predictive models in a pipeline results in reduction of the accuracy as the error rate of the former models will be propagated into the

later ones. Measuring the degree of usefulness of a fully Useful review, for example, if Aspect detection approach misses the aspect mention (i.e., false negative), the review will be labelled as Useless, though other usefulness factor extractors, such as Issue Report, User Action, and System Action, make correct decisions. This propagation of error rate from initial processing steps to the higher is a common problem in text mining, but it leads to reduction of the accuracy of the final system [213].

This is worth mentioning that, observing the outputs, in some cases, the Issue detection model detects Issues where there is no issue (i.e., false positive), but the selected review is a FR which is yet in the category of Useful reviews. Similarly, in some cases, the Issue detection model detects Issues when there is no issue (i.e., false positive), but the review does not contain any Aspect and will be discarded. For example: "Please improve your photo viewing experience and provide full screen option to see photos, videos and documents." OR "Need to have an option to select multiple documents (and/or folders) and forward them at the same time instead of opening each one individually!" is labelled as an Issue by the Issue detector model, but the review is a Feature Request in fact. So, there are wrong predictions in some modules compensated by the others when classifiers are placed in a pipeline.

7.4 Comparison with Baseline Studies

In this section, the results obtained from the component level evaluation phase are compared to the state-of-the-art to visualise how effective each proposed approach is in detection of the corresponding usefulness factor. There is no comparison against the system level results as, to the best of my knowledge, this is the first time such a system is proposed.

Related studies extracting Issues from mobile application reviews, discussed in Chapter 2, have also proposed supervised models for this task. In [80] (S52) and Guzman, et al. [62] (S18), deep learning models are learnt to identify problem reports. Maalej and Nabil, [60] (S21) have also trained a Naïve Bayes classifier to perform a similar task. Comparing to these baseline studies, the model proposed in this study achieves an acceptable accuracy but does not outperform them yet. Table 7.7 represents this comparison. Please see Appendix I to find the papers using study

IDs. It is noteworthy to mention that the accuracy reported for S21 in this section is the highest accuracy achieved by the authors' model focusing on the text of given reviews.

Table 7.7: Accuracy of the neural network used for identifying Issues from review text compared to the baseline studies

	Precision	Recall	F-Measure
S52 (NN + problem report)	0.46	0.60	0.52
S18 (NN + Bug report & Complaint)	0.64	0.42	0.51
S21 (NB + Bug report)	0.71	0.72	0.71
This approach (CNN + Issue Report)	0.60	0.76	0.67

Chapter 2 has also discussed studies investigated the task of extracting Feature Requests employing machine learning techniques. Supervised approaches have been used in (S21, S18) to extract this factor. Table 7.8 compares the accuracy achieved by our proposed model to the baseline approaches.

Table 7.8: Accuracy of the neural network used for identifying Feature Requests from review text compared to the baseline studies

	Precision	Recall	F-Measure
S18 (NN + User Requests)	0.71	0.39	0.50
S21 (NB + Feature Request)	0.76	0.68	0.72
This approach (CNN + Feature Request)	0.67	0.78	0.72

Aspect is the last usefulness factor that could be compared with baseline studies in terms of accuracy. The accuracy of our proposed approach for extraction of Aspects is compared with the baseline studies in Table 7.9. In the first baseline study [180] (S54), deep learning models are combined with linguistic patterns to extract Aspects from review text. Testing their approach, the authors used reviews for two types of products (i.e., laptop and restaurant) the average of which is reported on Table 7.9. Rana et al., [99] (S55) used sequential patterns and opinion lexicon, taken from [101] (S56) wherein no validation is reported, to extract noun phrases and opinion words. then, they measured frequency of Aspect candidates to find Aspect mentions.

As the table shows, the accuracy achieved in S55 is slightly higher than the proposed approach in this thesis. However, there are several drawbacks with the approach proposed in S55.

Firstly, the approach is an opinion mining technique and aims to extract user opinions on an Aspect of a product. The Aspects discussed technically without any opinion associated, therefore, are ignored using such approaches. Analysing app review datasets in this research, too many samples of Aspects used in neutral sentences were observed. A simple example for demonstrating ineffectiveness of

Table 7.9: Accuracy of the semantic rules used for identifying Aspects from review text compared to the baseline studies

	Precision	Recall	F-Measure
S54 (CNN + Linguistic patterns)	0.82	0.87	0.84
S55 (sequential patterns)	0.87	0.92	0.89
This approach (Semantic rules)	0.85	0.90	0.87

such approaches for analysing app reviews is the following review containing an Aspect and having a neutral sentiment:

"I use the navigation feature to travel overseas"

Secondly, another clue for identifying Aspects in their study or for pruning detected Aspects to increase the recall by minimizing false positives, they took the frequency of Aspect candidates' occurrences in the dataset to approve them as an Aspect mention. It might be useful when processing product reviews to find problematic components, for example, discussed in several reviews. Analysing ap reviews, however, it causes missing rarely mentioned Aspects. Particularly, investigating existence of Feature Requests, the proposed Aspect might be used only once in the whole dataset.

Thirdly, the analysed reviews in their approach are product reviews, such as DVD players, cameras, and MP3 players, rather than mobile app reviews. Obviously, type of the aspects used in these two domains of reviews vary significantly effecting the accuracy of the extraction technique.

Finally, as it has been discussed in Section 2.7.1.3, Dabrowski et al. [36] discovered in an empirical study that such approaches achieve lower effectiveness than reported originally. Therefore, in terms of processing app reviews, the effectiveness of the approach proposed in S55 is far below the expectations.

7.5 Conclusion

Validation of the proposed final system for measuring the usefulness of app reviews is reported in this chapter along with individually validating its components. The results are interpreted and justified across several sections and are compared to the baseline studies reported comparable approaches. As this is the first research attempt on automatically identifying User Action and System Action, there is no comparison with baseline approaches reported in this chapter.

As the most important finding of this chapter, the results demonstrate that the process of automating the task of extracting each of the usefulness factors proposed in this research and integrating them as a final system is practical and achievable.

Chapter 8 Research Limitations and Challenges

8.1 Introduction

The research reported in this thesis addresses several challenges and exposes several limitations that suggest future research. In this chapter, the limitations and challenges this research project was encountered during different phases are discussed.

This chapter consists of two sections. Section 8.2 discusses details on the limitations of this research. This is followed by a statement of the challenges our study was tackled with in Section 8.3.

8.2 Limitations of the Approach

Observations on various datasets show that domain dependency has an inevitable impact on the performance of app review processing approaches. Although reviews for six apps from different domains have composed the dataset used in this research, several domains of the existing apps are yet unexplored for this research, such as medical apps, games, and financial apps to name a few.

Each of the domains has their own terminology and jargons making factors extraction complicated. In some cases, the explanation of users in their review is very similar to one of the required factors, but they are just explaining their personal life rather than referring to the app functionality. The proposed system might not be able to capture all these cases due to the complexity of natural language and the unstructured reviewing platforms. The following fabricated review is an example of capturing User Action by the proposed approach while there is no User Action in the review.

Ali: "*When I started my travel, I realised that I had forgotten to pay for my booking. So, I used the app and paid it easily.*"

In the above example, the user is explaining his action, but it is not a User Action as described in Section 3.4.4. However, the proposed approach captures this as a User Action because it is explaining an action of the user, and several key terms such as ‘pay’, ‘booking’, and ‘travel’ are in the review that are components of an app in many domains such as travelling, booking, etc.

The aim of this study for proposing semantic rules for extracting Aspects is to discover reviews including an Aspect mention. These rules are crafted in a way to meet the project aim. Thus, such semantic rules might not perform appropriately when used in opinion mining tasks for detecting exact Aspect word/phrase without proper customisations and adaptation.

Another limitation of the proposed approach for detecting Aspects is dealing with negations. Negations do not considerably impact the machine learning techniques used for extracting other usefulness factors as the models trained using the manually coded data samples, and the word embeddings used have already considered such relationships between words. However, the semantic rule-based approach needs to be equipped with a negation detection technology, though observing data samples in the corpus reveals the minimal impact of negations on the accuracy of aspect detection rules.

Cross document processing is out of the scope of this study. Although we had no observation of interaction between reviewers causing distribution of usefulness factors across several reviews in the dataset, the current approach is unable to find such relations between different reviews in order to extract and link the factors across several revies. The following reviews are fabricated to show how a useful review will be considered useless using the proposed system for ignoring the relationship between revies.

Ali: “When I open the app, it freezes at the login page, and I cannot even enter my username and password”

Zeynab: “I have the same issue Ali, but mine accept username and password, then freezes”

One of the required usefulness factors for each review to be considered as useful is Aspect. In the reviews above the aspect mention 'login page' is in Ali's review, but it is referred to in Zeynab's review as well. However, the proposed approach is unable to capture Zeynab's review as useful.

The F-Measure of the proposed approaches for detecting Feature Requests, User Actions, and System Actions are currently state-of-the-art. Performance of the Issue and Aspect detector approaches are also considered as good with the F-Measures equal to 0.67 and 0.87, respectively. However, there is still between %10 to %33 wrong predictions with the approaches. Propagating these errors further has caused lower accuracy for the integrated final system.

8.3 Research Challenges

The research presented in this thesis was posed to a number of unique challenges, mainly related to the proposed approaches for extracting usefulness factors from app reviews. These challenges are discussed in the following paragraphs.

Sensitivity and complexity of Neural Network hyperparameters. The main technology used in this research is Neural Network which had better performance comparing to other supervised machine learning techniques used for processing natural language [179, 214]. However, in terms of variety of hyperparameters to be tuned, the networks are significantly sensitive and complicated. Configuring a CNN model for text processing, Zhang and Byron [191] analysed the sensitivity of Neural Network hyperparameters. They argued that CNN-based approaches are very sensitive to many free parameters causing difficulties in repeating experiments for practitioners.

Using invented abbreviations for writing reviews. Observing the dataset, a considerable percentage of reviews were found written in shortened word forms. Some people use to shorten the words into ungrammatical abbreviations whenever possible to type less characters. This trend initiated when there was text limitation in some platforms. There are several words that can be abbreviated in English such as That='dat', The = 'd', Please = 'plz', Friend = 'frnd', Your = 'ur', and Message = 'msg', to name a few. This type of writing defects the training and testing processes

in machine learning based approaches, causes pattern matching mistakes in rule-based ones, and results in redundant outputs. Unfortunately, as these abbreviations are invented by the reviewers, detecting them using existing NLP and text processing tools is almost impossible. In this research, a word replacing heuristic was developed to search for frequent abbreviations and correct them as many of them are frequently repeated. However, as the human brain is creative, several odd cases were not captured by the heuristic and manually corrected.

Conflicting app Aspects with real world objects. One of the challenges in extracting app Aspects is detecting aspects having similar names to real world things. For example, Pintrest has an Aspect called 'pin' which is expected to be captured as an Aspect when it is occurred in Pinterest reviews and not captured otherwise. Similarly, the word 'chat' is used as an Aspect in Whatsapp reviews but as conversation in other app reviews. In the following review, as another challenging example, although the word 'note' is one of the Evernote components, it is not an Aspect.

"Instead of having multiple paper notebooks or little notes everywhere, everything I need to remember or reference goes into Evernote. This will be my tool for going paperless."

As there are lots of these conflicting Aspects, detecting them is challenging because several text processing techniques such as keywords become ineffective. This problem was alleviated in this study defining semantic rules to use relationships between different words in a sentence to decide on the role of a word. However, the issue is yet challenging.

Error-prone nature of natural language. In this research, dependency relations between words are combined with part of speeches of them to define accurate rules for identifying Aspects. The reviews are written by human and might not go through grammar and spell-checking systems or are corrected by the automatic spell-checking systems inappropriately. This fact affects the accuracy of the proposed approach. For example, 'upload' is tagged as a Verb in the following review taken from our dataset because the word 'an' is used rather than the correct

word ‘and’. The POS tagger will tag ‘upload’ as a Noun if we correct the mistake and change the word ‘and’ to ‘an’.

“…also and upload to Facebook would be as equally wonderful for the one place stop to go to.”

Grammar mistakes. As the reviewing facility is provided by global platforms, English is second language of many users posting English reviews for mobile apps. Analysing the dataset used in this research, several sentences where found using incorrect grammar that have not been captured by grammar exception detectors. Our manual analysis reveals that the grammar used in the review is either taken from individuals’ first language which is not English, or due to the reviewer’s language barriers. The following review, for example, is not identified by automatic grammar correction tools.

“But there is one suggestion is you that have include chat functionality.”

8.4 Conclusion

The research experiment reported in this thesis has encountered several limitation and challenges. These limitations and challenges are discussed in this chapter to provide future practitioners and researchers from the same domain with insights and cautions while conducting experiments.

Chapter 9 Conclusion

9.1 Introduction

In this thesis, an approach for efficiently measuring usefulness of mobile application reviews has been defined. Carefully applying the discretion of mobile application developers and requirement engineers, five usefulness factors were defined to quantify the task of measuring usefulness of application reviews. For reality checking and investigating the feasibility of automating the process of detecting usefulness reviews using these factors, an NLP approach was proposed for extracting each factor and the results were checked against real world reviews and human judgements. In Chapter 1, a set of research questions is defined. How this research answered these research questions is discussed in Section 9.2. This is followed by a number of future research directions discussed in Section 9.3.

9.2 Research Questions

RQ1. How to effectively measure the usefulness of app reviews? What usefulness factors could be used?

In this research, the task of identifying usefulness factors was performed in two phases to fulfil adjective one. First, conducting a systematic literature review, related papers studying app review analysis for software development were selected and investigated to understand how researchers have measured the usefulness of an app review for software evolution. Besides, research outputs from projects investigating the quality of software artifacts, such as user needs, bug reports, app testing results, and requirement statements were analysed to identify any possible metric to be used for measuring the usefulness of natural language reviews from developers' viewpoint. The output of this phase was a preliminary list of 12 usefulness factors for measuring the usefulness of app reviews. Details of this phase are reported and discussed in Chapter 3.

The second phase of this task was to seek advice and suggestions from experienced developers and requirements engineers for any factors. This phase is discussed in sections 428 and 4.2.9.

RQ2. How to validate the proposed factors for measuring the usefulness of app reviews?

To fulfil objective two which is designed to address this research question, six well-skilled mobile app developers were invited to a focus group discussion. The list of identified factors was discussed and validated in the FGD. The outcome of this task was a set of five approved usefulness factors. Details of this phase are reported and discussed in Chapter 4.

RQ3. How to automatically extract the usefulness factors from the text of app reviews?

To address this research question, the annotated dataset discussed in Section 7.2 was carefully analysed to investigate possibility of extracting the usefulness factors automatically. Considering all specifics of each usefulness factor, proper extraction technique was proposed. To extract Issues and Feature requests, CNN models were applied with word embeddings as word representation. The similar approach was used for extracting User Action and System Action, but the analysis was at sentence level. Details of the extraction methods are explained in Sections 6.4.3 and 6.4.4. Finally, Abstracts were extracted using semantic rules discussed in Section 6.4.5.

RQ4. How to automatically detect useful reviews using the usefulness factors?

Objective 4 was designed in this project to address this research question. To fulfil objective 4, the individual approaches proposed for automatically extracting each of the usefulness factors gathered in a pipeline to form an integrated system for measuring the usefulness of given reviews. The measurement was done using the decision tree discussed in Section 6.5.

RQ5. How to evaluate the proposed automatic approaches?

To evaluate the individual extraction approaches, a ground truth dataset was created in Section 7.2. Each of the approaches were then tested against the ground truth datasets using the evaluation metrics (i.e., Precision, Recall, and F-Measure)

defined in Section 7.3.1. Results of evaluating the approaches are discussed in detail in Chapter 7. The results show that the approaches perform well.

The final system composed of these individual approaches for measuring the usefulness of given reviews was also tested on the ground truth dataset. Similar validation metrics were also used for the evaluation. Results show that the integrated system has a reasonable accuracy with potential for further improvement. An important finding at this point is that the automation task of extracting the proposed usefulness factors is feasible as the implementation phase of this research demonstrates.

9.3 Future work

9.3.1 Tool Extensions and Improvements

One direction of future work could be extending each individual approach proposed for extracting each of the usefulness factors to improve the accuracy and effectiveness. Integrating these individual approaches to form a system for identifying useful reviews, such improvements will increase the overall accuracy of the system.

The neural network-based models trained for extracting Issues, Feature Requests, User Actions, and System Actions could be extended further by identifying and engineering effective features from review text and using them as classification attributes to improve the accuracy of the classifier. Name entities, semantic roles, frequently repeated keywords, readability of reviews, and Bag of Words (BoW) are few examples of such features. These supervised models can also be further improved by employing other available types of information coming with a review. For example, rating given to the target app is one of the useful clues in identifying Issues as problematic apps usually receive low rates[18, 49]. Available data and metadata fields attached to a review are discussed in Section 5.2.

Integrating these supervised models with related complementary approaches to form a hybrid approach is another possibility of improvement. For example, results obtained from the CNN models could be further investigated using syntactic

rules and heuristics. The complementary approaches can also be computed on training data and fed into the model to enrich the training phase. The last suggested improvement for such convolutional neural network models is designing much deeper architectures for the networks. Comparing the performance of existing architectures with limited layers to the ones suggested by Conneau et al, [215] with more than 100 layers demonstrates the privilege of the deeper architectures designed for processing natural language text.

In terms of extending the sematic rule-based approach proposed in this research for identifying Aspects, one future improvement could be exploring more in-dept relationships among words, sentences, and semantics in reviews not only to refine existing rules accordingly, but also to define new rules. Similar to the supervised approaches discussed above, combining with other approaches, such as heuristics, statistical methods, and machine learning techniques, rule-based approaches can also be part of a hybrid approach to improve the performance.

9.3.2 Expanding the Current Research

One of the future research directions in identification of constructive user requirements for software development from user generated feedback is to explore other sources of user feedback. Apart from mobile application reviews posted on opinion sharing websites such as Google Play and App Store, other channels are available to be mined for mobile application constructive user feedback. In recent research efforts, relevant information for software companies is observed and investigated in mobile application related tweets [148, 216, 217], and software forums [149, 218]. The quality factors defined for extracting useful reviews, approaches proposed for identifying each factor, and the proposed system for extracting useful reviews could be adopted for mining and analysing such channels to identify useful information for mobile application evolution.

The performance evaluation of the proposed approach is reported in two levels (i.e., component level and system level) in Chapter 7. It is expected that either the final system, or any of its individual components can be used for analysing reviews in other languages. Although the grammar and semantic relationships used in other languages varies considerably, available tools and Python libraries to elicit

such syntactic roles and semantic dependencies facilitate the customisation of the system.

Facilitating the requirements engineering task, the next step after automating the process of extracting user needs from app reviews would be transforming them to requirements. Very similar to the requirements engineering phases, identified user needs are to be processed further (i.e., removing redundant needs, summarizing topics, and classifying them into further sub-categories). Finally, the prepared requirements are to be automatically prioritised to be incorporated in the next releases of the app versions. Effective automated approaches at this step is a promising future work for transforming user needs to requirement statements and prioritizing them with respect to developers' viewpoint

9.4 Summary

This thesis aims to address the issue of ignoring developers' perspective in identification of useful app reviews for software evolution. To achieve this, experienced mobile app developers were employed to identify five usefulness factors for measuring the usefulness of app reviews with respect to developers' viewpoint. To demonstrate how achievable is the automation of the usefulness factors, an NLP based approach was proposed for extracting each of the factors. Finally, the individual factor extractors were integrated to form a final system for measuring the usefulness of app reviews. In its first section, this chapter answers the research questions defined in Chapter 1. Several related chapters are cited for more details. Several future research directions are also provided in this chapter.

References

[1] S. Wilson, M. Bekker, P. Johnson, and H. Johnson, "Helping and hindering user involvement—a tale of everyday design," in *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems*, 1997, pp. 178-185.

[2] S. Kujala, M. Kauppinen, L. Lehtola, and T. Kojo, "The role of user involvement in requirements quality and project success," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005, pp. 75-84: IEEE.

[3] G. Ortega and A. Emitza, "Mining User Reviews from Mobile Applications for Software Evolution," Technische Universität München, 2015.

[4] J. Blomberg, M. Burrell, and G. Guest, "An ethnographic approach to design," *Human-Computer Interaction*, pp. 71-94, 2009.

[5] J. Karat, "Evolving the scope of user-centered design," *Communications of the ACM*, vol. 40, no. 7, pp. 33-38, 1997.

[6] E. Guzman and B. Bruegge, "Towards emotional awareness in software development teams," in *Proceedings of the 2013 9th joint meeting on foundations of software engineering*, 2013, pp. 671-674.

[7] S. Kujala, "User involvement: a review of the benefits and challenges," *Behaviour & information technology*, vol. 22, no. 1, pp. 1-16, 2003.

[8] L. Damodaran, "User involvement in the systems design process-a practical guide for users," *Behaviour & information technology*, vol. 15, no. 6, pp. 363-377, 1996.

[9] K. El Emam and N. H. Madhavji, "A field study of requirements engineering practices in information systems development," in *Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95)*, 1995, pp. 68-80: IEEE.

[10] H. Lieberman, F. Paternò, M. Klann, and V. Wulf, "End-user development: An emerging paradigm," in *End user development*: Springer, 2006, pp. 1-8.

[11] C. Jones, "End user programming," *Computer*, vol. 28, no. 9, pp. 68-70, 1995.

[12] K. El Emam, S. Quintin, and N. H. Madhavji, "User participation in the requirements engineering process: An empirical study," *Requirements engineering*, vol. 1, no. 1, pp. 4-26, 1996.

[13] B. Bruegge and A. H. Dutoit, "Object--Oriented Software Engineering. Using UML, Patterns, and Java," *Learning*, vol. 5, no. 6, p. 7, 2009.

[14] W. Maalej and D. Pagano, "On the socialness of software," in *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, 2011, pp. 864-871: IEEE.

[15] E. L. Wagner and G. Piccoli, "Moving beyond user participation to achieve successful IS design," *Communications of the ACM*, vol. 50, no. 12, pp. 51-55, 2007.

[16] A. Heydari, M. ali Tavakoli, N. Salim, and Z. Heydari, "Detection of review spam: A survey," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3634-3642, 2015.

[17] P. Mikalef, K. Sharma, I. O. Pappas, and M. Giannakos, "Seeking information on social commerce: An examination of the impact of user-and marketer-generated content through an eye-tracking study," *Information Systems Frontiers*, pp. 1-14, 2020.

[18] H. Xia, X. Pan, W. An, and Z. Zhang, "Can Online Rating Reflect Authentic Customer Purchase Feelings? Understanding How Customer Dissatisfaction Relates to Negative Reviews," *Journal of Computer Information Systems*, pp. 1-14, 2019.

[19] A. Golmohammadi, A. S. Mattila, and D. K. Gauri, "Negative online reviews and consumers' service consumption," *Journal of Business Research*, vol. 116, pp. 27-36, 2020.

[20] F. Dalpiaz and M. Parente, "RE-SWOT: from user feedback to requirements via competitor analysis," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2019, pp. 55-70: Springer.

[21] F. Palomba *et al.*, "Crowdsourcing user reviews to support the evolution of mobile apps," *Journal of Systems and Software*, vol. 137, pp. 143-162, 2018.

[22] W. Martin, "Causal impact for app store analysis," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 659-661.

[23] A. Heydari, M. Tavakoli, and N. Salim, "Detection of fake opinions using time series," *Expert Systems with Applications*, vol. 58, pp. 83-92, 2016.

[24] Z. Ismail, A. Heydari, M. Tavakoli, and N. Salim, "Incorporating Author's Activeness in Online Discussion in Thread Retrieval Model," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 2, pp. 473-479, 2016.

[25] A. Heydari, M. Tavakoli, Z. Ismail, and N. Salim, "Leveraging Quality Metrics in Voting Model Based Thread Retrieval," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, no. 1, pp. 117-123, 2016.

[26] M. Fernández-Gavilanes, T. Álvarez-López, J. Juncal-Martínez, E. Costa-Montenegro, and F. J. González-Castaño, "Unsupervised method for sentiment analysis in online texts," *Expert Systems with Applications*, vol. 58, pp. 57-75, 2016.

[27] S. Venkatakrishnan, A. Kaushik, and J. K. Verma, "Sentiment analysis on google play store data using deep learning," in *Applications of Machine Learning*: Springer, 2020, pp. 15-30.

[28] S. Ranjan and S. Mishra, "Comparative Sentiment Analysis of App Reviews," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, 2020, pp. 1-7: IEEE.

[29] D. Savage, X. Zhang, X. Yu, P. Chou, and Q. Wang, "Detection of opinion spam based on anomalous rating deviation," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8650-8657, 2015.

[30] L. You, Q. Peng, Z. Xiong, D. He, M. Qiu, and X. Zhang, "Integrating aspect analysis and local outlier factor for intelligent review spam detection," *Future Generation Computer Systems*, vol. 102, pp. 163-172, 2020.

[31] M. Z. Asghar, A. Ullah, S. Ahmad, and A. Khan, "Opinion spam detection framework using hybrid classification scheme," *Soft computing*, vol. 24, no. 5, pp. 3475-3498, 2020.

[32] M. Castelli, L. Manzoni, L. Vanneschi, and A. Popovič, "An expert system for extracting knowledge from customers' reviews: The case of Amazon. com, Inc," *Expert Systems with Applications*, vol. 84, pp. 117-126, 2017.

[33] R. Collado-Borrell, V. Escudero-Vilaplana, C. Villanueva-Bueno, A. Herranz-Alonso, and M. Sanjurjo-Saez, "Features and functionalities of smartphone apps related to COVID-19: systematic search in app stores and content analysis," *Journal of medical Internet research*, vol. 22, no. 8, p. e20334, 2020.

[34] A. Al-Subaihin, F. Sarro, S. Black, and L. Capra, "Empirical comparison of text-based mobile apps similarity measurement techniques," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3290-3315, 2019.

[35] K. Srisopha, C. Phonsom, M. Li, D. Link, and B. Boehm, "On Building an Automatic Identification of Country-Specific Feature Requests in Mobile App Reviews: Possibilities and Challenges," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 494-498.

[36] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Mining User Opinions to Support Requirement Engineering: An Empirical Study," in *International Conference on Advanced Information Systems Engineering*, 2020, pp. 401-416: Springer.

[37] A. AlSubaihin, F. Sarro, S. Black, L. Capra, and M. Harman, "App store effects on software engineering practices," *IEEE Transactions on Software Engineering*, 2019.

[38] A. A. Al-Subaihin and F. Sarro, "Exploring The Use of Genetic Algorithm Clustering for Mobile App Categorisation," in *International Symposium on Search Based Software Engineering*, 2020, pp. 181-187: Springer.

[39] M. Tavakoli, L. Zhao, A. Heydari, and G. Nenadić, "Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools," *Expert Systems with Applications*, 2018.

[40] A. Di Sorbo, G. Grano, C. Aaron Visaggio, and S. Panichella, "Investigating the criticality of user-reported issues through their relations with app rating," *Journal of Software: Evolution and Process*, p. e2316, 2020.

[41] C. Wang, M. Daneva, M. van Sinderen, and P. Liang, "A systematic mapping study on crowdsourced requirements engineering using user feedback," *Journal of software: Evolution and Process*, vol. 31, no. 10, p. e2199, 2019.

[42] N. Genc-Nayebi and A. Abran, "A Systematic Literature Review: Opinion Mining Studies from Mobile App Store User Reviews," *Journal of Systems and Software*, 16/11/2016 2016.

[43] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE Transactions on Software Engineering*, 2016.

[44] Barbara Kitchenham, Stuart Charters, David Budgen, Pearl Brereton, and M. Turner, "Guidelines for performing systematic literature reviews in software engineering," in *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*, 2007.

[45] H. Boeije, "A purposeful approach to the constant comparative method in the analysis of qualitative interviews," *Quality & quantity*, vol. 36, no. 4, pp. 391-409, 2002.

[46] E. Ha and D. Wagner, "Do android users write about electric sheep? examining consumer reviews in google play," in *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*, 2013, pp. 149-157: IEEE.

[47] C. Iacob, V. Veerappa, and R. Harrison, "What are you complaining about?: a study of online reviews of mobile applications," in *Proceedings of the 27th International BCS Human Computer Interaction Conference*, 2013, p. 29: British Computer Society.

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

[49] H. Khalid, "On identifying user complaints of iOS apps," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 1474-1476: IEEE.

[50] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?," *IEEE Software*, vol. 32, no. 3, pp. 70-77, 2014.

[51] M. Khalid, M. Asif, and U. Shehzaib, "Towards improving the quality of mobile app reviews," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 7, no. 10, p. 35, 2015.

[52] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "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*, 2013, pp. 1276-1284: ACM.

[53] A. Truelove, F. N. Chowdhury, O. Gnawali, and M. A. Alipour, "Topics of concern: identifying user issues in reviews of IoT apps and devices," in *2019 IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)*, 2019, pp. 33-40: IEEE.

[54] L. V. Galvis Carreno and K. Winbladh, "Analysis of user comments: an approach for software requirements evolution," in *Proceedings of the International Conference on Software Engineering*, 2013 pp. 582-591: IEEE Press.

[55] E. Platzer, "Opportunities of automated motive-based user review analysis in the context of mobile app acceptance," in *CECIIS-2011*, 2011.

[56] J. Oh, D. Kim, U. Lee, J.-G. Lee, and J. Song, "Facilitating developer-user interactions with mobile app review digests," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, 2013, pp. 1809-1814: ACM.

[57] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 767-778: ACM.

[58] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *2014 IEEE 22nd international requirements engineering conference (RE)*, 2014, pp. 153-162: IEEE.

[59] S. McIlroy, N. Ali, H. Khalid, and A. E. Hassan, "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1067-1106, 2016.

[60] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? on automatically classifying app reviews," in *2015 IEEE 23rd international requirements engineering conference (RE)*, 2015, pp. 116-125: IEEE.

[61] W. Maalej, Z. KurtanoviÄ‡, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Engineering*, vol. 21, no. 3, pp. 311-331, 2016.

[62] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble methods for app review classification: An approach for software evolution (N)," in *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, 2015, pp. 771-776: IEEE.

[63] M. Tavakoli, A. Heydari, Z. Ismail, and N. Salim, "A Framework for Review Spam Detection Research," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, no. 1, pp. 67-71, 2015.

[64] L. Wang, H. Nakagawa, and T. Tsuchiya, "Opinion Analysis and Organization of Mobile Application User Reviews," in *REFSQ Workshops*, 2020.

[65] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *Software Maintenance and Evolution (ICSME), IEEE International Conference on*, 2015 pp. 281-290: IEEE.

[66] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "ARdoc: app reviews development oriented classifier," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2016, pp. 1023-1027: ACM.

[67] J. Buchan, M. Bano, D. Zowghi, and P. Volabouth, "Semi-automated extraction of new requirements from online reviews for software product evolution," in *2018 25th Australasian Software Engineering Conference (ASWEC)*, 2018, pp. 31-40: IEEE.

[68] A. 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*, 2016, pp. 499-510: ACM.

[69] N. Jha and A. Mahmoud, "Mining User Requirements from Application Store Reviews Using Frame Semantics," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2017, pp. 273-287: Springer.

[70] N. Jha and A. Mahmoud, "MARC: A Mobile Application Review Classifier," presented at the International Working Conference on Requirements Engineering: Foundation for Software Quality, 2017.

[71] N. Jha and A. Mahmoud, "Using frame semantics for classifying and summarizing application store reviews," *Empirical Software Engineering*, vol. 23, no. 6, pp. 3734-3767, 2018.

[72] N. Jha and A. Mahmoud, "Mining non-functional requirements from App store reviews," *Empirical Software Engineering*, vol. 24, no. 6, pp. 3659-3695, 2019.

[73] Z. Kurtanović and W. Maalej, "Mining user rationale from software reviews," in *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, 2017, pp. 61-70: IEEE.

[74] F. Palomba *et al.*, "User reviews matter! tracking crowdsourced reviews to support evolution of successful apps," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, 2015, pp. 291-300: IEEE.

[75] A. Al-Hawari, H. Najadat, and R. Shatnawi, "Classification of application reviews into software maintenance tasks using data mining techniques," *Software Quality Journal*, pp. 1-37, 2020.

[76] I. Triguero *et al.*, "KEEL 3.0: an open source software for multi-stage analysis in data mining," 2017.

[77] J. Arunadevi, S. Ramya, and M. R. Raja, "A study of classification algorithms using Rapidminer," *International Journal of Pure and Applied Mathematics*, vol. 119, no. 12, pp. 15977-15988, 2018.

[78] R. Mans, W. M. van der Aalst, and H. Verbeek, "Supporting Process Mining Workflows with RapidProM," *BPM (Demos)*, vol. 56, 2014.

[79] N. Al Kilani, R. Tailakh, and A. Hanani, "Automatic Classification of Apps Reviews for Requirement Engineering: Exploring the Customers Need from Healthcare Applications," in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019, pp. 541-548: IEEE.

[80] C. Stanik, M. Haering, and W. Maalej, "Classifying multilingual user feedback using traditional machine learning and deep learning," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, 2019, pp. 220-226: IEEE.

[81] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Mining Software Repositories (MSR), 10th IEEE Working Conference on*, 2013 pp. 41-44: IEEE.

[82] M. van Vliet, E. C. Groen, F. Dalpiaz, and S. Brinkkemper, "Identifying and Classifying User Requirements in Online Feedback via Crowdsourcing," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2020, pp. 143-159: Springer.

[83] M. Glinz, "On non-functional requirements," in *15th IEEE International Requirements Engineering Conference (RE 2007)*, 2007, pp. 21-26: IEEE.

[84] C. W. Lee, S. Licorish, S. MacDonell, P. Patel, and T. Savarimuthu, "They'll Know It When They See It: Analyzing Post-Release Feedback from the Android Community," *InProc 21st Amer Conf. Info. Sys.-AMCIS. Puerto Rico: AISeL. pp. 1 2015 (Vol. 11).* 2015.

[85] S. Moghaddam, "Beyond sentiment analysis: mining defects and improvements from customer feedback," in *European Conference on Information Retrieval*, 2015, pp. 400-410: Springer.

[86] H. Yang and P. Liang, "Identification and Classification of Requirements from App User Reviews," in *ACM International conference on Software engineering and knowledge engineering (SEKE)*, 2015, pp. 7-12.

[87] J. Zhang, Y. Wang, and T. Xie, "Software feature refinement prioritization based on online user review mining," *Information and Software Technology*, vol. 108, pp. 30-34, 2019.

[88] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *Journal of the Association for Information Science and Technology*, vol. 63, no. 1, pp. 163-173, 2012.

[89] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993-1022, 2003.

[90] S. S. Sohail, J. Siddiqui, and R. Ali, "Feature extraction and analysis of online reviews for the recommendation of books using opinion mining technique," *Perspectives in Science*, vol. 8, pp. 754-756, 2016.

[91] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining User Opinions in Mobile App Reviews: A Keyword-Based Approach (T)," in *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, 2015, pp. 749-759: IEEE.

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

[93] H. Malik, E. M. Shakshuki, and W.-S. Yoo, "Comparing mobile apps by identifying 'Hot'features," *Future Generation Computer Systems*, vol. 107, pp. 659-669, 2020.

[94] S. Poria, E. Cambria, L.-W. Ku, C. Gui, and A. Gelbukh, "A rule-based approach to aspect extraction from product reviews," in *Proceedings of the second workshop on natural language processing for social media (SocialNLP)*, 2014, pp. 28-37.

[95] I. Cruz, A. F. Gelbukh, and G. Sidorov, "Implicit Aspect Indicator Extraction for Aspect based Opinion Mining," *Int. J. Comput. Linguistics Appl.*, vol. 5, no. 2, pp. 135-152, 2014.

[96] G. A. Miller, *WordNet: An electronic lexical database*. MIT press, 1998.

[97] E. Cambria, D. Olsher, and D. Rajagopal, "SenticNet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis," in *Proceedings of the twenty-eighth AAAI conference on artificial intelligence*, 2014, pp. 1515-1521.

[98] S. Poria, E. Cambria, and A. Gelbukh, "Aspect extraction for opinion mining with a deep convolutional neural network," *Knowledge-Based Systems,* vol. 108, pp. 42-49, 2016.

[99] T. A. Rana and Y.-N. Cheah, "A two-fold rule-based model for aspect extraction," *Expert systems with applications,* vol. 89, pp. 273-285, 2017.

[100] T. Ahmad Rana, "A Sequential Pattern Rule-Based Approach For Explicit And Implicit Aspect Extraction From Online Product Reviews," Universiti Sains Malaysia, 2017.

[101] T. A. Rana and Y.-N. Cheah, "Hybrid rule-based approach for aspect extraction and categorization from customer reviews," in *2015 9th International Conference on IT in Asia (CITA)*, 2015, pp. 1-5: IEEE.

[102] T. Guo, B. Guo, Y. Ouyang, and Z. Yu, "Mining and Analyzing User Feedback from App Reviews: An Econometric Approach," in *SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI*, 2018, pp. 841-848.

[103] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka Jr, and T. M. Mitchell, "Coupled semi-supervised learning for information extraction," in *Proceedings of the third ACM international conference on Web search and data mining*, 2010, pp. 101-110.

[104] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores," in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, 2012, pp. 108-111: IEEE Press.

[105] L. Hoon, R. Vasa, J.-G. Schneider, and K. Mouzakis, "A preliminary analysis of vocabulary in mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference*, 2012, pp. 245-248: ACM.

[106] R. Vasa, L. Hoon, K. Mouzakis, and A. Noguchi, "A preliminary analysis of mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference*, 2012, pp. 241-244: ACM.

[107] E. Noei, F. Zhang, and Y. Zou, "Too Many User-Reviews, What Should App Developers Look at First?," *IEEE Transactions on Software Engineering*, 2019.

[108] U. Grömping, "Relative importance for linear regression in R: the package relaimpo," *Journal of statistical software,* vol. 17, no. 1, pp. 1-27, 2006.

[109] B. E. Feldman, "Relative importance and value," *Available at SSRN 2255827,* 2005.

[110] H. Guo and M. P. Singh, "Caspar: extracting and synthesizing user stories of problems from app reviews," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 628-640.

[111] D. Cer *et al.*, "Universal sentence encoder," *arXiv preprint arXiv:1803.11175,* 2018.

[112] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 48-58.

[113] C. Gao *et al.*, "Emerging app issue identification from user feedback: experience on wechat," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019, pp. 279-288: IEEE.

[114] M. J. Zaki, "Scalable algorithms for association mining," *IEEE transactions on knowledge and data engineering,* vol. 12, no. 3, pp. 372-390, 2000.

[115] C. Tao, H. Guo, and Z. Huang, "Identifying security issues for mobile applications based on user review summarization," *Information and Software Technology,* vol. 122, p. 106290, 2020.

[116] Y. Zhou, Y. Su, T. Chen, Z. Huang, H. C. Gall, and S. Panichella, "User review-based change file localization for mobile applications," *IEEE Transactions on Software Engineering,* 2020.

[117] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* vol. 63, no. 2, pp. 411-423, 2001.

[118] A. Begel and T. Zimmermann, "Analyze this! 145 questions for data scientists in software engineering," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 12-23: ACM.

[119] Y. Liu, J. Jin, P. Ji, J. A. Harding, and R. Y. Fung, "Identifying helpful online reviews: a product designer's perspective," *Computer-Aided Design*, vol. 45, no. 2, pp. 180-194, 2013.

[120] J. Qi, Z. Zhang, S. Jeon, and Y. Zhou, "Mining customer requirements from online reviews: A product improvement perspective," *Information & Management*, vol. 53, no. 8, pp. 951-963, 2016.

[121] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of systems and software*, vol. 80, no. 4, pp. 571-583, 2007.

[122] W. Claes, R. Per, H. Martin, C. Magnus, R. Björn, and A. Wesslén, "Experimentation in software engineering: an introduction," *Online Available: http://books. google. com/books*, 2000.

[123] M. Gasparic and A. Janes, "What recommendation systems for software engineering recommend: A systematic literature review," *Journal of Systems and Software*, vol. 113, pp. 101-113, 2016.

[124] S. Krishnamoorthy, "Linguistic features for review helpfulness prediction," *Expert Systems with Applications*, vol. 42, no. 7, pp. 3751-3759, 2015.

[125] T. L. Ngo-Ye, A. P. Sinha, and A. Sen, "Predicting the helpfulness of online reviews using a scripts-enriched text regression model," *Expert Systems with Applications*, vol. 71, pp. 98-110, 2017.

[126] H. Zhang and M. A. Babar, "On searching relevant studies in software engineering," in *14th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2010, pp. 1-10.

[127] L. Li *et al.*, "Static analysis of android apps: A systematic literature review," *Information and Software Technology*, vol. 88, pp. 67-95, 2017.

[128] P. Kong, L. Li, J. Gao, K. Liu, T. F. Bissyandé, and J. Klein, "Automated testing of android apps: A systematic literature review," *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 45-66, 2018.

[129] Y. Chen, "Convolutional neural network for sentence classification," University of Waterloo, 2015.

[130] J. Liu, Y. Cao, C.-Y. Lin, Y. Huang, and M. Zhou, "Low-quality product review detection in opinion summarization," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[131] J. Jin and Y. Liu, "How to interpret the helpfulness of online product reviews: bridging the needs between customers and designers," in *Proceedings of the 2nd international workshop on Search and mining user-generated contents*, 2010, pp. 87-94: ACM.

[132] D. Chen, D. Zhang, and A. Liu, "Intelligent Kano classification of product features based on customer reviews," *CIRP Annals*, vol. 68, no. 1, pp. 149-152, 2019.

[133] A. J. Ko, B. A. Myers, and D. H. Chau, "A linguistic analysis of how people describe software problems," in *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on*, 2006, pp. 127-134: IEEE.

[134] A. Reyes-Menendez, J. R. Saura, and J. G. Martinez-Navalon, "The impact of e-WOM on hotels management reputation: exploring tripadvisor review credibility with the ELM model," *IEEE Access*, vol. 7, pp. 68868-68877, 2019.

[135] Z. Zhang, J. Qi, and G. Zhu, "Mining Customer Requirement from Helpful Online Reviews," in *Enterprise Systems Conference (ES), 2014*, 2014, pp. 249-254: IEEE.

[136] M. J. Muller, J. H. Haslwanter, and T. Dayton, "Participatory practices in the software lifecycle," in *Handbook of human-computer interaction*: Elsevier, 1997, pp. 255-297.

[137] H. Barki and J. Hartwick, "Rethinking the concept of user involvement," *MIS quarterly*, pp. 53-63, 1989.

[138] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 308-318: ACM.

[139] J. Wen, G. Zhang, H. Zhang, W. Yin, and J. Ma, "Speculative text mining for document-level sentiment classification," *Neurocomputing*, vol. 412, pp. 52-62, 2020.

[140] B. Ives and M. H. Olson, "User involvement and MIS success: A review of research," *Management science*, vol. 30, no. 5, pp. 586-603, 1984.

[141] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?," *IEEE Transactions on Software Engineering*, vol. 36, no. 5, pp. 618-643, 2010.

[142] D. Schuler and A. Namioka, *Participatory design: Principles and practices*. CRC Press, 1993.

[143] T. Hou, B. Yannou, Y. Leroy, and E. Poirson, "Mining customer product reviews for product development: A summarization process," *Expert Systems with Applications*, vol. 132, pp. 141-150, 2019.

[144] A. J. Ko, M. J. Lee, V. Ferrari, S. Ip, and C. Tran, "A case study of post-deployment user feedback triage," in *proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering*, 2011, pp. 1-8.

[145] N. Seyff, F. Graf, and N. Maiden, "Using mobile re tools to give end-users their own voice," in *2010 18th IEEE International Requirements Engineering Conference*, 2010, pp. 37-46: IEEE.

[146] M. Bano and D. Zowghi, "User involvement in software development and system success: a systematic literature review," in *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, 2013, pp. 125-130.

[147] Z. Kurtanović and W. Maalej, "On user rationale in software engineering," *Requirements Engineering*, vol. 23, no. 3, pp. 357-379, 2018.

[148] G. Williams and A. Mahmoud, "Mining twitter feeds for software user requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 1-10: IEEE.

[149] A. Heydari, M. Tavakoli, Z. Ismail, and N. Salim, "Leveraging quality metrics in voting model based thread retrieval," *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, no. 1, pp. 117-123, 2016.

[150] S. Lee and J. Y. Choeh, "Predicting the helpfulness of online reviews using multilayer perceptron neural networks," *Expert Systems with Applications*, vol. 41, no. 6, pp. 3041-3046, 2014.

[151] Z. Ismail, A. Heydari, M. Tavakoli, and N. Salim, "Incorporating Author's Activeness in Online Discussion in Thread Retrieval Model," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 2, pp. 473-479, 2015.

[152] M. Salehan and D. J. Kim, "Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics," *Decision Support Systems*, vol. 81, pp. 30-40, 2016.

[153] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst1990.

[154] A. Guzzi, A. Begel, J. K. Miller, and K. Nareddy, "Facilitating enterprise software developer communication with CARES," in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, 2012, pp. 527-536: IEEE.

[155] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. Van Deursen, "Communication in open source software development mailing lists," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, 2013, pp. 277-286: IEEE.

[156] E. Von Hippel, "Lead users: a source of novel product concepts," *Management science*, vol. 32, no. 7, pp. 791-805, 1986.

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

[158] E. Goldberg, "Bug writing guidelines."

[159] S. Sen and D. Lerman, "Why are you telling me this? An examination into negative consumer reviews on the web," *Journal of interactive marketing*, vol. 21, no. 4, pp. 76-94, 2007.

[160] X. Bai, "Predicting consumer sentiments from online text," *Decision Support Systems*, vol. 50, no. 4, pp. 732-742, 2011.

[161] A. Ghose and P. G. Ipeirotis, "Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics," *IEEE transactions on knowledge and data engineering*, vol. 23, no. 10, pp. 1498-1512, 2011.

[162] R. M. Schindler and B. Bickart, "Perceived helpfulness of online consumer reviews: The role of message content and style," *Journal of Consumer Behaviour*, vol. 11, no. 3, pp. 234-243, 2012.

[163] R. Barbado, O. Araque, and C. A. Iglesias, "A framework for fake review detection in online consumer electronics retailers," *Information Processing & Management*, vol. 56, no. 4, pp. 1234-1244, 2019.

[164] D. L. Morgan and M. T. Spanish, "Focus groups: A new tool for qualitative research," *Qualitative sociology*, vol. 7, no. 3, pp. 253-270, 1984.

[165] D. L. Morgan, *Focus groups as qualitative research*. Sage publications, 1996.

[166] J. Kitzinger, "The methodology of focus groups: the importance of interaction between research participants," *Sociology of health & illness*, vol. 16, no. 1, pp. 103-121, 1994.

[167] M. Bloor, *Focus groups in social research*. Sage, 2001.

[168] J.-E. Asbury, "Overview of focus group research," *Qualitative health research*, vol. 5, no. 4, pp. 414-420, 1995.

[169] M. D. C. Tongco, "Purposive sampling as a tool for informant selection," *Ethnobotany Research and applications*, vol. 5, pp. 147-158, 2007.

[170] M. Skovdal and F. Cornish, "Qualitative research for development," *Rugby: Practical Action Publishing*, 2015.

[171] R. A. Krueger, *Focus groups: A practical guide for applied research*. Sage publications, 1994.

[172] R. A. Krueger, *Focus groups: A practical guide for applied research*. Sage publications, 2014.

[173] D. L. Morgan and R. A. Krueger, "When to use focus groups and why," *Successful focus groups: Advancing the state of the art*, vol. 1, pp. 3-19, 1993.

[174] J. Cameron, "Focusing on the focus group," *Qualitative research methods in human geography*, vol. 2, no. 8, pp. 116-132, 2005.

[175] J. T. Bertrand, J. E. Brown, and V. M. Ward, "Techniques for analyzing focus group data," *Evaluation review*, vol. 16, no. 2, pp. 198-209, 1992.

[176] S. Elo and H. Kyngäs, "The qualitative content analysis process," *Journal of advanced nursing*, vol. 62, no. 1, pp. 107-115, 2008.

[177] Y. Zhang, "Incorporating phrase-level sentiment analysis on textual reviews for personalized recommendation," in *Proceedings of the eighth ACM international conference on web search and data mining*, 2015, pp. 435-440: ACM.

[178] E. Loper and S. Bird, "NLTK: the natural language toolkit," *arXiv preprint cs/0205028*, 2002.

[179] Y. Goldberg, "A primer on neural network models for natural language processing," *Journal of Artificial Intelligence Research*, vol. 57, pp. 345-420, 2016.

[180] S. Poria, E. Cambria, and A. Gelbukh, "Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 2539-2544.

[181] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of machine learning research*, vol. 12, no. ARTICLE, p. 2493− 2537, 2011.

[182] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer, "Polisis: Automated analysis and presentation of privacy policies using deep learning," in *27th {USENIX} Security Symposium ({USENIX} Security 18)*, 2018, pp. 531-548.

[183] F. Liu, N. L. Fella, and K. Liao, "Modeling language vagueness in privacy policies using deep neural networks," *arXiv preprint arXiv:1805.10393*, 2018.

[184] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[185] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146-162, 1954.

[186] Y. Goldberg, *Neural network methods in natural language processing*. Morgan & Claypool Publishers, 2017.

[187] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137-1155, 2003.

[188] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[189] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," *arXiv preprint arXiv:1606.01781*, 2016.

[190] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning*. MIT press Massachusetts, USA:, 2017.

[191] Y. Zhang and B. Wallace, "A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification," *arXiv preprint arXiv:1510.03820*, 2015.

[192] J. Brownlee, *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Machine Learning Mastery, 2017.

[193] K. Dashtipour, M. Gogate, J. Li, F. Jiang, B. Kong, and A. Hussain, "A hybrid Persian sentiment analysis framework: Integrating dependency grammar based rules and deep neural networks," *Neurocomputing*, vol. 380, pp. 1-10, 2020.

[194] S. Poria, E. Cambria, G. Winterstein, and G.-B. Huang, "Sentic patterns: Dependency-based rules for concept-level sentiment analysis," *Knowledge-Based Systems*, vol. 69, pp. 45-63, 2014.

[195] A. Boujelben and I. Amous, "A New Method For Rules Dependency Extraction," *Procedia Computer Science*, vol. 126, pp. 860-869, 2018.

[196] P. Gamallo and M. Garcia, "Dependency parsing with finite state transducers and compression rules," *Information Processing & Management*, vol. 54, no. 6, pp. 1244-1261, 2018.

[197] K. Toutanova and C. Manning, "Enriching the knowledge sources used in a maximum entropy part-of-speech tagger," in *Proceedings of the 2000 Joint SIGDAT Conference EMNLP/VLC, 63-71, 2000*, 2000.

[198] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*, 2003, pp. 173-180: Association for Computational Linguistics.

[199] P. Qi, Y. Zhang, Y. Zhang, J. Bolton, and C. D. Manning, "Stanza: A python natural language processing toolkit for many human languages," *arXiv preprint arXiv:2003.07082*, 2020.

[200] M.-C. De Marneffe *et al.*, "Universal Stanford dependencies: A cross-linguistic typology," in *LREC*, 2014, vol. 14, pp. 4585-4592.

[201] M.-C. De Marneffe and C. D. Manning, "Stanford typed dependencies manual," Technical report, Stanford University2008.

[202] J. Hirschberg and C. D. Manning, "Advances in natural language processing," *Science*, vol. 349, no. 6245, pp. 261-266, 2015.

[203] J. Yu, J. Jiang, and R. Xia, "Global inference for aspect and opinion terms co-extraction based on multi-task neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 1, pp. 168-177, 2018.

[204] K. Liu, L. Xu, and J. Zhao, "Co-extracting opinion targets and opinion words from online reviews based on the word alignment model," *IEEE Transactions on knowledge and data engineering*, vol. 27, no. 3, pp. 636-650, 2014.

[205] Y. Wu, Q. Zhang, X.-J. Huang, and L. Wu, "Phrase dependency parsing for opinion mining," in *Proceedings of the 2009 conference on empirical methods in natural language processing*, 2009, pp. 1533-1541.

[206] M. Hu and B. Liu, "Mining opinion features in customer reviews," in *AAAI*, 2004, vol. 4, no. 4, pp. 755-760.

[207] N. Gupta and S. Chandra, "Product Feature Discovery and Ranking for Sentiment Analysis from Online Reviews," 2013.

[208] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 168-177.

[209] H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva, "Getting more out of biomedical documents with GATE's full lifecycle open source text analytics," *PLoS computational biology*, vol. 9, no. 2, 2013.

[210] R. Artstein and M. Poesio, "Inter-coder agreement for computational linguistics," *Computational Linguistics*, vol. 34, no. 4, pp. 555-596, 2008.

[211] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557-572, 1999.

[212] J. Dąbrowski, E. Letier, A. Perini, and A. Susi, "Finding and analyzing app reviews related to specific features: A research preview," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, 2019, pp. 183-189: Springer.

[213] N. Miloševic, "A multi-layered approach to information extraction from tables in biomedical documents," 2018.

[214] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649-657.

[215] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for natural language processing," *arXiv preprint arXiv:1606.01781*, vol. 2, 2016.

[216] E. Guzman, R. Alkadhi, and N. Seyff, "A Needle in a Haystack: What Do Twitter Users Say about Software?," in *Requirements Engineering Conference (RE), 2016 IEEE 24th International*, 2016, pp. 96-105: IEEE.

[217] M. Nayebi, H. Cho, and G. Ruhe, "App store mining is not enough for app improvement," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2764-2794, 2018.

[218] J. Tizard, H. Wang, L. Yohannes, and K. Blincoe, "Can a Conversation Paint a Picture? Mining Requirements in Software Forums," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 17-27: IEEE.

Appendix I

List of the selected papers for SLR

Study ID	Citation	Title	Venue
S1	(Platzer et al. 2011)	Opportunities of automated motive-based user review analysis in the context of mobile app acceptance	Central European Conference on Information and Intelligent Systems
S2	(Harman et al. 2012)	App store mining and analysis: MSR for app stores	Proceedings of the 9th IEEE Working Conference on Mining Software Repositories
S3	(Hoon et al. 2012)	A preliminary analysis of vocabulary in mobile app user reviews	Proceedings of the 24th Australian Computer-Human Interaction Conference
S4	(Vasa et al. 2012)	A preliminary analysis of mobile app user reviews	Proceedings of the 24th Australian Computer-Human Interaction Conference
S5	(Fu et al. 2013)	Why people hate your app: Making sense of user feedback in a mobile app store	Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining
S6	(Ha & Wagner. 2013)	Do android users write about electric sheep? examining consumer reviews in google play	Consumer Communications and Networking Conference
S7	(Iacob et al. 2013)	What are you complaining about?: a study of online reviews of mobile applications	Proceedings of the 27th International BCS Human Computer Interaction Conference
S8	(Khalid 2013)	On identifying user complaints of iOS apps	35th International Conference on Software Engineering
S9	(Oh et al. 2013)	Facilitating developer-user interactions with mobile app review digests	CHI'13 Extended Abstracts on Human Factors in Computing Systems

Study ID	Citation	Title	Venue
S10	(Galvis & Winbladh. 2013)	Analysis of user comments: an approach for software requirements evolution	Proceedings of the International Conference on Software Engineering
S11	(Iacob & Harrison. 2013)	Retrieving and analyzing mobile apps feature requests from online reviews	10th IEEE Working Conference on Mining Software Repositories
S12	(Pagano & Maalej. 2013)	User feedback in the appstore: An empirical study	21st IEEE international requirements engineering conference
S13	(Chen et al. 2014)	AR-miner: mining informative reviews for developers from mobile app marketplace	Proceedings of the 36th ACM International Conference on Software Engineering
S14	(Guzman & Maalej. 2014)	How do users like this feature? a fine grained sentiment analysis of app reviews	22nd IEEE international requirements engineering conference
S15	(Khalid et al. 2014)	What do mobile app users complain about?	IEEE Software
S16	(Gu & Kim et al. 2015)	What Parts of Your Apps are Loved by Users?	30th IEEE/ACM International Conference on Automated Software Engineering
S17	(Guzman et al. 2015)	Retrieving diverse opinions from app reviews.	Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on, IEEE.
S18	(Guzman et al. 2015)	Ensemble methods for app review classification: An approach for software evolution	Automated Software Engineering
S19	(Khalid et al. 2015)	Towards improving the quality of mobile app reviews	International Journal of Information Technology and Computer Science
S20	(Lee et al. 2015)	They'll Know It When They See It: Analyzing Post-Release Feedback from the Android Community	InProc 21st Amer Conf. Info. Sys.-AMCIS. Puerto Rico
S21	(Maalej & Nabil. 2015)	Bug report, feature request, or simply praise? on automatically classifying app reviews	23rd IEEE international requirements engineering conference

Study ID	Citation	Title	Venue
S22	(Moghaddam et al. 2015)	Beyond sentiment analysis: mining defects and improvements from customer feedback	European Conference on Information Retrieval, Springer.
S23	(Palomba et al. 2015)	User reviews matter! tracking crowdsourced reviews to support evolution of successful apps	IEEE International Conference on Software Maintenance and Evolution
S24	(Vu et al. 2015)	Mining User Opinions in Mobile App Reviews: A Keyword-Based Approach	30th IEEE/ACM International Conference on Automated Software Engineering
S25	(Yang & Liang. 2015)	Identification and Classification of Requirements from App User Reviews	ACM International conference on Software engineering and knowledge engineering
S26	(Panichella et al. 2015)	How can i improve my app? classifying user reviews for software maintenance and evolution	IEEE International Conference on Software Maintenance and Evolution
S27	(Di Sorbo et al. 2015)	What would users change in my app? summarizing app reviews for recommending software changes	Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering
S28	(Genc-Nayebi & Abran . 2016)	A Systematic Literature Review: Opinion Mining Studies from Mobile App Store User Reviews	Journal of Systems and Software
S29	(Maalej et al. 2016)	On the automatic classification of app reviews	Journal of Requirements Engineering
S30	(Martin et al. 2016)	A survey of app store analysis for software engineering	IEEE Transactions on Software Engineering
S31	(McIlroy et al. 2016)	Analysing and automatically labelling the types of user issues that are raised in mobile app reviews	Empirical Software Engineering
S32	(Panichella et al. 2016)	ARdoc: app reviews development-oriented classifier.	Proceedings of 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering

Study ID	Citation	Title	Venue
S33	(Jha & Mahmoud. 2017)	Mining User Requirements from Application Store Reviews Using Frame Semantics	International Working Conference on Requirements Engineering: Foundation for Software Quality
S34	(Jha & Mahmoud. 2017)	MARC: A Mobile Application Review Classifier	International Working Conference on Requirements Engineering: Foundation for Software Quality
S35	(Al-Hawari et al. 2020)	Classification of application reviews into software maintenance tasks using data mining techniques	Software Quality Journal,
S36	(Al Kilani et al. 2019)	Automatic Classification of Apps Reviews for Requirement Engineering: Exploring the Customers Need from Healthcare Applications	Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS).
S37	(Guo, Hui and Munindar P Singh. 2020	Caspar: extracting and synthesizing user stories of problems from app reviews	Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering
S38	(Haroon et al. 2020)	Comparing mobile apps by identifying 'Hot' features	Future Generation Computer Systems
S39	(Gao et al. 2018)	Online app review analysis for identifying emerging issues	Proceedings of the 40th International Conference on Software Engineering
S40	(Gao et al. 2019)	Emerging app issue identification from user feedback: experience on wechat	IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)
S41	(Martijn et al. 2020)	Identifying and Classifying User Requirements in Online Feedback via Crowdsourcing	International Working Conference on Requirements Engineering: Foundation for Software Quality
S42	(Tao et al. 2020)	Identifying security issues for mobile applications based on user review summarization	Information and Software Technology
S43	(Jha & Anas 2018)	Using frame semantics for classifying and summarizing application store reviews	Empirical Software Engineering

Study ID	Citation	Title	Venue
S44	(Jha & Anas 2019)	Mining non-functional requirements from App store reviews	Empirical Software Engineering
S45	(Guo et al. 2018)	Mining and Analyzing User Feedback from App Reviews: An Econometric Approach	SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI
S46	(Wang et al. 2020)	Opinion Analysis and Organization of Mobile Application User Reviews	REFSQ Workshops
S47	(Buchan et al. 2018)	Semi-automated extraction of new requirements from online reviews for software product evolution	25th Australasian Software Engineering Conference (ASWEC)
S48	(Noei et al. 2019)	Too Many User-Reviews, What Should App Developers Look at First?	IEEE Transactions on Software Engineering
S49	(Truelove et al. 2019)	Topics of concern: identifying user issues in reviews of IoT apps and devices	IEEE/ACM 1st International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)
S50	(Zhou et al. 2020)	User review-based change file localization for mobile applications	IEEE Transactions on Software Engineering
S51	(Zhang et al. 2019)	Software feature refinement prioritization based on online user review mining	Information and Software Technology
S52	(Stanik et al. 2019)	Classifying multilingual user feedback using traditional machine learning and deep learning	IEEE 27th International Requirements Engineering Conference Workshops (REW)
S53	(Poria et al. 2014)	A rule-based approach to aspect extraction from product reviews	Proceedings of the second workshop on natural language processing for social media (SocialNLP)

Study ID	Citation	Title	Venue
S54	(Poria et al. 2016)	Aspect extraction for opinion mining with a deep convolutional neural network	Knowledge-Based Systems
S55	(Rana & Cheah, 2017)	A two-fold rule-based model for aspect extraction	Expert systems with applications
S56	(Rana & Cheah, 2015)	Hybrid rule-based approach for aspect extraction and categorization from customer reviews	9th International Conference on IT in Asia (CITA)

Appendix II

Number of the papers published in each channel

Publication Channel	Number of papers	Study ID(s)
IEEE/ACM International Conference on Software Engineering (ICSE)	6	S8, S10, S13, S37, S39, S40
IEEE international requirements engineering conference	4	S12, S14, S21, S52
International Working Conference on Requirements Engineering: Foundation for Software Quality	4	S33, S34, S41, S46
IEEE/ACM International Conference on Automated Software Engineering (ASE)	3	S16, S18, S24
IEEE Transactions on Software Engineering	3	S30, S48, S50
Empirical Software Engineering	3	S31, S43, S44
IEEE Working Conference on Mining Software Repositories	2	S2, S11
Australian Computer-Human Interaction Conference	2	S3, S4
IEEE International Conference on Software Maintenance and Evolution	2	S23, S26
ACM SIGSOFT International Symposium on Foundations of Software Engineering	2	S27, S32
Information and Software Technology	2	S42, S51
The Journal of Society for e-Business Studies	1	S1
ACM SIGKDD international conference on Knowledge discovery and data mining	1	S5
Consumer Communications and Networking Conference	1	S6
International BCS Human Computer Interaction Conference	1	S7
CHI'13 Extended Abstracts on Human Factors in Computing Systems	1	S9
IEEE Software	1	S15
ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)	1	S17
International Journal of Information Technology and Computer Science	1	S19
Australasian Software Engineering Conference (ASWEC)	1	S47
Amer Conf. Info. Sys.-AMCIS. Puerto Rico	1	S20
European Conference on Information Retrieval, Springer.	1	S22
International Conference on Social Networks Analysis, Management and Security (SNAMS)	1	S36
ACM International conference on Software engineering and knowledge engineering	1	S25
Journal of Systems and Software	1	S28
Journal of Requirements Engineering	1	S29
Knowledge-Based Systems	1	S54
Future Generation Computer Systems	1	S38
Software Quality Journal	1	S35

Publication Channel	Number of papers	Study ID(s)
workshop on natural language processing for social media (SocialNLP)	1	S53
IEEE/ACM International Workshop on Software Engineering Research & Practices for the Internet of Things (SERP4IoT)	1	S49
International Conference on IT in Asia (CITA)	1	S56
SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI	1	S45
Expert systems with applications	1	S55

APPENDIX III

Mobile app development topics discovered in the 56 selected studies.

Topic Category	Specific Topic	Study Id
General Comment	Entirely reviewing the app, Works/Doesn't work	S6
	Positive	S7
	Helpfulness, Praise	S12
	Rating	S21
	Feature strength, General Praise	S18
	Descriptions	S13
Negative comment	Works/Doesn't work - Uninstalled	S6
	Negative	S7
	Shortcoming, Dispraise, Dissuasion	S12
	General complaint	S13, S18
	Compatibility, App Crashing, Network Problem, Interface Design, Privacy and Ethical, Response Time, Uninteresting Content, Resource Heavy	S8, S31
App Comparison	Comparison with other apps	S6
	Comparative	S7
	Other app	S12
Issue Report	Feature/Functionality	S6
	Issue reporting	S7
	Content request, Feature, Bug report	S12
	Functional bug	S9, S31
	Performance flaw	S13
	Functional error, Feature Removal	S8, S13
	Bug report	S18, S21, S22, S33, S34
	Solution proposal	S26, S32
	Problem discovery	S26, S27, S32
	Feature shortcoming	S18
Feature Request	Aesthetics	S6
	Request for requirements	S7
	Improvement request	S12, S22
	Promise better rate for improvement	S12
	Functional demand, Non-Functional request	S9
	Feature Request	S8, S11, S12, S21, S26, S27, S32
	User Request	S18
	User Requirements	S33, S34
User Experiences	Tips (installation/usage)	S6
	Usability	S7
	Question (How to use)	S12, S13
	User experiences	S21
	Information Seeking, Information Giving	S26, S27, S32

	Opinion asking	S26, S32
	User scenario	S18
Updates	Updates (comparing to previous version)	S6
	Update issues	S31
	Versioning	S7
Price	Money (worth the money)	S6
	Price related	S7
	Hidden Cost	S8
	Additional Cost	S31
Recommendation	Recommending the app	S6
	Customer support	S7
	Recommendation	S12
Other	Additional Program needed, Number and content of ads, Company, Just downloaded, Not used yet, Device model, Permissions, Preinstalled app, Consumption of resources	S6

Appendix IIII

List of POS tags [3] used in this study

No.	Tag	Description
1	CC	Coordinating conjunction
2	CD	Cardinal number
3	DT	Determiner
4	EX	Existential *there*
5	FW	Foreign word
6	IN	Preposition or subordinating conjunction
7	JJ	Adjective
8	JJR	Adjective, comparative
9	JJS	Adjective, superlative
10	LS	List item marker
11	MD	Modal
12	NN	Noun, singular or mass
13	NNS	Noun, plural
14	NNP	Proper noun, singular
15	NNPS	Proper noun, plural
16	PDT	Predeterminer
17	POS	Possessive ending
18	PRP	Personal pronoun
19	PRP$	Possessive pronoun
20	RB	Adverb
21	RBR	Adverb, comparative
22	RBS	Adverb, superlative
23	RP	Particle
24	SYM	Symbol
25	TO	*to*
26	UH	Interjection
27	VB	Verb, base form
28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle
30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present

[3] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

No.	Tag	Description
32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner
34	WP	Wh-pronoun
35	WP$	Possessive wh-pronoun
36	WRB	Wh-adverb