Detection of Spam Review on Mobile App Stores, Evaluation of Helpfulness of User Reviews and Extraction of Quality Aspects Using Machine Learning Techniques

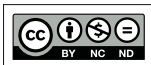
by

Necmiye GENC

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, "JUNE 04, 2019"

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC



BOARD OF EXAMINERS

THIS THESIS HAS BEEN EVALUATED

BY THE FOLLOWING BOARD OF EXAMINERS

Mr. Alain Abran, Thesis Supervisor
Department of Software and IT Engineering, École de technologie supérieure

Mr. Marc Thomas, President of the Board of Examiners
Department of Mechanical Engineering, École de technologie supérieure

Mr. François Coallier, Member of the jury
Department of Software and IT Engineering, École de technologie supérieure

Mr Mehmet Gumus, External Independent Examiner
Operations Management at the Desautels Faculty of Management, Mcgill University

THIS THESIS WAS PRESENTED AND DEFENDED

IN THE PRESENCE OF A BOARD OF EXAMINERS AND THE PUBLIC

ON "MAY 15, 2019"

AT ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

ACKNOWLEDGEMENTS

I dedicate this work to my 2 years old daughter, Grace who was named after Grace Hopper, one of the first programmers of the Harvard Mark I computer. Grace Hopper was a pioneer of computer programming and invented a computer language compiler, which led to the popular COBOL language. I hope one day, she becomes a great scientist as her namesake and changes the world and makes it a much better place.

I must express very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study. None of my accomplishment would have been possible without their support.

I appreciate deeply my thesis director Dr. Alain Abran who has inspired me with his intelligence and scientific insight. I am sincerely grateful to him for sharing his truthful and illuminating views on a number of issues related to the research. With his invaluable comments and directives, I was able to successfully conclude this study. He will be the one to follow as my role model in my future academic life.

Last but not the least, I would like to thank my husband and research partner Fatih Nayebi for his support and friendship throughout my Ph.D. studies. I am thankful for his inspiring guidance and invaluably constructive criticism through the process of research and writing this thesis.

Détection, sur les plates-formes de distribution des applications mobiles, des opinions frauduleuses, évaluation de l'utilité des avis des utilisateurs et extraction des aspects qualité en utilisant des techniques d'apprentissage automatique

Necmiye GENC

RÉSUMÉ

Alors que les appareils mobiles ont dépassé l'accès Internet fixe, les applications mobiles et les plates-formes de distribution ont pris de l'importance. Ces plates-forme de distribution permettent aux utilisateurs de rechercher et d'acheter des applications mobiles, puis de donner leurs opinions sous forme d'avis et de notes. Un avis peut contenir des informations critiques sur l'expérience utilisateur, les demandes de fonctionnalités et les rapports de bogues. Les avis des utilisateurs sont précieux non seulement pour les développeurs et les éditeurs de logiciels intéressés à connaître l'opinion de leurs clients, mais également pour les utilisateurs potentiels désireux de savoir ce que les autres pensent de l'application mobile.

Bien que certains chercheurs aient répertorié les techniques et méthodes d'analyse d'opinion, aucune étude systématique de la littérature n'a encore fait état de problèmes d'extraction d'opinion et de détection des opinions frauduleuses dans une plate-forme de distribution d'applications mobiles.

L'extraction d'opinions nécessite un pré-traitement au niveau du texte et du contenu, y compris le filtrage du contenu sans opinion et l'évaluation de la fiabilité et de l'authenticité des critiques. La prédiction de l'utilité des applications mobiles et les problèmes de détection des opinions frauduleuses n'ont guère retenu l'attention de la littérature universitaire. De plus, la pertinence des fonctionnalités extraites n'a pas fait l'objet d'une validation croisée avec les principaux concepts de génie logiciel.

Ce projet de recherche a d'abord fait une revue systématique de la littérature sur l'évaluation des études d'extraction des opinions sur les plates-formes de distribution d'applications mobiles. Pour combler les lacunes identifiées dans cette revue de littérature, nous avons ensuite utilisé un réseau de neurones convolutifs pour apprendre à représenter des documents pour la détection de révisions frauduleuses en caractérisant un jeu de données d'avis d'utilisateurs sur des plates-formes de distribution d'applications mobiles, ce qui inclut des analyses de vérités et de fraudes. Nos résultats ont révélé que notre méthode de détection basée sur un réseau de neurones atteignait une précision de 82,5%, tandis qu'un modèle de classification de type Machine Support Vector Machine (SVM) n'atteint que 70% de précision, malgré l'utilisation de diverses combinaisons de fonctions.

Nous avons ensuite comparé quatre modèles de classification afin d'évaluer l'utilité des avis des utilisateurs d'une plate-forme de distribution pour proposer un modèle prédictif reposant sur des méta-données des avis, ainsi que sur des caractéristiques structurelles et lexicales pour la prédiction de l'utilité fonctionnelle de ces avis.

Dans la dernière partie de cette étude, nous avons construit un jeu de données annotées des opinions sur des applications mobiles pour la tâche d'extraction d'aspects de qualité, basé sur la norme ISO 25010 - Normes d'évaluation de la qualité des produits et logiciels. Nous avons ensuite utilisé deux modèles de réseau neuronal profond pour l'extraction d'aspects à partir des avis d'utilisateurs: bidirectionnel long-court avec mémoire de termes et champ aléatoire conditionnel (Bi-LSTM + CRF) et réseaux de neurones à structure profonde et champ aléatoire conditionnel (CNN + CRF). Les deux modèles ont obtenu un score F_1 de près de 80% (moyenne pondérée de la précision et du rappel, qui prend en compte à la fois les faux positifs et les faux négatifs) avec l'appariement exact d'aspects, et un score F1 de 86% dans l'appariement partiel.

Mots-clés: Ingénierie des exigences, apprentissage automatique, exploration de textes, applications mobiles, détection de spam, évaluation de l'utilité des revues, l'extraction des aspects qualité

Detection of Spam Review on Mobile App Stores, Evaluation of Helpfulness of User Reviews and Extraction of Quality Aspects Using Machine Learning Techniques

Necmiye GENC

ABSTRACT

As mobile devices have overtaken fixed Internet access, mobile applications and distribution platforms have gained in importance. App stores enable users to search and purchase mobile applications and then to give feedback in the form of reviews and ratings. A review might contain critical information about user experience, feature requests and bug reports. User reviews are valuable not only to developers and software organizations interested in learning the opinion of their customers but also to prospective users who would like to find out what others think about an app.

Even though some surveys have inventoried techniques and methods in opinion mining and sentiment analysis, no systematic literature review (SLR) study had yet reported on mobile app store opinion mining and spam review detection problems. Mining opinions from app store reviews requires pre-processing at the text and content levels, including filtering-out non-opinionated content and evaluating trustworthiness and genuineness of the reviews. In addition, the relevance of the extracted features are not cross-validated with main software engineering concepts.

This research project first conducted a systematic literature review (SLR) on the evaluation of mobile app store opinion mining studies. Next, to fill the identified gaps in the literature, we used a novel convolutional neural network to learn document representation for deceptive spam review detection by characterizing an app store review dataset which includes truthful and spam reviews for the first time in the literature. Our experiments reported that our neural network based method achieved 82.5% accuracy, while a baseline Support Vector Machine (SVM) classification model reached only 70% accuracy despite leveraging various feature combinations.

We next compared four classification models to assess app store user review helpfulness and proposed a predictive model which makes use of review meta-data along with structural and lexical features for helpfulness prediction.

In the last part of this research study, we constructed an annotated app store review dataset for the aspect extraction task, based on ISO 25010 - Systems and software Product Quality Requirements and Evaluation standard and two deep neural network models: Bi-directional Long-Short Term Memory and Conditional Random Field (Bi-LSTM+CRF) and Deep Convolutional Neural Networks and Conditional Random Field (CNN+CRF) for aspect extraction from app store user reviews. Both models achieved nearly 80% F_1 score (the weighted average of precision and recall which takes both false positives and false negatives into account) in exact aspect matching and 86% F_1 score in partial aspect matching.

Keywords: Requirement engineering, machine learning, text mining, mobile applications, review spam detection, review helpfulness, extraction of quality aspects

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

LIST OF ABREVIATIONS

AMT	Amazon Mechanical Turk
ANOVA	Analysis of Variance
API	Application Program Interface
ASUM	Aspect and Sentiment Unification Model
AUC	Area Under the Curve
Bi-LSTM	Bidirectional Long-Short Term Memory
CART	Classification and Regression Tree
CBOW	Continuous Bag of Words
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRF	Conditional Random Field
ELMo	Embeddings from Language Models
FR	Functional Requirement
GP	Google Play
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HTML	Hypertext Markup Language
iOS	iPhone Operating System (Apple)
ISO/IEC	International Standard Organization and the International Electrotechnical Commission

JSON	JavaScript Object Notation
LDA	Latent Dirichlet Allocation
LIWC	Linguistic Inquiry and Word Count
LSA	Latent Semantic Analysis
LSTM	Long-Short Term Memory
MAP	Maximum A Posteriori
ME	Maximum Entropy
NFR	Non-Functional Requirement
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PhD	Doctor of Philosophy
POS	Part Of Speech
ROC	Receiver Operating Characteristic
RQ	Research Question
RSS	Rich Site Summary
SLR	Systematic Literature Research
SVM	Support Vector Machine
UI	User Interface
XML	Extensible Markup Language

INTRODUCTION

Customers now buy a very large number of goods and services directly from online websites and digital distribution platforms. They often rely on others' reviews or recommendations, either from online purchase web sites or review sites, to finalize their purchasing decisions. Given the sheer volume of reviews, reading and interpreting them all and analyzing the opinions and sentiments within the text would be time consuming and, sometimes, deceptive for end-users because of low-quality and opinion spam reviews.

Mobile devices bring significant advantages to their users in terms of portability, location awareness and accessibility. Improvements in hardware and software capabilities of hand-held devices and decreases in prices have led to a huge expansion of such devices and related markets. Individual developers or software organizations can easily distribute their applications via app stores. All major mobile operating system vendors, including Apple, Google and Microsoft, run their own app stores that give them control over the applications on their software distribution platforms. The increasing demand for various kinds of mobile apps running on different devices has led to a corresponding increase in the number of mobile developers and competition in app stores.

Searching mobile app stores is, for users, the preferred way of discovering new apps. According to Google (2015): (i) search is a driver of app discovery, and (ii) recommendations and also interest and fun level are top reasons to download an app. Given that the number of available apps in app stores is around 7 million Statista (2018), it gets more difficult for users to find the exact app they are looking for.

Ratings and reviews are therefore considered valuable tools that help users decide which apps to download. In addition, understanding user feedback, anticipating user experience and acceptance of the app may be a key success factor for mobile app developers and software organizations.

App stores provide a user feedback capability that is particularly useful and interesting in terms of software requirements evolution. User ratings and reviews are user-driven feedback that may help improve software quality and address missing application features. Besides, developers and software organizations build their app store reputation based on reviews and ratings. Hence, over the past several years, various techniques and automated systems have been proposed to mine, analyze and extract user opinions and sentiments from app store review text.

Opinion mining and sentiment analysis have long attracted researchers' interest. However, it is only after the early 2000s that it became a major part of the decision-making processes with the growing availability and popularity of opinion-rich resources such as online review sites, personal blogs and social media. The increasing interest in opinion mining and sentiment analysis is partly due to the variety of its application areas: business intelligence, recommendation systems and human-computer interactions with a wide range of problems to be solved such as lexicon creation, subjectivity and sentiment classification, aspect extraction and topic summarization.

A number of researchers have studied opinion mining since the late 90s; however, the introduction of machine learning techniques and annotated datasets such as customer review datasets Hu & Liu (2004), Ding *et al.* (2008), pros and cons dataset Ganapathibhotla & Liu (2008), Amazon product review dataset Jindal & Liu (2008) and blog author gender classification dataset Mukherjee & Liu (2010) have accelerated the research in the domain. With the emergence of different opinion mining domains such as social media (Twitter, app ecosystems, micro blogs, etc.), the focus of the studies has since shifted into short-length texts, contradiction analysis and spam detection.

Mobile app store reviews have some characteristics that set them apart from other kinds of texts, such as being short, informal and ungrammatical, consisting of incomplete sentences, elongations and abbreviations that make them difficult to handle. Traditional text mining techniques,

ranging from Part Of Speech (POS) tagging to dependency parsing, cannot be applied to this kind of short text. Besides, as short texts usually do not possess sufficient signals to support statistical text processing techniques based on term counts and term-by-document frequencies, new approaches must be introduced to better handle them.

Given the impact of user reviews on purchasing decisions and strong incentives of fraudsters to game the rating system, detecting opinion spam has become more critical than ever to protect product reputation. A deceptive opinion spam is a review with fictitious opinions that is deliberately written to appear authentic where the intention ranges from self-promotion to destroy a competitor's reputation. Within the corpus of mobile app stores, scammers use a great many bogus user accounts or computer bots in order to download applications multiple times and write fraudulent reviews. In this way, the applications begin appearing improperly on the top charts and have greater visibility in an app store search. In addition, there are numerous websites for purchasing reviews, such as Appthurst [1], AppSally [2] and Mobiaso [3].

Even though spam and deceptive reviews are widespread and may have significant manipulative effects on app store success, it is only recently that app store regulators have begun to crack down on spam reviews Ahn (2016). On the other hand, most of the time deceptive reviews are not easily identifiable by human readers; consequently, there are few good sources of labeled data for research purposes. Indeed, in the absence of a gold-standard dataset, earlier studies have had to utilize non-standard evaluation procedures Ott *et al.* (2011).

Unbiased or non-spam user reviews may be numerous but of varying quality. The terms such as 'Liked', 'Not recommend', 'OK app' do not convey any information about why users like an application or which aspects they like the most. Secondly, most reviews are poorly written and

[1] https://appthurst.com

[2] https://www.appsally.com

[3] https://mobiaso.com

the information they contain is often not useful, or highly personal and device- or technology-specific. Sophisticated ranking schemes, such as found in the Apple app store and Google Play, measure reviews by their 'helpfulness' as rated by users. In the Apple app store, the button under each user review allows other users to vote on whether the review is helpful or not: reviews may also be sorted from Most Helpful to Less Helpful based on these voting results. However, for newly written reviews or less popular applications, there would not be enough 'helpfulness' voting to be of any use.

Users prefer having comparisons of specific features of different products available rather than having to gather isolated opinions about a single product themselves. In addition to average rating on a five-star scale and corresponding ranking on the app store, users prefer learning about others' experience with the app, including which aspects/features they liked or disliked most. Each user has his/her own preferences and while one user might feel strongly about appearance, others may focus on functional or technical aspects. Hence, there is a need to extract and rate individual application features. However, to be able to make such comparisons, domain knowledge (ability to spot features) is required.

This doctoral research mine user opinions and extract targeted software features from app store user reviews for the use of both developers and end-users. To do these, we first develop:

- A model to detect opinion spam (deceptive) user reviews.
- A model to evaluate review usefulness.
- An automated system to extract application aspects for both user and developers.

These models aim to help:

- App ecosystem operators to identify spam and deceptive reviews as well as ranking frauds.
- Users to come up with an informed decision about applications.
- Developers to obtain users' feedback about most liked or expected features and bugs in mobile apps.

This research thesis is structured as follows:

- Chapter 1 summarizes the literature review results. It is based on five research questions to search the literature and evaluation criteria to select the relevant studies.
- Chapter 2 presents the problem statement, research motivation and objectives of the thesis. This chapter also presents the overall research methodology including the phases:
    1. Development of an app store crawler to populate the dataset.
    2. Construction of a model to identify opinion spam (deceptive) reviews.
    3. Development of a review helpfulness assessment model.
    4. Development of a deep learning model for aspect extraction mobile app features.
- Chapter 3 presents the details of opinion spam review detection model including:
    1. Background Information on Word Embeddings and Convolutional Neural Networks (CNN).
    2. Research Methodology.
    3. Implementation.
    4. Experiments and Results.

- Chapter 4 presents the development and the details of automatic assessment of review helpfulness, including the following steps:

    1. Feature Generation.
    2. Selection of the Model.
    3. Model Refinement.
    4. Experiments and Results.

- Chapter 5 presents the app aspect extraction task that includes:

    1. Background Information on Aspect Extraction and ISO 25010.
    2. Research Methodology.
    3. Annotation of Mobile App Aspects.
    4. Details of Aspect Extraction Models.
    5. Implementation.
    6. Experiments and Results.

- Chapter 6 presents a summary of the key contributions and suggests some future work.

CHAPTER 1

LITERATURE REVIEW

The success of the Apple app store has led to the launch of other similar stores and services, with an exponential growth both in the number of applications and revenues. Data mining and opinion aggregation from these platforms has therefore become an important research topic.

1.1 Need for a systematic review

There is quite a number of survey studies on opinion mining and sentiment analysis in the literature. Pang & Lee (2008) made a comprehensive contribution to opinion mining and sentiment analysis survey studies by covering applications, major tasks of opinion mining, extraction and summarization, sentiment classification and also the common challenges in this new research field. Tsytsarau & Palpanas (2012) surveyed the development of sentiment analysis and opinion mining research studies including spam detection and contradiction analysis. Their survey study provided 26 additional papers compared to Pang & Lee (2008)'s preliminary survey. The survey of Tang *et al.* (2009) had a narrower scope, examining the opinion mining problem only for customer reviews on the web sites that couple reviews with e-commerce like Amazon.com or from sites that specialize in collecting user reviews in a variety of areas like Rottentomates.com. Cambria *et al.* (2013) revealed the complexities involved in opinion mining with respect to current use along with future research directions.

Even though some surveys have reviewed the techniques and methods in opinion mining and sentiment analysis from text, no Systematic Literature Review (SLR) had yet reviewed the literature regarding mobile app store data mining, opinion aggregation and spam detection. Martin *et al.* (2015) provided an initial survey into the literature from 2000 to November 2015; however their survey is not a SLR and they were particularly interested in the studies that combine technical (Application Program Interface (API) usage, size, platform version, etc.) and non-technical attributes (category, rating, reviews, installs, etc.) of mobile apps.

The goal of our SLR was to methodically review and gather research results for specific research questions and to develop evidence-based guidelines for app store practitioners. We developed a set of five research questions to guide the literature review process and performed an extensive search to find publications that answer the research questions.

The objectives of this systematic literature review are to identify:

- Proposed solutions for mining online opinions in app store user reviews.
- Challenges and unsolved problems in the domain.
- Any new contributions to software requirements evolution, and
- Future research directions.

1.2 Systematic review methodology

The SLR was conducted following the guidelines of Kitchenham (2004). The activities performed in the course of the SLR were structured into three phases: (1) planning, (2) conducting the review, and (3) reporting- see Figure 1.1. The individual tasks performed in each activity are described in sub-sections 1.2.1–1.2.3.

1.2.1 Planning

The planning phase clarified the specific objectives of the SLR, that is: to identify mobile app store studies, the challenges faced when mining app store data, how these challenges have been overcome, and any unsolved challenges. In addition, we specified the following five research questions and the motivations behind the questions.

1.2.1.1 Research questions RQ)

RQ1: Which specific data mining techniques are used for reviews on software distribution platforms?

Motivation: App stores provide a wealth of information in the form of customer reviews. Opinion mining and sentiment analysis systems have been applied to various kinds of texts including newspaper headline, novels, emails, blogs, tweets and customer reviews. Different techniques and automated systems have been proposed over the years by researchers to extract user opinions and sentiments within the text over the years. Unlike documents or long length text, mobile app store reviews have some unique characteristics such as being short, informal and sometimes even ungrammatical consisting of incomplete sentences, elongations and abbreviations that make them difficult to handle. This RQ1 question targets approaches and techniques proposed particularly for app store user review mining and opinion extraction problems.

RQ2: How do the studies address the 'domain dependency' challenge for app store reviews?
Motivation: Vocabulary varies within different contexts and domains, and the same term might mean different opinions. An opinion classifier trained using opinionated words from one domain might perform poorly when it is applied to another domain: Not only the words and the phrases, but also that the language structure could differ from one domain to another. Hence, the language structure and linguistic context of opinion and sentiments terms play a key role in opinion mining: domain adaptation methods are also required to be considered while dealing with app store user reviews. This RQ2 question aims to identify how mobile app store opinion mining studies tackle the domain dependency problem.

RQ3: What criteria make a review useful?
Motivation: Quality varies from review to review and low quality reviews might not convey any signal to be used for information extraction. To tackle spam identification problem, it is critical to have a mechanism or criteria to assess the quality of reviews and to filter out low-quality or noisy reviews. While review helpfulness is assessed manually by users in mobile app stores, there also exists some automated systems to assess and rank reviews in accordance with their usefulness or helpfulness. This RQ3 question aims to identify the methods or criteria used to differentiate useful app store reviews from the others. Furthermore, this research question also searches for automated systems that evaluate review usefulness and helpfulness.

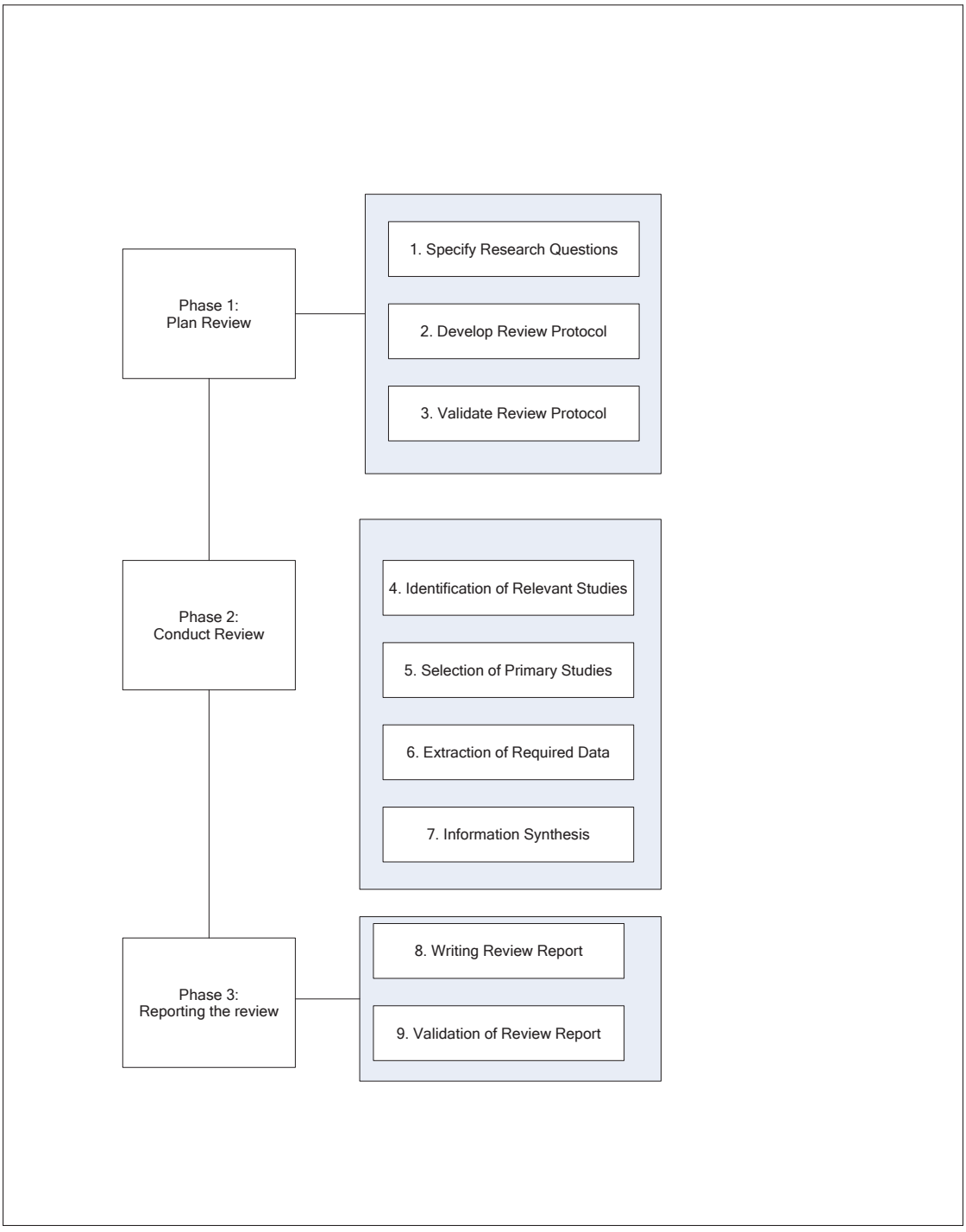


Figure 1.1 SLR Process

RQ4: How can spam reviews be differentiated from legitimate reviews?

Motivation: With the number of online reviews increasing, the number of fraudsters who

produce deceptive or untruthful reviews increases as well. It is an essential task to identify and filter out the opinion spam. Different studies and techniques have been proposed for the spam review detection problem. The opinion spam identification task has great impact on industrial and academia communities. Our objective with this RQ4 research question is to investigate spam review and ranking fraud detection methods and techniques for online stores and mobile app stores.

RQ5: Does the study extract targeted/desired software features from application reviews?
Motivation: Apart from app's average rating over a 5-star scale and its corresponding ranking in app store, users would like to learn about others' experience with the app and which aspect-s/features they liked or disliked most. The information obtained from mobile app reviews is also valuable for developers to get users' feedback about most liked or expected features (e.g., requirements elicitation) as well as bugs in the application (e.g., software quality and software evaluation). This RQ5 research question focuses on aspect-based opinion mining studies extracting application features and aims to identify the studies that make automated application feature extraction from user reviews.

1.2.1.2 Development and validation of the review protocol

The review protocol defines the activities required to carry out the literature review. A review protocol helps reduce researchers' biases and defines the source selection and searching processes, the quality criteria and the information synthesis strategies. This subsection presents the details of our review protocol.

The following digital libraries were used to search for primary studies:

- Science Direct
- IEEExplore
- ACM Digital Library

- Citeseer library (citeseer.ist.psu.edu)
- Springer Link
- Google Scholar

Our research covered the period between January 2010 and November 2017. The following search query was created by augmenting the keywords with possible synonyms. While conducting the review, we examined the reference list of primary studies to determine if there were additional studies not captured by our research query.

((mobile OR software) OR ((apps OR app OR application) OR (market OR ecosystem OR AppStore OR store))) AND ((data OR (online OR review) OR user OR (text OR comment OR vocabulary)) OR rating OR (opinion OR sentiment) OR (mining OR analysis OR processing) OR (feature OR requirement) OR request OR expectation OR (bug OR quality OR complain OR issue) OR (usefulness OR helpfulness))

Study Selection Procedure: We systematically selected the primary studies by applying the following four steps:

1. We examined the paper titles to eliminate studies unrelated to our research focus.
2. We reviewed the abstracts and keywords in the remaining studies. If either the abstracts or keywords did not provide the necessary information, we reviewed the results and conclusion sections to determine if the study was relevant.
3. We filtered the remaining studies in accordance with the inclusion and exclusion criteria given in Table 1.1.
4. We double-checked the reference list of the initial primary studies to identify additional studies that might be relevant to our search.

Table 1.1 SLR inclusion and exclusion criteria

Inclusion criteria	**Exclusion criteria**
Case studies and surveys of text analysis, opinion mining and sentiment analysis from app store reviews	Papers that present opinions without sufficient and reliable supporting evidence.
Preliminary analysis of mobile app store reviews, vocabulary, trends.	Studies not related to the research questions.
Papers searching application feature requests and bug reports within review text.	Papers that do not comply with the evaluation criteria in Table 1.2.
Papers that describe the criteria of what makes a review useful and helpful for readers.	Preliminary conference papers of journal papers by same author(s).
Papers that distinguish fake reviews and spams from legitimate ones.	

Table 1.2 SLR quality checklist (Adapted from Keele (2007))

No	**Question /Criteria**
1	Are the aims of the study stated clearly?
2	Is the basis of evaluative appraisal clear?
3	How defensible is the research design?
4	Are data collection methods described adequately?
5	Has the approach to, and formulation of, analysis been conveyed adequately?
6	Has the diversity of perspectives and contexts been explored
7	Has the approach to, and formulation of, analysis been conveyed adequately?
8	Is the reporting clear and coherent?
9	Has the research process been documented adequately?
10	Could the study be replicated?

We evaluated the quality of the primary studies using the checklist adapted from Keele (2007). Each study was evaluated according to the quality checklist questions given in Table 1.2. The studies that provided a ‘yes’ answer to at least seven questions from the checklist were selected.

1.2.2 Conducting the review

1.2.2.1 Identification and selection of relevant studies

We followed Wohlin (2014) snowballing procedure in order to identify relevant studies. In the first step called database search, we identified the keywords and formulated search string as given in subsection 1.2.1.2. Our research with the search query generated more than 500 hits to build up our start set. After examining the paper title, abstract, keywords, results and conclusions (if necessary) to filter out unrelated studies, 63 studies remained as the start set. We used the reference list of our start set papers to identify new papers to include. Afterwards, we went through the reference list and excluded the papers that did not fulfil the basic criteria such as title, language and publication venue. We also performed forward snowballing to identify new papers based on those papers citing the paper being examined. Each candidate citing the paper was examined by screening the information provided in Google Scholar. If this information was not sufficient for a decision, the citing paper was examined in more details. After implementing backward snowballing and forward snowballing steps, we ended up with 45 research papers. Using the inclusion and exclusion criteria and the quality checklist, the examination of the remaining literature produced 24 primary studies.

1.2.2.2 Extraction of data

We used the data extraction form in Table 1.3 to extract data from the 24 primary studies. Even though the same data items were searched with RQ1 and RQ4, opinion mining and spam analysis studies respectively, the results obtained are presented in distinct tables. See Tables 1.1 and 1.2 of Appendix I.

Table 1.3 SLR data extraction form

Search Focus	Data Item	Description
General	Identifier	Reference number given to the article
	Bibliography	Author, year, title, source
	Type of article	Journal/conference/technical report/etc.
	Study aims	Aims or goals of the study
RQ1 / RQ4	Text and data mining methods and techniques used	Algorithms, models and measures
	Selected or obtained review features	The subset of text features used or identified in the study
	Dataset	List of chosen applications, number of reviews
	Performance/Results	Precision, recall, accuracy/Obtained results
RQ2	Domain-specific text and data mining techniques used for app store reviews	App store and app review specific algorithms, methods
	Specific features used for app store reviews	App store and app review specific text features
	Performance improvement	Performance improvement compared to conventional opinion mining studies
RQ3	User review helpfulness/usefulness assessment framework	Predictors, variables, features that specify review quality
	Model used for automated usefulness task	Algorithms, models and measures
	Selected features	Subset of text features used in the study
	Performance	Precision, recall, accuracy
RQ5	Extracted app features	Mobile app features retrieved from online review text
	Method	Approaches, techniques used for automatically extracting application features
	Performance	Precision, recall, accuracy

1.2.2.3 Information synthesis

We read the 24 selected studies noting the methods and findings that were repeated. Inconsistencies and contradictions in the information were also recorded and are presented in the discussion and principal findings sections.

1.2.3 Reporting the review

Data extracted from the primary studies were used to answer our five research questions. The guidelines of Kitchenham (2004) were closely followed in the reporting of results.

1.3 Results

1.3.1 RQ1: Which specific data mining techniques are used for the reviews on software distribution platforms?

The analysis of the 24 primary studies identified a number of specific opinion mining and opinion extraction techniques used for reviews on software distribution platforms.

Chen & Liu (2011) identified useful app features:

- (i) static (e.g., application name, provider),
- (ii) dynamic (e.g., current rate, update date),
- (iii) comment (e.g., user rate, comment content)) to predict app popularity and trained a model for an automated popularity prediction task.

To create the dataset, they sampled 102,337 applications and a list of dynamic features were accumulated for the top 200 paid and free applications by tracking their daily ranking. They used a Classification and Regression Tree (CART) model as a popularity prediction model and leveraged static (app name, provider, category, etc.), dynamic (current rank, all version count, all version rate, etc.) app and app store features and also comment features (user rate, comment title and comment content). As a result, they found that the top-ranked (e.g., the search ranking on the app store that factors in average app store rating, rating/review volume, download and install counts and app usage statistics) paid applications were not closely related to customer ratings.

Vasa *et al.* (2012) and Hoon *et al.* (2012) made a preliminary analysis of mobile app user reviews. They initially analyzed the data using summary statistics with a one-way ANOVA test, box plots and cumulative distribution charts to confirm their hypothesis that rating and category have an effect on the length of the review. They analyzed 8.7 million reviews from 17,330 app. According to their analysis, users take the time to express their discontent by writing longer reviews, in contrast to short reviews when content with the application. The researchers also identified a strong correlation between positive-negative sentiments and one- and five- star ratings. Unexpectedly, more than 50% of the two and three-star rated user reviews did not include any sentiment.

Harman *et al.* (2012) mined the Blackberry app store using Spearman's Rank Correlation method and identified a strong correlation between application rating and number of downloads, whereas there was no correlation between price and rating, nor price and number of downloads. They tested their approach on the 32,108 non-zero priced apps. Iacob & Harrison (2013) manually analyzed reviews and identified nine classes of feedback: positive, negative, comparative, price related, request for requirements, issue reporting, usability, customer support and versioning. They first randomly chosed 169 apps and collected 3279 user reviews and then manually examined and classified reviews based on their content and then coded the categories, for example: aesthetics, company, comparison, feature/functionality, model, permissions, money, etc. They observed a correlation between review positivity and feature or functionality request.

Ha & Wagner (2013) manually analyzed Android users' reviews to see what they write about when reviewing Google Play applications. They crawled Google Play to collect information about 202,264 free applications and they selected 60 free applications with 556 reviews. As a result, they found a small subset of reviews addressing privacy and security implications, whereas the majority of the reviews focused on the quality of the applications.

Wano & Iio (2014) performed a manual text analysis and determined that review styles differ with software categories. Their study used the search API and also used Rich Site Summary

(RSS) Feed Generator by Apple. The number of targeted software was 500 and for each software, the targeted reviews are restricted up to 50 because of the API restriction. They concluded that consumers should pay attention to bias in reviews.

Gómez *et al.* (2015) mined reviews with LDA and error-suspicious permission patterns with the J48 decision tree algorithm (a Weka implementation of the C4.5 algorithm), identifying potential correlations between error-sensitive permissions and error-related reviews over time. They built a dataset that consists of a random sample of all the mobile apps available on Google Play Store. They collected 500 applications from 27 different categories.

Mojica Ruiz *et al.* (2015) made an overall evaluation of app stores and user rating schema and concluded that the current store rating of apps was not dynamic enough to capture the changing user satisfaction levels along with evolving application versions. Their dataset was extracted by crawling Google Play and this resulted in 242,089 app versions of 131,649 mobile apps. After the filtration, they ended up with 238,198 versions of 128,195 apps. They used hexbin plots to examine whether there would be a noticeable change in the store-rating of an app given a rise or drop in the rating of a specific version of that app.

Most studies identified within this research question are preliminary researches and based on either manual or statistical analysis of user reviews. The researchers used either the research API and RSS Feed Generator by Apple store or some scrapers script to collect app store data. The datasets are mostly created with random sampling of all the mobile apps available and there is not any specific or common app category preferred by researchers. Since Martin *et al.* (2015) presented empirical evidence that indicates that the partial nature of data available on App Stores could pose an important threat to the validity of findings, the obtained results from different App Store research studies could not be compared with one another. Star rating, category and review content are most common features collected within 87.5% of the studies.

We could not obtain any data regarding the average length of a review considered in the studies; however the dataset by Vasa *et al.* (2012) showed that user review length is highly skewed with an average of 110 characters. On the other hand, Fu *et al.* (2013) reported that the average

length of the comments is 71 characters, and median length is 47 characters. If the datasets used in these research studies would have been publicly available, it would have been feasible to validate these numbers. For this reason, researchers need to augment their findings with an argument to convince the reader that any sampling bias is unlikely to affect their research findings and conclusions. One of very recent studies by Gu & Kim (2015) indicated this app store sampling phenomenon as a threat to validity.

Table 1.1 of Appendix I presents the list of these 9 studies with their methods, details of datasets, features and performance as a response to RQ1.

1.3.2 RQ2: How do the studies remedy the 'domain dependency' challenge for app store reviews?

RQ2 looks at how domain dependency affects opinion mining from reviews. For example, a classifier trained in using opinionated words from one domain might perform poorly when applied to another domain, since the language structure in addition to words and phrases, may differ from one domain to another.

From the primary studies it was noted that while some researchers labelled data for the new domain and created their own dataset from scratch, other researchers used labelled data from one domain and unlabelled data from the target domain, and then made the domain adaptation by using general opinion words (Aue & Gamon (2005), Yang *et al.* (2006), Blitzer *et al.* (2007), Pan *et al.* (2010)). In order to overcome the domain barrier in opinion extraction, Cosma *et al.* (2014) proposed a generalized methodology by considering a set of grammar rules for the identification of opinion-bearing words.

In addition, online reviews have distinctive text features, including short length, unstructured phrases and abundant information. Short reviews bring new challenges to traditional research topics in text analytics, such as text classification, information extraction and sentiment analysis. In contrast to standard texts that include many words and phrases and their corresponding

statistics, short texts consist of few phrases and sentences. Several traditional text analytics methods have been proposed to tackle this data sparseness problem:

- The first is surface representation that uses phrases in the original text from different product aspects to maintain the contextual information. However, this method fails to produce a deep understanding of the text and the method does not make use of external knowledge, which has been found useful in dealing with the semantic gap in text representation Hu & Liu (2012). For example, this review from the app store: "This iOS 9 update. App crashing and ugly font", does not contain any words or phrases related to the reason for the crash and possible user interface (UI) design problem, while the words 'crash' and 'font' are related to software engineering concepts. Hence, it is difficult to use bag-of-words based models and methods to build semantic connections between the review text and software characteristics.
- Another approach is to enrich the context of basic text segments by searching the external sources. Such methods have been found effective in narrowing the semantic gap for different tasks (Gabrilovich & Markovitch (2007), Ureña-López *et al.* (2001). In the app store corpus, these external sources would be app crash reports, tweets, community blogs and code repositories.

Another important characteristic of online text, particularly in online reviews, is the use of colloquial language. When composing a review, users might use abbreviations or acronyms that seldom appear in conventional text. As an example, the phrases "superb" "Good 2go" "you do not buy the guarskldj; al b bbbbbbbbb,,,,,,,wke;" make it very difficult to identify the semantic meaning. With research question RQ2, we sought to discover how researchers tackled domain adaptation problems, how they dealt with distinctive features of the review text and what specific methods or algorithms and text features were used to improve performance. To answer this research question, we reviewed the primary studies to identify the training datasets, methods, text features and performance comparisons. The mobile app store researchers mentioned in Table 1.3 of Appendix I used their own annotated dataset rather than leveraging existing

online review datasets. Since they preferred to use conventional text mining methods such as Latent Dirichlet Allocation (LDA), Aspect and Sentiment Unification Model (ASUM), Naive Bayes classifier and statistical analysis, we cannot report any new method developed specifically for app store corpus. No new solutions or methods were proposed for examining the text characteristics (e.g., short length, unstructured phrases and colloquial language and challenges) of app store user reviews in the primary studies.

As in many real-word applications, topics revealed by LDA and ASUM must be verified by experts to ensure they are semantically meaningful within the domain analysis. Hence, four studies out of 24 leveraged truth sets to understand if the extracted features align with real app features and to minimize this threat to validity. Galvis Carreño & Winbladh (2013) used the manually classified data as a truth set. Since the second author is not a domain expert or is not involved in software development, they reported that the process is error-prone. Chen *et al.* (2014) collected the group truth labels of the training pool and test set according to pre-defined rules. Guzman & Maalej (2014) and Gu & Kim (2015) also used the truth set that was created with systematic assessment of review samples by human coders.

However, manual validation could dominate the time and cost of building high-quality topic models. To overcome this problem, some researchers proposed measuring topic quality with topic coherence and statistical methods (Mimno *et al.* (2011), Newman *et al.* (2009)). To tackle this problem, incorporating domain knowledge into Topic Modelling via Dirichlet Forest Priors Andrzejewski *et al.* (2009) would be useful. Dirichlet Forest Priors, when combined with LDA, allows the user to encode domain knowledge (must-links and cannot-links between words) into the prior on topic-word multi nominal P (word | topic). In this way, app store domain knowledge could be expressed by a set of Must-Links (e.g., two words u, v have similar probability within any topic) and Cannot-Links (e.g., two words u, v should not both have large probability within any topic).

1.3.3 RQ3: What criteria make a review useful?

Review quality varies from reviewer to reviewer, and low-quality reviews might not convey any useful information. App store regulators allow users to vote on the helpfulness of each review and then rank the reviews based on votes. While review helpfulness is usually assessed manually, there are automated systems that do this. For the manual review of usefulness, there are no defined criteria among users, so a review that appears helpful to one user may not be helpful for others, since they might be searching for different information or have differing priorities or biases. On the other hand, defined criteria would be valuable to differentiate useful reviews from others and reviews ranked in accordance with these criteria would provide better information extraction capability.

Studies examining online review helpfulness with manual assessments are as follows:

- Cheung *et al.* (2008) measured review quality in terms of completeness, timeliness, accuracy and relevance.
- Mudambi & Schuff (2010) found that review depth had a positive effect on the helpfulness of the review but product type affected the perceived helpfulness of reviews
- Pan & Zhang (2011) analyzed a large sample of reviews from Amazon to identify what determined information helpfulness and they reported that there was a high correlation between review length, positivity level of reviews and review usefulness.
- Korfiatis *et al.* (2012) reported that review readability and positive ratings affected the number of helpfulness votes given by users.

Studies on automatically assessing review helpfulness:

- Kim *et al.* (2006) trained an SVM (Support Vector Machine) regression model to learn the helpfulness function and then applied it to rank unlabelled reviews. They reported that the most important features were the length of the review, its uni-grams and its product rating.

- Liu *et al.* (2008) also modelled the helpfulness of reviews. They reported that helpfulness of a review depends on three important factors: reviewer expertise, writing style, and timeliness.

- Ghose & Ipeirotis (2011) used a Random Forest-based classifier and examined the relative importance of three feature categories: (i) reviewer related, (ii) reviewer subjectivity, and (iii) review readability. They reported that using any of the three feature category results provided the same performance as using all available features.

- Moghaddam *et al.* (2012) used a probabilistic graphical model based on Matrix Factorization and Tensor Factorization. These models are based on the assumption that the observed review ratings depend on latent features of the reviews, reviewers, raters and products. They reported that the latent factor models outperformed state-of-the-art approaches using textual and social features.

Even though the user review helpfulness evaluation problem has been studied for various platforms such Amazon, Yelp, TripAdvisor, we could not find any study that assessed review helpfulness either manually or automatically for the mobile app store corpus. As app store users could mark any review for any app as: 'Helpful', 'Unhelpful' and 'Spam' and the reviews could be ranked per their helpfulness at Google Play, some of App Store mining researchers such as Chen *et al.* (2014) and Park *et al.* (2015) from Table 1.6 preferred using only Helpful reviews or filter Unhelpful reviews out to train their models. According to Pagano & Maalej (2013)'s dataset only 67,143 (5.96%) reviews are rated by other users regarding their usefulness. From these, 38,519 (57.37%) are considered 100% helpful. Interestingly, 16,671 (24.83%) are rated completely useless. Even though we could not get enough information about the percentage of helpful or unhelpful reviews in other datasets, the need for filtering these reviews has been apparent for increasing data mining and information extraction capabilities.

1.3.4 RQ4: How could the spam reviews be differentiated from legitimate reviews?

As consumers increasingly rely on user reviews and ratings, there is greater incentive to create fraudulent reviews in order to boost sales and to damage competitor reputations on the market. Fraudulent reviews not only mislead customers into poor purchase decisions, but also degrade users' trust in online reviews. According to the study of Luca & Zervas (2016), 20% of all online reviews on Yelp.com are fake.

Most of the earlier research focused on detecting email and web spam. As the number of online reviews increases, as well as the number of fraudsters, additional studies and techniques have been proposed to detect spam reviews. Two main approaches are being used for opinion spam detection: behavioural and textual features. Behavioural features correspond to features such as review date, rating, and geo-location of the reviewer, while textual features refer to methods, such as part-of-speech patterns, word frequency, n-grams and cosine similarity.

Dellarocas (2000), the first to work on immunizing online reputation systems against unfair ratings and discriminatory behaviour, proposed a set of 'exception handling' techniques such as 'controlled anonymity' and 'cluster filtering'.

Kim *et al.* (2006) used SVM regression on different classes of features including structural (e.g., HTML tags, punctuation, review length), lexical (e.g., n-grams), syntactic (e.g., percentage of verbs and nouns), semantic and meta-data (e.g., star rating) features.

Jindal & Liu (2008) observed that spammers tended to create a small number of review templates and then copy them to spam a single product or several different products. To identify the replicated spam reviews, they used the two-gram review content comparison method, as in Kim *et al.* (2006).

Lim *et al.* (2010) trained a linear regression model to use four different spamming behaviour models as target products and groups, general rating deviation and early rating deviation. Wang *et al.* (2011) proposed a heterogeneous graph model to capture relations between reviewers, reviews and stores. Sandulescu & Ester (2015) presented two methods: (i) a semantic similarity

measure by extracting specific parts-of-speech (POS) patterns and (ii) an LDA model using bag-of-words and opinion phrases.

Within the corpus of mobile app stores, scammers use a great many bogus user accounts or bots in order to download applications multiple times and write fraudulent reviews. In this way, the applications begin appearing on the top charts and have greater visibility in an app store search. In addition, numerous sites allow purchasing of reviews: one example is the Fiverr site. Even though fake and opinion spam reviews are widespread and have significant manipulative effects on app store success, regulators have only recently begun to crack down on fake reviews Clover (2014). We found only a single app store review spam identification study in the literature:

- Chandy & Gu (2012) compared latent class graphical and decision tree models for classification of app spam and analyzed the preliminary results for clustering reviews. They used linear Gaussian parameterization on the labelled data, which achieved higher accuracy than a baseline decision tree model. As a result, they proposed a latent class model for the spam identification task. The details of this study are presented in Table 1.2 of Appendix I

1.3.5 RQ5: Extracted Application Features from User Reviews

App stores provide a user feedback capability that is particularly useful and interesting from the software requirements engineering point of view. User ratings and reviews correspond to user-driven feedback that may help improve software quality and address missing features.

With regard to extracted application features from app store user reviews, Iacob & Harrison (2013) identified nine different classes of feedback: positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer supports and versioning. Galvis Carreño & Winbladh (2013) adopted the ASUM, which incorporates both topic modelling and sentiment analysis, to obtain constructive feedback from user comments. They extracted various topics such as updates, features and developers from review text.

Pagano & Maalej (2013) identified topics in user app store reviews by grouping the information as follows:

- Community: References to other reviews or other applications.
- Requirements: All request types such as feature, content, improvement requests, shortcomings and bug reports.
- Rating: User intention to change his/her idea given certain improvements.
- User experience: Helpfulness in terms of application features and user interface.

In addition, they pointed out the correlation between overall app ratings and number of user reviews, app price and amount and type of feedback the application received. Fu *et al.* (2013) identified 10 top factors that affect the success of an app on mobile application ecosystems: attractiveness, stability, accuracy, compatibility, connectivity, cost, telephony, picture, media and spam. In addition, they identified 0.9% inconsistencies between user review texts and rating that may be caused by careless mistakes or intention to mislead.

Oh *et al.* (2013) developed a review digest system (using a SVM classifier) which was tested on 1,711,556 reviews mined from 24,000 Google Play apps. They automatically categorized user reviews into functional and non-functional requests, bug reports and produced a digest featuring the most informative reviews in each category. Chen *et al.* (2014) compared LDA and ASUM and reported that LDA presented many "non-informative or redundant topics" However, they validated their results on user reviews of only four Android apps, and it is not clear that the framework will attain similar good results when applied to other Android apps or other app stores.

Guzman & Maalej (2014) used topic-modelling techniques to group fine-grained explicit features into high-level features using topic modelling LDA and weighted-average techniques. In addition, they compared the relevance of the extracted features with app requirements and concluded that for the top 10 popular extracted features, the words (e.g., upload photo, file

exchange – for Dropbox, board pin, time search – for Pinterest) usually described actual app features and conveyed some clues about how the app was used. McIlroy *et al.* (2015b) and its counterpart studies Khalid (2013) and Khalid *et al.* (2015) automatically labelled the types of user issues raised in mobile app reviews, such as additional cost, functional complaint, compatibility issue, crashing, feature removal request, network problem, privacy and ethical issue, resource heaviness, response time, uninteresting content, update issue and user interface. They manually labelled a statistically representative sample of user reviews from the Apple app store and Google Play.

Vu *et al.* (2015) pursued a keyword based approach to collect and mine user opinion from app stores by extracting, ranking and grouping keywords based on semantic similarity. In addition, they provided a visualization tool that showed the occurrence of keywords over time and reported any unusual patterns. Park *et al.* (2015) developed a topic model AppLDA designed for use on app descriptions and user reviews. Their proposed method enables developers to inspect the reviews and find out important app features of apps. Panichella *et al.* (2015) presented a system for automatically classifying user reviews based on a predetermined taxonomy, in order to support software maintenance and requirements evolution. Gu & Kim (2015) proposed a SUR-Miner that is a review summarization and categorization tool, which evaluated 2000 sentences from the reviews of 17 Google Play apps. In addition to these studies, McIlroy *et al.* (2015a) examined this research problem from developers' respective and observed that there are positive effects to responding the reviews (users changed their earlier ratings 37.8% of the time) with a median increase of 20% in the rating.

Table 1.3 of Appendix I presents the list of studies relevant to RQ5 and identifies the related methods, extracted app features and performance.

1.4 Discussion

The mobile app ecosystem and user reviews contain a wealth of information about user experience and expectations. Developers and app store regulators could leverage the information

to better understand their audience. Mining app store data, and in particular user reviews, may provide valuable information for users to reach an informed decision about applications and their features; similarly, it would be valuable for developers to receive user feedback about most liked or expected features, as well as reported bugs in the applications. Mining opinions from app store reviews still requires pre-processing at the content level, including filtering out non-opinionated content and identifying the trustworthiness and genuineness of the opinion and its source. Even though, there is a limited number of research studies analyzing mobile app reviews, the direction and results obtained are promising. Hence, from the perspective of software requirements engineering, with further research, it is expected that app store meta-data will provide a more accurate picture of user choices and expectations. Developers and app store regulators could leverage reviews to better understand their audience. Here we present our principal findings from the SLR.

1.4.1 Principal Findings

- App store ecosystem is a new form of software repository with finer granularity including the information such as customer ratings and reviews, app price, version and features and app store user feedback mining has begun to attract the attention of researchers. Most of the studies were of an exploratory nature, based on manual classification and correlation analysis. The number of high-quality app store studies was very limited: we retrieved nine app store mining studies and only one app store spam identification study.

- The automated extraction of app features in online reviews does not consider the nature of the review text. As online app reviews have distinctive features of text (e.g., short length, unstructured phrases, colloquial language and abundant information), there is a need to develop a model specific for app store reviews in order to extract targeted app features rather than use conventional methods and techniques developed for different domains and contexts.

- Furthermore, the information requested by users and developers are different. Users are more interested in the opinion and experience of others about the application and which

aspects/features they liked or disliked most. Developers have a different point of view when using reviews to:

  - extract usability and user experience information,
  - elicit missing requirements and define requested application features, and
  - improve software quality.

- To deal with abundant information in reviews, external sources such as app crash reports, tweets, community blogs and code repositories could be used to enrich the data. In addition, integration of text from different data sources (such as social media profiles) would be helpful to ensure context level opinion mining, since in terms of preferences and needs, opinions are specific to each person or group.

- Opinion spam or fake review detection is one of the largest problems in the domain. In addition to spam reviews, there are various kinds of user reviews, some of which do not include any useful data for information extraction. Hence, it is necessary to merge multiple criteria not only to identify suspicious reviews but also to differentiate useful reviews from others so that reviews complying with the usefulness criteria can be processed for information extraction. Even though some automated systems have been introduced to identify fake and spam reviews and to evaluate review usefulness, these systems are very limited and not yet mature.

CHAPTER 2

PROBLEM STATEMENT & RESEARCH METHODOLOGY

2.1 Problem Statement

The greater variety of mobile apps and the ever-rising demand for the apps running on different mobile systems create a corresponding increase of mobile developers and a big competition in mobile app markets. With so much competition in the market, it gets more difficult for developers and software organizations to achieve the success they have targeted. The empirical studies mentioned in the previous chapter have shown that app store reviews include information that is useful to mobile app developers and analysts in regards to users requirements, bug reports, feature requests, and documentation of user experiences with specific app features Genc-Nayebi & Abran (2017). Hence, understanding users' feedbacks and anticipating users experience and user acceptance of the app are becoming key success drivers in app stores.

Since 2010, app store user feedback mining has begun to attract the attention of researchers. One of the challenges in app store opinion data mining is vocabulary, which can vary even with a single term having different meanings in different contexts and domains. As online app reviews have features distinctive of text (e.g., short length, unstructured phrases, colloquial language and abundant information), there is a need to develop app store specific model in order to extract targeted app features rather than using conventional methods and techniques developed for different domains and contexts.

Whereas app users rely on the reviews and ratings of others to formulate an informed decision about applications and their features before downloading them, reading all the reviews is time consuming and occasionally deceptive due to misleading or spam reviews. In addition to spam reviews, some reviews do not include useful data for information extraction. Even though some automated systems have been introduced to identify fake and spam reviews and evaluate usefulness, these systems are limited.

Lastly, extracted features relevance has not been cross-validated with main software engineering concepts: for example, a user comment about the app's simplicity and easy to use manner could imply 'user experience' and 'usability' of the software, while user's expectancy about a feature that makes the app contextual and relevant could refer to 'requirements analysis' and 'use cases' Genc-Nayebi & Abran (2018). Hence, these aspects or topic modelling results would make more sense if they were mapped to software engineering concepts such as use-cases, software maintenance, quality assurance, Graphical User Interface (GUI) design, user experience, usability, etc.

2.1.1 Research Motivation

Our literature review study exposed that the app store ecosystem is a new form of software repository with finer granularity, since the information such as customer ratings and reviews, app price, version and features information makes app ecosystem a good place for data mining and empirical analysis. Also the limited number of studies identified in the literature did not focus on the domain and context dependency of the problem. Due to the unstructured nature of user reviews and colloquial language used, it is a challenging, but also a promising, task to extract information from app store user reviews.

Opinion spam or fake reviews is one of the biggest problems in the domain. As consumers increasingly rely on user reviews and ratings, there has been a strong incentive to create fraudulent reviews in order to boost sales and damage competitors' reputations. Since we found only a single app store review spam identification study in the literature, our research study will be one of the very first studies examining opinion spam detection problem for mobile app stores.

In addition to fake reviews, there exist various kinds of user reviews, some of which do not convey any useful data; also, unbiased or non-spam user reviews may be numerous but of varying quality. Hence, it would be helpful to automatically assess and rank reviews in accordance with their usefulness or helpfulness.

During the literature review it was also discovered that most of the studies are of an exploratory nature that are conducted for formulating app store opinion mining problem based on preliminary analysis of the data and unstructured research design. The studies that automatically extract app features do not provide enough related information from a requirements engineering point of view, because:

- user reviews often convey implicit information,
- new feature requests, quality issues and application aspects are not written explicitly in the reviews.

From the users' point of view, they would like to learn about others' experience with the app, what aspects/features they liked or disliked most and the features that distinguish the app from similar apps in the market. Each user has his/her own preferences and while one user might feel strongly about the appearance, others may focus on functional or technical aspects. Hence, app features that are extracted from review text should be presented differently to developers and users.

2.1.2 Research Objectives

The four objectives of this research are described through the following research hypotheses:

- H0a – It is possible to classify app store user review's legitimacy (being spam or truthful) by automatically discovering distributed feature representation of review data.
- H1a – All user reviews do not include useful information in regards to merits and drawbacks of the mobile app.
- H0b – Analysis of app store user reviews provides necessary information for software quality.
- H1b – It is possible to automatically retrieve app aspects from the review text.

To investigate the research hypotheses, this doctoral study will develop the following research tools:

- A deep learning model to classify deceptive and benign user reviews.
- A model to evaluate review usefulness.
- A deep learning model to extract app aspects from user reviews for the use of both developers and app users.

2.1.3 Target audiences

The targeted audiences for this research are the following:

- Mobile app users,
- Application design and development professionals,
- Software organizations,
- App store regulators,
- Artificial intelligence and machine learning researchers
- Human-Computer Interaction researchers,
- User experience design professionals and usability evaluation experts.

2.2 Research Methodology

This research study explores the hypotheses and the approaches regarding app store opinion mining, opinion spam detection and feature extraction problems by executing the following research phases:

- Phase 1: Development of an app store crawler
- Phase 2: Identification of opinion spam (deceptive) reviews
    1. Construction of a spam review dataset
    2. Development of an opinion spam detection model
    3. Performance evaluation of the opinion spam detection model
- Phase 3: Automatic assessment of review helpfulness
    1. Feature generation
    2. Selection of the model
    3. Model refinement
- Phase 4: Development a model to extract targeted app features
    1. Collection of Review Dataset
    2. Filtering Spam and Non-Helpful Reviews
    3. Annotation of Mobile App Aspects
    4. Development of aspect extraction models
    5. Performance evaluation of aspect extraction models

This research study begins with Phase 1: development of an app store crawler to populate mobile app store meta-data and user reviews. Crawled artifacts will be the inputs of the following phases.

Phase 2 proposes a deep learning model for opinion spam identification task.

Phase 3 presents an automatic helpfulness assessment of app store user reviews.

After filtering out non-opinionated content and evaluating the trustworthiness of the user reviews, Phase 4 of this research study proposes an aspect extraction model for the use of app developers and end-users.

Figure 2.1 presents an overview of the research methodology including the inputs, phases and outputs. The following subsections in this chapter provide more details about each phase of this research.

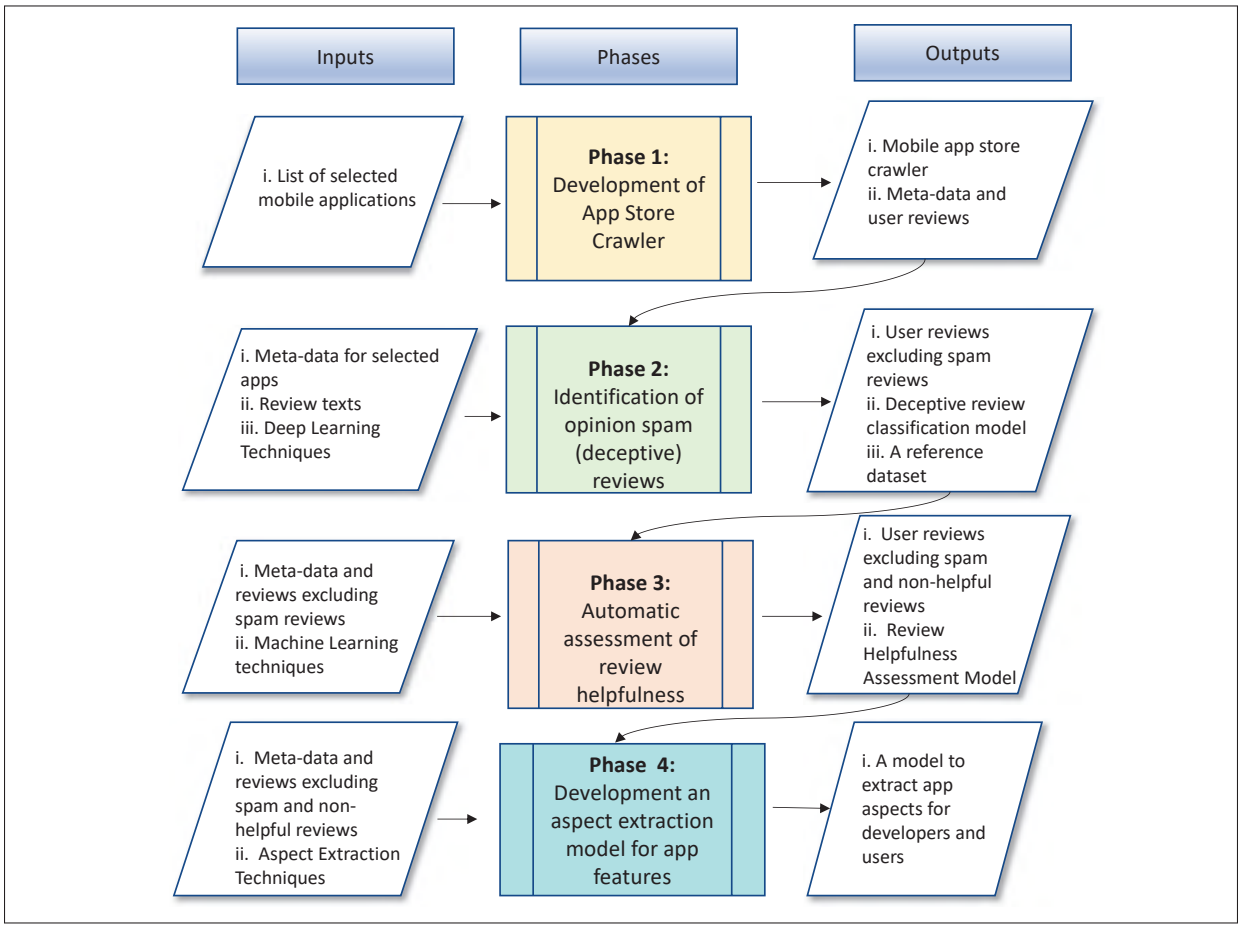


Figure 2.1 Research Methodology

2.2.1 Phase 1: Development of an app store crawler

In this first phase of the doctoral research, two crawlers, one for Apple App Store and the another one is for Google Play are developed to collect user reviews and app meta-data such as appId, description, version, genres, etc. from the subject app stores.

Apple provides different iTunes RSS feeds that retrieve varying results depending on the media type. One of the feeds is for retrieving customer reviews that require the following variables:

CODE (App Store territory code), APPID (9 digit number unique for each app) and FILETYPE (either Extensible Markup Language (XML) or Javascript Object Notation (JSON)) as given in the following URL format.

https://itunes.apple.com/CODE/rss/customerreviews/page=1/id=APPID/sortby=mostrecent/FILETYPE

```
[ {
    id: '3237000149',
    userName: 'J_L_P_A_0_1',
    userUrl:
      ↪ 'https://itunes.apple.com/us/reviews/id467774835',
    version: '1.134.1',
    score: 3,
    title: 'The new version is horrible',
    text: 'I love this game but the update that I just
      ↪ install on 9/27/18 I hate it. It doesn't allow
      ↪ you to play the game anymore as soon as you
      ↪ swipe. The computer already generate a new move
      ↪ for you, if doesn't made you think about your
      ↪ next move. I like it when it will give you 5 to
      ↪ 10 seconds before the computer help you on a
      ↪ move. This way it allows you to play the game the
      ↪ way that you think it's best. Can you return that
      ↪ option to the original settings'
} ]
```

Listing 1: Example of a review result for a specific app store application

Our iTunes crawler first fetches complete RSS response in JSON format and then parses it into the following data fields as id (10 digit number unique for each review), userName (reviewer user name), userURL, version (app version), score (rating given by the user), title (title of the review) and text (review body) for further analysis of the meta-data and the reviews.

On the other hand, Google Play (GP) loads data for customer reviews via AJAX requests to the URL https://play.google.com/store/getreviews using the HTTP POST method.

The data returned in JSON format are parsed into the following fields as id (review unique ID), userName (reviewer name), userImage (reviewer profile image), date (date of the review), score (rating given by the user), title (title of the review), text (review body), replyDate (date of developer response to review), replyText (developer response to review) by our Google Play crawler.

2.2.2 Phase 2: Identification of opinion spam (deceptive) reviews

Phase 2 of the research study focuses on automatically identifying deceptive user reviews. The following sub-sections present the detailed information about:

- Construction of a spam review dataset,
- Development of an opinion spam detection model,
- Performance evaluation of the opinion spam detection model

2.2.2.1 Construction of a spam review dataset

Data is a major part of any machine learning based model, and even though a massive volume of app reviews is available on the app stores, collecting and labeling a sufficient number of reviews to train a deceptive review classifier is a difficult task. Besides, it is hard to manually label a review as deceptive or benign by simply reading the review, because a spammer can carefully craft a non-legitimate review to promote or to degrade the reputation of the app.

To generate spam user reviews, we first tried a text bot, the Markov-bot, that relies on Markov Chains stochastic model Gilks *et al.* (1996). However, the obtained results are not realistic (an example of Markov-bot generated reviews is presented as follows):

*I always use my gift card and then when I click on *just* that hyperlink. Today this options it just closes the app always works for the app multiple times with no improvement to performance.*

Thus, we develop a three-step review synthesis method including the following steps:

1. First a review repository for the target application is created. The reviews that belong to the target app and 4 other apps from the same app category (e.g., Shopping, Game, Entertainment) are crawled from App Store.

2. Afterwards, our review synthesizer breaks the reviews into individual sentences per specific topic and then mixes them up to generate a new review as minimum 100 and maximum 300 characters long, given that the average length is 170 characters for app store user reviews.

3. Finally, human evaluators are asked to read generated reviews and identify whether the reviews appear realistic or not.

2.2.2.2 Development of an opinion spam detection model

The objective of a spam identification task is to distinguish whether the review is spam or a truthful, a 2-category classification problem. To tackle the spam identification task behavioural and textual features are typically being used within supervised machine learning models. Behavioural features correspond to features such as review date, rating and geo-location of the reviewer, while textual features refer to methods, such as part-of-speech patterns, word frequency, n-grams and cosine similarity. Most related studies focus on designing effective features to enhance the classification performance. Even though feature engineering is useful and brings strong performance to the classifier, the sparsity of the features makes it difficult to capture non-local semantic information over a sentence or discourse.

Neural network based models in Natural Language Processing (NLP) tasks currently achieve highly competitive results Kim (2014), Kalchbrenner *et al.* (2014), Johnson & Zhang (2015). Neural models use dense hidden layers for automatic feature combinations: they can capture complex global semantic information that is difficult to express by using traditional discrete manual features. Hence, in this part of the doctoral research we will investigate empirically the

effectiveness of learning dense document representations with convolutional neural networks for opinion spam detection task. In the first stage of our convolutional neural network system, sentence representations are produced from word representations, while a document representation is constructed from the sentence vectors in the second stage of the system. Finally, this document representation is used as features to identify app store opinion spams.

2.2.2.3 Performance evaluation of the opinion spam detection model

The performance of our neural network based spam classification will be compared with a baseline classification model, SVM, by creating different feature combinations. The classification results of two groups of experiments (CNN and Support Vector Machine (SVM)) are evaluated for the accuracy measure in 5-fold cross validation. The features extracted and then leveraged within the SVM model are presented in Table 2.1 :

Table 2.1 Text features extracted and then leveraged within the SVM model

Review-centric features	**Details**
Structural features	Length of the review, average word length, number of sentences, average sentence length, percentage of numerals, percentage of capitalized words.
POS percentages	The percentage of each POS tagging in each review.
Semantic features	The percentages of positive and negative opinion-bearing words in each review.
n-grams	Top 100 uni-grams and bi-grams that have the most different percentages (in terms of ratios) in fake and non-fake reviews.

2.2.3 Phase 3: Automated assessment of review helpfulness

In Phase 3 of this research study, the task of automatically predicting app store user reviews' helpfulness is investigated. Different aspects of a review text such as lexical, sentential and

meta-data and their effects on influencing perceived usefulness (helpfulness) of the reviews are explored. We will first formulate the learning task and then investigate several features for assessing review helpfulness.

2.2.3.1 Feature Generation

The dataset of reviews is populated and filtered with noise removal, stemming, tokenization and stop-word removal processes in this part of the research study. Afterwards, various features organized in three classes as Structural, Lexical, and Meta-data are extracted:

- **Structural:** Structural features are observations on the document structure and formatting. Properties such as review length and average sentence length are hypothesized to relate structural complexity to helpfulness.
- **Lexical:** Lexical features capture the words observed in the reviews. We will experiment with two sets of features:

  Unigram (UGR): The tf-idf statistic of each word occurring in a review.

  Bigram (BGR): The tf-idf statistic of each bigram occurring in a review.
- **Meta-Data Features:** Meta-data features capture observations which are independent of the text (i.e., unrelated with linguistic features).

2.2.3.2 Selection of the model

This part of the research explores the different binary classifiers and assess how *good* or how *accurate* the model is at predicting the class label of each instance (e.g., in this study, application) in the test dataset.

2.2.3.3 Model Refinement

New features are added to the existing feature set to enhance the prediction performance. The model parameters are set through grid search and cross-validation results obtained with new feature set are also presented in this part of our research study.

2.2.4 Phase 4: Extraction of Mobile App Features

The objective of Phase 4 is to automatically extract application features (aspects) from user reviews.

2.2.4.1 Collection of a Dataset of Reviews

In this part of the research, a dataset of reviews is populated from Apple App Store and Google Play. Both app stores categorize mobile apps and games into different categories based on the main function or subject matter of the app. The rationale behind populating the review dataset through different app stores and categories is to evaluate our aspect extraction models against the reviews that contain different vocabularies, and describing various app aspects.

2.2.4.2 Filtering Opinion Spam and Non-Helpful Reviews

In this part of the research, the reviews are first classified as truthful and deceptive with opinion spam detection model that was trained in Phase 2. Afterwards, the helpfulness of the reviews are predicted with the review helpfulness evaluation model earlier presented in Phase 3.

2.2.4.3 Development of Aspect Extraction Models

Deep neural network models can learn text representations from data without careful engineering of features and capture semantic relations between aspect and context words in a more scalable way. Hence, we employ two deep neural net models that can generalize well even when limited training data is available in this part of the research.

2.2.4.4 Performance Evaluation

In the last part of our research study, we evaluate the proposed aspect extraction techniques where precision, recall, and F_1-score are employed as our evaluation criteria.

CHAPTER 3

OPINION SPAM DETECTION

3.1 Introduction

Neural network based models in Natural Language Processing (NLP) tasks currently achieve highly competitive results Kim (2014), Kalchbrenner *et al.* (2014), Johnson & Zhang (2015). Neural models use dense hidden layers for automatic feature combinations: they can capture complex global semantic information that is difficult to express by using traditional discrete manual features. Hence, in this research phase, we investigate empirically the effectiveness of learning dense document representations with CNNs for opinion spam detection task. In the first stage of our CNN system, sentence representations are produced from word representations, while a document representation is constructed from the sentence vectors in the second stage of the system. Finally, this document representation is used as features to identify app store opinion spams.

Most of the time deceptive reviews are not easily identifiable by human readers; consequently, there are a few sources of labeled dataset for the research purposes. Indeed, in the absence of a standard dataset, earlier studies have been forced to utilize non-standard evaluation procedures Ott *et al.* (2011). Thus, one of the contributions of this research study is the creation of first large-scale and publicly available electronic dataset that contains 400 truthful and 400 deceptive reviews for the research on app store opinion spam problem.

This chapter which addresses Phase 2 of our research study is structured as follows: Section 3.2 presents background information about the necessary concepts including word embedding and neural networks. Section 3.3 describes the details of our methodology. Section 3.4 presents the details of the implementation. Section 3.5 reports on the performance results of our neural network model against a SVM baseline model. Section 3.6 presents the implications of this opinion spam detection model.

3.2 Background Information

3.2.1 Spam Review Detection

There are various data and features pertaining to a review that can be used to detect opinion spam, such as:

- review content,
- meta-data including reviewer's identity, time of the review and information about the product Liu (2012).

A variety of methods and techniques have been proposed by researchers to leverage these data and features for the review spam detection problem. The very first technique proposed by Jindal & Liu (2008) studies the duplication of review content (body) to detect spam reviews. A probabilistic language model was proposed next by Lai *et al.* (2010) to calculate a similarity score between two reviews to figure out if one review was generated from another.

Ott *et al.* (2011) reported that using content-based features, such as Linguistic Inquiry and Word Count (LIWC) with bi-grams, significantly improve the performance of spam reviews detection methods. The fake reviews used in this study were generated by an Amazon Mechanical Turk (AMT) crowd sourcing service.

Sun *et al.* (2013) reported on a review synthesis method that automatically generates fake reviews from a collection of truthful reviews. The authors claim that not only human judges, but also state-of-the-art review spam detection techniques, had over 40% error in detecting synthesized reviews task.

Li *et al.* (2011) used review features and reviewer features as two views in employing a co-training algorithm to detect spam reviews. Mukherjee & Liu (2010) classified individual reviewers into spam and non-spam groups by using a frequent-pattern mining technique and var-

ious spamming behaviours and spam indicators, such as content similarity between members, number of reviews in a time interval, deviation of product rating, etc.

While detection of email spam and opinion spam for rating sites (e.g., Yelp, TripAdvisor) or marketplaces (e.g., Amazon) has received considerable attention from researchers, there has been little work to date within the app store corpus. We found only a single app store review spam identification study in the literature where Chandy & Gu (2012) compared latent class graphical and decision tree models for classification of app store spam reviews and analyzed the preliminary results for clustering reviews. Their study used linear Gaussian parameterization on the labelled data and they reported higher accuracy than their baseline decision tree model.

3.2.2 Convolutional Neural Networks (CNNs)

CNNs are a family of deep networks that can exploit the spatial structure of data (e.g., images and texts) to learn about the data, so that the algorithm could provide an useful output. Neural network models have also been exploited to learn dense feature representation for a variety of NLP tasks such as part-of-speech tagging (POS), chunking, name entity recognition and sentence modeling Collobert *et al.* (2011), Kalchbrenner *et al.* (2014) and as well as for text classification Yin *et al.* (2017). Distributed word representations Mikolov *et al.* (2013) have been used as the basic building block by most models for NLP tasks. However, as convolutions and pooling operations lose information about the local order of words, POS Tagging or Entity Extraction is harder to fit into a pure CNN architecture without adding positional features to the input Chiu & Nichols (2015).

Since deceptive opinion spam detection is a complex text classification task due to reviews' short length and varied quality, deep features of the deceptive review text are required to be extracted along with textual semantics and emotional polarity features that have been widely used in text analytics. Hence, deceptive opinions detection is now an important application of CNN models and they have been used extensively in recent years due to their superiority in

terms of good fault tolerance, parallel processing and self-learning abilities for the spam review detection task Ren & Ji (2017), Li *et al.* (2017).

3.2.3 One-Layer CNN for Text Classification Task

The architecture of a typical one-layer CNN for text classification task that is depicted in Figure 3.1. It consists of three consecutive operations:

- convolution layer with activation function,
- pooling,
- softmax function for classification.

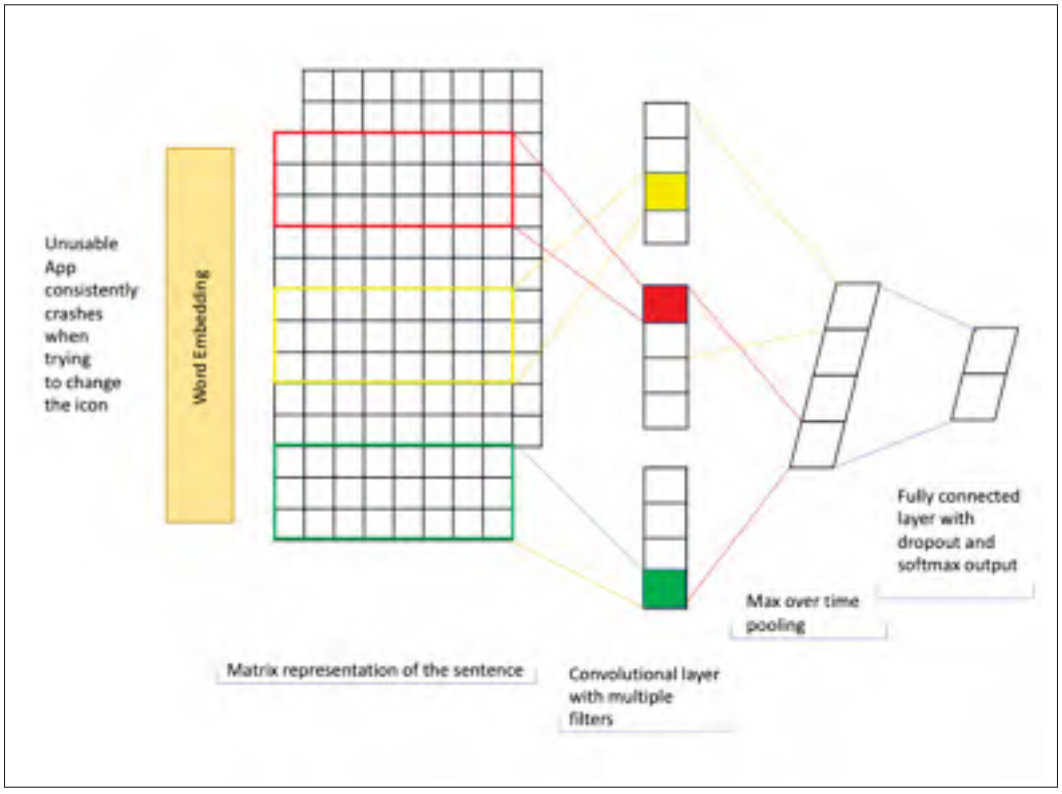


Figure 3.1 A typical one-layer CNN for text classification task (Adapted from Kim (2014))

Convolution later acts like a sliding window function applied over the sentence matrix that filters the current region where the calculated value of the matrix depends upon the activation function used. The activation function maps the feature value to a specific continuous real value range. Activation functions usually range from element wise matrix multiplication to non-linear functions such as the Rectified Linear Unit or hyperbolic tangent (tanh) chosen as the activation function in our CNN.

The resultant matrix of the filters applied on convolutional layer is not uniformed in size: hence, pooling is applied to get the uniform output. At subsequent classification layer (e.g., fully connected softmax layer) outputs N by using the softmax activation function, the number of class labels in the classification problem, where spam review detection task N=2. The function in Equation 3.1 adapted from Bishop (2006) gives the probabilistic value of the classes to which an input text belongs.

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}} \tag{3.1}$$

3.2.4 Pre-trained Word Embeddings

Word embedding is building a low-dimensional vector representation from a corpus of text, which preserves the contextual similarity of words. Since word embeddings are numerical representations of contextual similarities between words in regards to e.g., gender, tense and geography, they can be manipulated arithmetically just like any other vectors. Thus word embeddings provide a good generalization to unseen words and they could capture general syntactic as well as semantic properties of words. Word embedding techniques include Word2Vec, GloVe, FastText two of which, Word2Vec and GloVe are discussed in the following sub-sections.

3.2.4.1 Word2Vec Model

One of the best known algorithms for producing word embedding models is Word2Vec which is a predictive model used to produce distributed representations of words. Word2Vec contains two distinct models, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model Mikolov *et al.* (2013) as presented in Figure 3.2.

Algorithmically, these models are similar, except that CBOW predicts target words (e.g. 'phone') from source context words (' I just got a new *phone*'), while the skip-gram does the inverse and predicts source context-words from the target words.

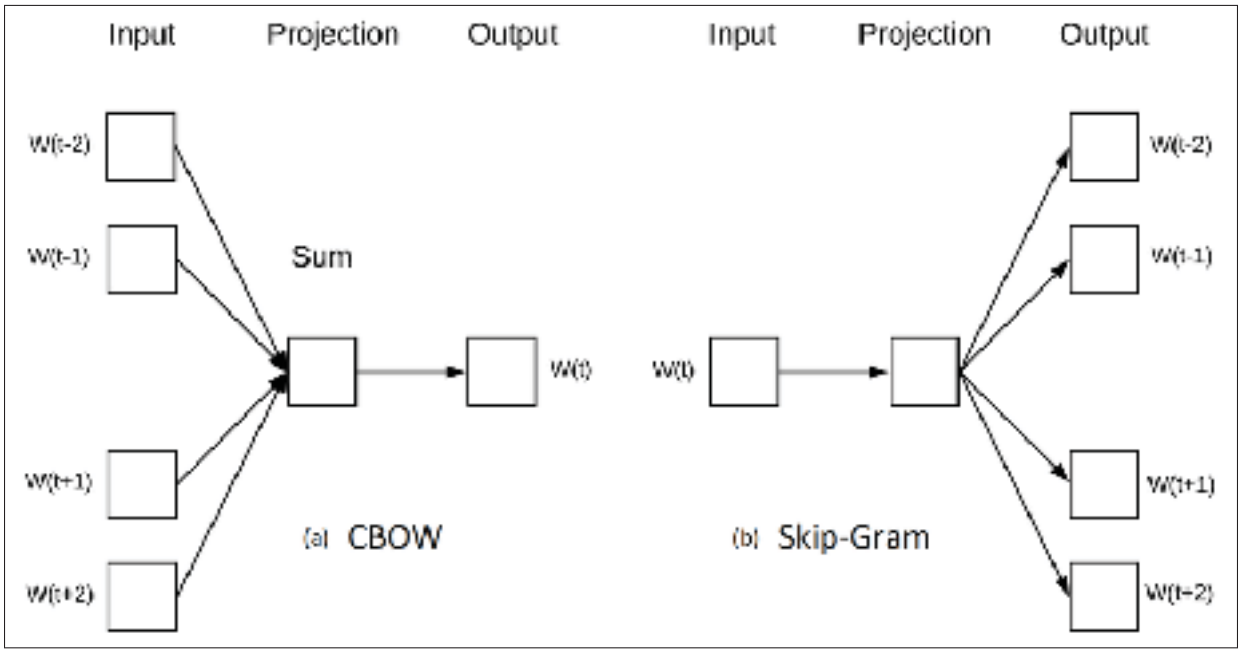


Figure 3.2 CBOW Architecture vs Skip-gram Architecture (Taken from Mikolov *et al.* (2013))

3.2.4.2 GloVe Model

The GloVe proposed by Pennington *et al.* (2014) is essentially a log-bilinear model with a weighted least-squares objective. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence.

Since the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well.

For instance, consider the co-occurrence probabilities for target words *ice* and *steam* with various probe words from the vocabulary. As presented in Table 3.1 [1], ice co-occurs more frequently with solid than it does with gas, whereas steam co-occurs more frequently with gas than it does with solid. Both words co-occur with their shared property water frequently, and both co-occur with the unrelated word fashion infrequently. Only in the ratio of probabilities does noise from non-discriminative words like water and fashion cancel out, so that large values correlate well with properties specific to ice, and small values correlate well with properties specific of steam. In this way, the ratio of probabilities encodes some raw form of meaning associated with the abstract concept of thermodynamic phase.

Table 3.1 Some words' actual probabilities from a 6 billion word corpus

Probability and Ratio	**k=*solid***	**k=*gas***	**k=*water***	**k=*fashion***
P(k | ice)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(k | steam)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k | ice) / P (k | steam)	8.9	8.5×10^{-2}	1.36	0.96

3.3 Methodology

Our research methodology for opinion spam detection includes three main tasks: (i) Automated Review Generation, (ii) Implementation and (iii) Experiments steps as presented in Figure 3.3. In following sub-sections, we present the details of automated spam review generation method and neural network based models to learn document representation for deceptive spam review detection.

[1] https://nlp.stanford.edu/projects/glove/

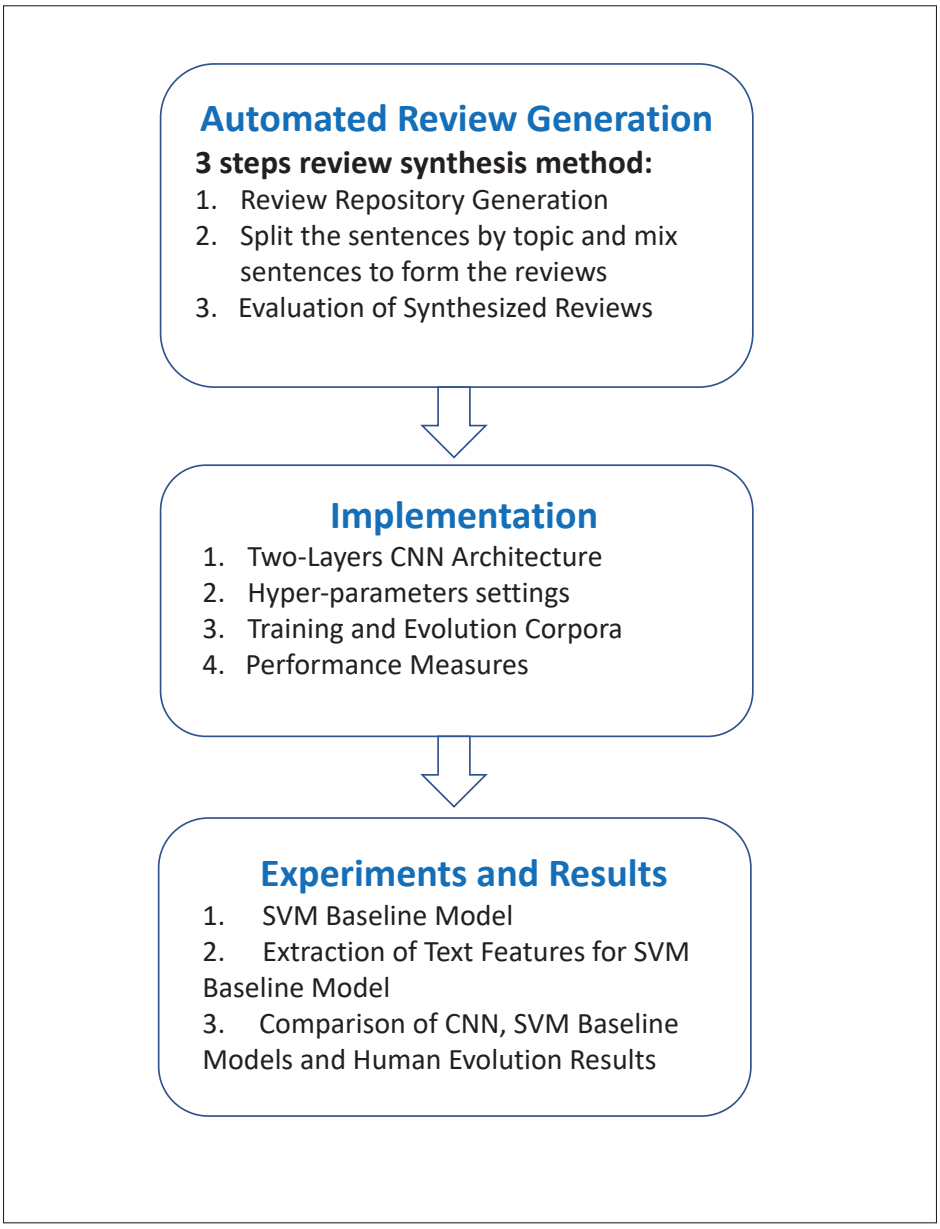


Figure 3.3 Opinion Spam Detection Methodology (Phase 2)

3.3.1 Automated Review Generation

To generate deceptive user reviews, we first tried a text bot, the Markov-bot, that relies on a stochastic model, Markov Chains Gilks *et al.* (1996). However, the obtained results are not realistic as seen below; therefore, we develop our own three-steps review synthesis method:

*I always use my gift card and then when I click on *just* that hyperlink. Today this options it just closes the app always works for the app multiple times with no improvement to performance.*

1. First, a review repository for the target application is created. The reviews that belong to the target app and 4 other apps from the same app category (Shopping, Game, Entertainment etc) are crawled from App Store. The list of 10 mobile apps from which we populated truthful user reviews, is presented in Table 3.2. For each app, 40 positive and 40 negative user reviews are chosen by random sampling. Table 3.3 presents the list of 40 apps (10 x 4 apps per category) from which we generated deceptive user reviews. Our synthesizer generated 40 positive deceptive and 40 negative deceptive reviews for each reference application.

2. Next, whole reviews are split into sentences by topic and our review synthesizer mixes the sentences from the review repository for each topic to generate full review text that is minimum 100 and maximum 300 characters long, given that the average length is 170 characters for app store user reviews.

3. Finally, 2 human annotators are allocated to evaluate the generated user review. Our human evaluators are asked to read generated reviews and identify whether they are realistic or not. One of the human annotators is PhD student with 10 years of software development expertise, while the second human annotator, a Post Doctoral researcher, has 15 years of software development expertise.

As a result, per our human evaluators' assessment:

- Only 37.5% of automatically generated reviews by following our review synthesis method are categorized as unrealistic,

This result indicates that an opinion spam detection task is challenging not only for machines but for human evaluators as well.

Table 3.2 List of the apps for truthful review dataset

ID	Category	Name	Version	# Truthful Positive Reviews	# Truthful Negative Reviews
1	Shopping	Amazon - Shopping made easy	11.14.0	40	40
2	Game	Battlelands Royale	0.5.8	40	40
3	Books	Audible: Listen to audio books	2.36	40	40
4	Health-Fitness	Fitbit	2.75	40	40
5	Game	Holeio: Daily Brain Games	1.3.1	40	40
6	Food-Drink	Uber Eats: Food Delivery	1.149. 10001	40	40
7	Finance	Paypal	6.28.0	40	40
8	Entertainment	TikTok	8.4.0	40	40
9	Magazine	The New York Times	7.2.0	40	40
10	Photo-Video	Youtube	13.33	40	40

Table 3.3 List of the apps for deceptive review dataset

Ref App & Category	Apps from the Category	Version	# Positive Deceptive Reviews	# Negative Deceptive Reviews
Amazon - Shopping	1.a Groupon 1.b OfferUp - Buy. Sell. Simple. 1.c Poshmark 1.d Wish - Shopping Made Fun	18.10 2.55.0 2.124 4.9.0	40	40
Battlelands Royale - Game	2.a Bumper.io 2.b Hello Stars 2.c Tomb of the Mask 2.d Ultra Sharp	1.1.4 1.7.7 1.5.1 1.0	40	40
Audible - Books	3.a Amazon Kindle 3.b Hooked 3.c Wattpad 3.d OverDrive: eBooks & audiobooks	6.9 3.10.6 6.85.0 3.7.5	40	40
Fitbit - Health - Fitness	4.a MyFitnessPal 4.b BetterMe 4.c 30 days fitness challenge 4.d Sweatcoin	18.7.5 2.5.4 3.3.19 9.1	40	40

Table 3.3 List of the apps for deceptive review dataset (continued)

Ref App & Category	**Apps from the Category**	**Version**	**# Positive Deceptive Reviews**	**# Negative Deceptive Reviews**
Holeio - Game	5.a Go Fish! 5.b Axe Climber	1.0.3 1.6	40	40
Holeio - Game	5.c Baseball Boy! 5.d 8 Ball Pool	1.7.1 4.0.1	40	40
Ubereats - Food - Drink	6.a McDonaldś 6.b DoorDash 6.c Starbucks 6.d Grubhub: Local Food Delivery	1.2.9 3.0.93 4.11 7.11	40	40
Paypal - Finance	7.a Cash App 7.b Venmo: Send & Receive Money 7.c Chase Mobile 7.d Bank of America Mobile Banking	2.30 7.20.1 2.675 7.8	40	40
TikTok - Entertainment	8.a Netflix 8.b Hulu: Watch TV Shows & Movies 8.c Amazon Prime 8.d Celebrity Voice	11.4.0 5.27 5.7.1 2.0.7	40	40
The New York Times - Magazine	9.a The Wall Street Journal 9.b USA TODAY 9.c PressReader 9.d B&W Photography Magazine	11.7.1 5.11.0 5.3.3 6.0.0	40	40
Youtube - Photo - Video	10.a Snapchat 10.b 1 Second Everyday: Video Diary 10.c Twitch: Live Game Streaming 10.d Unfold — Create Stories	10.37.1.1 2.0.12 6.6 3.3.2	40	40

3.3.2 Two-layer CNN Architecture

We propose to use two-layer CNN architecture as depicted in Figure 3.4 for opinion spam review detection task. The first convolution layer produce sentence representations from word representations, while the second convolutional layer of the model which is called the document convolution transforms sentence vectors into a document vector. Finally the soft-max

classification layer use the document vector representation as features to identify deceptive spam review.

We first create the word embeddings that are low dimensional, continuous and real-valued vectors and pre-trained from a large corpus of text with a learning algorithm. Denote a sentence consisting of n words as $\{w_1, w_2, \ldots, w_i, \ldots w_n\}$ where each word w_1 is mapped to the embedding representation $\mathrm{e}(\mathrm{w_i}) \in \mathrm{R^D}$.

Let $\mathrm{D1}, \mathrm{D2}, \mathrm{D3}$ be the width of the convolutional filters. We set $\mathrm{D1} = 1,\ \mathrm{D2} = 2$ and $\mathrm{D3} = 3$ for representing uni-grams, bi-grams and tri-grams, respectively. Taking D1 for example, the input of a linear layer is the concatenation of word embeddings in a fixed-length window size D1, which is denoted as $\mathrm{I_{1,i}} = [\mathrm{e}(\mathrm{w_i}); \mathrm{e}(\mathrm{w_{i+1}}); \ldots; \mathrm{e}(\mathrm{w_{i+D1-1}})]\, \varepsilon \mathrm{R^{DxDl}}$:

$$\mathrm{H_{1,i}} = \mathrm{W_1} \cdot \mathrm{I_{1,i}} + \mathrm{b_1} \tag{3.2}$$

where $\mathrm{W_1}\varepsilon \mathrm{R^{locxDxD1}}$ is the output size of the linear layer.

Average pooling layer is used to merge the varying number of outputs $\{\mathrm{H_{1,1}}, \mathrm{H_{1,2}}, \ldots, \mathrm{H_{1,n}}\}$ from convolution layer into a vector with fixed dimensions.

$$H_1 = \frac{1}{n} \sum_{i=1}^{n} \mathrm{H_{1,}}_i \tag{3.3}$$

To incorporate nonlinearity, an activation function tanh is used to obtain the output O1 of this filter.

$$\mathrm{O_1} = \tanh(\mathrm{H_1}) \tag{3.4}$$

Similarly, O2 and O3 are obtained for the other two convolutional filters with width 2 and 3, respectively. The outputs of three filters are lastly averaged to generate sentence representation.

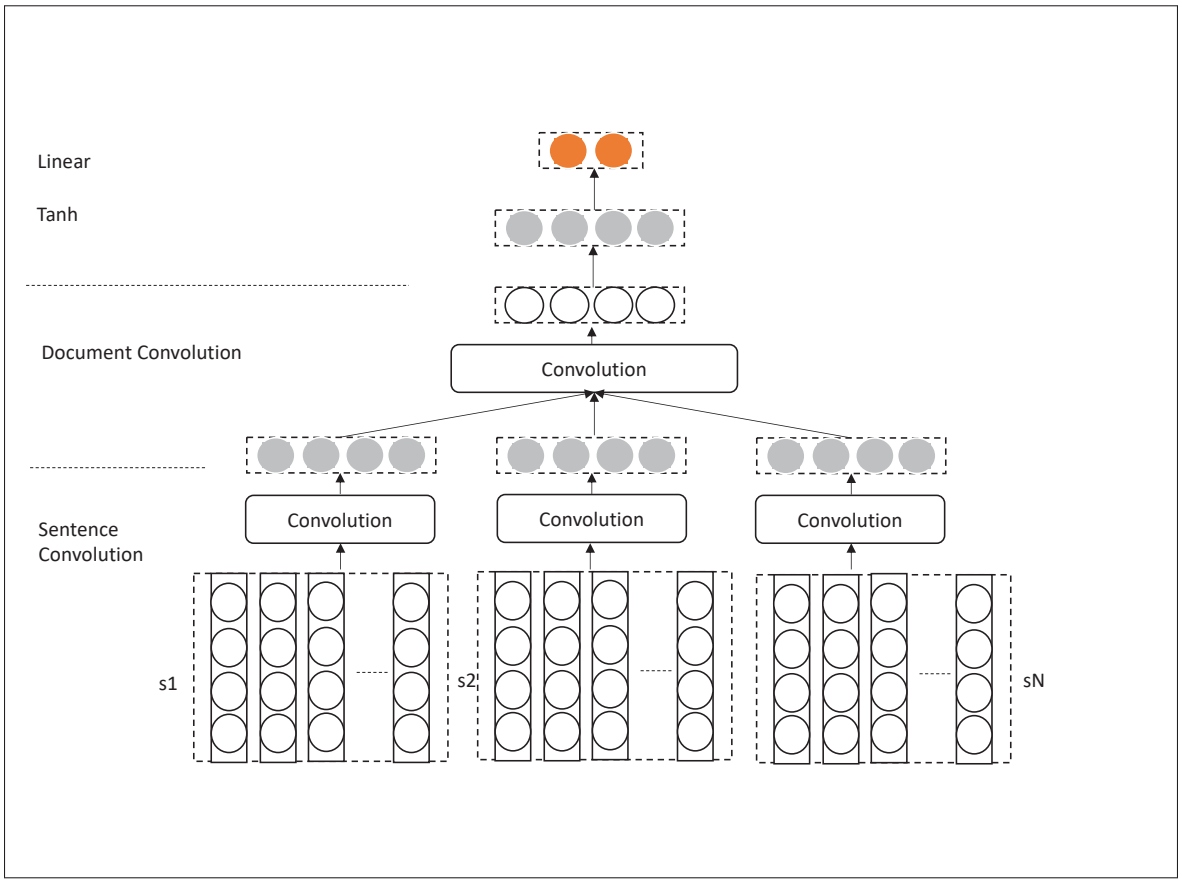


Figure 3.4 CNN Architecture for deceptive opinion spam detection
(Adapted from Li *et al.* (2017))

The document convolution transforms sentence vectors s1, s2, .., sm into a document vector. Finally, we use the document representation as features for identifying deceptive opinion spam. More specifically, a linear layer is added to transform the document vector into a real-valued vector, whose length is class number C. A softmax function is added to convert real vector to conditional probability for document classification.

3.4 Implementation

The model was implemented in Tensorflow, an open-source software library for dataflow programming across a range of tasks. Tensorflow can break up the graph of computations into

several chunks and run them in parallel across multiple CPUs and GPUs. Our computations for CNN model are run on CPU 3.1 GHz Intel Core i7.

We first build the vocabulary of our dataset and also a list that contains the number of words in each review. We then reconstruct the input vector representation by replacing each word in sentence by its Word2Vec representation with Tensorflow's embedding lookup function. The function takes in two arguments, one for the embedding matrix, and one for the ids of each of the words which is the input data in our case.

Next a convolution, max pooling and a tanh activation are created for each filter width. The result of each filter width is concatenated and then a dropout is added to that. We finally obtain the output: the vector representation of each of the sentence.

The document convolution layer is similar to the sentence convolution layer and the output of the document convolution layer gives us the document vector representation. On the top of the document convolution layer, we add a softmax classification layer to identify the deceptive spam review.

3.4.1 Hyper-parameters

Hyper-parameters for deep neural networks are the variables which determine the network structure e.g., number of hidden units and the variables which determine how the network is trained e.g., learning rate. Table 3.4 presents the hyper-parameters for our two-layer CNN opinion spam detection model.

Table 3.4 Hyper-Parameters of the Two-Layer CNN Model

Argument	**Explanation**	**Value**
wordVectors	The Word2Vec model	Google News dataset
embedding_size	The number of convolutional filters for the sentence convolution layer	300

Table 3.4 Hyper-Parameters of the Two-Layer CNN Model (continued)

Argument	**Explanation**	**Value**
filter_widths_sent_-conv	An array that contains the widths of the convolutional filters for the sentence convolution layer	[3,4,5]
num_filters_sent_conv	The number of convolutional filters for the sentence convolution layer	100
filter_widths_doc_conv	An array that contains the widths of the convolutional filters for the document convolution layer	[3,4,5]
num_filters_doc_conv	The number of convolutional filters for the document convolution layer	100
dropout_keep_prob	The probability of any given neuron's output to be preserved (as opposed to dropped, that is zeroed out.)	0.5
l2_reg_lambda	lambda value defined for l2 regularizer for final prediction layer	0
learning method	Kingma & Ba (2014)	ADAM
learning rate	The amount that the weights are updated during training	0.001
num_epochs	The number of complete passes through the training dataset	100
batch_size	The number of training samples to work through before the model's internal parameters are updated	32

3.4.2 Training and Evaluation Corpora

We used the review dataset depicted in sub-section 3.3.1 that consists of:

- 400 truthful positive and 400 truthful negative reviews crawled from App Store,
- 400 deceptive positive and 400 deceptive negative reviews generated as explained in Section 4.3.

Our dataset is split into 80% training (the sample of data used to fit the model), 10% validation (the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters) and 10% test (the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset) subsets.

3.4.3 Performance Measures

We use accuracy, precision, recall and F_1 as the evaluation metrics to evaluate the performance of opinion spam detection model Powers (2011). Accuracy, precision and recall are then defined as:

$$Accuracy = \frac{Tp + Tn}{Tp + Tn + Fp + Fn} \tag{3.5}$$

$$Precision = \frac{Tp}{Tp + Fp} \tag{3.6}$$

$$Recall = \frac{Tp}{Tp + Fn} \tag{3.7}$$

where Tp is the number of items correctly labeled as belonging to the positive class, Tn is the number of items correctly labeled as belonging to the negative class, Fp is the number of items where the model incorrectly predicts the positive class. And a Fn is an outcome where the model incorrectly predicts the negative class.

The F_1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$\text{F1} = 2^{\star}\left(\text{ Recall } {}^{\star} \text{ Precision}\right) / \left(\text{ Recall } + \text{ Precision }\right) \tag{3.8}$$

3.5 Experiments and Results

We compare the performance of our CNN based opinion spam detection model with a baseline SVM model using the same review dataset. The features extracted and then used in the SVM model are presented in Table 3.5.

Table 3.5 Text features extracted and used in the SVM model

Review-centric features	**Details**
Structural features	Length of the review, average word length, number of sentences, average sentence length, percentage of numerals, percentage of capitalized words.
POS percentages	The percentage of each POS tagging in each review.
Semantic features	The percentages of positive and negative opinion-bearing words in each review.
N-grams	Top 100 uni-grams and bi-grams that have the most different percentages (in terms of ratios) in fake and nonfake reviews
LIWC dimensions	Self-references (I, me, my), Social words, Overall cognitive words, Articles (a, an, the), Big words (> 6 letters)

Table 3.6 shows the results obtained from the SVM baseline method, the human evaluators as well as our CNN model. SVM model gets the best result with n-grams, structural and LIWC features, while the human evaluators accuracy to spot the deceptive opinion reviews are inferior not only to SVM model but also to our neural network model.

Table 3.6 SVM, Human Evaluators and CNN performance results

Model	**Features**	**Accuracy**	**P**	**R**	F_1
SVM	(n-grams + structural features)	0.6750	0.6842	0.65	0.6667
	(POS + n-grams)	0.5813	0.5844	0.5625	0.5732
	(n-grams + LIWC)	0.6938	0.7067	0.6625	0.6839

Table 3.6 SVM, Human Evaluators and CNN performance results (continued)

Model	**Features**	**Accuracy**	**P**	**R**	F_1
SVM	**(n-grams + structural features + LIWCs)**	**0.70**	**0.80**	**0.6667**	**0.7273**
	n-grams + LIWC + semantic features	0.6438	0.6625	0.6625	0.6503
Human Evaluators	Evaluator 1	0.7125	0.6932	0.7625	0.7262
	Evaluator 2	0.7250	0.7143	0.75	0.7317
CNN		**0.8250**	**0.8514**	**0.7875**	**0.8182**

Per our human evaluators assessment as presented in Table 3.7 and Table 3.8, only the 37.5% (in average) of the synthesized reviews are categorized as unrealistic (spam):

- Evaluator 1 has predicted 32 reviews as True Negative out of 80 spam reviews (%40),
- Evaluator 2 has predicted 28 reviews as True Negative out of 80 spam reviews (%35),

while, 21.875% (in average) of the truthful reviews are classified also unrealistic (spam) by the same evaluators in average:

- Evaluator 1 has predicted 19 reviews as False Positive out of 80 truthful reviews (%23.75),
- Evaluator 2 has predicted 16 reviews as False Positive out of 80 truthful reviews (%20),

On the other hand, our CNN model with optimized hyperparameters achieved 82.5% accuracy that gives better results by capturing relationships between local sentences.

3.6 Conclusion

App store user reviews are very diverse in form and most of the time no language structure is followed: people tend to use slangs, short forms and emoji. Therefore, it is difficult to use hand-crafted text features derived through tedious feature extraction and feature extraction processes

Table 3.7 Confusion Matrix for Evaluator 1 (N=160)

		Actual		
		Truthful	Spam	Total
Prediction	Truthful	61	48	109
	Spam	19	32	51
	Total	80	80	160

Table 3.8 Confusion Matrix for Evaluator 2 (N=160)

		Actual		
		Truthful	Spam	Total
Prediction	Truthful	64	52	116
	Spam	16	28	44
	Total	80	80	160

for use of traditional text classification algorithms. Hence, the deep learning techniques have potential for effective text classification problems like spam detection since these models work effectively on raw data by learning high level features on its own.

In the second phase of our research study, we proposed to use a novel convolutional neural network to learn document representation for deceptive spam review detection. Furthermore, we characterized a review dataset including truthful and spam reviews acquired by crawling App Store and synthesized automatically for the first time. Additionally, we do the comparison between neural network based method and baseline classification model, SVM, by making different feature combinations: from the reported experiments, our two-layer CNN model gives a higher Accuracy (82.5%) in comparison to the SVM classifier (70%).

Traditional machine learning methods require huge amount of training data and subsequently absence of enough training data brings a higher risk of overfitting, a common problem in machine learning. As we employ text synthesis method to generate reviews and use them for both training and test, our deep learning model might have memorized the peculiarities in the data that leads to our model to achieve super-human performance on the datasets.

Besides, deep learning models require a lot of hyper-parameter tuning, and the values of these hyper-parameters can make the difference between the state of the art and a nonfunctioning model. Tuning hyper-parameters in a deep learning application can be a challenging process given that it requires not only the execution of an experiment but the evaluation of the results against other versions of the model. Since we pursued a manual tuning process based on trial and error experiments to finalize the hyper-parameters, the exact unconditional P values for non-inferiority (superiority) of our DL model to other state of the art models could not be validated completely.

CHAPTER 4

AUTOMATED ASSESSMENT OF REVIEW HELPFULNESS

4.1 Introduction

User-supplied reviews are widely and increasingly used to enhance online retail and other websites. However, the quality of the reviews has usually suffered from the anonymity of reviewers and absence of repercussion of what they write. As the reviews are numerous and varying in quality, it becomes more critical to detect low-quality reviews and eliminate their possible negative effect on customers' purchasing decision Ghose & Ipeirotis (2011).

Online retailers also have an increasing interest in displaying high-quality or helpful reviews more prominently than low-quality ones. A key challenge when ranking reviews is to determine which reviews the customers find helpful. To tackle this challenge, some retailers such App Stores, Amazon.com, Yelp, etc. let users rate the reviews as Helpful or Unhelpful and then provide them the capability to sort the reviews according the categories such as Most Helpful, Most Recent, etc.

Helpfulness is often assessed manually by human evaluators who are asked for their opinion by posing the question: "Was this review helpful to you?", as shown in Figure 4.1. However, this kind of manual assessment suffers from a number of limitations including:

- **Biases:** Human review evaluators are subject to a number of biases. For instance, reviews with many positive helpfulness votes are displayed prominently: that results in being read by many users and receiving even more helpfulness votes. Hence, the "winner circle" phenomenon makes it hard for new reviews to be ranked appropriately Liu *et al.* (2007).
- **Sparseness:** Many reviews have received no helpfulness votes at all; many more have not received enough votes to reliably compute their helpfulness.

- **No Instant Evaluation:** After a review has been posted, it takes a significant amount of time until it has been evaluated by a sufficient number of other users. As a consequence, it is impossible to immediately rank a review in terms of its helpfulness.

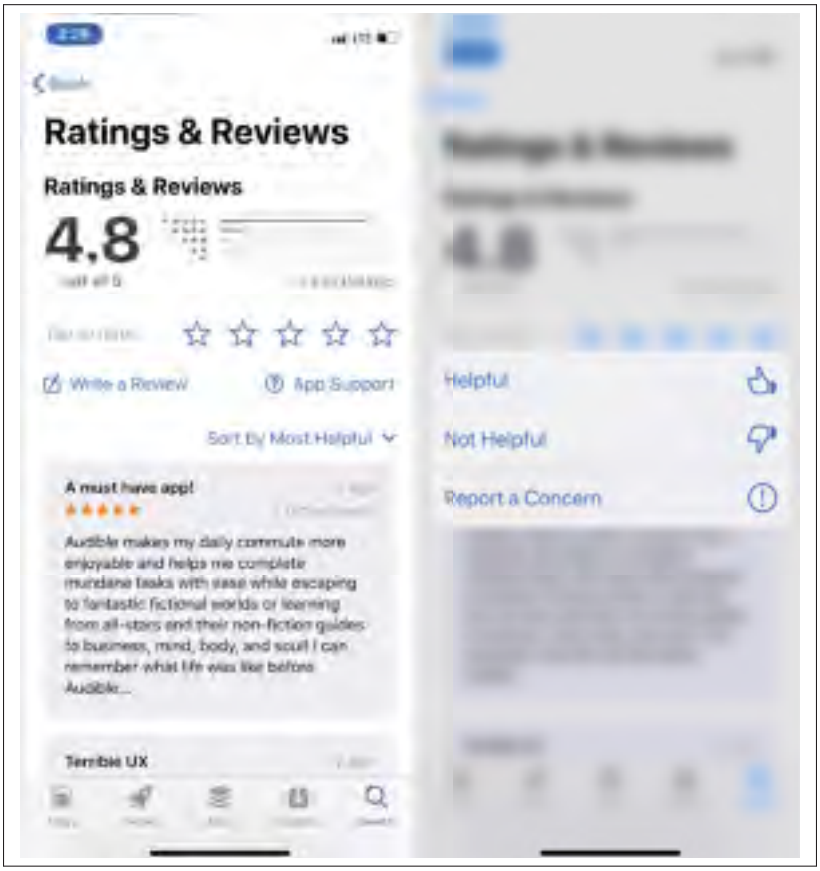


Figure 4.1 Review helpfulness rating option for an App Store app

On the other hand, user review helpfulness evaluation problem has been studied for various platforms such Amazon, Yelp, TripAdvisor; however, no study that assessed review helpfulness either manually or automatically for the mobile app store corpus has been identified in our systematic literature review presented in Section 1.3.3.

To overcome these limitations, we first populate a mobile app store user review dataset including helpful and non-helpful labelling. Afterwards, a binary review helpfulness classifier is trained and tuned using the grid parameter and k-fold validation for the review helpfulness

assessment task. The performance of the binary classification model predicting the helpfulness of reviews is improved by adding text features capturing the presence and occurrence frequencies of the app aspects (features) as given in app store description of the mobile app. Thus, app store regulators and user would be able to obtain a helpfulness assessment result automatically for every single review, at the moment it is posted and less affected by human biases.

This chapter which addresses the phase 3 of our research study is structured as follows: Section 4.2 presents the workflow of our methodology for automated assessment of review usefulness. Section 4.3 describes the details about pre-processing including dataset creation, dataset cleaning and feature generation steps. Section 4.4 presents the details of benchmarked binary classifiers of Logistic Regression, AdaBoosting, Gaussian Naive Bayes and Support Vector Machine (SVM). Section 4.5 evaluates the performance of the chosen binary classifier. Section 4.6 presents the implications of automated app store review helpfulness assessment model.

4.2 Methodology

Our research methodology for automated assessment of review helpfulness includes the main tasks: (i) Pre-Processing, (ii) Selection of Appropriate Classifier and (iii) Classifier Refinement steps as presented in Figure 4.2:

4.3 Pre-Processing

The pre-processing phase includes (i) dataset creation, (ii) dataset cleaning, (iii) feature generation steps as follows:

4.3.1 Dataset Creation

The dataset used for this part of the research is populated with the app store crawler developed in Section 2.2.1. In Apple App Store and Google Play, reviews could be retrieved sorted by their Helpfulness (Most Helpful), Post Date (Most Recent) and Rating. The crawlers are run

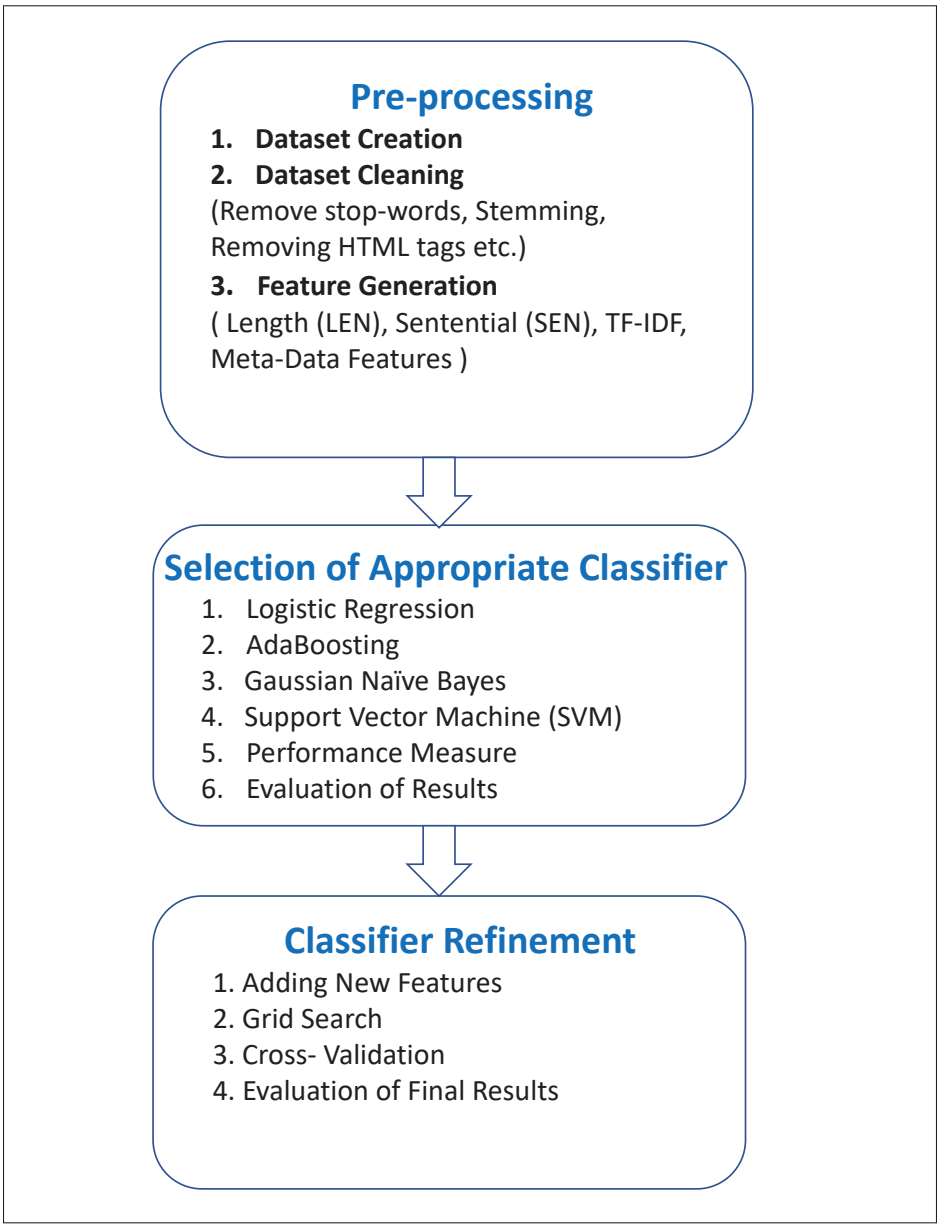


Figure 4.2 Automatic Assessment of Review Helpfulness Methodology (Phase 3)

twice: first to pull the most helpful reviews and then the most recent user reviews. Hence, the entire review dataset includes both Helpful and Non-Helpful reviews as tagged by users.

The data retrieved in JSON format is converted to a pandas data frame for the use of following sections. Each review in the data frame has the following information:

- id - ID of the review, e.g. 2182577253
- userName - userName of the reviewer, e.g. Mmmmmmmmmhhhhhhhhhhhhhhhhz
- userUrl - url for the reviewer
- version - app version, e.g. 1.118.0
- score - rating given by the user
- title - title of the review
- text - review body

4.3.2 Dataset Cleaning

The dataset is cleaned through the following steps: (i) noise removal e.g., whitespace, punctuation and non-text characters such as HTML tags, (ii) stemming, (iii) tokenization and (iv) stop-word removal. Afterwards, the data columns that will not be used as model features are removed, only the *reviewText* to generate text features, the *score* to generate the Meta-Data feature and the helpfulness score (0 or 1) to generate classification labels are kept.

The pre-processing steps are detailed as follows:

- **Noise Removal:** The text is converted to lower case and also whitespace and non-text characters such as HTML tags are removed.
- **Stemming:** Stemming means removing morphological affixes from words, leaving only the word stem e.g., words such as "download" and "downloading" would both be represented as "download" that allows classification algorithms to be more accurate at finding trends in the meanings of sentences and stemming also reduces the total amount of features generated.
- **Tokenization:** Token is a single entity (words) that is building blocks for sentence or paragraph. Given a character sequence and a defined document unit, tokenization is the task of

chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation.

- **Stop-word removal:** A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. Natural Language Toolkit (NLTK) Loper & Bird (2002) English corpus stop-word list is used to remove the stop-words from review corpus along with tokenization.

4.3.3 Feature Generation

Machine learning approaches heavily rely on features selection apart from the choice of machine learning algorithm and data. Therefore, in machine learning based opinion mining and sentiment analysis studies, it is necessary to convert a piece of text into a feature vector in order to represent the most salient and important features available.

In this part of our research, various features organized in three classes are extracted: Structural, Lexical, and Meta-data.

- **Structural Features:** Structural features are observations of the document structure and its formatting:
  - **Length (LEN):** *The total number* of tokens in a syntactic analysis of the review.
  - **Sentential (SEN):** Observations of the sentences, including *the number of sentences*, *the average sentence length*, *the percentage of question sentences*, and *the number of exclamation marks*.
- **Lexical Features:** Lexical features capture the words observed in the reviews and two sets of lexical features are examined in this research study:
  - **Uni-gram (UGR):** The *Term Frequency – Inverse Document Frequency (TF-IDF)* statistic of each word occurring in a review.

- **Bi-gram (BGR):** The *Term Frequency – Inverse Document Frequency (TF-IDF)* statistic of each bigram occurring in a review.

  Term Frequency – Inverse Document Frequency (TF-IDF) features that is the product of two statistics, term frequency (Equation 4.1) and inverse document frequency (Equation 4.2) as given in Equation 4.3.

$$\text{tf}(t,d) = f_{t,d} \tag{4.1}$$

$$\text{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{4.2}$$

$$\text{tidf}(t,d,D) = \text{tf}(t,d) \cdot \text{idf}(t,D) \tag{4.3}$$

  with:

  - N: total number of documents in the corpus $N = |D|$,
  - $|\{d \in D : t \in d\}|$ number of documents where the term t appears (i.e.,$\text{tf}(t,d) \neq 0$) . If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

4.4 Selection of Appropriate Classifier

At this step of the research, the review data is shuffled and split the data into 80% training and 20% testing that allows the simulated evaluation of how well the model is performing before using it in the real world to make predictions. Afterwards, the performances of different binary classifiers are evaluated for the features generated in Section 4.3.3.

The following sub-sections present the details of chosen binary classifiers:

- Logistic Regression

- AdaBoosting
- Gaussian Naive Bayes
- Support Vector Machine (SVM)

4.4.1 Logistic Regression

Logistic Regression is one of the most used machine learning algorithms for binary classification that outputs the probability of occurrence of an event as its prediction Hosmer Jr *et al.* (2013).

Logistic regression can be expressed as:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X \tag{4.4}$$

where, the left hand side is called the $logit$ or $log-odds$ function where $p(x)/(1-p(x))$ is called as odds. The odds signifies the ratio of probability of success to probability of failure. Therefore, in Logistic Regression, the linear combination of inputs are mapped to the $log(odds)$ where the output being equal to 1.

The inverse of the function Equation 4.4 gives a *Sigmoid function* with a S-shaped curve that always gives a probability ranging from $0 < p < 1$:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \tag{4.5}$$

The regression coefficients are usually estimated using maximum likelihood estimation as expressed in the following function:

$$\mathscr{L}(\beta;y) = \prod_{i=1}^{N} \left(\frac{\pi_i}{1-\pi_i} \right)^{y_i} (1-\pi_i) \tag{4.6}$$

4.4.2 AdaBoosting

AdaBoost classifier Freund & Schapire (1995) is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus on more difficult cases.

Given a data set containing n points, where -1 denotes the negative class, while 1 represents a positive one:

$$x_i \in \mathbb{R}^d, y_i \in \{-1, 1\} \tag{4.7}$$

Initialize the weight for each data point as:

$$w(x_i, y_i) = \frac{1}{n}, i = 1, \ldots, n \tag{4.8}$$

For iteration m=1,. . . ,M:

(1) Fit the weak classifiers to the data set and select the one with the lowest weighted classification error:

$$\varepsilon_m = E_{w_m}\left[1_{y \neq f(x)}\right] \tag{4.9}$$

(2) Calculate the weight for the m_th weak classifier:

$$\theta_m = \frac{1}{2} \ln \left(\frac{1-\varepsilon_m}{\varepsilon_m} \right) \tag{4.10}$$

(3) Update the weight for each data point as:

$$w_{m+1}(x_i, y_i) = \frac{w_m(x_i, y_i)\exp[-\theta_m y_i f_m(x_i)]}{Z_m} \tag{4.11}$$

where Z_m is a normalization factor that ensures the sum of all instance weights is equal to 1.

After M iterations, the final prediction is obtained by summing up the weighted prediction of each classifier.

4.4.3 Gaussian Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable Zhang *et al.* (2015). Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n :

$$P(y|x_1,\ldots,x_n) = \frac{P(y)P(x_1,\ldots x_n|y)}{P(x_1,\ldots,x_n)} \tag{4.12}$$

Using the naive conditional independence assumption that:

$$P(x_i|y,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = P(x_i|y) \tag{4.13}$$

for all i, this relationship is simplified to

$$P(y|x_1,\ldots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,\ldots,x_n)} \tag{4.14}$$

Since $P(x_1,\ldots,x_n)$ is constant given the input, the following classification rule is used:

$$P(y|x_1,\ldots,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \tag{4.15}$$

$$\hat{y} = \arg\max_{y} P(y)\prod_{i=1}^{n} P(x_i|y) \tag{4.16}$$

and we can use Maximum A Posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$, the former is then the relative frequency of class y in the training set. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

The likelihood of the features is assumed to be Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \tag{4.17}$$

The parameters σ_y and μ_y are estimated using maximum likelihood.

4.4.4 Support Vector Machine (SVM)

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. A Support Vector Machine (SVM) performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors Vapnik (1995).

SVM training involves the minimization of the error function:

$$\frac{1}{2}w^T w + C\sum_{i=1}^{N} \xi_i \tag{4.18}$$

subject to the constraints:

$$y_i \left(w^T \phi \left(x_i \right) + b \right) \geq 1 - \xi_i \text{ and } \xi_i \geq 0, i = 1, \ldots, N \tag{4.19}$$

where C is the capacity constant, w is the vector of coefficients, b is a constant, and represents parameters for handling non-separable data (inputs). The index i labels the N training cases. Note that $y \in \pm 1$ represents the class labels and x_i represents the independent variables.

The kernel is used to transform data from the input (independent) to the feature space. It should be noted that the larger the C, the more the error is penalized. Thus, C should be chosen with care to avoid over fitting.

4.4.5 Performance Measure

The area under the Receiver Operating Characteristic (ROC) curve, or Area Under Curve (AUC), is used as the measure of performance of our supervised classification models.

The ROC curve is created by plotting the True Positive rate (TPR) against the False Positive rate (FPR) at various threshold settings. The probabilistic interpretation of ROC score is that if a positive case and a negative case are randomly chosen, the probability that the positive case outranks the negative case according to the classifier is given by the ROC. An example of a plot of a ROC curve for a specific class is represented in Figure 4.3.

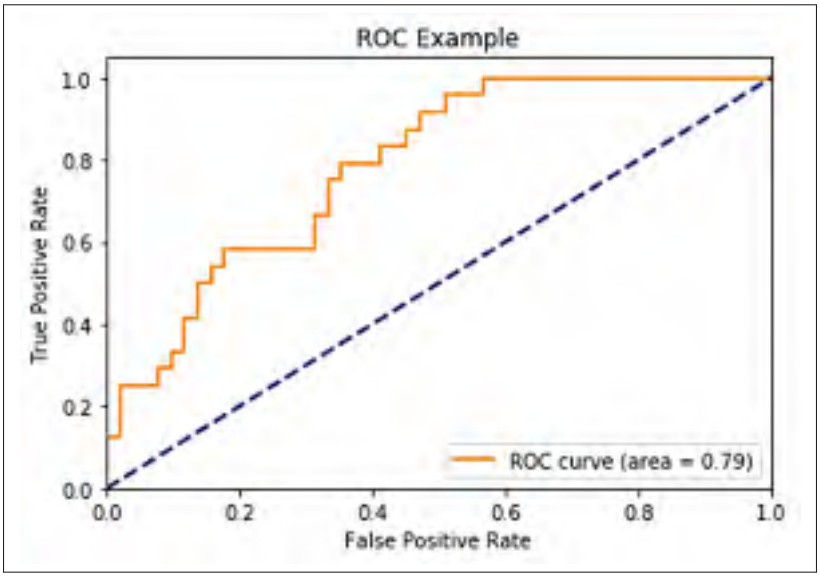


Figure 4.3 ROC Curve example for a specific class

4.4.6 Evaluation Results

Each of the algorithms listed above in Section 4.4 are tested using the train/test split (%80 / %20) and five different sizes of data (1000, 5000, 10000, 15000, 21064) and the results are given in Figure 4.4.

Per classification results summarized in Table 4.1, the Logistic Regression is the best algorithm in terms of ROC for the test sizes 1000, 5000, 10000, 15000 and 21064. However, its final score for the area under the ROC curve = 0.7588 is the same with Support Vector Machine (SVM) model.

The training speed and prediction speed:

- 0.1924 seconds and 0.0032 seconds respectively for Logistic Regression with a sample size of 21064.
- 1723.9000 seconds and 42.6865 seconds respectively SVM with a sample size of 21064.

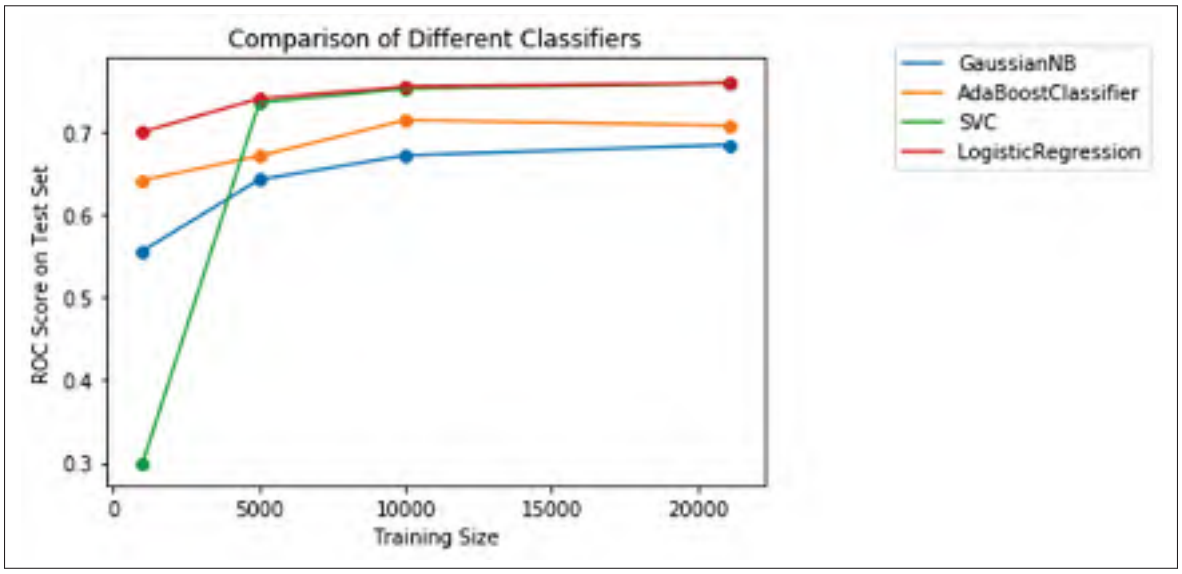


Figure 4.4 Comparison of Different Review Helpfulness Classification Models

As the automatic review helpfulness assessment system needs to consider the trade off between accuracy and speed, the Logistic Regression algorithm seems to be the ideal model for our benchmark. Surprisingly, the Gaussian Naive Bayes algorithm did quite poorly in our tests and is less accurate than the AdaBoost algorithm.

Table 4.1 Benchmarking Results for Binary Classifiers

Model	**Training Size**	**Training Time (sec)**	**Prediction Time (sec)**	**ROC_AUC for Training Set**	**ROC_AUC for Test Set**
GaussianNB	1000	0.0135	0.0603	0.8778	0.5560
	5000	0.0618	0.0650	0.8147	0.6423
	10000	0.1194	0.0498	0.7597	0.6715
	21064	0.2367	0.1345	0.7385	0.6842
AdaBoost	1000	0.4974	0.2315	0.9693	0.6415
	5000	3.3465	0.0898	0.8357	0.6708
	10000	6.2303	0.1450	0.7916	0.7141
	21064	11.9485	0.1406	0.7621	0.7076
SVM	1000	3.8297	3.1750	0.0802	0.2995
	5000	76.9110	12.5610	0.8166	0.7353
	10000	325.0635	30.2081	0.8006	0.7522

Table 4.1 Benchmarking Results for Binary Classifiers (continued)

Model	**Training Size**	**Training Time (sec)**	**Prediction Time (sec)**	**ROC_AUC for Training Set**	**ROC_AUC for Test Set**
SVM	21064	**1723.90**	**42.6865**	0.7876	**0.7588**
Logistic Regression	1000	0.0038	0.0038	0.9612	0.6999
	5000	0.0281	0.0035	0.8580	0.7402
	10000	0.0619	0.0036	0.8226	0.7546
	21064	**0.1924**	**0.0032**	0.7984	**0.7588**

4.5 Experiments

This section gives the details of new features added to the existing feature set, grid search procedure to generate all possible parameter combinations and cross-validation results obtained with (i) new feature set and (ii) optimized model parameters.

4.5.1 Adding Meta-data Features

In order to improve our results, our feature set is updated by adding the following meta-data features:

- **App Features:** The features of apps that are written in app store description and occur in the review text , e.g., stream music, connect and share for Youtube. This feature counts the number of lexical matches that occur in the review for each app feature.

- **General-Inquirer (GIW):** Positive and negative sentiment words describing products or product features (e.g., “easy to share music” and “amazing design”). The intuition is that reviews that analyze product features are more helpful than those that do not. We try to capture this feature by extracting sentiment words using the publicly available list of positive and negative sentiment words from the General Inquirer Dictionaries [1].

[1] http://www.wjh.harvard.edu/ inquirer/homecat.htm

4.5.2 Grid Search

The grid search technique works by generating a grid of all possible provided parameter combinations. It then evaluates a model using a validation set based on every combination of parameters in the grid. It is used to find the optimum set of parameters for a learning algorithm given a data set.

The k-folds validation training technique creates multiple testing and training sets and trains a model on each, averaging the results. The splitting method is to divide the data into separate bins (for example k = 5), train on bins 1 to k-1 and test on bin k. The next fold trains on bins 2 to k and tests on bin 1 and so on, until all of the bins have acted as a test bin. This effectively trains and tests the models on all of the data without over-fitting to the data.

The best classifier is tuned using a parameter grid and tested with k=5-fold validation. Afterwards, the obtained results are compared to the label results generated in Section 4.4.6 using the AUC_ROC.

4.5.3 Results

Per 5-fold cross-validation results, our optimized logistic regression model using Structural + Lexical and Meta-Data Features was able to score a value of 0.8158 for the area under the ROC curve. Compared to initial benchmark model scoring 0.7588, 6% improvement was achieved with grid-search and addition of meta-data features. Figure 4.5 presents the ROC curves for optimized logistic model with and without meta-data features. Optimized logistic model with structural, lexical features obtained the area under the ROC value = 0.7781, while same model was able to score a value of 0.8158 for the area under the ROC curve with addition of meta-data features.

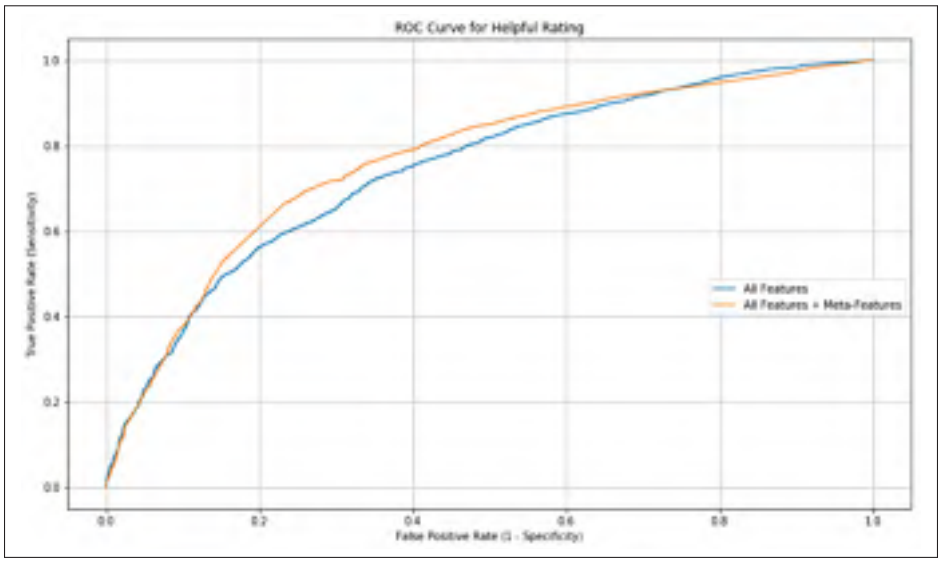


Figure 4.5 ROC curve for Review Helpfulness evaluation model with meta-data features

4.6 Conclusion

Online product review helpfulness modelling is a multi-faceted task that involves using text content and context information to understand the components of the helpfulness. Although significant advances have been made on finding hand-crafted features for helpfulness prediction, effective comparisons between proposed approaches have been hindered by the absence of standard evaluation datasets, well-defined baselines, and feature extraction studies.

In this third phase of our research study, we built (i) a dataset including app store reviews with helpful and non-helpful labels and (ii) a baseline binary review helpfulness classifier that can help app store regulators to understand the components of review helpfulness. In addition, the model provides an automated review assessment capability for every single review regardless of its publication date and time and its given helpfulness votes for the use of both mobile app users, app store owners and regulators.

As indicated in results, adding app related and GIW features to the feature set improved the performance of our classifier 6% compared to initial benchmark model. The evoked ROC gain

in prediction performance shows that the distribution of mobile app aspect words in the review text is an indicator of the perceived helpfulness of the review by users.

CHAPTER 5

EXTRACTION OF MOBILE APP FEATURES

5.1 Introduction

Sentiment analysis, known as opinion mining, is a NLP research topic that has received much attention in the past few years. With the advent of user-generated content as a rich source of subjective information, there has been active research to analyze and classify the nature and opinion polarity of written text to fine-grained levels. Aspect extraction is one of the key tasks in fine-grained sentiment analysis that aims to extract entity aspects on which opinions have been expressed Liu (2012).

Previous works for aspect extraction can be categorized into three: frequency-based, supervised, and unsupervised approaches. The most well-known approach featuring the frequency-based approach for aspect detection task was proposed by Hu & Liu (2004) to find the most comprehensive reviews with respect to a certain aspect by their frequencies. However, a major shortcoming of most of the frequency-based methods is that nouns and noun phrases that naturally have a high frequency can be mistakenly seen as aspects. Scaffidi *et al.* (2007) addressed this limitation by comparing the frequency of a prospective aspect with baseline statistics gathered from a corpus of 100 million words.

Supervised learning approaches that are characterized by the use of classifiers built from linguistic resources within the text have dominated the research landscape for some time. A substantial number of studies in SemEval 2014 [1] and SemEval 2016 [2] leveraged popular classifiers such as Maximum Entropy (ME), Conditional Random Field (CRF) and Support Vector Machine (SVM) for aspect and sentiment polarity detection tasks. While supervised learning approaches are quite efficient to solve aspect based sentiment analysis problems, they have certain weaknesses including the requirement of relatively large annotated datasets, reliance

[1] alt.qcri.org/semeval2014/

[2] http://alt.qcri.org/semeval2016/

on manually crafted features and annotated training data and non-replicable results for other domains Fernández-Gavilanes *et al.* (2016).

Most of unsupervised machine learning approaches proposed for aspect based sentiment analysis use LDA, a topic model proposed by Blei *et al.* (2003) and its variants Titov & McDonald (2008), Brody & Elhadad (2010). One of the main drawbacks of LDA models is that the generated topics are unlabeled, preventing a direct correspondence between topics and specific aspects or entities. In addition, since LDA was initially designed to operate on the document level, employing it for fine-grained aspect based sentiment analysis is not a straightforward task Schouten & Frasincar (2016).

Deep learning (DL) architectures and algorithms have already made impressive advances in the fields of computer vision and pattern recognition. Following this trend, NLP research is now increasingly focusing on the use of new deep learning methods. In the past few years, neural networks based dense vector representations have been producing superior results on various NLP tasks Collobert *et al.* (2011), Lample *et al.* (2016), Yin & Schütze (2015), Rush *et al.* (2015). Deep neural network models can learn text representation from data without careful engineering of features and capture semantic relations between aspects and context words in a more scalable way than feature-based supervised models Young *et al.* (2017).

Mobile app stores allow users to provide feedback in form of ratings and reviews towards installed apps, which actually serve as an effective communication channel between app developers and users. Mobile app stores receive enormous amounts of reviews every day making the manual analysis unfeasible and it is necessary to develop automated methods for mining the review text. App store opinion mining techniques usually rely on the textual attributes of user reviews to classify them into fine-grained aspects such as *feature requests* and *bug reports*.

The techniques applied to app store user review mining range from detecting the presence and absence of certain terms (e.g.,'crash','bug') to more computationally expensive methods that rely on topic modelling and supervise or unsupervised classification techniques. However, the approaches proposed for mobile app aspect extraction task do not usually consider the nature

of the review text and the relevance of the extracted features has not been cross-validated with the main software engineering concepts Genc-Nayebi & Abran (2017). Considering the huge amount of user reviews and the distinctive features of the review text text (e.g., short length, unstructured phrases, colloquial language and abundant information), there is a certain need to leverage deep neural networks in order to extract targeted app aspects.

App Store aspect extraction studies use different techniques, review datasets and annotation guidelines; therefore, the results reported in earlier studies are not directly comparable to each other. In addition, when the extracted aspects contain many short and frequent app aspects that are easy to detect but not enough informative not only for the developers but also for the app users, the evaluation results would be artificially high. A large amount of manually annotated corpora is available for fine-grained sentiment analysis and opinion mining in other domains e.g., Amazon reviews by Hu & Liu (2004), movie reviews by Maas *et al.* (2011) and SemEval datasets. However, there is no standard app review dataset that clearly identifies the mobile app aspects.

To address these challenges, we first annotate a review dataset that follows the dataset annotation guideline of SemEval Pontiki *et al.* (2015). The annotated aspects are based on ISO/IEC 25010:2011, the quality model that determines which quality characteristics will be taken into account when evaluating the properties of a software product. Second, we employ two deep neural net models: (i) Bidirectional Long Short Term Memory (bi-LSTM) Recurrent Neural Network (RNN) with Conditional Random Fields (CRF), (ii) a deep Convolutional Neural Network (CNN) model with CRF that can generalize well even when limited training data is available.

This chapter which addresses the phase 4 of our research study is structured as follows: Section 5.2 presents the necessary definitions and related work. Section 5.3 presents the details of our research methodology and aspect extraction models. Section 5.4 presents the details for implementation. Section 5.5 reports the experiment results. Lastly, Section 5.6 presents the implications of our mobile app feature extraction model.

5.2 Definitions and Related Work

This section first presents the definitions adopted in this chapter, followed by the related work.

5.2.1 Definitions

Functional Requirement: specifies a function that a system or system component must be able to perform.

Non-Functional Requirement: are any other requirement than functional requirements. These are the requirements that specify the criteria that can be used to judge the operation of a system, rather than specific behaviours.

Functional Suitability: This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is composed of the following sub-characteristics: (i) Functional completeness, (ii) Functional correctness, (iii) Functional appropriateness. (ISO/IEC 25010:2011)

Performance Efficiency: This characteristic represents the performance relative to the amount of resources used under stated conditions. This characteristic is composed of the following sub-characteristics: (i) Time Behaviour, (ii) Resource Utilization, (iii) Capacity. (ISO/IEC 25010:2011)

Compatibility: Degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment. (ISO/IEC 25010:2011)

Usability: The extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. (ISO/IEC 25010:2011)

Reliability: Degree to which a system, product or component performs specified functions under specified conditions for a specified period of time. This characteristic is composed of the following sub-characteristics: (i) Maturity, (ii) Availability, (iii) Fault tolerance, (iv) Recoverability. (ISO/IEC 25010:2011)

Portability: Degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another. This characteristic is composed of the following sub-characteristics: (i) Adaptability, (ii) Installability, (iii) Replaceability. (ISO/IEC 25010:2011)

Praises and Complaints: We refer to a positive sentence as a praise. The sentence might have a positive connotation with supplemental information, answering the question of why a topic or aspect is positive. Similarly, We refer to a negative sentence as a complaint.

Feature Request: denotes a request to add a new function to the software or to modify the existing functionality.

Bug Reports: describe problems with the app which should be corrected, such as a crash or an erroneous behaviour.

5.2.2 Related Work

Several sentiment analysis competitions have been organized in the context of workshops and conferences focusing on different sentiment analysis problems in recent years. Such competitions provide training datasets and the opportunity for direct comparison of different approaches on common test sets Pontiki *et al.* (2014), Pontiki *et al.* (2015), Pontiki *et al.* (2016). Aspect Based Sentiment Analysis (ABSA) detects fine-grained opinions expressed about different aspects of a given entity, on user-generated comments. The need for identifying aspect terms and their respective polarity have given rise to research in ABSA. Following this particular interest and sentiment analysis competitions, the technology performing ABSA becomes more and more mature.

Most of the systems dedicated to ABSA use machine learning algorithms e.g., Conditional random fields (CRFs) Toh & Wang (2014), Liu *et al.* (2015) and SVMs Chang & Lin (2011) which are often combined with semantic lexical information, n-gram models, and sometimes more fine-grained syntactic or semantic information. Both of these approaches have their own limitations: they are in need of a large number of features to work well. The syntactic or semantic features need to be crafted by hand, and they crucially depend on the grammatical accuracy and completeness of the sentences.

Deep neural networks have been applied to learn better features for supervised aspect extraction, e.g., using LSTM first introduced by Hochreiter & Schmidhuber (1997) and attention mechanism Wang *et al.* (2016). Convolutional Neural Networks (CNN), a category of neural networks, have become more popular in the past few years. The approaches using deep CNNs showed significant performance improvement over the state-of-the-art methods on the tasks such as part-of-speech (POS) tagging, semantic role labeling and sentence and text classification Collobert *et al.* (2011), Kim (2014), Zhang *et al.* (2015), Gehring *et al.* (2017). For ABSA task, Poria *et al.* (2016) used CNN together with manual features to extract aspect terms. Liu *et al.* (2015) proposed a general class of discriminative models based on recurrent neural networks (RNNs) and word embeddings. Xu *et al.* (2018) enhanced a pure CNN-based sequence labeling model with double embedding mechanism (general-purpose embeddings and domain-specific embeddings) for aspect extraction.

Similar to conventional online markets (e.g., Amazon and eBay) and customer review sites (e.g., Yelp, Tripadvisor), app stores enable their customers to share their app experience in the form of user reviews and star ratings. Realizing the technical and business value of app stores feedback, research on mining user reviews in mobile app stores has noticeably advanced in the past few years. The main objective of this line of research is to extract useful information that can help app developers in software maintenance and release planning tasks.

Various methods and tools are proposed to automatically analyze user reviews with the aim of acquiring valuable information from the app store user reviews. However, the aspects or topics

extracted from user reviews vary depending on the choice and scope and also the methodology used by the researcher. (See Table 1.3, Appendix I). We are aware of two studies that have classified user reviews in accordance with quality aspects, and these researches by Groen *et al.* (2017) and Lu & Liang (2017) were published nearly simultaneously to our SLR, presented earlier in Section 1. Groen *et al.* (2017) used manual tagging and language patterns to identify quality attributes, while Lu & Liang (2017) did feature engineering and leveraged three machine learning algorithms Naive Bayes, J48, and Bagging to classify user reviews.

In addition, we identified two published annotation guidelines associated with annotated app review datasets. Guzman & Maalej (2014) annotation guideline defines an app feature as (i) a description of specific app functionality visible to the user (such as uploading files or sending emails), (ii) a specific screen of the app, (iii) a general quality of the app (such as time needed to load or size of storage) or (iv) a specific technical characteristic (e.g. a network protocol or HTML5). The annotation guideline prepared by Sänger *et al.* (2016) is in German and the annotated aspects in the study are both app features, subjective phrases and relationships between them. On the other hand, since we target to identify ISO/IEC 25010:2011 quality attributes within the review text, the annotated validation dataset proposed by and Sänger *et al.* (2016) could not be used in our study.

5.3 Methodology

Our research methodology for extraction of mobile app aspects includes the main research steps of: (i) collection of review dataset, (ii) filtering of non-helpful and spam reviews with the pre-trained models proposed in Chapter 4 and Chapter 5, (iii) annotation of review dataset, (iv) aspect extraction with Bi-LSTM +CRF and a Deep CNN+CRF models as presented in Figure 5.1. The details of each step are presented in the following sub-sections.

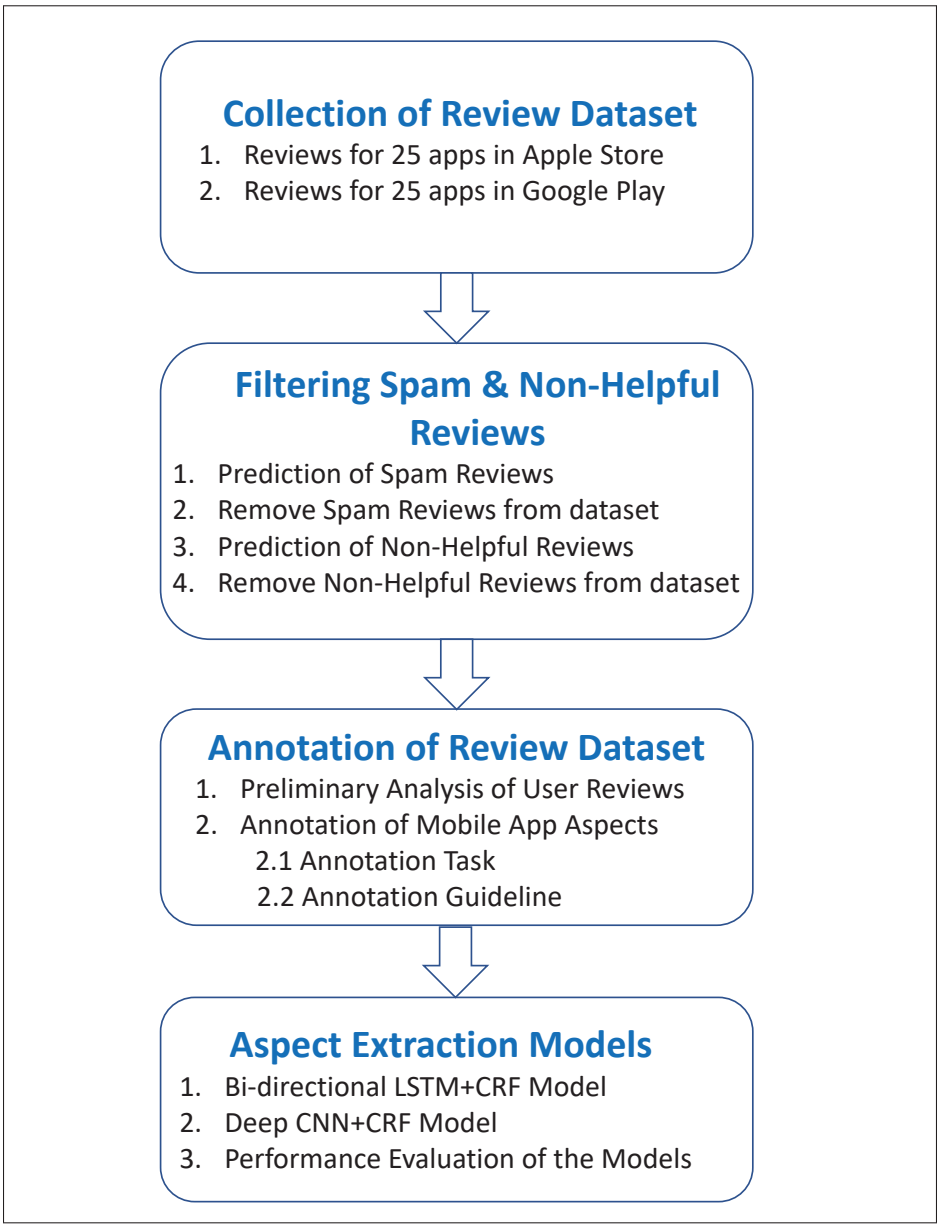


Figure 5.1 Extraction of Mobile App Features Methodology (Phase 4)

5.3.1 Collection of Review Dataset

Our review dataset consists of user reviews that are collected from Apple App Store and Google Play. Both app stores categorize mobile apps and games into different categories such as *Books*, *Business*, *Entertainment*, etc. based on main function or subject matter of the app. As of

the date of this writing, App Store has 25 different categories, while Google Play offers 30 categories of apps and 17 categories for games.

We chose 25 apps by random sampling from each app category in Apple App Store, Canada. On the other hand, 24 apps are sampled out of 320 apps (from 32 app categories) and 1 game is sampled out of 170 games (from 17 game categories) that makes 25 apps in total for Google Play, Canada. Table 5.1 and Table 5.2 show category, name, version, price tag (free or paid), total number of English reviews as of December, 31 2018 and number of reviews retrieved for the selected apps.

Table 5.1 List of App Store apps (N=25)

ID	**Category**	**Name**	**Version**	**Free or Paid**	**Total Reviews (English)**	**#Reviews Retrieved**
1	Overall	Fitbit	2.84	Free	99,600	20,000
2	Overall	Instagram	75.0	Free	846,658	20,000
3	Books	Google Play Books	5.1.1	Free	4,060	4,060
4	Business	Indeed Job Search	11	Free	15,661	15,661
5	Education	Lumosity: Daily Brain Games	9.50.1	Free	31,446	20,000
6	Entertainment	Face Secret – Face Reader 2019	1.5.5	Free	1,176	1,176
7	Finance	Debt Manager	1.7.1	Paid	686	686
8	Food & Drink	SkipTheDishes - Food Delivery	3.9.11	Free	1,984	1,984
9	Games	Fortnite	7.10.1	Free	279,302	20,000
10	Health & Fitness	vívofit jr	3.3	Free	1,020	1,020
11	Lifestyle	Live Wallpapers Now	3.0.0	Free	10,040	10,040
12	Magazines & Newspapers	Black Box - Movie Listing	2.6.10	Free	220	220
13	Medical	Maple - 247 Online Doctors	3.8.0	Free	56	56
14	Music	FL Studio Mobile	3.2.05	Paid	1,324	1,324
15	Navigation	Google Maps - Transit & Food	5.7	Free	156,979	20,000
16	News	CBC News	4.2.11	Free	4,547	4,547
17	Photo & Video	Facetune	2.7.3	Paid	14,045	14,045

Table 5.1 List of App Store apps (N=25) (continued)

ID	Category	Name	Version	Free or Paid	Total Reviews (English)	#Reviews Retrieved
18	Productivity	Fantastical 2 for iPhone	2.10.3	Paid	9,842	9,842
19	Reference	Bible	8.7.1	Free	355,610	20,000
20	Shopping	Flipp - Weekly Shopping	9.3.1	Free	12,665	12,665
21	Social Networking	Messenger	196.0	Free	344,445	20,000
22	Sports	Hockey Scoreboard - Universal Hockey Scorekeeping	2.0.5	Paid	130	130
23	Travel	MarineTraffic - Ship Tracking	3.9.2	Paid	3,626	3,626
24	Utilities	AdBlock	4.1	Paid	7,706	7,706
25	Weather	CARROT Weather	4.9	Paid	1,716	1,716

Table 5.2 List of Google Play Apps (N=25)

ID	Category	Name	Version	Free or Paid	Total # Reviews (English)	# Reviews Retrieved
26	Overall	Netflix	2018.12.13	Free	481,876	20,000
27	Casual-Games	Candy Crush Saga	1.140.0.5	Free	1,680,209	20,000
28	Art & Design	ibis Paint X	5.5.5	Free	2,733	2,733
29	Beauty	Hairstyle Try On - Hair Styles and Haircuts	5.1	Free	66	66
30	Books & Reference	YouVersion Bible App + Audio & Daily Verse	8.7.0	Free	119,825	20,000
31	Business	Uber Driver	2018.12.18	Free	57,092	20,000
32	Comics	How To Draw Comics	1.0.11	Free	981	981
33	Communications	WhatsApp Messenger	2018.12.13	Free	4,720,021	20,000
34	Education	Toca Life: Neighborhood	1.0.1-play	Paid	346	346
35	Events	Ticketmaster Event Tickets	1.27.2	Free	12,717	12,717
36	Finance	RBC Mobile	2018.12.10	Free	10,825	10,825

Table 5.2 List of Google Play Apps (N=25) (continued)

ID	Category	Name	Version	Free or Paid	Total # Reviews (English)	# Reviews Retrieved
37	House & Home	Universal Smart TV / IR TV Remote Control PREMIUM	1.0.16	Paid	41	41
38	Libraries & Demo	SYMA-FPV	5.2	Free	562	562
39	Lifestyle	Hue Pro	2.4.11	Paid	918	918
40	Maps & Navigation	BackCountry Navigator TOPO GPS PRO	2018.12.21	Paid	2,702	2,702
41	Medical	Monash Uni Low FODMAP Diet	2.0.7	Paid	544	544
42	Music & Audio	djay 2	2.3.4	Paid	2,147	2,147
43	Parenting	FamilyAlbum - Easy Photo & Video Sharing	2018.12.27	Free	620	620
44	Personalization	ZEDGE Ringtones & Wallpapers	2018.12.19	Free	179,198	20,000
45	Productivity	Plague Inc: Scenario Creator	1.1.6	Paid	373	373
46	Shopping	Kijiji: Buy, Sell and Save on Local Deals	2018.12.19	Free	25,379	20,000
47	Sports	iHunter Ontario	2.0.41	Paid	46	46
48	Tools	Google Translate	2018.11.08	Free	258,018	20,000
49	Travel & Local	Road to Hana GyPSy Drive Tour	2.2	Paid	266	266
50	Video Players & Editors	Video & TV Cast | Ultimate Edition	1.1	Paid	246	246

The reason of populating the review dataset through different app stores, categories and price points is to evaluate our approach against reviews that contain diverse vocabularies, describing different app aspects, and written by different users. Since users of a broad spectrum of mobile apps have different expectations and interactions with technology in various manners, they would possibly express their opinions and experiences in different ways. In addition, choosing popular apps from top charts increases the probability that the human annotators would be familiar with the apps that reduces the manual feature extraction effort and minimize the errors.

Given that popular apps are also more likely to have more reviews, an automated analysis for these apps would provide more realistic and useful aspects.

As earlier explained in Section 2.2.1, we collect review id, userName, version (app version), score, title and review text for the analysis of the meta-data and user reviews with the Apple Store and Google Play crawlers. The crawlers are run between the period of 01 June 2018 and 31 December 2018 for the apps depicted in Table 5.1 and 5.2.

Apple Store iTunes RSS feeds returns only a list of 50 reviews up to 10 pages (500 reviews in total) and Google Play app page displays 40 reviews at most, we were not able to retrieve enough amount of user reviews for some apps. To overcome this limitation, we use Appbot[3] service which tracks mobile app store reviews and ratings for all versions since their first launch. Even though our Appbot subscription has another limitation that a maximum 20,000 reviews per app could be exported (as in Table 5.1 and 5.2), the number and the quality of reviews is sufficient for aspect based sentiment analysis.

5.3.2 Filtering Opinion Spam and Non-Helpful Reviews

In this part of the research, we first classified the reviews as truthful and deceptive with the CNN based opinion spam detection model that was trained in Section 3. We then predict the helpfulness of the reviews with the logistic regression model earlier presented in Section 4.

Table 5.3 presents the number of App Store user reviews filtered by our opinion spam detection and user review helpfulness assessment models per app.

Table 5.3 Number of App Store Reviews Before and After Filtering

ID	#Total Reviews	#Deceptive Reviews	#Non-Helpful Reviews	#Final Reviews
1	20,000	95	1,017	18,888
2	20,000	153	1,346	18,501
3	4,060	51	244	3,765
4	15,661	92	442	15,127
5	20,000	83	570	19,347

[3] https://appbot.co/

Table 5.3 Number of App Store Reviews Before and After Filtering (continued)

ID	#Total Reviews	#Deceptive Reviews	#Non-Helpful Reviews	#Final Reviews
6	1,176	51	173	952
7	686	7	271	408
8	1,984	68	193	1,723
9	20,000	1,671	1,307	17,022
10	1,020	50	144	826
11	10,040	452	2,098	7,490
12	220	7	53	160
13	56	0	5	51
14	1,324	99	306	919
15	20,000	363	2,471	17,166
16	4,547	150	328	4,069
17	14,045	386	4,352	9,307
18	9,842	461	1,693	7,688
19	20,000	58	1,057	18,885
20	12,665	73	907	11,685
21	20,000	161	1,486	18,353
22	130	1	9	120
23	3,626	11	312	3,303
24	7,706	36	562	7,108
25	1,716	41	306	1,369
Total	**230,504**	**4,620**	**21,652**	**204,232**

Table 5.4 shows the number of Google Play user reviews filtered by our opinion spam detection and user review helpfulness assessment models for each app.

Table 5.4 Number of Google Play Reviews Before and After Filtering

ID	# Total Reviews	# Deceptive Reviews	# Non-Helpful Reviews	# Final Reviews
26	20,000	201	2,051	17,748
27	20,000	150	3,131	16,719
28	2,733	68	782	1,883
29	66	1	25	40
30	20,000	94	2,028	17,878
31	20,000	182	1,837	17,981
32	981	4	76	901
33	20,000	262	2,865	16,873
34	346	5	72	269
35	12,717	16	2,295	10,406
36	10,825	90	1,728	9,007
37	41	2	11	28
38	562	11	164	387
39	918	24	281	613
40	2,702	32	486	2,184
41	544	7	89	448

Table 5.4 Number of Google Play Reviews Before and After Filtering (continued)

ID	# Total Reviews	# Deceptive Reviews	# Non-Helpful Reviews	# Final Reviews
42	2,147	29	361	1,757
43	620	13	264	343
44	20,000	82	2,223	17,695
45	373	6	95	2,72
46	20,000	338	5,197	14,465
47	46	2	19	25
48	20,000	212	3,577	16,211
49	266	0	23	243
50	246	5	54	187
Total	**196,133**	**1,836**	**29,734**	**164,563**

5.3.3 Annotation of Mobile App Aspects

The software requirements are distinguished into three categories: (i) functional requirements (FRs), (ii) constraints and (iii) quality requirements or non-functional requirements (NFRs) Pohl (2010). According to Khalid *et al.* (2015), FRs could be refined by analyzing online reviews for complaints and praises, while new requirements can be derived from feature requests Villarroel *et al.* (2016). Constraints that affect and limit software projects and application are often beyond the user's awareness or explanation capabilities Groen *et al.* (2017), Palomba *et al.* (2017). Since app users are directly affected by quality characteristics of the app such as performance efficiency, reliability, and portability, it is possible that online reviews contain statements about product qualities.

To investigate the assumption that app store user reviews contain statements about FRs and quality characteristics, the preliminary analysis of user reviews are performed by 2 software development experts through manual analysis. 1250 reviews (25 reviews per app) have been chosen with random sampling and the software development experts are asked to categorize the topic presented in the review. As the examples for topic types, ISO/IEC 25010:2011 quality characteristics: (i) Functional suitability, (ii) Performance Efficiency, (iii) Compatibility, (iv) Usability, (v) Reliability , (vi) Security, (vii) Maintainability, (viii) Portability and the other aspects reported by earlier studies (See Section 1) (ix) Feature Request, (x) Praises and Complaints, (xi) Questions and also (xii) Price are given to the software development experts. If

the aspect in the review does not fit into any of these categories, the evaluators are asked to chose the aspect 'Other'. Since the review could contain more than one aspect, evaluators are not restricted to choose single aspect per review.

Table 5.5 Descriptive Statistics: Occurrence of the aspects for 50 apps

Quality Aspect	Mean	Median	Standard Dev	Range	Min	Max
Functional Suitability	5.7	5	3.54	14	0	14
Performance Efficiency	0.62	0	1.05	5	0	5
Compatibility	0.42	0	1.29	7	0	7
Usability	1.86	1	1.86	8	0	8
Reliability	2.72	1.5	3.13	12	0	12
Security	0.1	0	0.30	1	0	1
Maintainability	0	0	0	0	0	0
Portability	0.98	1	1.60	10	0	10
Feature Request	1.16	1	1.34	5	0	5
Praises & Complaints	9.86	9	4.90	25	0	25
Questions	0.3	0	0.84	4	0	4
Price	1.06	0	2.71	14	0	14
Other	1.86	1.5	2.15	12	0	12

As a result, the evaluators report that the top 5 topics in the user reviews are "Praises and Complaints" (%37), "Functional Suitability" (%21), "Reliability" (%10), "Usability" (%7) and "Other" (%5). The reviews categorized as "Other" are not directly to the app but the service or very specific concerns or comments e.g., *"Why can't children not have a Fitbit?"* or simply do not convey any useful information e.g., *"My business address is not right and I put the correct one this is a xx"*. There is no review identified only for the maintainability aspect among the other aspects. Figure 5.2 shows the distribution of the topics explored in user reviews. Table 5.5 gives the preliminary analysis descriptive statistics for the occurrence of aspects in sampled 25 reviews within 50 apps and Table 5.6 presents the examples of each type of NFRs topics.

Table 5.6 Examples of Each Type of NFRs and Others

NFR Type	Example from User Reviews
Functional Suitability	*"It hardly sends me notification for messages and phone calls."*

Table 5.6 Examples of Each Type of NFRs and Others (continued)

NFR Type	**Example from User Reviews**
Reliability	*"The app loaded but the screen froze and I could neither access my library nor my ebooks."*
Performance Efficiency	*"I'm trying to edit my resume too much too slow and I can not take it."*
Compatibility	*"I can't use this app with my Viano Smart TV."*
Usability	*"What an easy app to use!"*
Portability	*"It is bad because it only works on certain devices which is really silly"*
FR Type	**Example from User Reviews**
Feature Request	*"There's no option for overtime on the periods slot for the customization. Please fix asap"*
Other	**Example from User Reviews**
Praises and Complaints	*"Do not waste your time on this app, unless you want to spend money and time."*

The preliminary analysis results in Figure 5.2 indicate that significant amount of user reviews are related to quality attributes (aspects) of the app and also "Feature Requests", it is useful to automatically extract these aspects from the review text for the use of both developers and app users. However, as every machine learning model needs data for training, validation and testing, the following sections provide the details on how these datasets are prepared for the use of our deep neural network aspect extraction models explained in Section 5.3.4.

5.3.3.1 Annotation Task

Annotation was performed by two human annotators, one of the annotators is PhD student while the second annotator is a post doctoral researcher. The training of the the annotators and optimization of the app review annotation guideline (Section 5.3.3.2) has been conducted in two iterations. In each iteration, 50 reviews were randomly sampled from the complete review corpus and given to the annotators. In the first iteration, the agreement between the annotators reached a value of %64. After the first round, the problems and ambiguities in the annotations of were discussed with the annotators. An agreement of %88 has been achieved in the second iteration. After completion of this training phrase, the actual annotation was

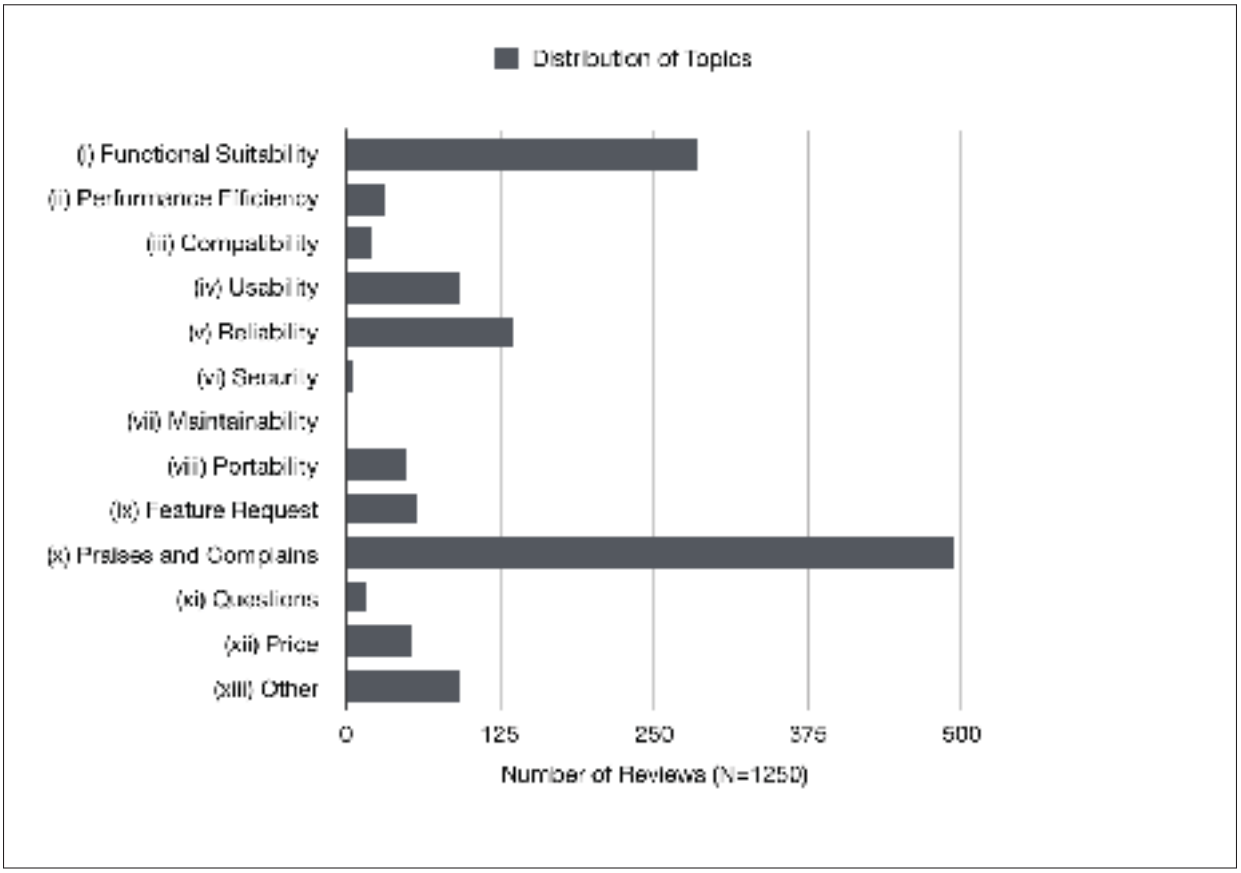


Figure 5.2 Distribution of the topics explored in user reviews with preliminary analysis

performed. The annotation took place in December 2018 and February 2019 over a period of six weeks. Annotators were asked to identify the aspect terms, aspect term polarity and aspect category and document them in XML format as presented as in Listing 2.

Each annotator worked on 2,000 reviews (total 4,000) randomly sampled among 368,795 reviews to obtain sufficient amount of annotated aspect in each category. As a result, 2170 aspects in 7 categories covered within 1685 sentences are annotated by our annotators.

5.3.3.2 App Review Annotation Guideline

For a given target entity which is a mobile app, the task of the annotator is to identify the following types of information:

```
<sentence id="1-2:10">
                <text>The android wear maps would not
                  ↪ load and the function was buggy and
                  ↪ unusable.</text>
                <Opinions>
                    <Opinion target="function"
                      ↪ category="RELIABILITY#QUALITY"
                      ↪ polarity="negative" from="45"
                      ↪ to="53"/>
                </Opinions>
            </sentence>
```

Listing 2: Example of an annotated review where the aspect term is *function*, aspect category is *RELIABILITY*, the aspect term polarity is *negative* and the aspect term expands between the character positions 45 and 53.

Aspect terms: Single or multiword terms naming particular aspects of the target entity. For example, in *"It's the most reliable app on my phone! The design is simple and effective and easy to read"*, the aspect terms are "app", and "design"; in *"User interface is cluttered. Impossible to adjust goals without losing all old data"* the only aspect term is "user interface".

Aspect term polarity: Each aspect term has to be assigned one of the following polarities based on the sentiment that is expressed in the sentence about it:

- positive
- negative
- neutral (neither positive nor negative sentiment)

Aspect category: The task of the annotator is to identify the aspect categories discussed in a sentence given the following five aspect categories:

- Functional Suitability
- Reliability

- Performance Efficiency
- Compatibility
- Usability
- Portability
- Feature Request

A sentence may be classified into one or more aspect categories based on its overall meaning. For example, the sentence *"Design is ok; usually opens up fast enough and reacts quickly"* discusses the aspect categories USABILITY and PERFORMANCE.

What should be annotated as aspect term?: Nominal phrases explicitly mentioning aspects. Notice that in (i) the aspect term is "inaccurate time error", not simply "error". In (ii) there is only one aspect term: the "quality of the photos & documents", since this is a single aspect, rather than two separate aspect terms "quality of the photos" and "quality of the documents".

(i) *No matter how many times i've updated it will give inaccurate time error even my mobile time was accurate.*

(ii) *Since last update, it is reducing the quality of the photos & documents that we send.*

What should NOT be annotated as aspect term?

(A) References to the target entity as a whole, mentions of other entities and service associated with the app. No aspect term should be annotated in the following sentences.

(i) *Great video streaming service.*

(ii) *This is better than Messenger.*

(B) The name or the type of the app (e.g., "Netflix", "Whatsup") or the name of the devices. No aspect term should be annotated in the following sentences:

(i) *I use a Galaxy Tab S2 for most of my media consumption, and it recently had some severe issues that I was only able to fix with a custom ROM*

(ii) *My Netflix just stopped working on the phone.*

(C) Pronouns (e.g.,"it", "they", "this") even if they refer to an aspect. For example, "it" should not be annotated below.

(i) *I love this app, it is amazing.*

5.3.4 Aspect Extraction Models

5.3.4.1 Bi-directional LSTM+CRF Model

Our first model relies on the deep learning architecture proposed in (Ma & Hovy (2016), where the authors combine two aspects previously exploited in different studies: i) the use of a character-level representation Chiu & Nichols (2015); ii) the additional output layer based on CRF Huang *et al.* (2015).

The overall architecture of Bi-directional LSTM+CRF model is presented in Figure 5.3. The character-level representation is computed by the CNN, as shown in Figure 5.3. Then the character-level representation vector is concatenated with the word embedding vector to feed into the Bi-directional LSTM network. Finally, the output vectors of Bi-directional LSTM are fed to the CRF layer to decode the best label sequence. Dropout layers are applied on both the input and output vectors of Bi-directional LSTM to prevent overfitting Srivastava *et al.* (2014). The details for individual layers are presented in the following paragraphs.

5.3.4.1.1 CNN for Character-level Representation

CNN and LSTM are often being used to construct character-level word embeddings in the literature Ma & Hovy (2016), Lample *et al.* (2016). Since the CNN has fewer parameters to train than BiLSTM network that is better in terms of training efficiency, we use CNN to extract

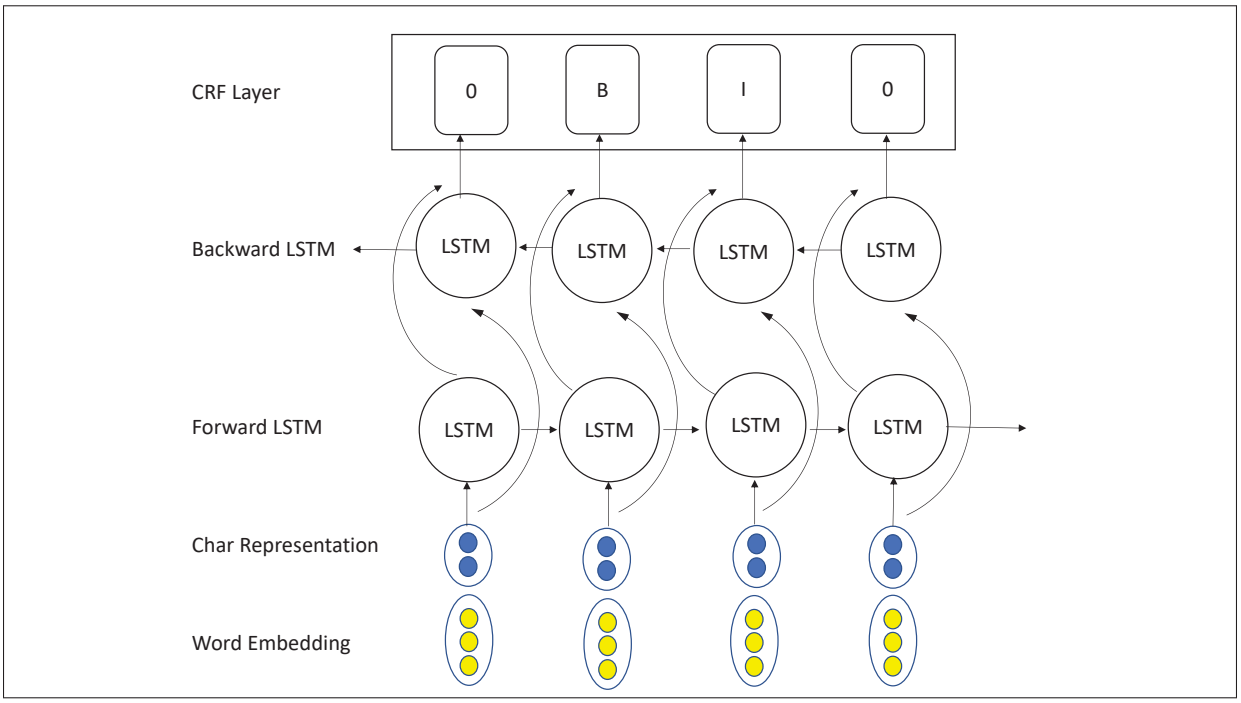


Figure 5.3 Bi-directional LSTM+CRF Model (Adapted from Ma & Hovy (2016))

character-level representation of a given word (Figure 5.4) as in the original architecture of Ma & Hovy (2016).

A dropout layer as indicated with dashed arrows is applied before feeding the CNN with character embeddings Srivastava *et al.* (2014). Then the character embeddings are concatenated with the word embeddings to form the input for the Bi-directional LSTM layer.

5.3.4.1.2 Bi-directional LSTM Network

LSTM networks are a specialized type of recurrent neural network (RNN)—a neural network architecture used for modelling sequential data and often applied to NLP tasks. The advantage of LSTMs over traditional RNNs is that they retain information for long periods of time, allowing for important information learned early in the sequence to have a larger impact on model decisions made at the end of the sequence.

Figure 5.5 illustrates the architecture of LSTM with more focus on a standard LSTM cell.

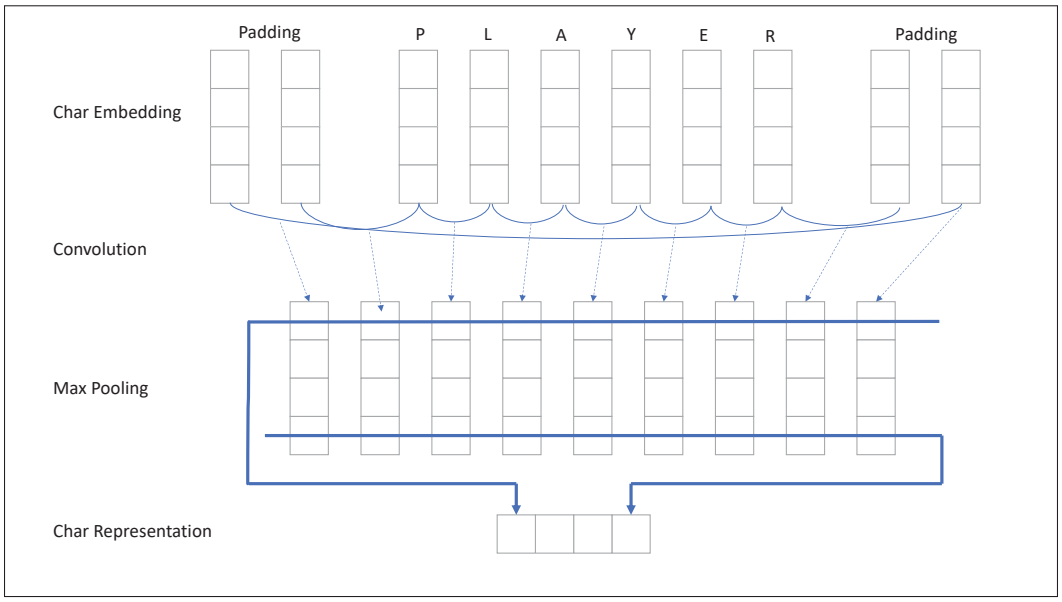


Figure 5.4 The convolution neural network for extracting character-level representations of words (Adapted from Ma & Hovy (2016))

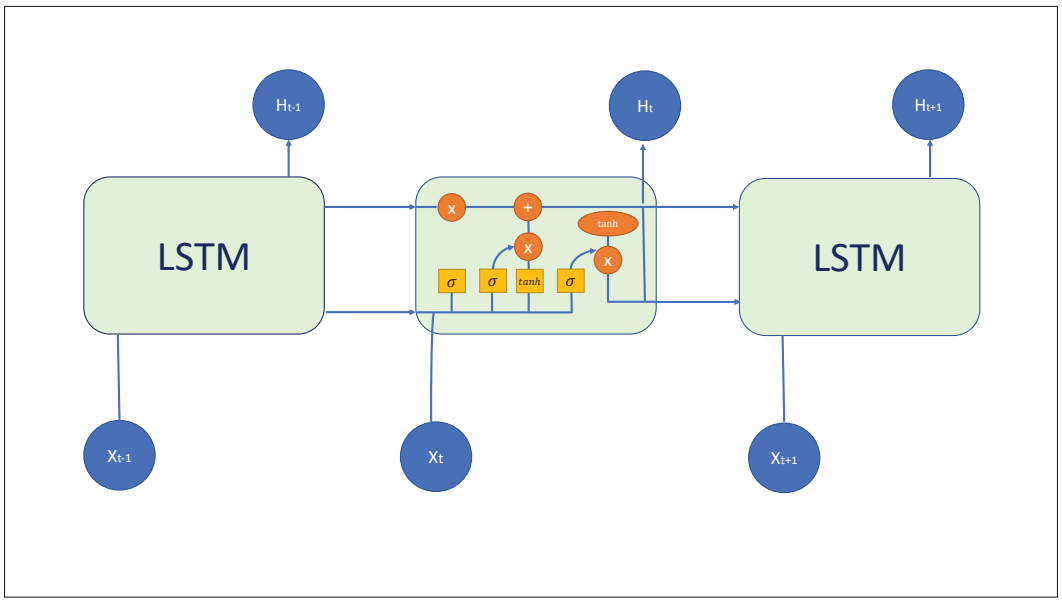


Figure 5.5 Detailed schematic of a Long Short-Term Memory block (Adapted from Greff *et al.* (2015))

In a standard LSTM cell:

- First, the forget gate controls what information is maintained from the previous state. This takes in the previous cell output h_{t-1} and the current input x_t band applies a sigmoid activation layer (σ) to get values between 0 and 1 for each hidden unit. This is followed by element-wise multiplication with the current state.
- Next, an update gate updates the state based on the current input. This passes the same input h_{t-1} and x_t into a sigmoid activation layer (σ) and into a *tanh* activation layer and performs element-wise multiplication between these two results. Next, element wise addition is performed with the result and the current state after applying the forget gate to update the state with new information.
- Finally, output gate controls what information gets passed to the next state. We run the current state through activation layer *tanh* and perform element-wise multiplication with the cell input h_{t-1} and x_t and run through a sigmoid layer (σ) that acts as a filter on what we decide to output. This output h_{t-1} is then passed to the LSTM cell for the next input of our sequence and also passed up to the next layer of our network.

More formally, each cell in LSTM is computed as follows:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{5.1}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{5.2}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{5.3}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{5.4}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5.5}$$

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{5.6}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{5.7}$$

where $W_i, W_f, W_o \in \mathbb{R}^{d \times 2d}$ are the weighted matrices and $b_i, b_f, b_o \in \mathbb{R}^d$ are biases of LSTM to be learned during training, parameterizing the transformations of the input, forget and output gates respectively. σ is the sigmoid function and $*$ stands for element-wise multiplication. x_t includes the inputs of LSTM cell unit and the vector of hidden layer is represented with h_t.

We regard the last hidden vector hN as the representation of sentence and put hN into a softmax layer after linearizing it into a vector whose length is equal to the number of class labels.

Bi-directional LSTM networks Schuster & Paliwal (1997) are extensions to single LSTM networks. They are capable of learning long-term dependencies and maintain contextual features from past and future. They comprise two separate hidden layers that feed forward to the same output layer (Figure 5.3). A Bi-directional LSTM calculates the forward hidden sequence $\vec{h}$, the backward hidden sequence $\overleftarrow{h}$ and the output sequence y by iterating over the following equations:

$$\vec{h}_t = \sigma\left(W_{x\vec{h}}x_t + W_{\vec{h}\vec{h}}\vec{h}_{t-1} + b_{\vec{h}}\right) \tag{5.8}$$

$$\overleftarrow{h}_t = \sigma\left(W_{x\overleftarrow{h}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}\right) \tag{5.9}$$

5.3.4.1.3 Conditional Random Fields (CRF)

For sequence labeling (or general structured prediction) tasks, it is beneficial to consider the correlations between labels in neighbourhoods and jointly decode the best chain of labels for a given input sentence. We thus feed the outputs of the previous bidirectional LSTM layer into a CRF layer as the unary potentials.

Given an input sentence $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T \rangle$ and a label sequence $\mathbf{y} = \langle \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T \rangle$ the score of the label predictions for x is calculated as:

$$\text{score}(\mathbf{x},\mathbf{y}) = \sum_{t=0}^{T} T_{y_t,y_{t+1}} + \sum_{t=1}^{T} H_{t,y_t} \tag{5.10}$$

where T is the transition matrix denoting the probabilities of one tag transiting to another, and H is the matrix stacked by the Bi-LSTM outputs.

The probability for the label sequence y given x is then computed from score $(\mathbf{x},\mathbf{y})$ using a softmax transformation.

$$p(\mathbf{y}|\mathbf{x}) = \frac{e^{s(\mathbf{x},\mathbf{y})}}{\sum_{\hat{\mathbf{y}}\in\mathbf{Y_x}} e^{s(\mathbf{x},\hat{\mathbf{y}})}} \tag{5.11}$$

where $\mathbf{Y_x}$ is the set containing all conceivable assignments of sequence labels for $\mathbf{X}$.

The network parameters are chosen to minimize the negative log-likelihood of the gold tag sequence for an input x:

$$\text{L}(\theta) = -\sum_{\mathbf{x},\mathbf{y}} \log(p(\mathbf{y}|\mathbf{x})) \tag{5.12}$$

While inferencing, we find the best label sequence $\mathbf{Y}^*$ that gives a maximum probability, which can be calculated efficiently by Viterbi algorithm.

5.3.4.2 Deep CNN+CRF Model

A CNN is a neural-based approach which represents a feature function that is applied to constituting words or n-grams to extract higher-level features. The resulting abstract features have been effectively used for sentiment analysis, machine translation, and question answering, among other tasks. Collobert & Weston (2008) are among the first researchers to apply CNN-based frameworks to NLP tasks. The goal of their method was to transform words into a vector

representation via a look-up table, which resulted in a primitive word embedding approach that learn weights during the training of the network (Figure 5.6).

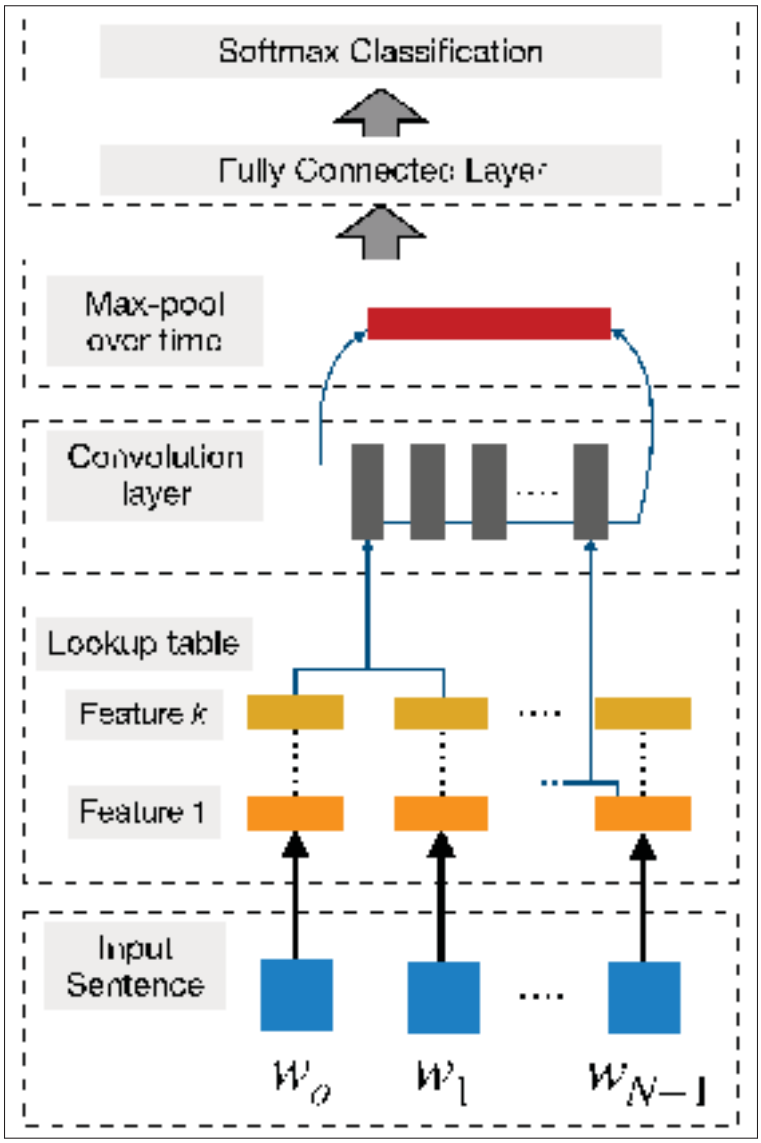


Figure 5.6 CNN framework used to perform word wise class prediction (Taken from Collobert & Weston (2008))

In order to perform sentence modelling with a basic CNN, sentences are first tokenized into words, which are further transformed into a word embedding matrix (i.e., input embedding layer) of d dimension. Then, convolutional filters are applied on this input embedding layer which consists of applying a filter of all possible window sizes to produce what is called a

feature map. This is then followed by a max-pooling operation which applies a max operation on each filter to obtain a fixed length output and reduce the dimensionality of the output. And that procedure produces the final sentence representation.

The ultimate goal of word-level classification is generally to assign a sequence of labels to the entire sentence. In such cases, structured prediction techniques such as conditional random field (CRF) are employed to better capture dependencies between adjacent class labels and finally generate a cohesive label sequence giving a maximum score to the whole sentence.

The window approach proposed by Collobert *et al.* (2011) assumes the tag of a word depends mainly on its neighbouring words. As the features of an aspect term depend on its surrounding words, we use a window of 5 words around each word in a sentence. We formed the local features of that window and considered them to be features of the middle word. Then, the feature vector was fed to a CNN. Our CNN network that is borrowed from Poria *et al.* (2016) contains one input layer, two convolution layers, two max-pool layers, and a fully connected CRF layer for the output. The output of each convolution layer is computed using a non-linear function: in our case it is the hyperbolic tangent.

5.4 Implementation

The models are implemented in Tensorflow, an open source software library for numerical computation and fine-tuned large scale machine learning. Tensorflow can break up the graph of computations into several chunks and run them in parallel across multiple CPUs and GPUs. The computations for both models are run on CPU 3.1 GHz Intel Core i7.

5.4.1 Hyper-parameters

In the following sub-sections, the hyper-parameters set for our two deep learning models are presented.

5.4.1.1 Bi-directional LSTM+CRF Model

Word Embeddings: Pre-trained word embeddings are meant to be particularly useful on problems where little training data is available. Hence, we use pre-computed GloVe embeddings [4] from 2014 English Wikipedia that contains 300-dimensional embedding vectors, 840B tokens and 2.2M vocabulary from Pennington *et al.* (2014).

Character Embeddings: Character embeddings are initialized with uniform samples from $\left[-\sqrt{\frac{10}{dim}}, +\sqrt{\frac{10}{\text{dim}}}\right]$, where *dim* is set as 100.

Weight Matrices and Bias Vectors: Matrix parameters are randomly initialized with uniform samples from $\left[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}\right]$ where r and c are the number of rows and columns in the structure Glorot & Bengio (2010). Bias vectors are initialized to zero, except the bias for the forget gate in LSTM, which is initialized to 0.1 Jozefowicz *et al.* (2015).

Optimization Algorithm: Adam optimizer Kingma & Ba (2014) is used to minimize the loss with initial learning rate 0.001. Due to the relatively small size the training and development sets, overfitting poses a considerable challenge for our system. To make sure that our model learns significant representations, we resort to dropout Srivastava *et al.* (2014) to mitigate overfitting. We find that dropout with a fixed rate of 0.5 decreases overfitting and improves the overall performance of our system. We also employ early stopping Graves *et al.* (2013) to mitigate overfitting by monitoring the model's performance on the development set.

Tuning Hyper-Parameters: Table 5.7 summarizes the chosen hyper-parameters for Bi-directional LSTM+CRF model including charactering embedding layer (CNN). We tune the hyper-parameters on the development sets by random search. Due to time constrains it is unfeasible to do a random search across the full hyper-parameter space. We initially set the state size of LSTM to 300, tuning this parameter did not significantly impact the performance of our model. For character embedding layer, we set charEmbeddingsSize (the dimension for characters) = 30, charFilterSize (filter Size) = 30, charFilterLength (the filter length) = 3.

[4] https://nlp.stanford.edu/projects/glove/

Table 5.7 Bi-directional LSTM+CRF Model Hyper-Parameters

Layer	Hyper-Parameter	Value
CNN	charFilterLength	3
	Number of filters	100
	charEmbeddingsSize	30
	charFilterSize	30
LSTM	Word embedding dimension	300
	Initial state	0
	Dimension of hidden layer	300
	Learning method	ADAM
	Learning rate	0.001
	Dropout	0.5
	Number of Epochs	200
	Batch Size	32

5.4.1.2 Deep CNN+CRF Model

Word Embeddings: We use pre-computed GloVe embeddings [5] from 2014 English Wikipedia that contains 300-dimensional embedding vectors, 840B tokens and 2.2M vocabulary Pennington *et al.* (2014).

Optimization Algorithm: Adam optimizer Kingma & Ba (2014) is used to minimize the loss with initial learning rate. To make sure that our model learns significant representations, we resort to dropout Srivastava *et al.* (2014) to mitigate overfitting. We also employ early stopping Graves *et al.* (2013) to mitigate overfitting by monitoring the model's performance on the development set.

Tuning Hyper-Parameters: Table 5.8 summarizes the chosen hyper-parameters for deep CNN+CRF model. We tune the hyper-parameters on the development sets by random search. Due to time constrains it is infeasible to do a random search across the full hyper-parameter space.

[5] https://nlp.stanford.edu/projects/glove/

Table 5.8 Deep CNN +CRF Model Hyper-Parameters

Layer	**Hyper-Parameter**	**Value**
CNN - Layer 1	Window size	[1, 2, 3]
	Number of filters	300
CNN - Layer 2	Window size	5
	Number of filters	300
	Word embedding dimension	300
	Learning method	ADAM
	Learning rate	0.001
	Dropout	0.5
	Number of Epochs	100
	Batch Size	32

5.4.2 Training and Evaluation Corpora

For training and evaluation of the model, we use the corpora presented in Section 5.3.3. Our dataset is split into 80% training, 10% validation and 10% test subsets.

The annotations in the corpora is encoded according to IOB2, a widely used coding scheme for representing sequences. In this encoding, the first word of each chunk starts with a "B-Type" tag, "I-Type" is the continuation of the chunk and "O" is used to tag a word which is out of the chunk. In our experiments, we are interested to determine whether a word or chunk is an aspect, so we only have "B–A", "I–A" and "O" tags for the words. Figure 5.7 presents an example of IOB2 tags.

5.5 Experiments and Results

In this part of the research, validation and prediction (testing) performance of our aspect extraction models are evaluated. We use the Precision (P), Recall (R), F_1 measures that are widely-used in prior aspect extraction studies to evaluate the prediction performance of aspect extraction models. We also report partial matches score that holds true positive if actual and prediction overlap by at least one token.

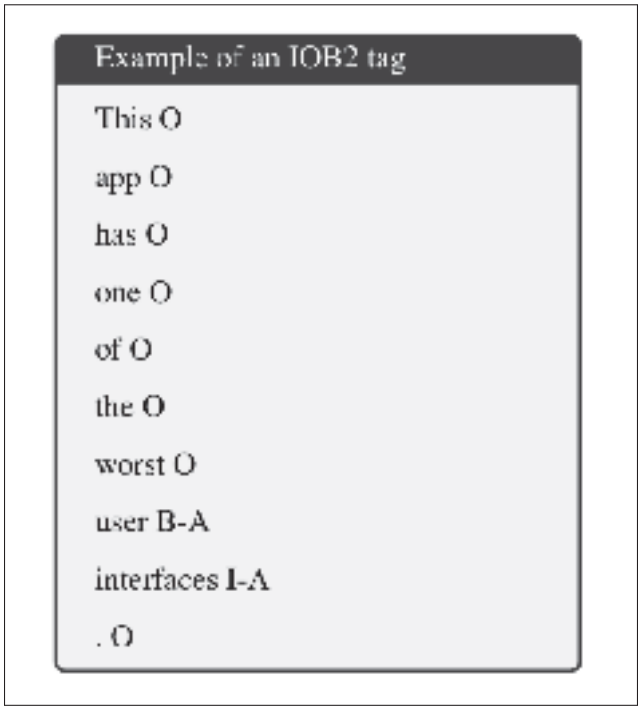


Figure 5.7 An example of an IOB2 tag

Figure 5.8 depicts the training loss graphs for Bi-LSTM+CRF and Deep CNN+CRF models. At Epoch 57 out of 200, the training has been stopped as there was not any further improvement happening for Bi-LSTM+CRF model. The training has been stopped for Deep CNN+CRF model at Epoch 73 out of 200.

Figure 5.9 presents the validation precision values per epoch for Bi-LSTM+CRF and Deep CNN+CRF models.

Figure 5.10 presents the validation recall values per epoch for Bi-LSTM+CRF and Deep CNN+CRF models.

Figure 5.11 presents the validation F_1 values per epoch for Bi-LSTM+CRF and Deep CNN+CRF models.

Table 5.9 Test Results for Aspect Extraction Models

Model	**P**	**R**	F_1	**Partial P**	**Partial R**	**Partial** F_1
Bi-LSTM+CRF	80.73	76.70	78.67	91.75	81.90	86.12
Deep CNN+CRF	80.19	78.37	79.27	90.60	83.47	86.62

5.6 Conclusion

Traditional approaches to aspect extraction relied on hand-crafted features, regular expressions and word dictionaries. Conditional random fields (CRF) and support vector machines (SVM) have been widely used to tackle the aspect extraction problem. At the moment deep learning methods are seen as the most promising choice for NLP. Distributed word representations are becoming a standard tool in the field of natural language processing. Such representations are able to capture semantic features of words and significantly improve results for different tasks.

In this fourth phase of our research study, we first created an annotated app store review dataset for the aspects extraction task, based on ISO 25010 - Systems and Software Product Quality Requirements and Evaluation standard. Next, we applied current state of the art neural network based (Bi-LSTM+CRF) model for mobile app aspect extraction problem. The proposed model consists of three main layers: character embedding which was built using convolutional neural networks (CNN), bi-directional LSTM and CRF. Our experiments demonstrated that the Bi-LSTM+CRF model was slightly better than the Deep CNN+CRF baseline model. To the best of our knowledge, this is the first study that reports on a deep neural network models for aspect extraction of app reviews.

Figure 5.8 Training Loss for Bi-LSTM and Deep CNN+CRF Models

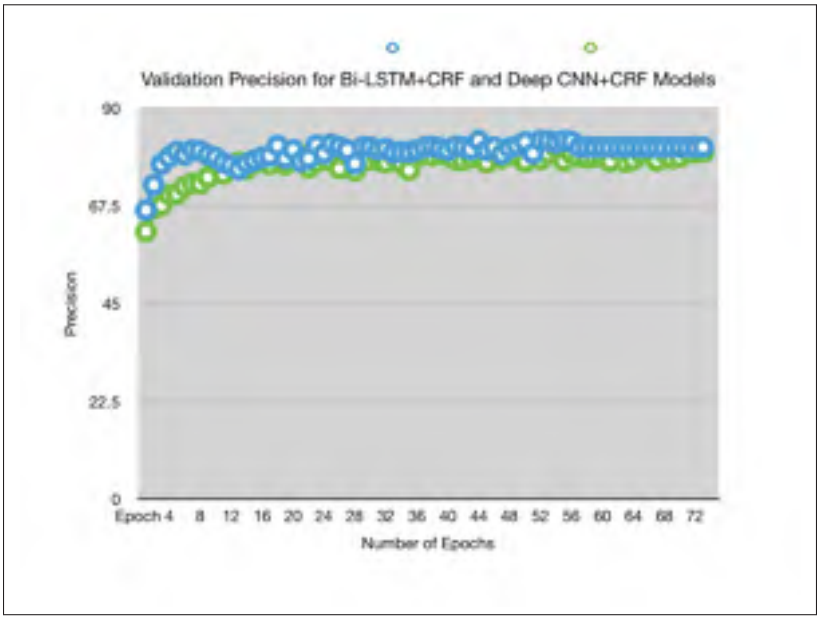


Figure 5.9 Validation precision values per epoch for Bi-LSTM+CRF and Deep CNN+CRF models

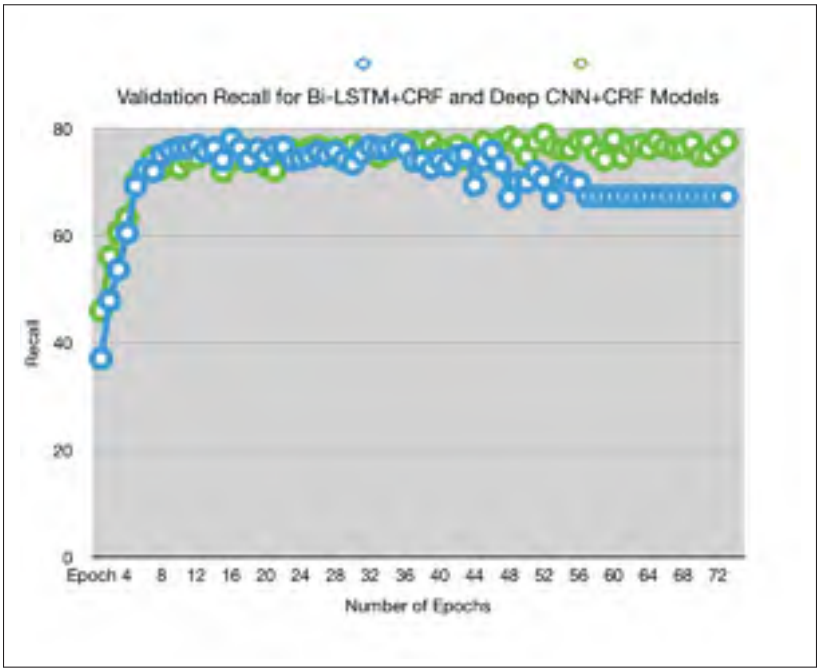


Figure 5.10 Validation recall values per epoch for Bi-LSTM+CRF and Deep CNN+CRF models

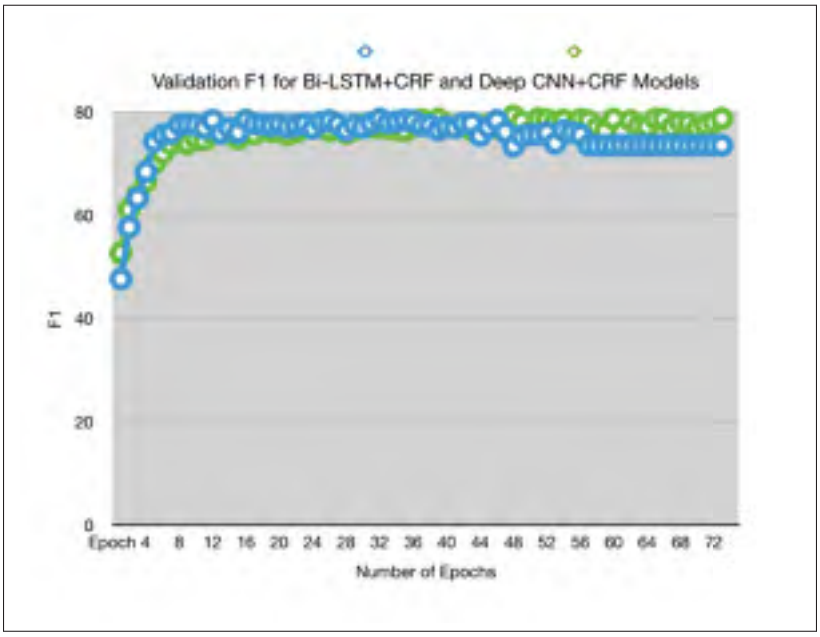


Figure 5.11 Validation F_1 values per epoch for Bi-LSTM+CRF and Deep CNN+CRF models

CHAPTER 6

CONTRIBUTIONS AND FUTURE WORK

This chapter presents a summary of the key contributions of this study and proposes some future work.

6.1 Contributions

The key contributions in this this doctoral research are the followings:

1. Proposing an app store opinion spam detection model: our systematic literature review has revealed that there was only one single publication in the literature specifically targeting the app store opinion spam detection problem. This publication dates back to 2012. Since app stores and deep neural networks have evolved tremendously over the last 7 years, our research has filled the gap in the literature by proposing a novel deep neural network architecture for spam review detection.

2. Construction and validation of four different machine learning based review helpfulness assessment models: Even though the user review helpfulness evaluation problem has been studied for various platforms such Amazon, Yelp, TripAdvisor, our SLR could not identify any study that assessed review helpfulness either manually or automatically for the mobile app store corpus.

   Our research study builds (i) a dataset including app store reviews with helpful and non-helpful labels and (ii) a baseline binary review helpfulness classifier that can help app store regulators to predict the helpfulness of the every single review regardless of its publication date and time and its given helpfulness votes.

3. Creation of an annotated app store review dataset for the aspect extraction task, based on ISO 25010 - Systems and software Product Quality Requirements and Evaluation standard: the systematic literature review has revealed that earlier app store aspect extraction studies

used different techniques, review datasets and annotation guidelines; therefore, the results reported in earlier studies are not directly comparable to each other.

4. Construction and validation of two Deep Neural Network Models: Bi-LSTM+CRF and Deep CNN+CRF for aspect extraction from app store user reviews. To the best of our knowledge, this research is the first study that reports on deep neural network models for aspect extraction of app store reviews.

5. The proposed models can be used to address individual app store problems of spam review detection, review helpfulness prediction and extraction of mobile app features. But also the combination of app store crawlers and three individual models would form a complete app review processing platform, as the retrieved app store reviews would be first filtered out by their helpfulness and authenticity. Then only helpful and non-spam reviews would be processed to extract significant app aspects from the review text (Figure 6.1 - Outputs).

   The initial findings of the phases 1 and 4 of this research have been published in the following articles:

   - Genc-Nayebi, N. & Abran, A. (2017). A systematic literature review: Opinion mining studies from mobile app store user reviews. Journal of Systems and Software, 125, 207–219.
   - Genc-Nayebi, N. & Abran, A. (2018). A Measurement Design for the Comparison of Expert Usability Evaluation and Mobile App User Reviews. 28th International Workshop on Software Measurement (IWSM) and the 13th International Conference on Software Process and Product Measurement (MENSURA), IWSM-Mensura 2018, Beijing, China, September 18-20, 64-76.
   - Genc-Nayebi, N. & Abran, A. (2019). An Empirical Study on the Comparison of Expert Usability Evaluation and Mobile App User Reviews. Empirical Software Engineering. (Submission:EMSE-D-18-00342)

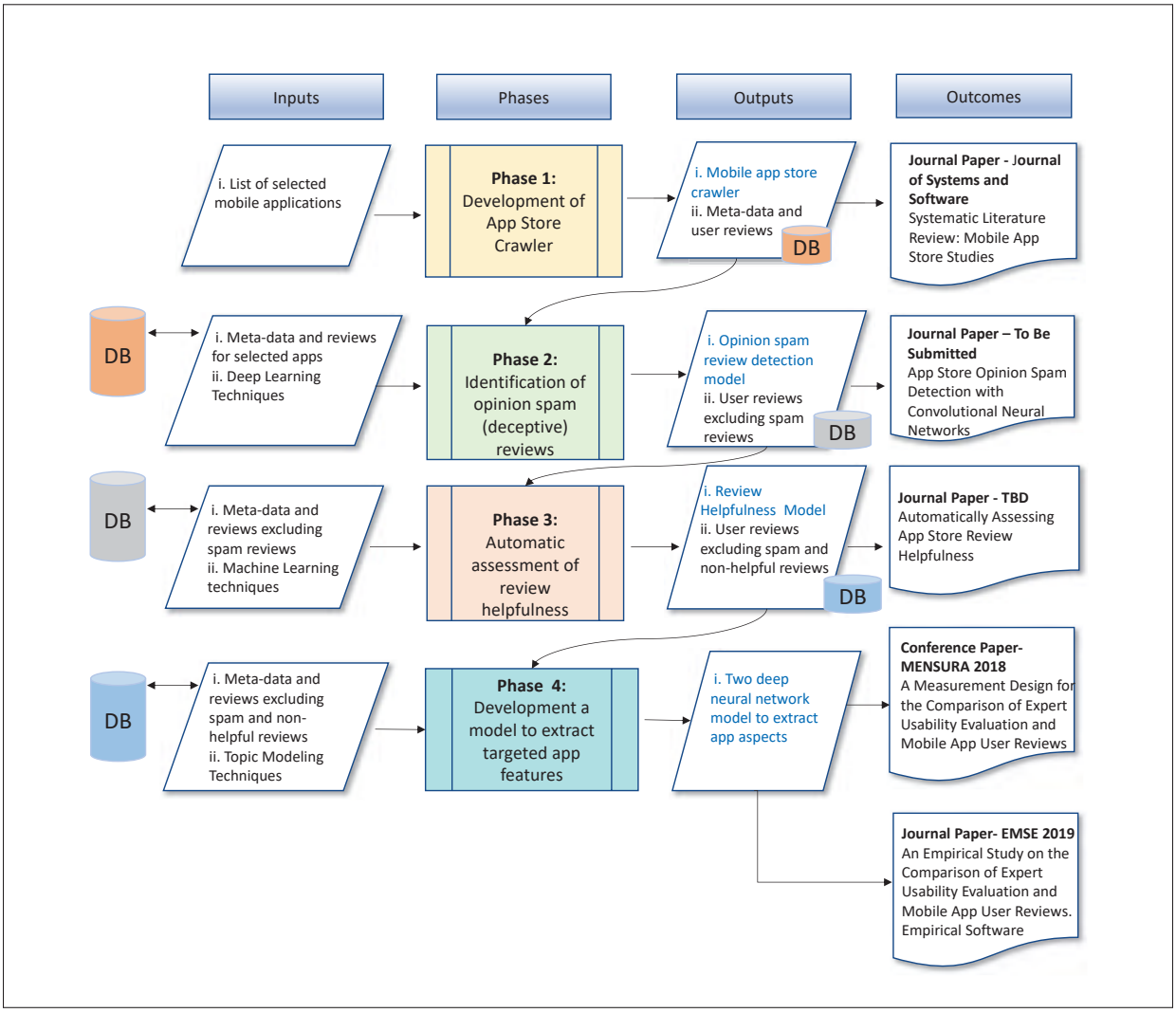


Figure 6.1 Outcomes of the Research Study

6.2 Future Work

Our research has concentrated on using different machine learning algorithms and mining useful features from either the content or the "meta-data" of the reviews. We investigated a variety of features from Apple store reviews, and found that structural (e.g., review length), lexical (e.g., inverse term frequency of uni-grams and bi-grams) and also meta-data features (e.g., app description) are most useful in measuring review helpfulness. As presented in Sections 3 and 5, many researchers have shown that deep learning methods are effective on a variety of NLP

tasks. As a future work, we will explore the possibility of employing a deep neural network model to assess review helpfulness.

Most of modern NLP architectures adopted word embeddings and giving up bag-of-word (BoW), LDA, LSA, etc. We also leveraged Word2Vec and Glove embeddings in our opinion spam detection and mobile app feature extraction models presented in Chapter 3 and Chapter 5. ELMo is a new deep contextualized word representation that models complex characteristics of word use and how these uses vary across linguistic contexts Peters *et al.* (2018). Adding ELMo to existing NLP systems could significantly improve the performance of many state-of-the-art NLP models. Thus, we intend to investigate ELMo embeddings in our opinion spam detection and aspect extraction models in the near future.

There is a growing demand for Artificial Intelligence (AI), which not only performs well, but is also transparent, interpretable and trustworthy. However, understanding neural-network predictions is notoriously difficult due to the fact that each prediction arises from a web of decisions made by hundreds to thousands of individual nodes. In our research, the proposed models are primarily evaluated by how much time they take to run, and by the accuracy of their resulting decision. As a future research direction, we would like to explore our models' resilience to corrupted or inconsistent training data and transparency of the resulting decisions.

CONCLUSION AND RECOMMENDATIONS

User reviews and ratings are an important source of information to support buying decisions for app users. Developers and development companies build their app store reputation based on reviews and ratings. The quantity of opinionated text data on web and mobile app stores increases tremendously along with other types of big data. While the volume of the big data increases, so do the complexity and relationships underneath the data.

As consumers increasingly start to rely on user reviews and ratings, the incentives also increase to create fraudulent reviews to boost sales and to damage competitors' reputation on the market. Fraudulent reviews not only mislead customers into wrong purchase decisions, but also degrade users' trust in online reviews. In addition, review quality varies from reviewer to reviewer and low quality reviews might not convey any necessary information. Thus, collecting opinions requires concept or semantic level processing and filtering out non-opinionated text data.

Automated systems are useful to identify, classify and summarize the opinions and extract application features and also to detect opinion spams in the review text. On the other hand, NLP has advanced over the past several years primarily due to advances in unsupervised Learning approaches that create word embeddings which is a continuation of the Word2Vec and Glove approaches.

This research study has aimed to explore and test the following hypotheses:

- H0a – It is possible to classify app store user review's legitimacy (being spam or truthful) by automatically discovering distributed feature representation of review data:

  Our two-layers novel CNN model using Word2Vec embedding (low dimensional and distributed representations of the text) gave superior result (%82.5 accuracy) compared to baseline SVM model for spam review detection. Per the experiment results, it is possible to classify app store user review's legitimacy (being spam or truthful) by automatically

discovering distributed feature representation of review data without any feature engineering.

- H1a – All user reviews do not include useful information in regards to merits and drawbacks of the mobile app:

  The preliminary analysis of user reviews performed by 2 software development experts through manual analysis shows that top 5 topics in the user reviews are "Praises and Complaints" (%37), "Functional Suitability" (%21), "Reliability" (%10), "Usability" (%7) and "Other" (%5). The reviews categorized as "Other" are not directly to the app but the service or very specific concerns. Hence, this true hypothesis claims that all user reviews do not include useful information in regards to merits and drawbacks of the mobile app.

- H0b – Analysis of app store user reviews provides necessary information for software requirement evolution:

  Distribution of the topics explored in user reviews with the preliminary analysis proves that analysis of app store user reviews provides necessary information for quality characteristics e.g., functional suitability, performance efficiency, compatibility, usability.

- H1b – It is possible to automatically retrieve app aspects from the review text:

  In the course of this research study, two deep neural network models developed for the extraction of mobile app aspects have achieved nearly 80% F_1 score in exact aspect matching and 86% F_1 score in partial aspect matching. Hence, the models and obtained results demonstrate that it is possible to automatically retrieve app aspects from the review text.

To achieve our research objectives, the following research phases have been followed:

Phase 0 - Identification of app store opinion mining studies: A systematic literature review has been conducted to identify earlier mobile app store opinion mining studies and related ar-

tifacts. The SLR methodology, the studies and techniques identified for each research question and the principal findings were presented in Chapter 1.

Phase 1- Development of an app store crawlers: Two app store crawlers, one for Apple App Store and the another one is for Google Play were developed to collect user reviews and app meta-data such as appId, description, version, genres, etc. from the subject app stores. The details for the app store crawlers are presented in sub-section 2.1.

Phase 2- Detection of opinion spam (deceptive) reviews: For deceptive spam review detection task, a two-layer CNN model was leveraged to learn document representation from sentence representations. The performance of the model was investigated against a baseline SVM model. The CNN model which does not require any annotated review dataset and hand-crafted features has better prediction performance compared to the baseline SVM model. A review dataset including truthful and spam reviews was acquired by crawling Apple Store and synthesized with a novel but an affective approach. Chapter 3 provides the details of two-layer CNN model and also the baseline SVM model along with the features used in the model.

Phase 3- Assessment of Review Helpfulness: Four different machine learning classification models have been examined to construct a review helpfulness prediction model. The best performing model was tuned using the grid search and then its performance has been enhanced by adding meta-data features which capture the presence and occurrence frequencies of the app aspects (features) given in app store description of the mobile app. Chapter 4 provides the details of the investigated classification models, the experiment results for grid search and meta-data features.

Phase 4- Extraction of Mobile App Features: An annotated app store review dataset was created for the aspect extraction task, based on ISO 25010 - Systems and software Product Quality Requirements and Evaluation standard. The Bi-LSTM+CRF model that was originally

developed for sequence tagging in the aspect extraction task was investigated in the extraction of mobile app features. The model performance was compared against a deep CNN model that has obtained good performance results for the task of aspect extraction. To the best of our knowledge, this is the first study that reports a deep neural network models for aspect extraction of app reviews. Chapter 5 presents the details for Bi-LSTM+CRF and Deep CNN+CRF models and also the experiment results.

APPENDIX I

LITERATURE REVIEW RESULTS

Table 1.1 List of mobile app store data mining studies

No	Reference	Study Name	Method	Features	Dataset	Performance Results
1	Chen & Liu (2011)	Predicting Popularity of Online Distributed Applications	CART	Static features, Dynamic features, Comment features	200 paid applications sampled from 102,237 applications	Preliminary observations were presented as results
2	Vasa *et al.* (2012), Hoon *et al.* (2012)	A Preliminary Analysis of Mobile App User Reviews, A preliminary analysis of vocabulary in mobile app user reviews	Summary statistics, Box plots, Distribution charts	Word frequencies	8.7 million reviews from 17,330 apps	Top 20 most frequent words were presented as results
3	Harman *et al.* (2012)	App Store Mining and Analysis: MSR for App Stores	Correlation analysis and greedy algorithm for extraction and grouping of features	Price, Rank of downloads, Rating mean	32,108 non-zero priced apps	Correlation between customer rating and the rank of app downloads is presented
4	Iacob & Harrison (2013)	What Are You Complaining About: A Study of Online Reviews of Mobile Applications	Manual analysis	Positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer supports and versioning	Randomly selected 161 apps and 3279 reviews from Google Play Store	Distribution of code classes is given in results
5	Ha & Wagner (2013)	Do Android Users Write About Electric Sheep?	Manual classification	PAdjective (positive/negative), ads (positive/negative), aesthetics, company, comparison, feature/-functionality, model, money, permissions, preinstalled, recommendations, resources, tips, uninstalled, used to be, work/doesn't work	556 reviews from 59 applications	Results include percentage of how often broad topics appeared in reviews

Table 1.1 List of mobile app store data mining studies (continued)

No	Reference	Study Name	Method	Features	Dataset	Performance Results
6	Wano & Iio (2014)	Relationship between Reviews at App Store and the Categories for Software	Manual text analysis	N/A	500 applications from various categories	Review styles are different with software categories
7	Gómez *et al.* (2015)	A Recommender System of Buggy App Checkers	LDA (for topic mining) and J48 for learning patterns	N/A	User reviews from 46,644 reviews	N/A
8	Mojica Ruiz *et al.* (2015)	An Examination of the Current Rating System used in Mobile App Stores	Hexbin plot	242,089 app versions of 131,649	User reviews from 46,644 reviews	Store rating is very resilient to changes in the version rating

Table 1.2 List of mobile app store spam identification studies.

No	Reference	Study Name	Method	Features	Dataset	Performance Results
1	Chandy & Gu (2012)	Identifying Spam in the iOS App Store	Latent Class graphical model	Decision tree model and latent class graphical model	User average rating, user number of reviews, application average rating, app number of reviews, number of instances with 2, 3, 4 stars, developer number of applications, developer average rating, binary class indicators	6.4% classification error with false positive rate 6.3% and false negative rate 40.9.

Table 1.3 List of mobile app feature extraction studies

No	Reference	Study Name	Method	Extracted app features	Performance Results
1	Iacob & Harrison (2013)	Retrieving and Analyzing Mobile Apps Feature Requests from Online	LDA	positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer support and versioning	N/A
2	Galvis Carreño & Winbladh (2013)	Analysis of User Comments: An Approach for Software Requirements Evolution	ASUM	They presented sample topics identified per application. As an example for Facebook: 'Updates', 'Developer', 'Messages', 'Photos'	For K=24 Precision: 62.5 Recall: 20.83 F-Measure: 31.44 For K=48 Precision: 86.67 Recall: 54.16 F-Measure: 66.64 K=150 Precision: 90 Recall: 75 F-Measure: 80
3	Pagano & Maalej (2013)	User Feedback in the AppStore: An Empirical Study	Statistical Analysis	Community, requirements, rating, user experience	2 "Rating" is revealed as most frequent them with the frequency of over 77%. Requirements30%, Community - 13%.
4	Fu *et al.* (2013)	Why People Hate Your App – Making Sense of User Feedback in a Mobile App Store	Statistical Analysis	Attractiveness, stability, accuracy, compatibility, connectivity, cost, telephony, picture, media and spam	91% (Precision), 73% (Recall)
5	Oh *et al.* (2013)	Facilitating developer-user interactions with mobile app review digests	SVM	Functional Bug, Functional Demand, Non-functional Request	0.8981 (Precision), 0.8165 (Recall), 0.8553 (F-Measure)
6	Chen *et al.* (2014)	AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace	EMNB (Expectation Maximization for Naive Bayes)	They presented sample topics identified per application. As an example for Swiftkey: 'more theme', 'swype feature', 'space bar', 'more option', 'like keyboard' and etc	F-measure: 0.764 - SwiftKey 0.877 - Facebook 0.797 - TempleRun2 0.761 -TopFish

Table 1.3 List of mobile app feature extraction studies (continued)

No	Reference	Study Name	Method	Extracted app features	Performance Results
7	Guzman & Maalej (2014)	How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews	LDA	Functionality related topics were extracted	0.59 (Precision), 0.51 (Recall)
8	McIlroy *et al.* (2015b)	Analyzing and Automatically Labelling the Types of User Issues that are Raised in Mobile App Reviews	Naïve Bayes, Decision Tree, SVM	Additional cost, functional complaint, compatibility issue, crashing, feature removal, feature request, network problem, privacy and ethical Issue, resource heavy, response time, uninteresting content, update issue, user interface	Average 59% (Accuracy), 44% (Exact Match), 65%(Precision), 64% (F-measure micro) and 56% (F-measure macro))
9	Khalid (2013)	On Identifying User Complaints of iOS Apps	Manual Tagging	Hidden Cost, Functional Error, Compatibility, App Crashing, Feature Removal, Feature Request, Network Problem, Privacy and Ethical, Resource Heavy, Unresponsive App, Uninteresting Content, Interface Design	N/A
10	Khalid *et al.* (2015)	What Do Mobile App Users Complain About?	Manual Tagging	App Crashing, Compatibility, Feature Removal, Feature Request, Functional Error, Hidden Cost, Interface Design, Network Problem, Privacy and Ethics, resource Heavy, Uninteresting Content, Unresponsive App	N/A
11	Vu *et al.* (2015)	Mining User Opinions In Mobile App Reviews	Keyword extraction, grouping and ranking	Battery, versioning, unrecoverable error, snapchat, authentication, facebook	Average: 83.11% Accuracy

Table 1.3 List of mobile app feature extraction studies (continued)

No	Reference	Study Name	Method	Extracted app features	Performance Results
12	Park *et al.* (2015)	Leveraging User Reviews to Improve Accuracy for Mobile App Retrieval	AppLDA	Top Topics by LDA: log, upgrade, purchase, note, account, battle, refund, support	NDCG (Normalized Discounted Cumulative Gain) @3 =0.651, @5=0.656, @7=0.627, @20=0.634
13	Panichella *et al.* (2015)	How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution	Bayes, SVM, Logictic Regression, J48 and ADTree	Information Giving, Information Seeking, Feature Request, Problem Discovery, Others	Precision = 0.79, Recall = 0.719, F-Measure = 0.672, best results obtained with the features Natural Language Processing (NLP), Text Analysis (TA) and Sentiment Analysis (SA)
14	Gu & Kim (2015)	What parts of your apps are loved by users?	SUR-Miner (POS tag, Parsing Tree and Semantic Dependence Graph (SDG)	Aspect Evaluation, Praises, Feature Request, Bug Reports and Others	F_1-score = 0,81

APPENDIX II

PUBLICATIONS

1. A Systematic Literature Review: Opinion Mining Studies from Mobile App Store User Reviews

The Journal of Systems and Software 125 (2017) 207–219

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A systematic literature review: Opinion mining studies from mobile app store user reviews

Necmiye Genc-Nayebi*, Alain Abran

Department of Software Engineering and Information Technologies, École de technologie supérieure, University of Québec, 1100 Rue Notre-Dame O, Montreal, QC H3C 1K3, Montréal, Canada

ARTICLE INFO

Article history:
Received 22 April 2016
Revised 13 November 2016
Accepted 14 November 2016
Available online 17 November 2016

Keywords:
Mobile application
App stores opinion mining
Systematic literature review
Requirements engineering

ABSTRACT

As mobile devices have overtaken fixed Internet access, mobile applications and distribution platforms have gained in importance. App stores enable users to search for, purchase and install mobile applications and then give feedback in the form of reviews and ratings. A review might contain information about the user's experience with the app and opinion of it, feature requests and bug reports. Hence, reviews are valuable not only to users who would like to find out what others think about an app, but also to developers and software companies interested in customer feedback.

The rapid increase in the number of applications and total app store revenue has accelerated app store data mining and opinion aggregation studies. While development companies and app store regulators have pursued upfront opinion mining studies for business intelligence and marketing purposes, research interest into app ecosystem and user reviews is relatively new. In addition to studies examining online product reviews, there are now some academic studies focused on mobile app stores and user reviews.

The objectives of this systematic literature review are to identify proposed solutions for mining online opinions in app store user reviews, challenges and unsolved problems in the domain, any new contributions to software requirements evolution and future research direction.

1. Introduction

With the rapid development of web and mobile devices, customers can now buy goods and services directly from online websites and digital distribution platforms. Users often rely on others' reviews or recommendations either from online purchase web sites or review sites to finalize their purchasing decisions. However, reading all reviews is time consuming and, sometimes, deceptive for users because of misleading or spam reviews. Therefore, researchers are looking into developing automated systems to identify, classify and summarize the opinions or sentiments and also to detect spam in an online text. Various researchers have studied opinion mining since the late 90s; however, the introduction of Machine Learning techniques and annotated datasets such as customer review datasets (Hu and Liu, 2004; Ding et al., 2008), pros and cons datasets Ganapathibhotla and Liu (2008), Amazon product review data (Jindal and Liu, 2008) and blog author gender classification dataset (Mukherjee and Liu, 2010) accelerated the research in the domain. With the emergence of different opinion mining domains such as social media (Facebook, Twitter, Instagram, App Store), app ecosystems, micro blogs, etc.), the focus of studies has since shifted into short-length texts, spam detection and contradiction analysis.

There also exists quite a number of survey studies on opinion mining and sentiment analysis in the literature. Pang and Lee (2008) made a comprehensive contribution into opinion mining and sentiment analysis survey studies by covering applications, major tasks of opinion mining, extraction and summarization, sentiment classification and also the common challenges in the research field. Tsytsarau and Palpanas (2012) surveyed the development of sentiment analysis and opinion mining research studies including spam detection and contradiction analysis. Their survey study provided 26 additional papers compared to Pang and Lee's (2008) preliminary survey. The survey of Tang et al. (2009) has a narrower scope, examining the opinion mining problem only for customer reviews on the web sites that couple reviews with e-commerce like Amazon.com or the sites that specialize in collecting user reviews in a variety of areas like Rottentomates.com. Cambria et al. (2013) revealed the complexities involved in opinion

* Corresponding author.
E-mail addresses: necmiye.genc.1@ens.etsmtl.ca (N. Genc-Nayebi), alain.abran@etsmtl.ca (A. Abran).
URL: https://www.etsmtl.ca (A. Abran)

http://dx.doi.org/10.1016/j.jss.2016.11.027
0164-1212/

mining with respect to current demand along with future research directions.

Along with internet and world wide web, mobile devices have gained popularity because of their portability, accessibility, and location awareness. Concurrently, the ever increasing demand for various kinds of mobile apps running on different devices has led to a corresponding increase in mobile developers and competitive mobile app markets. App ecosystem opinion mining studies did not start until the early 2010s, soon after the launch of the Apple app store, the first application distribution platform, in July 2008. The success of the Apple app store has led to the launch of other similar stores and services, with an exponential growth both in number of applications and revenue. The Apple app store generated over 10 billion dollar in revenue for developers in 2014 and currently offers about three million apps (Statista, 2014). Data mining and opinion aggregation from these platforms has therefore become a serious research topic.

User ratings and reviews are user-driven feedback that may help improve software quality and address missing application features. However, it is difficult for an individual to read all the reviews and reach an informed decision due to the ever growing amount of textual review data. Hence, over the last several years, various techniques and automated systems have been proposed to mine, analyze and extract user opinion and sentiment from app store review text. Our first research question aims to reveal the data mining techniques used for reviews on software distribution platforms.

One challenge in app store opinion data mining is vocabulary, which can vary, with the same term having different meanings in different contexts and domains. For example, even though "unpredictable' may have a positive meaning for a movie or book review, it could indicate a negative opinion in a mobile app review and be associated with a possible bug or a quality issue. Since the linguistic context of terms used in reviews plays a key role in opinion mining, domain adaptation and transfer learning aspects should also be considered. Secondly, the reviews found in app ecosystems are relatively short (71 characters on average) and have different vocabulary compared to other commodity marketplaces (Fu et al., 2013). Harman et al. (2012) have pointed out that app ecosystems are a new form of software repository and very different from traditional repositories. The granularity in an app store ecosystem is finer and the information collected (such as price, customer rating, number of downloads and application features, in addition to user reviews) allows empirical analysis. Our second research questions looks for research studies that explore this domain dependency challenge.

Unbiased or non-spam user reviews may be numerous but of varying quality. The terms such as 'Liked', "Not recommend', "OK app" do not convey any information about why users like an application or which aspects they like the most. Secondly, most reviews are poorly written and the information they contain often not useful, or highly personal and device- or technology-specific. Sophisticated ranking schemes, as found in the Apple app store and Google Play, measure reviews by their "helpfulness" as rated by users. In the Apple app store, the button under each user review allows other users to vote on whether the review is helpful or not; reviews may also be sorted from Most Helpful to Less Helpful based on these voting results. However, for newly written reviews or less popular applications, there would not be enough "helpfulness" voting to be of any use. Our third research question searches for studies that automatically asses and rank reviews in accordance with their usefulness or helpfulness.

As consumers increasingly rely on user reviews and ratings, there has been a stronger incentive to create fraudulent reviews in order to boost sales and damage competitors" reputations. Fraudulent reviews not only mislead customers into poor purchase decisions, but also degrade user trust in online reviews. Various studies and techniques have been proposed for detecting spam reviews. In an app ecosystem, spam app developers and opinion spammers (including those who would like to gain monetary profit or leak valuable user data such as contact lists or credit card information) tend to post spam reviews using Internet bots and puppet user accounts (Chandy and Gu, 2012). Despite some existing studies on opinion spam, the identification of spam in app stores has become another promising topic for researchers. Our fourth research question investigates spam identification and ranking fraud detection methods and techniques.

Users prefer having comparisons of specific features of different products available rather than having to gather isolated opinions about a single product themselves. In addition to average rating on a five-star scale and corresponding ranking on the app store, users prefer learning about others' experience with the app, including which aspects/features they liked or disliked most. Each user has his/her own preferences and while one user might feel strongly about the appearance, others may focus on functional or technical aspects. Hence, there is a need to extract and rate individual application features. However, to be able to make such comparisons, domain knowledge (ability to spot features) and commonsense knowledge about how to identify text polarity are required. Our last research question searches for aspect-based opinion mining studies extracting application features from mobile app store reviews.

Even though some surveys have reviewed the techniques and methods in opinion mining and sentiment analysis from text, no SLR has reviewed the literature regarding mobile app store data mining, opinion aggregation and spam detection. Martin et al. (2016) provided an initial survey into literature that covers the period of 2000 to November 27,2015, however their survey is not a SLR and they particularly interested in studies that combine technical (Application Program Interface (API) usage, size, platform version and etc) and non-technical attributes (category, rating, reviews, installs and etc) of mobile apps. The goal of our SLR is to methodically review and gather research results for specific research questions and to develop evidence-based guidelines for app store practitioners. We developed a set of five research questions to guide the literature review process and performed an extensive search to find publications that answer the research questions.

The rest of this paper is structured as follows. Section 2 presents our research methodology, including the research questions. Section 3 presents the results obtained by our SLR and identifies the challenges and avenues in this new field. Section 4 presents discussions about the mobile app store opinion mining studies.

2. Research methodology

The SLR was conducted following the guidelines of Kitchenham (2004). The activities performed in the course of the SLR were structured into three phases: (1) planning, (2) conducting the review, and (3) reporting. See Fig. 1. The individual tasks performed in each activity are described in Sections 2.1–2.3 .

2.1. Planning

The planning phase clarified the specific objectives of the SLR, that is, to identify mobile app store studies, the challenges faced when mining app store data, how these challenges have been overcome, and any unsolved challenges. In addition, we specified the following five research questions and the motivations behind the questions.

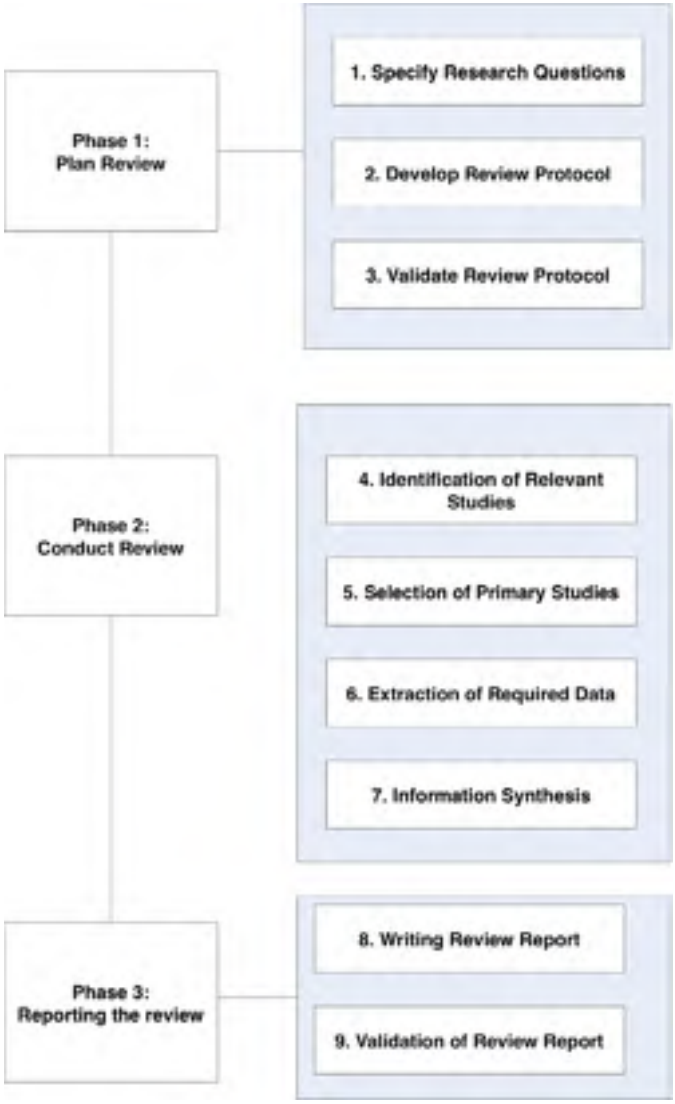


Fig. 1. SLR process.

2.1.1. Research questions

RQ1: Which specific data mining techniques are used for reviews on software distribution platforms?

Motivation: App stores provide a wealth of information in the form of customer reviews. Opinion mining and sentiment analysis systems have been applied to various kinds of texts including newspaper headline, novels, emails, blogs, tweets and customer reviews. Different techniques and automated systems have been proposed by researchers to extract user opinions and sentiments within the text over the years. Unlike documents or long length text, mobile app store reviews have some unique characteristics such as being short, informal and sometimes even ungrammatical consisting of incomplete sentences, elongations and abbreviations that make them difficult to handle. This question targets to present approaches and techniques proposed particularly for app store user review mining and opinion extraction problems.

RQ2: How do the studies remedy the 'domain dependency' challenge for app store reviews?

Motivation: Vocabulary varies within different context and domains, and same term might mean different opinions. An opinion classifier trained using opinionated words from one domain might perform poorly when it is applied to another domain. The reason is that not only the words and the phrases, but also the fact that language structure could differ from one domain to another. Hence, language structure and linguistic context of opinion and sentiments terms plays a key role in opinion mining, domain adaptation methods are also required to be considered while dealing with app store user reviews. This research question aims to reveal how mobile app store opinion mining studies tackle domain adaptation problem.

RQ3: What criteria make a review useful?

Motivation: Quality varies from review to review and low quality reviews might not convey any necessary signals to be used for information extraction. To tackle spam identification problem, it is critical to have a mechanism or a criterion that assesses the quality of reviews and filter out low-quality/noisy reviews. While review helpfulness is assessed manually by users in mobile app stores, there also exists some automated systems that assess and rank reviews in accordance with their usefulness or helpfulness. This research question aims to expose the methods or criteria used to differentiate useful app store reviews from the others. Besides, this research question also searches for automated systems that evaluate review usefulness and helpfulness.

RQ4: How can spam reviews be differentiated from legitimate reviews?

Motivation: As number of online reviews increased and fraudsters who produce deceptive or untruthful reviews emerged, tit is an essential task to identify and filter out the opinion spam. Different studies and techniques have been proposed for spam review detection problem. The opinion spam identification task has great impacts on industrial and academia communities. Our objective with this research question is to investigates spam review and ranking fraud detection methods and techniques for online stores and mobile app stores.

RQ5: Does the study extract targeted/desired software features from application reviews?

Motivation: Apart from app's average rating over 5-star scale and its corresponding ranking on app store, users would like to learn about others' experience with the app and which aspects/features they liked or disliked most. The information obtained from mobile app reviews is also valuable for developers to get user feedback about most liked or expected features (Requirements Elicitation) and bugs on the application (Software Quality and Software Evaluation). This research question focuses on aspect-based opinion mining studies extracting application features and aims to reveal the studies that make automated application feature extraction and rating in the face of user reviews.

2.1.2. Development and validation of the review protocol

The review protocol defines the activities required to carry out the literature review. A review protocol helps reduce researcher bias and defines the source selection and searching processes, quality criteria and information synthesis strategies. This subsection presents the details of our review protocol.

The following digital libraries were used to search for primary studies:

- Science Direct
- IEEExplore
- ACM Digital Library
- Citeseer library (citeseer.ist.psu.edu)
- Springer Link
- Google Scholar

The following search query was created by augmenting the keywords with possible synonyms. While conducting the review, we examined the reference list of primary studies to determine if there were additional studies not captured by our research query.

((mobile OR software) OR ((apps OR app OR application) OR (market OR ecosystem OR AppStore OR store))) AND ((data OR (on-

Table 1
Inclusion and exclusion criteria.

Inclusion criteria	Exclusion criteria
Case studies and surveys of text analysis, opinion mining and sentiment analysis from app store reviews.	Papers that present opinions without sufficient and reliable supporting evidence.
Preliminary analysis of mobile app store reviews, vocabulary, trends.	Studies not related to the research questions.
Papers searching application feature requests and bug reports within review text.	Papers that do not comply with the evaluation criteria in Table 2.
Papers that describe the criteria of what makes a review useful and helpful for readers.	Preliminary conference papers of journal papers by same author(s).
Papers that distinguish fake reviews and spams from legitimate ones.	

Table 2
Quality checklist Keele (2007).

No	Question
1	Are the aims of the study stated clearly?
2	Is the basis of evaluative appraisal clear?
3	How defensible is the research design?
4	Are data collection methods described adequately?
5	Has the approach to, and formulation of, analysis been conveyed adequately?
6	Has the diversity of perspectives and contexts been explored
7	Are there any links between data, interpretation and conclusions?
8	Is the reporting clear and coherent?
9	Has the research process been documented adequately?
10	Could the study be replicated?

line OR review) OR user OR (text OR comment OR vocabulary)) OR rating OR (opinion OR sentiment) OR (mining OR analysis OR processing) OR (feature OR requirement) OR request OR expectation OR (bug OR quality OR complain OR issue) OR (usefulness OR helpfulness))

Study Selection Procedure: We systematically selected the primary studies by applying the following four steps:

1. We examined the paper titles to eliminate studies unrelated to our research focus.
2. We reviewed the abstracts and keywords in the remaining studies. If either the abstracts or keywords did not provide the necessary information, we reviewed the results and conclusion sections to determine if the study was relevant.
3. We filtered the remaining studies in accordance with the inclusion and exclusion criteria given in Table 1.
4. We double-checked the reference list of the initial primary studies to identify additional studies that might be relevant to our search.

We evaluated the quality of the primary studies using the checklist adapted from Keele (2007). Each study was evaluated according to the quality checklist questions given in Table 2. The studies that provided a 'yes' answer to at least seven questions from the checklist were selected.

2.2. Conducting the review

2.2.1. Identification and selection of relevant studies

We followed Wohlin's (2014) snowballing procedure in order to identify relevant studies. In the first step called database search, we identified the keywords and formulated search string as given in Session 2.1.2. Our research with the search query generated more than 500 hits that will build up our start set. After examining the paper title, abstract, keywords, results and conclusions (if necessary) to filter out unrelated studies, 63 studies remained as start set. We used the reference list of our start set papers to identify new papers to include. Afterwards, we went through the reference list and exclude the papers that do not fulfil the basic criteria such as title, language and publication venue. We also performed forward snowballing to identify new papers based on those papers citing the paper being examined. Each candidate citing the paper is examined by screening the information provided in Google Scholar. If this information is not sufficient enough for a decision, the citing paper is examined in more details. After implementing backward snowballing and forward snowballing steps, we ended up with 45 research papers. Using the inclusion and exclusion criteria and quality checklist, examination of the remaining literature produced 24 primary studies.

2.2.2. Extraction of data

We used the data extraction form in Table 3 to extract data from the 24 primary studies. Even though the same data items were searched with RQ1 and RQ4, opinion mining and spam analysis studies respectively, the results obtained are presented in distinct tables. See Tables 4 and 5 in the Appendix.

2.2.3. Information synthesis

We read the 24 selected studies noting the methods and findings that were repeated. Inconsistencies and contradictions in the information were also recorded and are presented in the discussion and principal findings sections.

2.3. Reporting the review

Data extracted from the primary studies were used to answer our five research questions. The guidelines of Kitchenham (2004) were closely followed in the reporting of results.

3. Results

3.1. RQ1: Which specific data mining techniques are used for the reviews on software distribution platforms?

Analysis of the 24 primary studies identified a number of specific opinion mining and opinion extraction techniques used for reviews on software distribution platforms.

Chen and Liu (2011) identified useful app features ((i) static (e.g., application name, provider), (ii) dynamic (e.g., current rate, update date) and (iii) comment (e.g., user rate, comment content)) to predict app popularity and trained a model for an automated popularity prediction task. To create the dataset, they sampled 102,337 applications and a list of dynamic features were accumulated for top 200 paid and free applications by tracking their daily ranking. They used a Classification and Regression Tree (CART) model as a popularity prediction model and leveraged static (app name, provider, category, etc.), dynamic (current rank, all version count, all version rate, etc.) app and app store features and also comment features (user rate, comment title and comment content). As a result, they found that the top-ranked (the search ranking on the app store that factors in average app store rating, rating/review volume, download and install counts and app usage

Table 3
Data extraction form.

Search focus	Data item	Description
General	Identifier	Reference number given to the article
	Bibliography	Author, year, title, source
	Type of article	Journal/conference/technical report/etc.
	Study aims	Aims or goals of the study
RQ1 / RQ4	Text and data mining methods and techniques used	Algorithms, models and measures
	Selected or obtained review features	The subset of text features used or identified in the study
	Dataset	List of chosen applications, number of reviews
	Performance/Results	Precision, recall, accuracy/Obtained results
RQ2	Domain-specific text and data mining techniques used for app store reviews	App store and app review specific algorithms, methods
	Specific features used for app store reviews	App store and app review specific text features
	Performance improvement	Performance improvement compared to conventional opinion mining studies
RQ3	User review helpfulness/usefulness assessment framework	Predictors, variables, features that specify review quality
	Model used for automated usefulness task	Algorithms, models and measures
	Selected features	Subset of text features used in the study
	Performance	Precision, recall, accuracy
RQ5	Extracted app features	Mobile app features retrieved from online review text
	Method	Approaches, techniques used for automatically extracting application features
	Performance	Precision, recall, accuracy

Table 4
List of mobile app store data mining studies.

No	Reference	Study Name	Method	Features	Dataset	Performance Results
1	Chen and Liu (2011)	Predicting Popularity of Online Distributed Applications	CART	Static features, Dynamic features, Comment features	200 paid applications sampled from 102,237 applications	Preliminary observations were presented as results
2	Vasa et al. (2012), Hoon et al. (2012)	A Preliminary Analysis of Mobile App User Reviews, A preliminary analysis of vocabulary in mobile app user reviews	Summary statistics, Box plots, Distribution charts	Word frequencies	8.7 million reviews from 17,330 apps	Top 20 most frequent words were presented as results
3	Harman et al. (2012)	App Store Mining and Analysis: MSR for App Stores	Correlation analysis and greedy algorithm for extraction and grouping of features	Price, Rank of downloads, Rating mean	32,108 non-zero priced apps	Correlation between customer rating and the rank of app downloads is presented
4	Iacob et al. (2013)	What Are You Complaining About: A Study of Online Reviews of Mobile Applications	Manual analysis	Positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer supports and versioning	Randomly selected 161 apps and 3279 reviews from Google Play Store	Distribution of code classes is given in results
5	Ha and Wagner (2013)	Do Android Users Write About Electric Sheep?	Manual classification	PAdjective (positive/negative), ads (positive/negative), aesthetics, company, comparison, feature/functionality, model, money, permissions, preinstalled, recommendations, resources, tips, uninstalled, used to be, work/doesn't work	556 reviews from 59 applications	Results include percentage of how often broad topics appeared in reviews
6	Wano and Iio (2014)	Relationship between Reviews at App Store and the Categories for Software	Manual text analysis	N/A	500 applications from various categories	Review styles are different with software categories
7	Gómez et al. (2015)	A Recommender System of Buggy App Checkers	LDA (for topic mining) and J48 for learning patterns	N/A	User reviews from 46,644 reviews	N/A
8	Mojica Ruiz et al. (2015)	An Examination of the Current Rating System used in Mobile App Stores	Hexbin plot	N/A	242,089 app versions of 131,649	Store rating is very resilient to changes in the version rating

Table 5
List of mobile app store spam identification studies.

No	Reference	Study Name	Method	Features	Dataset	Performance/Results
1	Chandy and Gu (2012)	Identifying Spam in the iOS App Store	CART	Decision tree model and latent class graphical model	User average rating, user number of reviews, application average rating, app number of reviews, number of instances with 2, 3, 4 stars, developer number of applications, developer average rating, binary class indicators	6.4% classification error with false positive rate 6.3% and false negative rate 40.9.

statistics) paid applications were not closely related to customer ratings.

Vasa et al. (2012) and Hoon et al. (2012) made a preliminary analysis of mobile app user reviews. They initially analyzed the data using summary statistics with a one-way ANOVA test, box plots and cumulative distribution charts to confirm their hypothesis that rating and category have an affect on the length of the review. They analyzed 8.7 million reviews from 17,330 app and according to their analysis, users take the time to express their discontent by writing longer reviews, in contrast to short reviews when content with the application. They also identified a strong correlation between positive-negative sentiments and one- and five- star ratings. Unexpectedly, more than 50% of the two and three-star rated user reviews did not include any sentiment.

Harman et al. (2012) mined the Blackberry app store using Spearman's Rank Correlation method and identified a strong correlation between application rating and number of downloads, whereas there is no correlation between price and rating, nor price and number of downloads. They tested their approach to the 32,108 non-zero priced apps. Iacob and Harrison (2013) manually analyzed reviews and identified nine classes of feedback: positive, negative, comparative, price related, request for requirements, issue reporting, usability, customer support and versioning. They first randomly choose 169 apps and collected 3279 user reviews and then manually examined and classified reviews based on their content and then coded the categories, for example: aesthetics, company, comparison, feature/functionality, model, permissions, money, etc. They observed a correlation between review positivity and feature or functionality request.

Ha and Wagner (2013) manually analyzed Android users' reviews to see what they write about when reviewing Google Play applications. They crawled Google Play to collect information about 202,264 free applications and they selected 60 free applications with 556 reviews. As a result, they found that small subset of reviews had pointed privacy and security implications, whereas the majority of the reviews focused on the quality of the applications. Wano and Iio (2014) performed a manual text analysis and determined that review styles differ with software categories. The study used the search API and also used RSS Feed Generator by Apple. The number of targeted software is 500 and for each software, the targeted reviews are restricted up to 50 because of the API restriction. They concluded that consumers should pay attention to bias in reviews.

Gómez et al. (2015) mined reviews with LDA and error-suspicious permission patterns with the J48 decision tree algorithm (a Weka implementation of the C4.5 algorithm), revealing potential correlations between error-sensitive permissions and error-related reviews over time. They built a dataset that consists of a random sample of all the mobile apps available on Google Play Store. They collected 500 applications from 27 different categories.

Mojica Ruiz et al. (2015) made an overall evaluation of app stores and user rating schema and concluded that the current store rating of apps was not dynamic enough to capture the changing user satisfaction levels along with evolving application versions. Their dataset was extracted by crawling Google Play and this resulted in 242,089 app versions of 131,649 mobile apps. After the filtration, they ended up with 238,198 versions of 128,195 apps. They used hexbin plots to examine whether there would be a noticeable change in the store-rating of an app given a rise or drop in the rating of a specific version of that app.

Most of studies identified within this research question are preliminary researches and based on either manual or statistical analysis of user reviews. The researchers used either the research API and RSS Feed Generator by Apple store or some scrapers script to collect app store data. The datasets are mostly created with random sampling of all the mobile apps available and there is not any specific or common app category preferred by researchers. Since Martin et al. (2015) presented empirical evidence that indicates that the partial nature of data available on App Stores could pose an important threat to the validity of findings, the obtained results from different App Store research studies could not be compared with one another. Star rating, category and review content are most common features collected within 87.5% of the studies.

We could not obtain any data regarding average length of a review considered in the studies, however the dataset by Vasa et al. (2012) showed that user review length is highly skewed with an average of 110 characters. On the other hand, Fu et al. (2013) stated in their paper that average length of the comments is 71 characters, and median length is 47 characters. If the datasets used in the research studies would be publicly available, we will have the chance to validate these numbers. For this reason, researchers need to augment their findings with an argument to convince the reader that any sampling bias is unlikely to affect their research findings and conclusions. One of very recent studies by Gu and Kim (2015) indicated this app store sampling phenomenon as a threat to validity.

Table 4 presents the list of 9 studies with their methods, details of datasets, features and performance as a response to RQ1.

3.2. RQ2: How do the studies remedy the 'domain dependency' challenge for app store reviews?

RQ2 looked at how domain dependency affects opinion mining from reviews. A classifier trained in using opinionated words from one domain might perform poorly when applied to another domain since not only words and phrases but also language structure may differ from one domain to another.

From the primary studies it was noted that some researchers labelled data for the new domain and created their own dataset from scratch, whereas other researchers used labelled data from one domain and unlabelled data from the target domain, and then made the domain adaptation by using general opinion words (Aue and Gamon, 2005; Yang et al., 2006; Blitzer et al., 2007; Pan et al., 2010). In order to overcome the domain barrier in opinion extraction, Cosma et al. (2014) proposed a generalized methodology by considering a set of grammar rules for the identification of opinion-bearing words.

In addition, online reviews have distinctive text features, including short length, unstructured phrases and abundant information. Short reviews bring new challenges to traditional research topics in text analytics, such as text classification, information extraction and sentiment analysis. As opposed to standard texts, which include many words and phrases and their corresponding statistics, short texts consist of few phrases and sentences. Several traditional text analytics methods have been proposed to tackle the data sparseness problem:

- The first is surface representation that uses phrases in the original text from different product aspects to maintain the contextual information. However, this method fails to produce a deep understanding of the text and the method does not make use of external knowledge, which has been found useful in dealing with the semantic gap in text representation (Hu and Liu, 2012). For example, this review from the app store: "This iOS 9 update. App crashing and ugly font", does not contain any words or phrases related to the reason for the crash and possible user interface (UI) design problem, while the words 'crash' and 'font' are related to software engineering concepts. Hence, it is difficult to use bag-of-words based models and methods to build semantic connections between the review text and software characteristics.
- Another approach is to enrich the context of basic text segments by searching the external sources. Such methods have been found effective in narrowing the semantic gap for different tasks (Gabrilovich and Markovitch, 2007; Alfonso Ureña-López et al., 2001). In the app store corpus, these external sources would be app crash reports, tweets, community blogs and code repositories.

Another important characteristic of online text, particularly in online reviews, is the use of colloquial language. When composing a review, users might use abbreviations or acronyms that seldom appear in conventional text. As an example, the phrases "superb" "Good 2go" "you do not buy the guarskldj; al b bbbbbbbbb,,,,,,,wke;" make it very difficult to identify the semantic meaning. With research question RQ2, we sought to discover how researchers tackled domain adaptation problems, how they dealt with distinctive features of the review text and what specific methods or algorithms and text features were used to improve performance. To answer this research question, we reviewed the selected studies to identify the training datasets, methods, text features and performance comparisons. The mobile app store researchers mentioned in Table 6 used their own annotated dataset rather than leveraging existing online review datasets. Since they preferred to use conventional text mining methods such as Latent Dirichlet Allocation (LDA), Aspect and Sentiment Unification Model (ASUM), Naive Bayes classifier and statistical analysis, we cannot present any new method developed for the app store corpus. No new solutions or methods were proposed for examining the text characteristics (e.g., short length, unstructured phrases and colloquial language and challenges) of app store user reviews.

As in many real-word applications, topics revealed by Latent Dirichlet Allocation (LDA) and Aspect and Sentiment Unification Model (ASUM) are needed to be verified by experts to ensure they are semantically meaningful within the domain analysis. Hence, 4 studies out of 24 leveraged truth sets to understand if the extracted features align with real app features and to minimize the threat to validity. Galvis˜Carreño and Winbladh (2013) used the manually classified data as a truth set. Since the second author is not domain expert or not involved in software development, they reported that the process is error-prone. Chen et al. (2014) collected the group truth labels of the training pool and test set according to pre-defined rules. Guzman and Maalej (2014) and Gu and Kim (2015) also used the truth set that was created with systematic assessment of review samples by human coders.

However, manual validation could dominate the time and cost of building high-quality topic models. To overcome this problem, some researchers proposed measuring topic quality with topic coherence and statistical methods (Mimno et al., 2011; Newman et al., 2009). We propose incorporating domain knowledge into Topic Modelling via Dirichlet Forest Priors (Andrzejewski et al., 2009). Dirichlet Forest Priors, when combined with LDA, allows the user to encode domain knowledge (must-links and cannot-links between words) into the prior on topic-word multi nominal P(word| topic). In this way, app store domain knowledge could be expressed by a set of Must-Links (Two words u, v have similar probability within any topic) and Cannot-Links (Two words u, v should not both have large probability within any topic).

3.3. RQ3: What criteria make a review useful?

Review quality varies from reviewer to reviewer, and low-quality reviews might not convey any useful information. App store regulators allow users to vote on the helpfulness of each review and then rank the reviews based on votes. While review helpfulness is usually assessed manually, there are automated systems that do this. For the manual review of usefulness, there are no defined criteria among users. A review that appears helpful to one user may not be helpful for others, since they might be searching for different information or have differing priorities or biases. On the other hand, standard defined criteria would be valuable to differentiate useful reviews from others. Reviews chosen in accordance with these criteria for data mining and opinion extraction studies would yield the maximum capability for information extraction. Studies examining online review helpfulness are as follows:

- Cheung et al. (2008) measured review quality in terms of completeness, timeliness, accuracy and relevance.
- Mudambi and Schuff (2010) found that review depth had a positive effect on the helpfulness of the review but product type affected the perceived helpfulness of reviews.
- Pan and Zhang (2011) analyzed a large sample of reviews from Amazon to identify what determined information helpfulness and found that review length and positive reviews had a direct correlation with review usefulness.
- Korfiatis et al. (2012) discovered that review readability and positive ratings affected the number of helpfulness votes.

Studies on automatically assessing review helpfulness:

- Kim et al. (2006) trained an SVM (Support Vector Machine) regression model to learn the helpfulness function and then applied it to rank unlabelled reviews. They found that the most important features were the length of the review, its unigrams and its product rating.
- Liu et al. (2008) also modelled the helpfulness of reviews. They showed that helpfulness of a review depends on three important factors: reviewer expertise, writing style, and timeliness.
- Ghose and Ipeirotis (2011) used a Random Forest-based classifier and examined the relative importance of three feature categories: (i) reviewer related, (ii) reviewer subjectivity, and (iii) review readability. They found that using any of the three feature category results provided the same performance as using all available features.
- Moghaddam et al. (2012) used a probabilistic graphical model based on Matrix Factorization and Tensor Factorization. These models are based on the assumption that the observed review ratings depend on latent features of the reviews, reviewers, raters and products. They reported that the latent factor models outperform state-of-the-art approaches.

Table 6
List of mobile app feature extraction studies.

No	Reference	Study Name	Method	Extracted app features	Performance/Results
1	Iacob and Harrison (2013)	Retrieving and Analyzing Mobile Apps Feature Requests from Online	LDA	positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer support and versioning	N/A
2	Galvis˜Carreño and Winbladh (2013)	Analysis of User Comments: An Approach for Software Requirements Evolution	ASUM	They presented sample topics identified per application. As an example for Facebook: 'Updates', 'Developer', 'Messages', 'Photos'	For K=24 Precision: 62.5 Recall: 20.83 F-Measure: 31.44 For K=48 Precision: 86.67 Recall: 54.16 F-Measure: 66.64 K=150 Precision: 90 Recall: 75 F-Measure: 80
3	Pagano and Maalej (2013)	User Feedback in the AppStore: An Empirical Study	Statistical Analysis	Community, requirements, rating, user experience	"Rating" is revealed as most frequent them with the frequency of over 77%. Requirements-30%, Community - 13%.
4	Fu et al. (2013)	Why People Hate Your App – Making Sense of User Feedback in a Mobile App Store	Statistical Analysis	Attractiveness, stability, accuracy, compatibility, connectivity, cost, telephony, picture, media and spam	91% (Precision), 73% (Recall)
5	Oh et al. (2013)	Facilitating developer-user interactions with mobile app review digests	SVM	Functional Bug, Functional Demand, Non-functional Request	0.8981 (Precision), 0.8165 (Recall), 0.8553 (F-Measure)
6	Chen et al. (2014)	AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace	EMNB (Expectation Maximization for Naive Bayes)	They presented sample topics identified per application. As an example for Swiftkey: 'more theme', 'swype feature', 'space bar', 'more option', 'like keyboard' and etc.	F-measure: 0.764 - SwiftKey 0.877 - Facebook 0.797 - TempleRun2 0.761 -TopFish
7	Guzman and Maalej (2014)	How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews	LDA	Functionality related topics were extracted	0.59 (Precision), 0.51 (Recall)
8	McIlroy et al. (2015a)	Analyzing and Automatically Labelling the Tyes of User Issues that are Raised in Mobile App Reviews	Naïve Bayes, Decision Tree, SVM	Additional cost, functional complaint, compatibility issue, crashing, feature removal, feature request, network problem, privacy and ethical Issue, resource heavy, response time, uninteresting content, update issue, user interface	Average 59% (Accuracy), 44 percent (Exact Match), 65 percent (Precision), 64% (F-measure micro) and 56% (F-measure macro))
9	Khalid (2013)	On Identifying User Complaints of iOS Apps	Manual Tagging	Hidden Cost, Functional Error, Compatibility, App Crashing, Feature Removal, Feature Request, Network Problem, Privacy and Ethical, Resource Heavy, Unresponsive App, Uninteresting Content, Interface Design	N/A
10	Khalid et al. (2015)	What Do Mobile App Users Complain About?	Manual Tagging	App Crashing, Compatibility, Feature Removal, Feature Request, Functional Error, Hidden Cost, Interface Design, Network Problem, Privacy and Ethics, resource Heavy, Uninteresting Content, Unresponsive App	N/A
11	Vu et al. (2015)	Mining User Opinions In Mobile App Reviews	Keyword extraction, grouping and ranking	Battery, versioning, unrecoverable error, snapchat, authentication, facebook	Average: 83.11% Accuracy
12	Park et al. (2015)	Leveraging User Reviews to Improve Accuracy for Mobile App Retrieval	AppLDA	Top Topics by LDA: log, upgrade, purchase, note, account, battle, refund, support	NDCG (Normalized Discounted Cumulative Gain) @3 =0.651, @5=0.656, @7=0.627, @20=0.634
13	Panichella et al. (2015)	How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution	Bayes, SVM, Logictic Regression, J48 and ADTree	Information Giving, Information Seeking, Feature Request, Problem Discovery, Others	Precision = 0.79, Recall = 0.719, F-Measure = 0.672, best results obtained with the features Natural Language Processing (NLP), Text Analysis (TA) and Sentiment Analysis (SA)
14	Gu and Kim (2015)	What parts of your apps are loved by users?	SUR-Miner (POS tag, Parsing Tree and Semantic Dependence Graph (SDG)	Aspect Evaluation, Praises, Feature Request, Bug Reports and Others	F1-score = 0,81

For the mobile app store corpus, we could not find any study that assessed app store review helpfulness either manually or automatically. As app store users could mark any review for any app as: 'Helpful', 'Unhelpful' and 'Spam' and the reviews could be ranked per their helpfulness at Google Play, some of App Store mining researchers such as Chen et al. (2014) and Park et al. (2015) from Table 6 preferred using only Helpful reviews or filter Unhelpful reviews out to train their models. According to Pagano and Maalej (2013)'s dataset only 67,143 (5.96%) reviews are rated by other users regarding their usefulness. From these, 38,519 (57.37%) are considered 100% helpful. Interestingly, 16,671 (24.83%) are rated completed useless. Even though we could not get enough

information about the percentage of helpful or unhelpful reviews in other datasets, the need for filtering these reviews has been apparent to maximize the information extraction capability.

3.4. RQ4: How could the spam reviews be differentiated from legitimate reviews?

As consumers increasingly rely on user reviews and ratings, there is greater incentive to create fraudulent reviews in order to boost sales and to damage competitor reputations on the market. Fraudulent reviews not only mislead customers into poor purchase decisions, but also degrade user trust in online reviews. According to a Harvard Business School study (Luca and Zervas, 2013), 20% of all online reviews on Yelp.com are fake.

Most of the earlier research focused on detecting email and web spam. As the number of online reviews increases, as well as the number of fraudsters, different studies and techniques have been proposed to detect spam reviews. Two main approaches are being used for opinion spam detection: behavioural and textual features. Behavioural features correspond to features such as review date, rating, and geo-location of the reviewer, while textual features refer to methods, such as part-of-speech patterns, word frequency, n-grams and cosine similarity.

Dellarocas (2000), the first to work on immunizing online reputation systems against unfair ratings and discriminatory behaviour, proposed a set of 'exception handling' techniques such as 'controlled anonymity' and 'cluster filtering'. Kim et al. (2006) used SVM regression on different classes of features including structural (e.g., html tags, punctuation, review length), lexical (e.g., n-grams), syntactic (e.g., percentage of verbs and nouns), semantic and metadata (e.g., star rating) features.

Jindal and Liu (2008) observed that spammers tended to create a small number of review templates and then copy them to spam a single product or several different products. To identify the replicated spam reviews, they used two-gram review content comparison method, as in Kim et al. (2006).

Lim et al. (2010) trained a linear regression model to use four different spamming behaviour models as target products and groups, general rating deviation and early rating deviation. Wang et al. (2011) proposed a heterogeneous graph model to capture relations between reviewers, reviews and stores. Sandulescu and Ester (2015) presented two methods: (i) a semantic similarity measure by extracting specific parts-of-speech (POS) patterns and (ii) an LDA model using bag-of-words and opinion phrases.

Within the corpus of mobile app stores, scammers use a great many bogus user accounts or bots in order to download applications multiple times and write fraudulent reviews. In this way, the applications begin appearing on the top charts and have greater visibility in an app store search. In addition, there are numerous sites that allow purchasing of reviews. One example such a site is Fiverr. Even though fake and opinion spam reviews are widespread and have significant manipulative effects on app store success, regulators have only recently begun to crack down on fake reviews (Clover, 2014). We found only a single app store review spam identification study in the literature:

- Chandy and Gu (2012) compared latent class graphical and decision tree models for classification of app spam and analyzed the preliminary results for clustering reviews. They used linear Gaussian parameterization on the labelled data, which achieved higher accuracy than a baseline decision tree model. As a result, they proposed a latent class model for the spam identification task. The details of this study are presented in Table 5.

3.5. RQ5: extracted application features from user reviews

App stores provide a user feedback capability that is particularly useful and interesting from the software requirements engineering point of view. User ratings and reviews are user-driven feedback that may help improve software quality and address missing features.

With regard to extracted application features from app store user reviews, Iacob and Harrison (2013) identified nine different classes of feedback: positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer supports and versioning. Galvis˜Carreño and Winbladh (2013) adopted the Aspect and Sentiment Unification model (ASUM), which incorporates both topic modelling and sentiment analysis to obtain constructive feedback from user comments. They extracted various topics such as updates, features and developers from review text.

Pagano and Maalej (2013) identified topics in user app store reviews by grouping the information as follows:

- Community: References to other reviews or other applications.
- Requirements: All request types such as feature, content, improvement requests, shortcomings and bug reports.
- Rating: User intention to change his/her idea given certain improvements.
- User experience: Helpfulness in terms of application features and user interface.

In addition, they pointed out the correlation between overall app ratings and number of user reviews, app price and amount and type of feedback the application received. Fu et al. (2013) identified 10 top factors that affect the success of an app on mobile application ecosystems: attractiveness, stability, accuracy, compatibility, connectivity, cost, telephony, picture, media and spam. In addition, they identified 0.9% inconsistencies between user review texts and rating that may be caused by careless mistakes or intention to mislead.

Oh et al. (2013) developed a review digest system (SVM classifier) which was tested on 1,711,556 reviews mined from 24,000 Google Play apps. They automatically categorized user reviews into functional and non-functional requests, bug reports and produced a digest featuring the most informative reviews in each category. Chen et al. (2014) compared the Latent Dirichlet Allocation (LDA) and Aspect and Sentiment Unification Model (ASUM) and found that LDA presented many "non-informative or redundant topics" However, they validated their results on user reviews of only four Android apps, and it is not clear that the framework will attain similar good results when applied to other Android apps or other app stores.

Guzman and Maalej (2014) used topic-modelling techniques to group fine-grained explicit features into high-level features using topic modelling LDA and weighted-average techniques. In addition, they compared the relevance of the extracted features with app requirements and concluded that for the top 10 popular extracted features, the words (e.g., upload photo, file exchange – for Dropbox, board pin, time search – for Pinterest) usually described actual app features and conveyed some clues about how the app was used. McIlroy et al. (2015a) and its counterpart studies Khalid (2013) and Khalid et al. (2015) automatically labelled the types of user issues raised in mobile app reviews, such as additional cost, functional complaint, compatibility issue, crashing, feature removal request, network problem, privacy and ethical issue, resource heaviness, response time, uninteresting content, update issue and user interface. They manually labelled a statistically representative sample of user reviews from the Apple app store and Google Play.

Vu et al. (2015) pursued a keyword based approach to collect and mine user opinion from app stores by extracting, ranking and grouping keywords based on semantic similarity. In addition, they

provided a visualization tool that showed the occurrence of keywords over time and reported any unusual patterns. Park et al. (2015) developed a topic model AppLDA that is designed for use on app descriptions and user reviews. Their proposed method enables developers to inspect the reviews and find out important app features of apps.Panichella et al. (2015) presented a system for automatically classifying user reviews based on a predetermined taxonomy, in order to support software maintenance and requirement evolution. Gu and Kim (2015) proposed a SUR-Miner that is a review summarization and categorization tool, which evaluated 2000 sentences from the reviews of 17 Google Play apps. In addition to these studies, McIlroy et al. (2015b) examined this research problem from different perspective, developers' respective and observed that there are positive effects to responding the reviews (users changed their earlier ratings 37.8% of the time) with a median increase of 20% in the rating.

Table 6 presents the list of studies that apply to RQ5 and identifies the related methods, extracted app features and performance.

4. Discussion

The mobile app ecosystem and user reviews contain a wealth of information about user experience and expectations. Developers and app store regulators could leverage the information to better understand their audience. Mining app store data, and in particular user reviews, may provide valuable information for users to reach an informed decision about applications and their features; similarly, it would be valuable for developers to receive user feedback about most liked or expected features, as well as reported bugs in the applications. Mining opinions from app store reviews still requires pre-processing at the content level, including filtering out non-opinionated content and identifying the trustworthiness and genuineness of the opinion and its source. Even though, to date, there is a limited number of research studies analyzing mobile app reviews, the direction and results obtained are promising. Hence, from the perspective of software requirements engineering, with further research, it is expected that app store meta-data will provide a more accurate picture of user choices and expectations. Developers and app store regulators could leverage reviews to better understand their audience. Here we present our principal findings from the SLR.

4.1. Challenges

Challenges in mining app store reviews fall into two main categories:

- The unstructured nature of user reviews and the colloquial language used make the task of extracting application features and user issues from those reviews a challenge, albeit potentially rewarding. Even though some studies useful for data mining user opinion and application features from review texts exist in the literature, the domain and context dependency aspect of the opinion mining problem has not yet been studied for the app ecosystem. Furthermore, the relevance of extracted features has not been cross-validated with the main software engineering concepts.
- Whereas app users rely on the reviews and ratings of others to formulate an informed decision about applications and their features before downloading them, reading all the reviews is time consuming and occasionally deceptive due to misleading or spam reviews. In addition to spam reviews, some reviews do not include useful data for information extraction. Even though some automated systems have been introduced to identify fake and spam reviews and evaluate usefulness, these systems are limited and not yet mature.

4.2. Principal findings

- App store user feedback mining has begun to attract the attention of researchers. Most of the studies selected were of an exploratory nature, based on manual classification and correlation analysis. The number of high-quality app store studies was very limited: we retrieved nine app store mining studies and only one app store spam identification study.
- The automated extraction of app features in online reviews does not consider the nature of the review text. As online app reviews have distinctive features of text (including short length, unstructured phrases, colloquial language and abundant information), there is a need to develop a unique model specific for app store reviews in order to extract targeted app features rather than use conventional methods and techniques developed for different domains and contexts.
- Furthermore, the information requested by users and developers are different. Users are more interested in the opinion and experience of others about the application and which aspects/features they liked or disliked most. Developers have a different point of view when using reviews to:
  - extract usability and user experience information,
  - elicit missing requirements and define requested application features, and
  - improve software quality.
- To deal with abundant information in reviews, external sources such as app crash reports, tweets, community blogs and code repositories could be used to enrich the data. In addition, integration of text with different data sources (such as social media profiles) would be helpful to ensure context level opinion mining, since in terms of preferences and needs, opinions are specific to each person or group.
- Opinion spam or fake review detection is one of the largest problems in the domain. In addition to spam reviews, there are various kinds of user reviews, some of which do not include any useful data for information extraction. Hence, it is necessary to merge multiple criteria not only to identify suspicious reviews but also to differentiate useful reviews from others so that reviews complying with the usefulness criteria can be processed for information extraction. Even though some automated systems have been introduced to identify fake and spam reviews and to evaluate review usefulness, these systems are very limited and not yet mature.

4.3. Future research directions

Our predictions about future of mobile app stores are as following:

We envision that the scale of opinionated text data on Web and mobile app stores will increase tremendously along with other types of big data. While the volume of the big data increases, so do the complexity and relationships underneath the data. Collecting opinions requires concept or semantic level processing and filtering out non-opinionated text data. Users generally prefer to compare specific features of different products. To make such comparisons, researchers need to construct comprehensive common-knowledge bases to spot product features and text polarity. Future opinion-mining systems need broader and deeper commonsense knowledge bases.

On the other hand, the ubiquity of sentiment or opinion analysis as a service (SaaS or OaaS) will make it easy and cheap to embed a SaaS into every application, mobile device and digital experience. Opinion mining and sentiment analysis are inextricably bound to the affective sciences that understand human emotions. Hence, neuroscience and cognitive sciences will inform how opinion mining researchers should measure, analyze and report the

emotions within the text. As there will be more data about a person under one single index, opinion mining will be more specific to user's preferences and needs, predictive sentiment analysis will be another research area to denote the approach in which sentiment analysis is used to predict the changes in the phenomenon of interest.

Our predictions about future of mobile app stores are as following:

Cross-platform and Cross-device Development Creating mobile apps that work easily on multiple platforms (iOS, Android and etc) and devices is presently a challenging task that will not be allowed to persist. Although there is no "one size fits all" approach for mobile app development, we envision the rise in cross-platform mobile development tools. As HTML5 evolves and matures in last couple of year, the future of mobile app development will also make greater use of it to build hybrid mobile apps that will work well across different platforms and devices.

Mobile App Development for Internet of Things (IoT): The future of mobile app development will not be simply about mobile phones and tables, but IoT. As the example of IoT products such as the self-driving cars, the thermostats, the fridges that read the tweets and etc increases and devices start to get more interconnected, the opportunity for software to add value to these smart devices will become even greater.

Search Ads in the App Store: Apple recently began inviting developers to test the App Store's new Search ads that will come to the U.S App Store with IOS 10 in Fall 2016. The introduction of App Store Search Ads will give developers another way to have their apps appear at the top of the results through paid advertisement. This change also brings forward the concern that larger developers could bid more often and win more search ads that will lead their apps to the higher ranks than small developers " apps. However, it is apparent that app users will need others' feedback and reviews more than before to understand if the mobile app really appeals to them, since actual search results will be modified via paid search ads.

References

Andrzejewski, D., Zhu, X., Craven, M., 2009. Incorporating domain knowledge into topic modeling via dirichlet forest priors. In: Proceedings of the 26th Annual International Conference on Machine Learning. ACM, New York, NY, USA, pp. 25–32. doi:10.1145/1553374.1553378.

Aue, A., Gamon, M., 2005. Customizing sentiment classifiers to new domains: a case study. In: Submitted to RANLP-05, the International Conference on Recent Advances in Natural Language Processing, Borovets, BG. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=65430.

Blitzer, J., Dredze, M., Pereira, F., 2007. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In: In ACL, pp. 187–205.

Cambria, E., Schuller, B., Xia, Y., Havasi, C., 2013. New avenues in opinion mining and sentiment analysis. IEEE Intell. Syst. 28 (2), 15–21. doi:10.1109/MIS.2013.30.

Chandy, R., Gu, H., 2012. Identifying spam in the iOS App Store. In: Proceedings of the 2Nd Joint WICOW/AIRWeb Workshop on Web Quality. ACM, New York, NY, USA, pp. 56–59. doi:10.1145/2184305.2184317.

Chen, M., Liu, X., 2011. Predicting popularity of online distributed applications: ITunes app store case analysis. In: Proceedings of the 2011 iConference. ACM, New York, NY, USA, pp. 661–663. doi:10.1145/1940761.1940859.

Chen, N., Lin, J., Hoi, S.C.H., Xiao, X., Zhang, B., 2014. AR-miner: mining Informative Reviews for Developers from Mobile App Marketplace. In: Proceedings of the 36th International Conference on Software Engineering. ACM, New York, NY, USA, pp. 767–778. doi:10.1145/2568225.2568263.

Cheung, C.M., Lee, M.K., Rabjohn, N., 2008. The impact of electronic word-of-mouth: the adoption of online opinions in online customer communities. Internet Res. 18 (3), 229–247.

Clover, J., 2014. MacRumors. URL: http://www.macrumors.com/2014/06/13/apple-fake-app-store-reviews/.

Cosma, A.C., Itu, V.-V., Suciu, D.A., Dinsoreanu, M., Potolea, R., 2014. Overcoming the domain barrier in opinion extraction. In: Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on. IEEE, pp. 289–296.

Dellarocas, C., 2000. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In: Proceedings of the 2Nd ACM Conference on Electronic Commerce. ACM, New York, NY, USA, pp. 150–157. doi:10.1145/352871.352889.

Ding, X., Liu, B., Yu, P.S., 2008. A holistic lexicon-based approach to opinion mining. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. ACM, New York, NY, USA, pp. 231–240. doi:10.1145/1341531.1341561.

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., Sadeh, N., 2013. 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. ACM, New York, NY, USA, pp. 1276–1284. doi:10.1145/2487575.2488202.

Gabrilovich, E., Markovitch, S., 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1606–1611. URL: http://dl.acm.org/citation.cfm?id=1625275.1625535

Galvis Carreño, L.V., Winbladh, K., 2013. Analysis of user comments: an approach for software requirements evolution. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, Piscataway, NJ, USA, pp. 582–591. URL: http://dl.acm.org/citation.cfm?id=2486788.2486865

Ganapathibhotla, M., Liu, B., 2008. Mining opinions in comparative sentences. In: Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 241–248. URL: http://dl.acm.org/citation.cfm?id=1599081.1599112

Ghose, A., Ipeirotis, P.G., 2011. Estimating the helpfulness and economic impact of product reviews: mining text and reviewer characteristics. Knowl. Data Eng., IEEE Trans. 23 (10), 1498–1512.

Gu, X., Kim, S., 2015. "what parts of your apps are loved by users?" (T). In: 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9–13, 2015, pp. 760–770. doi:10.1109/ASE.2015.57.

Guzman, E., Maalej, W., 2014. How do users like this feature? A fine grained sentiment analysis of app reviews.. In: Gorschek, T., Lutz, R.R. (Eds.), RE. IEEE Computer Society, pp. 153–162. URL: http://dblp.uni-trier.de/db/conf/re/re2014.html#GuzmanM14

Gómez, M., Rouvoy, R., Monperrus, M., Seinturier, L., 2015. A recommender system of buggy app checkers for app store moderators. In: Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems. IEEE Press, Piscataway, NJ, USA, pp. 1–11. URL: http://dl.acm.org/citation.cfm?id=2825041.2825043

Ha, E., Wagner, D., 2013. Do Android users write about electric sheep? Examining consumer reviews in Google Play. In: Consumer Communications and Networking Conference (CCNC), 2013 IEEE, pp. 149–157. doi:10.1109/CCNC.2013.6488439.

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

Hoon, L., Vasa, R., Schneider, J.-G., Mouzakis, K., 2012. A preliminary analysis of vocabulary in mobile app user reviews. In: Proceedings of the 24th Australian Computer-Human Interaction Conference. ACM, New York, NY, USA, pp. 245–248. doi:10.1145/2414536.2414578.

Hu, M., Liu, B., 2004. Mining and summarizing customer reviews. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 168–177. doi:10.1145/1014052.1014073.

Hu, X., Liu, H., 2012. Text analytics in social media. In: Mining text data. Springer, pp. 385–414.

Iacob, C., Harrison, R., 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In: Proceedings of the 10th Working Conference on Mining Software Repositories. IEEE Press, Piscataway, NJ, USA, pp. 41–44. URL: http://dl.acm.org/citation.cfm?id=2487085.2487094

Iacob, C., Veerappa, V., Harrison, R., 2013. What are you complaining about?: A study of online reviews of mobile applications. In: Proceedings of the 27th International BCS Human Computer Interaction Conference. British Computer Society, Swinton, UK, UK, pp. 29:1–29:6. URL: http://dl.acm.org/citation.cfm?id=2578048.2578086

Jindal, N., Liu, B., 2008. Opinion spam and analysis. In: Proceedings of the 2008 International Conference on Web Search and Data Mining. ACM, New York, NY, USA, pp. 219–230. doi:10.1145/1341531.1341560.

Keele, S., 2007. Guidelines for performing systematic literature reviews in software engineering. Technical report, Ver. 2.3 EBSE Technical Report. EBSE.

Khalid, H., 2013. On identifying user complaints of ios apps. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, Piscataway, NJ, USA, pp. 1474–1476. URL: http://dl.acm.org/citation.cfm?id=2486788.2487044

Khalid, H., Shihab, E., Nagappan, M., Hassan, A., 2015. What do mobile app users complain about? Software, IEEE 32 (3), 70–77. doi:10.1109/MS.2014.50.

Kim, S.-M., Pantel, P., Chklovski, T., Pennacchiotti, M., 2006. Automatically assessing review helpfulness. In: Proceedings of the 2006 Conference on empirical methods in natural language processing. Association for Computational Linguistics, pp. 423–430.

Kitchenham, B., 2004. Procedures for Performing Systematic Reviews. Keele University Technical Report TR/SE-0401. Department of Computer Science, Keele University, UK.

Korfiatis, N., García-Bariocanal, E., Sánchez-Alonso, S., 2012. Evaluating content quality and helpfulness of online product reviews: the interplay of review helpfulness vs. review content. Electron. Commer. Res. Appl. 11 (3), 205–217.

Alfonso Ureña-López, L., Buenaga, M., Gómez, J.M., 2001. Integrating linguistic resources in tc through wsd. Comput. Hum. 35 (2), 215–230. URL: http://www.jstor.org/stable/30204851

Lim, E.-P., Nguyen, V.-A., Jindal, N., Liu, B., Lauw, H.W., 2010. Detecting product review spammers using rating behaviors. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management. ACM, New York, NY, USA, pp. 939–948. doi:10.1145/1871437.1871557.

Liu, Y., Huang, X., An, A., Yu, X., 2008. Modeling and predicting the helpfulness of online reviews. In: Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on, pp. 443–452. doi:10.1109/ICDM.2008.94.

Luca, M., Zervas, G., 2013. Fake It Till You Make It: Reputation, Competition, and Yelp Review Fraud. Harvard Business School Working Papers. Harvard Business School. URL: https://ideas.repec.org/p/hbs/wpaper/14-006.html

Martin, W., Harman, M., Jia, Y., Sarro, F., Zhang, Y., 2015. The app sampling problem for app store mining. In: Proceedings of the 12th Working Conference on Mining Software Repositories. IEEE Press, Piscataway, NJ, USA, pp. 123–133. URL: http://dl.acm.org/citation.cfm?id=2820518.2820535

Martin, W., Sarro, F., Jia, Y., Zhang, Y., Harman, M., 2016. A Survey of App Store Analysis for Software Engineering. Technical Report. University College London. URL: http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/research/Research_Notes/RN_16_02.pdf

McIlroy, S., Ali, N., Khalid, H., E. Hassan, A., 2015. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empir. Softw. Eng. 1–40. doi:10.1007/s10664-015-9375-7.

McIlroy, S., Shang, W., Ali, N., Hassan, A., 2015. Is it worth responding to reviews? A case study of the top free apps in the google play store. IEEE Softw. PP (99). doi:10.1109/MS.2015.149. 1–1

Mimno, D., Wallach, H.M., Talley, E., Leenders, M., McCallum, A., 2011. Optimizing semantic coherence in topic models. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 262–272. URL: http://dl.acm.org/citation.cfm?id=2145432.2145462

Moghaddam, S., Jamali, M., Ester, M., 2012. ETF: extended tensor factorization model for personalizing prediction of review helpfulness. In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining. ACM, New York, NY, USA, pp. 163–172. doi:10.1145/2124295.2124316.

Mojica Ruiz, I., Nagappan, M., Adams, B., Berger, T., Dienst, S., Hassan, A., 2015. An examination of the current rating system used in mobile app stores. Software, IEEE PP (99). doi:10.1109/MS.2015.56. 1–1

Mudambi, S.M., Schuff, D., 2010. What makes a helpful online review? A study of customer reviews on amazon.com. MIS Q. 34 (1), 185–200. URL: http://dl.acm.org/citation.cfm?id=2017447.2017457

Mukherjee, A., Liu, B., 2010. Improving gender classification of blog authors. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 207–217. URL: http://dl.acm.org/citation.cfm?id=1870658.1870679

Newman, D., Karimi, S., Cavedon, L., 2009. External evaluation of topic models. In: Proc. of ADCS 2009, pp. 11–18.

Oh, J., Kim, D., Lee, U., Lee, J., Song, J., 2013. Facilitating developer-user interactions with mobile app review digests. In: 2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27–May 2, 2013, Extended Abstracts, pp. 1809–1814. doi:10.1145/2468356.2468681.

Pagano, D., Maalej, W., 2013. User feedback in the appstore: an empirical study.. In: RE. IEEE Computer Society, pp. 125–134. URL: http://dblp.uni-trier.de/db/conf/re/re2013.html#PaganoM13

Pan, S.J., Ni, X., Sun, J.-T., Yang, Q., Chen, Z., 2010. Cross-domain sentiment classification via spectral feature alignment. In: Proceedings of the 19th International Conference on World Wide Web. ACM, New York, NY, USA, pp. 751–760. doi:10.1145/1772690.1772767.

Pan, Y., Zhang, J.Q., 2011. Born unequal: a study of the helpfulness of user-generated product reviews. J. Retailing 87 (4), 598–612. http://dx.doi.org/10.1016/j.jretai.2011.05.002.

Pang, B., Lee, L., 2008. Opinion mining and sentiment analysis. Found. Trends Inf. Retr. 2 (1–2), 1–135. doi:10.1561/1500000011.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C., Canfora, G., Gall, H., 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In: Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on, pp. 281–290. doi:10.1109/ICSM.2015.7332474.

Park, D.H., Liu, M., Zhai, C., Wang, H., 2015. Leveraging user reviews to improve accuracy for mobile app retrieval. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, New York, NY, USA, pp. 533–542. doi:10.1145/2766462.2767759.

Sandulescu, V., Ester, M., 2015. Detecting singleton review spammers using semantic similarity. In: Proceedings of the 24th International Conference on World Wide Web. ACM, New York, NY, USA, pp. 971–976. doi:10.1145/2740908.2742570.

Statista, 2014. Statistics and facts about App Stores. URL: http://www.statista.com/topics/1729/app-stores/.

Tang, H., Tan, S., Cheng, X., 2009. A survey on sentiment detection of reviews. Expert Syst. Appl. 36 (7), 10760–10773. doi:10.1016/j.eswa.2009.02.063.

Tsytsarau, M., Palpanas, T., 2012. Survey on mining subjective data on the web. Data Min. Knowl. Discov. 24 (3), 478–514. doi:10.1007/s10618-011-0238-6.

Vasa, R., Hoon, L., Mouzakis, K., Noguchi, A., 2012. A preliminary analysis of mobile app user reviews. In: Proceedings of the 24th Australian Computer-Human Interaction Conference. ACM, New York, NY, USA, pp. 241–244. doi:10.1145/2414536.2414577.

Vu, P.M., Nguyen, T.T., Pham, H.V., Nguyen, T.T., 2015. Mining user opinions in mobile app reviews: a keyword-based approach. CoRR abs/1505.04657. URL: http://arxiv.org/abs/1505.04657

Wang, G., Xie, S., Liu, B., Yu, P.S., 2011. Review graph based online store review spammer detection. In: Proceedings of the 2011 IEEE 11th International Conference on Data Mining. IEEE Computer Society, Washington, DC, USA, pp. 1242–1247. doi:10.1109/ICDM.2011.124.

Wano, M., Iio, J., 2014. Relationship between reviews at app store and the categories for software. In: Network-Based Information Systems (NBiS), 2014 17th International Conference on, pp. 580–583. doi:10.1109/NBiS.2014.51.

Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. ACM, New York, NY, USA, pp. 38:1–38:10. doi:10.1145/2601248.2601268.

Yang, H., Callan, J., Si, L., 2006. Knowledge Transfer and Opinion Detection in the TREC 2006 Blog Track. In: Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, USA, November 14–17, 2006. URL: http://trec.nist.gov/pubs/trec15/papers/cmu.blog.final.pdf

Necmiye Genc-Nayebi is currently pursuing her Ph.D. degree at the École de Technologie Supérieure (ETS) - Université du Québec, Software Engineering Department. Her research topics are Text Mining, Natural Language Processing, Machine Learning and Spam Detection.
She has 11 years of industry expertise on ***Distributed Network Architecture, Service Oriented Architecture (SOA), Virtualization and Integration Concepts (EAI)***. She has hands on experience on software application development, architecture and testing. Embedded, Mobile and Secure Software Development, Algorithm Design and Implementation are some of her major competencies.

Dr. Alain Abran is a Professor and the Director of the Software Engineering Research Laboratory at the École de Technologie Supérieure (ETS) - Université du Québec.
He is currently Co-executive editor of the Guide to the Software Engineering Body of Knowledge project. He is also actively involved in international software engineering standards and is Co-chair of the Common Software Metrics International Consortium (COSMIC).
Dr. Abran has more than 20 years of industry experience in information systems development and software engineering. The maintenance measurement program he developed and implemented at Montreal Trust, Canada, received one of the 1993 Best of the Best awards from the Quality Assurance Institute.

2. A Measurement Design for the Comparison of Expert Usability Evaluation and Mobile App User Reviews

A Measurement Design for the Comparison of Expert Usability Evaluation and Mobile App User Reviews

Necmiye Genc-Nayebi and Alain Abran

Department of Software Engineering and Information Technology,
Ecole de technologie superieure (ETS) – University of Quebec (Montreal, Canada)

necmiye.genc.1@ens.etsmtl.ca,alain.abran@etsmtl.ca

Abstract. Usability and user experience (U&UX) as important components of software quality are now more critical than ever for mobile app store success. Usability experts use different protocols to evaluate the usability of mobile apps while app store user reviews also produce valuable related information. Our research study proposes a measurement design to compare user reviews and expert-based usability evaluation results that includes an exploratory analysis and topic modeling of user reviews. This design is structured to investigate whether mobile app usability features extracted from user reviews align with subject-matter expert usability evaluation results.
Keywords: Usability measurement, Measurement Design, App Store Reviews, Text Mining

1 Introduction

Usability evaluations have been performed traditionally by subject-matter experts and end users, usually in laboratory and field contexts. However, such evaluations can cover only a limited time-span of the applications and re-doing the same usability evaluation for all available app versions would typically lead to large evaluation costs. Furthermore, evaluating some usability dimensions, such as understandability or learnability, requires more complex procedures and indicators [1]. In addition, studies have demonstrated that both experts and end-users are effective in revealing different usability problems [2,3].

With the advent of mobile ecosystems including mobile apps and related meta-data such as ratings and user reviews, app stores now contain a wealth of information about user experience and expectations. However, it is difficult to manually extract this information due to various factors such as the large quantity of reviews, their lack of structure and varying quality.

In this paper, we present a measurement design to compare the findings from subject-matter expert usability evaluations and corresponding app store user reviews. The measurement design proposes two different approaches: one is through an exploratory analysis and the other is through, first, a semi-supervised topic modeling to extract

usability aspects from user reviews and next, comparing these findings with the results from a prior expert-based usability evaluation. To the best of our knowledge, our work is the first to propose topic modeling techniques to automatically extract usability and user experience (U&UX) information from app store user reviews and to compare usability evaluation results of experts and end-users.

2 Related Work

Usability evaluation of mobile apps is an emerging research area that faces a variety of challenges due to the limitations of mobile devices such as processing capacity, screen size, connectivity, and a lack of a consensus on a usability evaluation methodology [4]. Over the years, different methods and techniques have been proposed for usability evaluation. The leading traditional methods fall into two main categories: inspection methods without end users and test methods with end users [5].

App stores are valuable repositories of app and user data where app users can give feedback about different aspects of an app such as its functionality, design or value. A previous study has reported that app store user reviews are valuable to understand user experience and usability aspects [6], while another study reported that 13%-49% of the content of user reviews contains U&UX information that could be used to improve the software quality [7]. However, these reviews permit a limited number of studies on end-user evaluation of usability. For example:

- mining the app store review corpus identified nine different classes of feedback: positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer supports and versioning [8];
- using a support vector machine (SVM) algorithm to classify five main dimensions of usability: memorability, learnability, efficiency, errors/effectiveness and satisfaction [9].

However, we could not identify any related work comparing user reviews and expert-based usability evaluation results.

3 Measurement Design

The first objective of the proposed measurement design is to extract usability related information from user reviews. The second objective is to compare expert-based usability results with user usability evaluation through reviews.

The overview of the proposed measurement design is presented in Figure 1 where Parts 1 and 2 address the first research objective and Part 3 the second research objective. Pre-defined usability keyword frequency analysis of a corpus of reviews was performed in Part 1, while topic modeling was performed in Part 2 to automatically extract usability related topics. The outputs obtained in Parts 1 and 2 and prior subject-matter expert usability evaluation findings are compared in Part 3. The details of the proposed measurement design are presented in the following sub-sections.

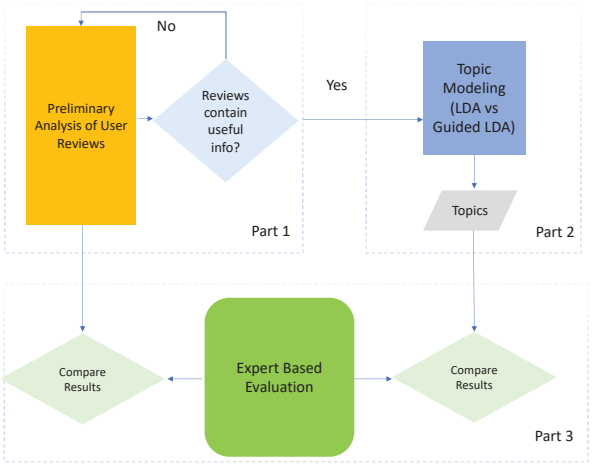


Fig. 1. Overview of the measurement design

3.1 Part 1. Preliminary Analysis of User Reviews for a Set of Apps

In the first part of the research, a preliminary data analysis was performed to discover usability related keyword frequencies in the review corpus through the following four steps:

- Step 1. Selection of apps with the information available to build a review corpus. A reference review corpus was identified and selected, which contained for the same apps both the results of expert-based usability evaluation and user reviews.
- Step 2. Converting usability attributes to aspect words. After the review corpus was populated for the selected versions of the apps, the usability attributes or heuristics were converted into a bag-of-words (BOW). Step 3. Stemming. The Porter stemming algorithm [10] was run to remove affixes from the words and then stemmed versions of the aspect words were searched in the review corpus.
- Step 4. Querying the review corpus. The stemmed words were next queried within individual review corpuses per app and their term frequencies recorded. Query results (e.g., usability aspect term frequencies) were analyzed to understand if (i) user reviews convey good information about usability aspects and (ii) user reviews align with expert-based usability evaluation results.

3.2 Part 2. Usability Topic Modeling

In part 2 of the measurement design, a topic modeling technique was used to help identify individual topics in the document and understand the document corpora in an automated manner. However, unsupervised topic models often lead to topics that are not completely meaningful and/or topics discovered in an unsupervised way that may not

match the true topics in the data. To address this limitation, we leveraged the guided latent Dirichlet allocation (LDA) topic model [11] given in Eqs. 1 to 3 that use Gibbs sampling as an inference method and usability related seed words to improve topic-word distribution.

Step 1: For $k = 1 \ldots \ldots . K$: (1)

(a) Choose regular topic $\emptyset^k{}_r \sim Dirichlet(\beta\_r)$

(b) Choose seed topic $\emptyset^k{}_s \sim Dirichlet(\beta_s)$

Step 2*:* For each seed set $s = 1 \ldots \ldots . S$, (2)

(a) Choose group-topic distribution $\psi_d \sim Dirichlet(\alpha)$

Step 3*:* For each document d: (3)

(a) Choose a binary vector $\vec{b}$ of length S

(b) Choose a document-group distribution $\zeta^d \sim Dirichlet(\tau\vec{b})$

(c) Choose a group variable $g \sim Multinominal(\zeta^d)$

(d) Choose $\theta_d \sim Dirichlet(\psi_g)$

(e) For each token $i = 1 \ldots \ldots . N_d$,

(1) Select a topic $z_i \sim Multinominal(\ \theta_d)$
(2) Select an indicator $x_i \sim Binomial(\ \pi_{zi})$
(3) If x_i is 0
Select a word $w_i \sim Multinominal(\ \theta_{zi}{}^r)$ //choose from LDA style topic
(4) If x_i is 1
Select a word $w_i \sim Multinominal(\ \theta_{zi}{}^s)$

The generative process for a document collection D under the guided LDA model is as follows – see Figure 2:

1. First, the T topic-word distribution $\emptyset^k$ and group-topic distribution ψ_{ds} were generated.
2. Then for each document, a list of seed sets allowed for the document, represented as a binary vector $\vec{b}$, was generated, and then
3. $\vec{b}$ was populated based on the document words, and hence treated as an observed variable.

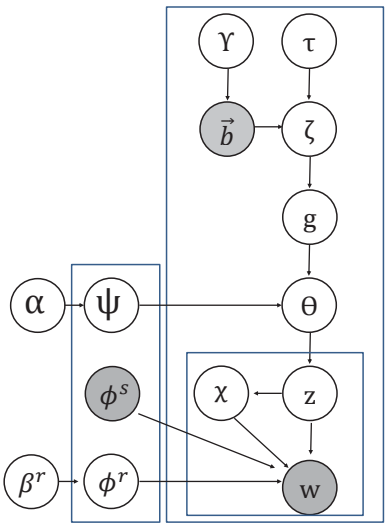


Fig. 2. Graphical model representation of a guided LDA [12]

Step 1. Pre-Processing on Review Corpus.

The review corpus generated in Part 1 was used in the topic modeling. To reduce the dimensionality of the document term matrix, certain data pre-processing and cleaning steps were carried out before proceeding with topic modeling. This pre-processing consisted of:

I. tokenization to segment the review corpus into its atomic elements using the Natural Language Toolkit (NLTK) *tokenize.regexp* module;
II. lower case conversion; and
III. stop-word (e.g. 'the', 'and', 'or', 'a'), punctuation and non-alphabetic phrase removal.

Step 2. Guided LDA Modeling

The guided LDA library1 developed in Python was used in this study. The document term matrix that was generated in the pre-processing step was given as an input to the LDA model. The training step required the input parameters, such as seed topics, seed confidence, be set at 0.15 to bias the seeded words by 15% towards the seeded topic, the number of topics be set at 5, 10 and 20, chunksize at 2000, refresh at 20 and iterations at 100.

[1] https://github.com/vi3k6i5/GuidedLDA

Step 3. Model Accuracy

The guided LDA model performance was tested on a complete review corpus where (i) the number of topics K = 5, 10 and 15 and (ii) LDA model was taken as the baseline. The topic model was run on individual corpuses per app.

3.3 Part 3. Evaluation of Results

In this part of the research, usability aspects extracted from user reviews were compared with expert-based usability evaluation results for the most frequent and the less frequent usability attributes:

- If the user reviews had more positive associations for the usability term, its user review evaluation rating were accepted as *positive*, corresponding to 4-5 stars given by usability experts in prior evaluations.
- If the user reviews had more negative associations for the usability term, its user review evaluation rating was accepted as *negative*, corresponding to 1-2 stars given by usability experts in prior evaluations.
- Equal numbers of positive and negative reviews were given a *neutral* evaluation rating, corresponding to 3 stars given by usability experts in prior evaluations.

4 Work in Progress and Future Work

In this proposed measurement design, usability aspects for expert-based usability evaluation questionnaires were first extracted and then converted into a BOWs in order to trace them back in user reviews. In addition, a guided LDA topic modeling was developed to automatically capture usability aspects intrinsic to the review texts. Expert-based usability evaluation results were compared with user evaluations expressed through reviews and the identified alignment and differences reported. We believe that this proposed measurement design is useful for supporting developers, U&UX designers and researchers to better understand user experience and opinion on mobile application usability aspects, which, finally, can lead to improved software quality.

In future work, we will explore the performance of our guided LDA topic model vs LDA topic model as baseline. Our topic model will also be run on individual review corpuses per app to find the percentage of clustered words that are directly related to usability and usability aspects such as efficiency, errors/effectiveness, etc. Next, for the top 10 and 10 lowest frequency terms identified with preliminary analysis in Part 1 and topic modeling in Part 2, user and expert evaluation ratings will be compared to determine possible alignments or differences between two different usability evaluation methods. A reference review corpus has already been selected which contains for the same apps both the results of expert-based usability evaluation as well as user reviews. This is the app dataset from [13] that includes a set 99 mobile apps evaluated by three usability experts. Since 19 out of the 99 apps from the study are no longer available in the Apple app store and there is no review available for five (5) other mobile apps

within this reference set, our review corpus will be populated with 75 mobile app selected versions of the app dataset.

References

1. T. Grossman, G. Fitzmaurice, and R. Attar, "A survey of software learnability: metrics, methodologies and guidelines," *SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 649–658.
2. P.-Y. Yen and S. Bakken, "A Comparison of Usability Evaluation Methods: Heuristic Evaluation versus End-User Think-Aloud Protocol - An Example from a Web-based Communication Tool for Nurse Scheduling," *AMIA Annual Symposium Proceedings,* 2009, pp. 714–8.
3. L. Hasan, A. Morris, and S. Probets, "A Comparison of Usability Evaluation Methods for Evaluating E-Commerce Websites," *Behaviour & Information Technology*, July 2012, vol. 31, no. 7, pp. 707–737.
4. D. Zhang and B. Adipat, "Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications," *International Journal of Human–Computer Interaction*, 2005, vol. 18, no. 3, pp. 293–308.
5. A. Holzinger, "Usability Engineering Methods for Software Developers," *Communications of the ACM*, Jan. 2005, vol. 48, no. 1, pp. 71–74.
6. N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews", *Journal of System & Software*, 2017, vol. 125, no. Supplement C, pp. 207–219.
7. S. Hedegaard and J. G. Simonsen, "Extracting Usability and User Experience Information from Online User Reviews," *SIGCHI Conference on Human Factors in Computing Systems*, 2013, New York, NY, pp. 2089–2098.
8. C. Iacob and R. Harrison, "Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews," *10th Working Conference on Mining Software Repositories*, Piscataway, NJ, 2013, pp. 41–44.
9. E. Bakiu and E. Guzman, "Which Feature is Unusable? Detecting Usability and User Experience Issues from User Reviews," *IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 182–187.
10. M. F. Porter, "Readings in Information Retrieval," K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 313–316.
11. J. Jagarlamudi, H. Daumé III, and R. Udupa, "Incorporating Lexical Priors into Topic Models," *13th Conference of the European Chapter of the Association for Computational Linguistics*, 2012, Stroudsburg, PA, pp. 204–213.
12. D. M. Blei, A. Y. Ng, M. I. Jordan, and J. Lafferty, "Latent dirichlet allocation," *Journal of Machine Learning Research*, 2003, vol. 3, p. 2003.
13. F. Nayebi, "iOS application user rating prediction using usability evaluation and machine learning," PhD Thesis, École de technologie supérieure, University of Quebec, Montreal, Canada, 2015, p. 203.

3. An Empirical Study on the Comparison of Expert Usability Evaluation and Mobile App User Reviews

Empirical Software Engineering

An Empirical Study on the Comparison of Expert Usability Evaluation and Mobile App User Reviews

--Manuscript Draft--

Manuscript Number:	EMSE-D-18-00342
Full Title:	An Empirical Study on the Comparison of Expert Usability Evaluation and Mobile App User Reviews
Article Type:	Journal First Paper
Keywords:	app store mining, usability evaluation, user experience, user feedback, topic modelling
Corresponding Author:	Necmiye Genc Ecole de technologie superieure Montreal, Quebec CANADA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Ecole de technologie superieure
Corresponding Author's Secondary Institution:	
First Author:	Necmiye Genc
First Author Secondary Information:	
Order of Authors:	Necmiye Genc
	Alain Abran
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	The rapid deployment of mobile devices and the proliferation of mobile apps have dramatically heightened competition in the development of high-quality applications. Usability and user experience (U\&UX), already important components of software quality, have become even more critical for app marketplace success. While app store user reviews include valuable information about U\&UX issues, it is usually difficult to extract this information because of the large quantity of unstructured user reviews where usability is often an implicit aspect of quality well hidden within the review text. This paper reports on the empirical analysis using the earlier research design that is based on two different approaches, exploratory and guided topic modelling to extract usability aspects from user reviews. First, a corpus of reviews for 70 mobile apps are analyzed in terms of word frequencies to verify whether or not the corpus contains useful information about usability aspects. Next, U\&UX related issues and users' praises are extracted automatically with a guided Latent Dirichlet Allocation (LDA) model by incorporating sets of seed words (lexical priors) that usability experts find useful as representative of the usability topics in a corpus. In this way, the LDA model improves word-topic distributions and topic-document distributions towards usability. Finally, exploratory analysis and topic modelling results are compared with usability evaluation of the experts for same mobile app versions. From the analysis of over fifty thousands user reviews in the reference dataset, only 14\% of user reviews were directly associated with the usability aspects of mobile apps. Results show that user reviews address usability aspects similar to those reported in earlier expert usability evaluation studies. Analyzing user reviews as an end-user think-aloud protocol identifies more obstacles to functionality and task performance while expert evaluations reveal more general interface and design problems. In addition, the lexical priors, usability seed words, incorporated into our topic modelling significantly improved word clustering performance.

Noname manuscript No.
(will be inserted by the editor)

An Empirical Study on the Comparison of Expert Usability Evaluation and Mobile App User Reviews

Necmiye Genc-Nayebi · Alain Abran

Received: date / Accepted: date

Necmiye Genc-Nayebi
Department of Software Engineering and Information Technologies, École de technologie supérieure, University of Québec, Montréal, Canada
E-mail: necmiye.genc.1@ens.etsmtl.ca

Alain Abran
Department of Software Engineering and Information Technologies, École de technologie supérieure, University of Québec, Montréal, Canada
E-mail: alain.abran@etsmtl.ca

Abstract The rapid deployment of mobile devices and the proliferation of mobile apps have dramatically heightened competition in the development of high-quality applications. Usability and user experience (U&UX), already important components of software quality, have become even more critical for app marketplace success. While app store user reviews include valuable information about U&UX issues, it is usually difficult to extract this information because of the large quantity of unstructured user reviews where usability is often an implicit aspect of quality well hidden within the review text.

This paper reports on the empirical analysis using the earlier research design that is based on two different approaches, exploratory and guided topic modelling to extract usability aspects from user reviews. First, a corpus of reviews for 70 mobile apps are analyzed in terms of word frequencies to verify whether or not the corpus contains useful information about usability aspects. Next, U&UX related issues and users' praises are extracted automatically with a guided Latent Dirichlet Allocation (LDA) model by incorporating sets of seed words (lexical priors) that usability experts find useful as representative of the usability topics in a corpus. In this way, the LDA model improves word-topic distributions and topic-document distributions towards usability. Finally, exploratory analysis and topic modelling results are compared with usability evaluation of the experts for same mobile app versions.

From the analysis of over fifty thousands user reviews in the reference dataset, only 14% of user reviews were directly associated with the usability aspects of mobile apps. Results show that user reviews address usability aspects similar to those reported in earlier expert usability evaluation studies. Analyzing user reviews as an end-user think-aloud protocol identifies more obstacles to functionality and task performance while expert evaluations reveal more general interface and design problems. In addition, the lexical priors, usability seed words, incorporated into our topic modelling significantly improved word clustering performance.

Keywords app store mining · usability evaluation · user experience · user feedback · topic modelling

1 Introduction

Mobile devices and applications provide significant advantages to their users, such as portability, accessibility and location awareness, while at the same time being constrained by limited resources, including screen size, hardware and network connectivity. These limitations, together with varying usability needs, lead to complex usability requirements that are more critical than ever for app store success.

Usability evaluations have traditionally been performed by subject-matter experts and end-users, usually in laboratory and field contexts. However, such evaluations typically cover only a limited time-span within the application's life, and the cost of re-doing the same evaluation for subsequent versions would be costly. Furthermore, evaluating some aspects of usability such as understandability or learnability not only requires more complex procedures and indicators but also is more time consuming [1]. Studies have also demonstrated that while both experts and end-users are effective in revealing usability problems, they capture different usability perspectives [2] [3].

Mobile ecosystems contain valuable information such as bug reports, feature requests or user experience [4]. However, it is difficult to extract this information manually, due to various factors such as the large quantity of reviews, on the order of thousands per day for some popular mobile applications, their lack of structure and their varying quality.

Usability aspects are usually not referred to explicitly in user reviews, while other aspects such as cost, network problem or functional error often are [5]. When users are asked to express their experience with an app and their opinion on usability, they do not use direct terms such as effectiveness, efficiency or cognitive load [6]. Hence, different information extraction approaches need to be developed to extract implicit usability-related aspects from the review text.

Since reading and analyzing huge amount of review corpora is labor intensive, topic models have great potential for helping users understand the corpora. However, this potential is often impeded by the purely unsupervised nature of the topic models that often leads to non-contexual topic results [7]. The researchers address this problem by proposing various statistical models to extract and categorize aspect terms automatically given some seeds in the user interested categories [8],[9], [10].

This paper presents an empirical analysis results using the earlier research design [11] that is based on two different approaches, one exploratory, , the other a guided topic modelling to extract usability aspects from user reviews. A corpus of reviews for a reference dataset is analyzed by querying usability related terms to verify whether or not the corpus contains useful information about usability aspects in the first part of this empirical study.

Next, U&UX related issues and users' praises are extracted automatically with a guided Latent Dirichlet Allocation (LDA) model in the second part of the empirical study. Guided LDA model incorporates sets of seed words (priors) that usability experts find useful as representative of the usability topics in a corpus. In this way, our LDA model leverages seed words to improve word-topic distributions and topic-document distributions through usability.

Finally, the exploratory analysis and topic modelling results are compared with usability evaluation of the experts for the same mobile apps. To the best of our knowledge, our work is the first (i) to leverage topic modelling techniques to automatically extract U&UX information from app store user reviews and (ii) to compare evaluation results of expert and end-user usability information extracted from user reviews.

Our contributions in this empirical study are three-fold:

1. Usability aspects are extracted from an expert based usability evaluation questionnaire and then converted into a bag-of-words (BOWs) to be able to trace them back to user reviews.
2. A guided LDA topic modelling is developed to automatically capture usability aspects that are often implicit in the review text.
3. Expert based usability evaluation results are compared with user evaluations that are expressed through reviews, and the identified alignment and differences are reported.

The paper is structured as follows. Section 2 introduces the related work, including the definitions adopted in this paper. Section 3 describes the details of our

proposed research method. Section 4 discusses the results of the empirical analysis and related findings. Finally, Section 5 presents our conclusions.

2 DEFINITIONS AND RELATED WORK

This section first presents the definitions adopted in this paper, followed by the related work.

2.1 Definitions

For this research the following definitions are adopted:

Usability: The extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [12].

User experience: A person's perceptions and responses that result from the use and/or anticipated use of a product, system or service [12].

User evaluation: As a part of usability evaluation, users are invited to offer their opinions on the application or system being evaluated. User evaluations are important because they measure how the app is perceived by the end-user, whose opinion matters a great deal.

Expert evaluation: Examination of specific aspects of a system interface, related to its effective, efficient and satisfactory interaction with users, by experts in the field. Expert evaluation is also called heuristic evaluation.

Usability attributes: Usability attributes are those features representing usability concepts such as learnability, efficiency, memorability, errors or satisfaction. A usability feature is called an attribute when the feature is evaluated by usability experts.

Usability aspects (features): Aspect-based opinion mining systems take an input text corpus about a product and mine the aspects (features) of the product [13]. A usability feature is referred to as an aspect when it is mined from a corpus of reviews.

2.2 Related Work

Usability attributes are various features used to measure the quality of applications. Based on the ISO 9241 standard, human-computer interaction handbooks, and existing usability studies on mobile applications, nine generic usability attributes have been identified [14].

Over the years different methods and techniques have been proposed for usability evaluation. The leading methods fall into two main categories: (i) inspection

methods without end users and (ii) test methods with end users [15]. Usability evaluation of mobile apps is an emerging research area that faces a variety of challenges due to the limitations of mobile devices such as processing capacity, screen size and connectivity, as well as lack of an agreed usability evaluation methodology. Nayebi [16] proposed a model and related criteria for usability evaluation of iOS applications that included the artifacts Apple human interface guidelines (iOS HIG), related ISO standards and literature defined usability measures and standardized usability questionnaires.

App stores are valuable repositories of app and user data where users can give feedback about different aspects of the app such as functionality, design or value. A previous study has shown that app store user reviews are valuable to understand user experience and usability aspects [5], while another study found that 13%-49% of the user review content contains U&UX information that could be used to improve the software quality [17]. However, there are only a limited number of studies that capture end-user evaluation of usability from the reviews; these include:

- Iacob and Harrison [18] mined the app store review corpus and identified nine different classes of feedback: positive, negative, comparative, price related, missing requirements, issue reporting, usability, customer supports and versioning.
- Bakiu and Guzman [19] used support vector machine (SVM) to classify the five main dimensions of usability memorability, learnability, efficiency, errors and effectiveness and satisfaction.
- Genc-Nayebi and Abran [11] proposed a measurement design to compare user reviews and expert-based usability evaluation results. This design proposed (i) automatic usability aspect extraction and (ii) a comparison procedure without testing it with empirical data.

3 PROPOSED METHOD

3.1 Research Objectives

The first objective of this study was to extract usability related information from user reviews. To accomplish this objective, the following research activities are defined:

- preliminary analysis of user reviews through term frequency analysis,
- topic modelling to automatically extract usability-related topics from a corpus of reviews.

The second objective of this empirical study was to compare expert based usability results with users' evaluation through reviews. The overview of the research method proposed earlier in [11] is presented in Figure 1 and the details are given in Section 3.2 and 3.3.

3.2 Preliminary Analysis of User Reviews

In the first part of the research, a preliminary data analysis was performed after converting usability attributes to a BOWs in order to search for them in the

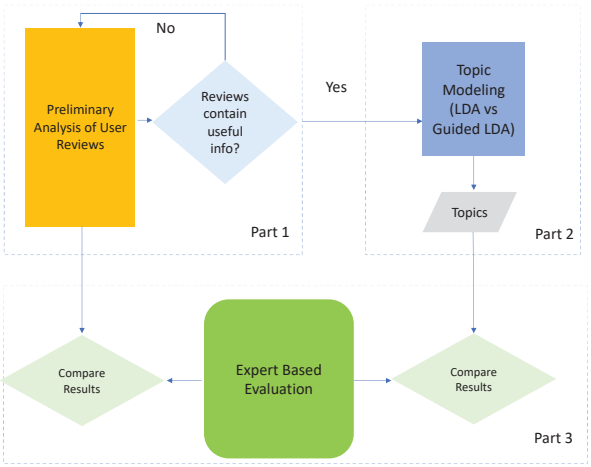


Fig. 1 Overview of Research Method [11]

review corpus and find their normalized term frequency which is the number of times a word appears in a document, divided by the total number of words in that document. Query results (usability aspects' term frequencies) were analyzed to understand whether or not:

- user reviews convey good piece of information about usability aspects,
- user reviews align with expert based usability evaluation results.

The overview of our 5-step research process for the preliminary analysis of user reviews is presented in Figure 2.

3.2.1 Step 1: Review Corpus Generation

The app dataset from an earlier study [16] that included a set 99 mobile apps was selected as the reference set for this study. A number of leading usability heuristics were structured into a questionnaire. See Appendix I, Table 4 for usability evaluation.

Our review corpus was generated by populating it with the user reviews from the same set used in [16]. The reviews were exported from appbot[1] which tracks app store reviews and ratings for all versions since their first launch. Since at the time of our study 14 out of the 99 apps were no longer available in the Apple App Store and there was no review available for fifteen (15) other mobile app versions, only 70 mobile apps had all of the information necessary for our study and were therefore used for populating our review corpus.

3.2.2 Step 2: Converting Usability Attributes to Aspect Words

After the review corpus was populated for the selected versions of the apps, the usability attributes or heuristics were converted into a BOWs. As an example, the

[1] https://appbot.co/

question A.1 ('If App uses visual weight and balance to show users the relative importance of onscreen elements") which is related to the simplicity heuristic was converted into the aspect words simple (A.1) and balance (A.1) and then traced in the user reviews. If an aspect word is derived from multiple questions, this is indicated within parentheses such as (C.0, D.1) or if it is associated with an explicit negative meaning in the question, this is shown with a Negative tagging. In the end, 157 terms were obtained under 21 usability attribute categories.

3.2.3 Step 3: Stemming

The Porter stemming algorithm [20] was run in Step 3 to remove affixes from the terms and then the stemmed versions of the aspect words are searched in the review corpus rather than querying all the different forms of the same words.

3.2.4 Step 4: Querying Review Corpus

The stemmed words in the review corpus were next queried by app and their normalized term frequencies are recorded (Step 4).

3.2.5 Step 5: Comparison of Results

In this part of the research usability aspects extracted from user reviews were compared with expert-based usability evaluation results, as presented in Figure 1, for (i) most frequent usability attributes, (ii) the apps with every good (5 star) or very bad (1 star) expert usability evaluation ratings.

- If the user reviews had more positive associations for the usability term, its user review evaluation rating is accepted Positive, which corresponds to 4-5 stars given by usability experts in the prior evaluation.

- If the user reviews had more negative associations for the usability term, its user review evaluation rating is accepted Negative, which corresponds to 1-2 stars given by usability experts in the prior evaluation.

- Equal number of positive and negative reviews led to an evaluation rating Neutral, which corresponds to 3 stars given by usability experts in the prior evaluation.

3.3 Usability Topic Modeling

In the second part of this empirical study, topic modelling was used to identify individual topics in the document in order to understand the document corpora in an automated manner. However, unsupervised topic models often lead to topics that are not completely meaningful and also topics discovered in an unsupervised way may not match the true topics in the data. Hence, we leverage guided latent Dirichlet allocation (LDA) topic model [10] that uses Gibbs sampling as an inference method, and usability related seed words to improve topic-word distribution.

In the following sub-sections, the details of:

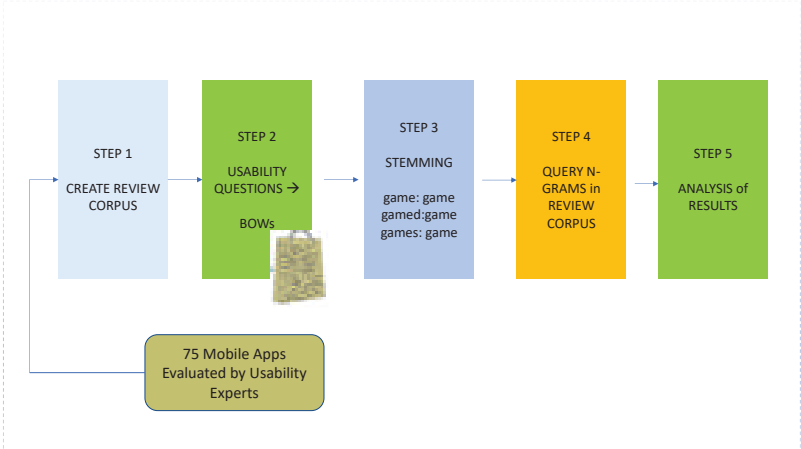


Fig. 2 The 5-step process of the preliminary analysis of user reviews

– LDA Topic Model,
– Guided LDA Model,
– Performance Measure Selection are presented.

3.3.1 LDA Topic Model

Manual elicitation of usability aspects from user reviews is possible but very time-intensive. Hence LDA, a generative probabilistic topic model, was used [21] to extract the topics from the corpus. The basic idea behind the model is that documents are represented as random mixtures of latent topics, where each topic is characterized by a distribution over words. The plate notation of this model is shown in Figure 3. The generative process for a collection D of documents (e.g. a corpus) under the LDA model is as follows:

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
   (b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

The key problem in topic modelling is posterior inference. This refers to reversing the defined generative process and learning the posterior distributions of the latent variables in the model given the observed data.

Gibbs sampling [22], one of the leading inference techniques for LDA models, is based on sampling from conditional distributions of the variables of the posterior. Implementing an LDA collapsed Gibbs sampler is straightforward as it involves setting up the requisite count variables, randomly initializing them and then running a loop over the desired number of iterations where on each loop a topic is sampled for each word instance in the corpus as given in Equation 1.

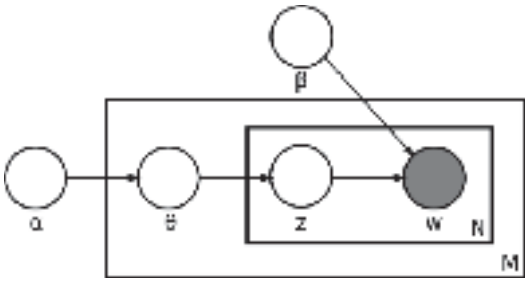


Fig. 3 Plate Notation of LDA [21]

$$P(z_i = j \mid z_{-i}, w_i, d_i) = \frac{C^{WT}_{w_i j} + \eta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\eta} \times \frac{C^{DT}_{d_i j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha} \tag{1}$$

where:

$P(z_i = j)$: The probability that token i is assigned to topic j
z_{-i} : Represents topic assignments of all other tokens
w_i : Word (index) of the i_{th} token
d_i : Document containing the i_{th} token
C^{WT} :Word-topic matrix, the wt matrix we generated.
$\sum_{w=1}^{W} C^{WT}_{wj}$: Total number of tokens (words) in document i.
η : Parameter that sets the topic distribution for the words, the higher the parameter value the more spread out the words will be across the specified number of topics (K).
α : Parameter that sets the topic distribution for the documents, the higher the parameter value the more spread out the documents will be across the specified number of topics (K).
W : Total number of words in the set of documents
T : Number of topics, equivalent of the K we defined earlier.

3.3.2 Guided LDA Model

Due to the unsupervised nature of LDA topic modelling, a user has no way to specify the intended topics in the corpus. To address this problem, seed topics and a guided-LDA model that improves the topic-word probability and document-topic distribution was leveraged in this part of the empirical research.

The guided-LDA model [10] has similar variables semantic as the LDA, presented earlier in Section 3.3.1:

1. For each $k = 1 \cdots \mathrm{T}$,
   (a) Choose regular topic $\phi^r_k \sim \mathrm{Dir}\,(\beta_r)$.
   (b) Choose seed topic $\phi^s_k \sim \mathrm{Dir}\,(\beta_s)$.
   (c) Choose $\pi_k \sim \mathrm{Beta}(1, 1)$.
2. For each *seed* set $s = 1 \cdots \mathrm{S}$,
   (a) Choose group-topic distribution $\psi_s \sim \mathrm{Dir}(\alpha)$.
3. For each document d,

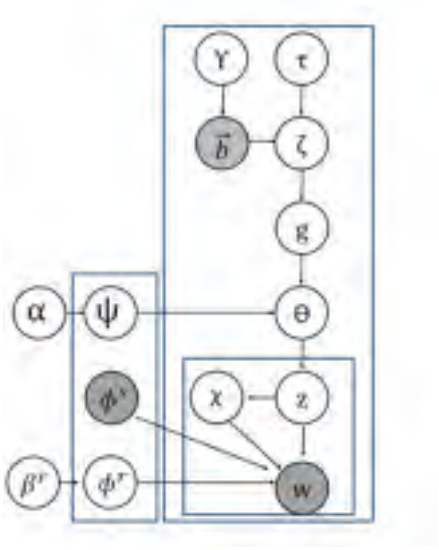


Fig. 4 Graphical model representation of Guided LDA [10]

(a) Choose a binary vector $\vec{\mathbf{b}}$ of length S.
(b) Choose a document-group distribution $\zeta^d \sim \text{Dir}(\tau\vec{\mathbf{b}})$.
(c) Choose a group variable $g \sim \text{Mult}\left(\zeta^d\right)$.
(d) Choose $\theta_d \sim \text{Dir}\left(\psi_g\right)$.
(e) For each token $i = 1 \cdots N_d$:
  (i) Select a topic $z_i \sim \text{Mult}\left(\theta_d\right)$.
  (ii) Select an indicator $x_i \sim \text{Bern}\left(\pi_{z_i}\right)$.
  (iii) if x_i is 0
    – Select a word $w_i \sim \text{Mult}\left(\phi^r_{z_i}\right)$
  (iv) if x_i is 1
    – Select a word $w_i \sim \text{Mult}\left(\phi^s_{z_i}\right)$

The generative process for a document collection D under the guided-LDA model is as follows – see Figure 4:

1. The T topic-word distribution ϕ_k and group-topic distribution ψ_{ds}.
2. Then for each document, a list of seed sets that are allowed for the document is generated. This is represented as $\vec{\mathbf{b}}$ and this binary vector is treated as observed variable, and
3. $\vec{\mathbf{b}}$ is then populated based on the document words and hence it is treated as an observed variable.

3.3.3 Performance Measure Selection

There are different measures used to evaluate topic models in the literature [23]. The most common evaluation measure is the **log-likelihood** of a held-out test set. For an LDA model, a test set is a collection of unseen documents $\boldsymbol{w}_d$ and

the model is described by the topic matrix $\boldsymbol{\phi}$ and the hyper-parameter α is for topic-distribution of documents. Another LDA parameter θ is not taken into consideration as it represents the topic-distributions of documents from the training set, and can therefore be ignored while computing the log-likelihood of unseen documents. The log-likelihood of a model is represented by Equation 2:

$$\mathcal{L}(\boldsymbol{w}) = \log p(\boldsymbol{w}|\boldsymbol{\Phi}, \alpha) = \sum_{d} \log p(\boldsymbol{w}_d|\boldsymbol{\Phi}, \alpha). \tag{2}$$

Another evaluation indicator for topic models that is derived from log-likelihood, is the **perplexity** of held-out documents $\boldsymbol{w}_d$. Perplexity quantifies the modelling power by calculating the inverse log-likelihood of unobserved documents that leads to a decreasing function where tthe higher the likelihood the better the model. Perplexity is defined by Equation 3 for test sample size N.

$$\text{perplexity(test set } \boldsymbol{w}) = \exp\left\{-\frac{\mathcal{L}(\boldsymbol{w})}{\text{count of tokens}}\right\} \tag{3}$$

However, [7] have shown that predictive likelihood or perplexity and human judgment are often not correlated, and even sometimes slightly anti-correlated. Hence, we chose to rely on user judgment to evaluate our guided LDA model performance:

- the model was run on our complete review dataset,
- the top five frequent words in each topic were taken,
- one of the five was replaced with a word that occurs very low in the sequence and
- a human evaluator was asked to spot the 'intruder-one-out' as in [7]

In this way, the model precision turns out to be the fraction of subjects agreeing with the model – Equation 4:

$$MP_k^m = \sum_{s} 1\left(i_{k,s}^m = \boldsymbol{w}k^{\boldsymbol{m}}\right)/\mathbf{S} \tag{4}$$

where $\boldsymbol{w}_k^{\boldsymbol{m}}$ is the index of the intruding word among the words generated from the k^{th} topic inferred by model m, $i_{k,s}^m$ is the intruder selected by subject s on the set of words generated from the k^{th} topic inferred by model m, S denotes the number of subjects.

3.4 Topic Modeling Implementation

The review corpus generated in section 3.2.1 was used in our topic modelling. Since the results of topic models are completely dependent on the features (terms) present in the corpus, reducing the dimensionality of the document term matrix can improve the performance of topic modelling. To reduce the dimensionality of our document term matrix, certain data pre-processing and cleaning steps were pursued before proceeding with topic modelling. Our pre-processing consisted of:

- **Noise Removal:** The text was converted to lower case and also whitespace, punctuation and non-text characters such as HTML tags and punctuations are removed.

- **Tokenization:** A tokenizer divided text into a sequence of tokens, which roughly correspond to words.
- **Lemmatization:** Lemmatization means removing morphological affixes from words, leaving only the word stem, e.g. words such as "download" and "downloading" would both be represented as "download". This reduced the total number of unique words in the dictionary and the number of columns in the document-term matrix.

3.4.1 LDA Implementation

Scikit-learn [24] ***LatentDirichletAllocation*** topic model which implements an online variational Bayes algorithm and supports both online and batch update method was used in the study.

When LatentDirichletAllocation is applied on a "document-term" matrix, the matrix is decomposed into a "topic-term" matrix and a "document-topic" matrix. While "topic-term" matrix is stored as components_ in the model, "document-topic" matrix can be calculated from transform method. To initialize the model, the initial parameters presented Listing 1 wee set and the results as Log Likelihood= -12.7482 and Perplexity= 1042.69 obtained with following parameters:

- batch_size: 128,
- doc_topic_prior: None,
- batch_size: 128,
- evaluate_every: -1,
- learning_decay: 0.7,
- learning_method: online,
- learning_offset: 10.0,
- max_doc_update_iter: 100,
- max_iter: 10,
- mean_change_tol: 0.001,
- n_components: 20,
- n_jobs: -1,
- n_topics: None,
- perp_tol: 0.1,
- random_state: 100,
- topic_word_prior: None,
- total_samples: 1000000.0,
- verbose: 0

3.4.2 Grid Search

The most important tuning parameter for LDA models is *n_components* (number of topics) and search *learning_decay* which controls the learning rate. Hence, these parameters were tuned with grid search that then constructs multiple LDA models for all possible combinations of parameter values in the *param_grid* dictionary.

As a result of grid search, best model parameters are determined as *learning_decay:0.9* and *n_components:10* as presented in Figure 5 and as a result of grid search, Log Likelihood= -4.5810 and Perplexity= 882.13 values were significantly improved.

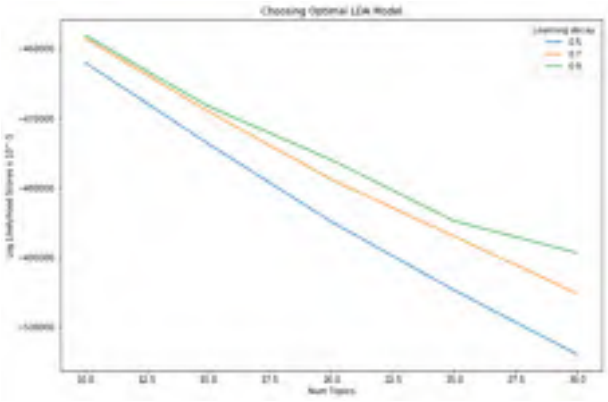


Fig. 5 Grid Search Results for LDA Model

3.4.3 Guided LDA Implementation

The guided LDA library[2] was used in this empirical study to improve topic generation and increase accuracy in topic-word distribution. The document-term matrix generated in the pre-processing step was used as an input to our guided LDA model. The model required the following input parameters:

- seed topics presented in Appendix I, Table 3,
- seed confidence= 0.15 (to bias the seeded words by 15% more towards the seeded topic),
- number of topics=10, 15, 20 and 25,
- chunk size=2000,
- refresh=20,
- iterations=100.

The model performance was tested on the complete review corpus where (i) the number of topics topics=10, 15, 20 and 25 and (ii) the non-supervised LDA model is taken as baseline. The best precision result was obtained through the guided LDA model for 10 topics, as presented in Table 1.

Table 1: Model Performance LDA vs Guided LDA

Topic Model	**Number of Topics**	**Precision**	**Log Likelihood**	**Perplexity**
LDA	10	0.560	**-4.581**	**882.13**
	15	**0.725**	-4.693	926.20
	20	0.618	-4.766	964.39
	25	0.535	-4.844	997.98
Guided LDA	10	**0.794**	**-3.056**	918.61
	15	0.715	-3.076	**918.10**
	20	0.630	-3.061	926.32
	25	0.695	-3.057	926.42

4 RESEARCH RESULTS

4.1 Exploratory Analysis Results

The review corpus is first analyzed to identify unique tokens and also the frequency of usability terms within the corpus. As a result:

- 14825 unique tokens out of 104443 total (%14) are directly associated with usability aspects. The statistics for total unique tokens and usability terms for 70 apps is presented in Figure 6.
- The most frequent usability terms (out of 157 terms in total) in the review corpus were related to *Ease of user input data*, *Searching* and *Physicality and*

[2] https://github.com/vi3k6i5/GuidedLDA

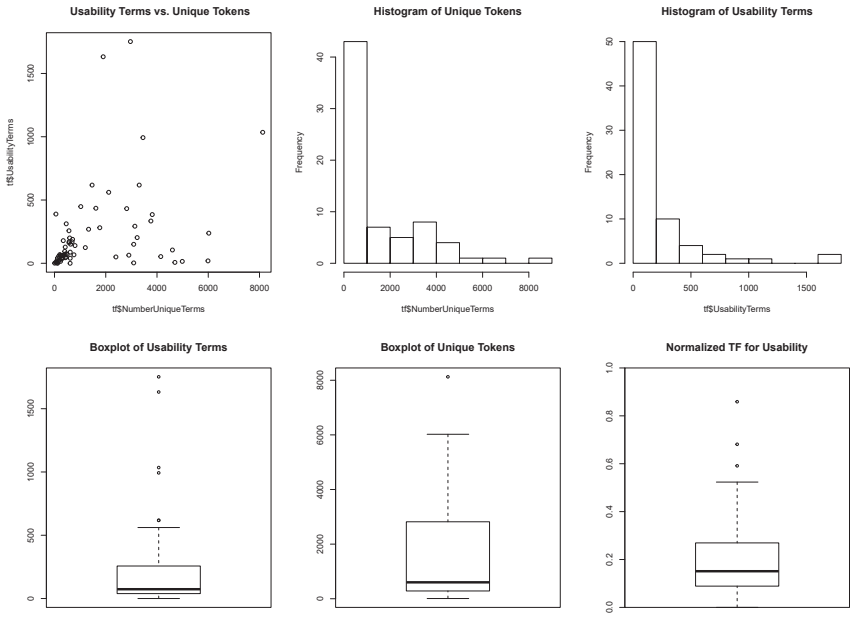


Fig. 6 Usability Terms vs Unique Tokens statistics for review corpus

Realism; whereas less frequent usability terms belong to the concepts that require more detailed expert judgment and evaluation such as *Linguistic clarity*, *Understandability*, *User interface consistency* and *User interface structure* etc. - see Table 2.

Next, user reviews were searched to find the occurrence and associated sentiment (positive or negative or neutral) value for specific usability terms:

Table 2: High Frequency, Top-10 Explicit and Low Frequency Usability Terms

	Usability Term	**Usability Attribute Group**
High Frequency	Easy (*A.1*,*E.1*)	*A.1-Simplicity, E.1- Ease of user input data*
	List *(E.2, I.3)*	*E.2-Ease of User Input Data, I.3-Searching*
	Form *(E.4, M.3)*	*E.4- Ease of User Input data, M.3-Physicality and realism*
	Documentation *(R.2)*	*R.2-Help*
	View *(B.4)*	*B.4-User control and navigation*

	Design *(N.5)*	*N.5-Aesthetic integrity*
	Missing Feature *(V.1)*	*V.1-Missing functionalities*
	Search *(I.1)*	*I.1-Searching*
	Information *(E.1)*	*E.1- Ease of user input data*
	Accurate *(K.4)*	*K.4-User interface structure*
Top-10 Explicit	Accurate *(K.4)*	*K.4-User interface structure*
	Information *(E.1)*	*E.1- Ease of user input data*
	Search *(I.1)*	*I.1-Searching*
	Missing Feature *(V.1)*	*V.1-Missing functionalities*
	Design *(N.5)*	*N.5-Aesthetic integrity*
	User Interface *(K.1)*	*K.1-User interface structure*
	Screen *(C.5)*	*C.5- Understandibility*
	Bug *(S.1)*	*S.1- Error Correction and Prevention*
	UI Control *(L.6)*	*L.6- User Interface Consistency*
	Color Code *(N.3)*	*N.3-Aesthetic Integrity*
Low Frequency	Abbreviations *(D2)*	*D2-Linguistic clarity*
	Hierarchy *(C.5)*	*C.5-Understandability*
	Non-Interactive *(L.7)*	*L.7- User interface consistency*
	Shortcut *(K.3)*	*K.3-User interface structure*
	Silhouette *(M.3)*	*M.3-Physicality and realism*
	Context *(R.4)*	*R.4-Help*
	Grammar *(J.1)*	*(J.1-Application description*
	Input Validation *(E.4)*	*E.4-Ease of user input data*
	Aesthetic *(N.1)*	*N.1-Aesthetic integrity*
	Phrase *(D.1)*	*D.1-Linguistic clarity*

– For the comparison of user and expert usability results, we did not go through 157 term x 70 apps. Instead, we identified 30 mobile apps (out of 70) for which top 10 explicit usability features are more frequent in the review corpus. When user and expert evaluation ratings are compared for these usability attributes/aspects, a 68% alignment is observed in terms of the evaluation results. The details of selected mobile apps and the comparison results are presented in Appendix 1, Table 5
– Another set of 25 mobile apps was identified to which usability experts gave either Very Good (5 star) and Very Bad(1 Star) or Bad (2 Star) ratings for

top 10 explicit usability terms. The first subset of the entire dataset contains 10 individual apps with Very Bad (1 Star) or Bad (2 Star) expert evaluation rating, while the second subset comprises 15 individual apps with Very Good (5 Star) expert evaluation rating. Within these

(i) For the first subset, the vast majority (%80) of user reviews related to the usability aspects were *Positive* opposing to expert usability evaluation results.

(ii) For the second subset, %87.7 of user reviews related to the usability aspects were *Positive* that are in alignment with expert usability evaluation results.

(iii) As the usability term frequency for the second subset was %25 higher than the second subset, this might indicate that the presence of an usability attribute is identified by end-users more easily than its absence.

(iv) Finally, a 60% match was observed between user and expert evaluation results for the entire set.

The details of selected mobile apps and the comparison results are presented in Appendix 1, Table 6.

4.2 Topic Modeling Results

Topic-word distributions and document-topic distributions of the topic modelling were visualized with PyLDAVis [25], a Python library for interactive topic model visualization as shown in Figure 7.

The circles in the left panel shows the top 10 topics, while the bar chart in the right panel displays the top 30 most salient terms in blue and estimated term frequency within the selected topic in red. The relevance of a term to a topic is determined by a weight parameter λ - small values of λ (near 0) highlights potentially rare, but exclusive terms for the selected topic. The distance between the center of circles indicate the similarity between topics, the topics.

The results obtained were in alignment with our earlier experiments and Finding #1 in Section 4.1:

- Only 12% of the topic-words were directly related to usability. The most prevalent usability aspects were about *(i) Errors*, *(ii) Satisfaction (iii) Efficiency* and *(iv) Simplicity*.
- Significant amount of the other topic-words, 55% was about *Functionality (praises and complaints)*, *Feature Demand* and *Task Performance*.
- %33 of topic-words could not be assigned to any specific topic.
- There were no specific topic detected for *(i) Learnability* and *(ii) Memorability* aspects of usability. Figure 8 shows the the overlapping usability topics retrieved from user reviews and expert usability evaluation results.

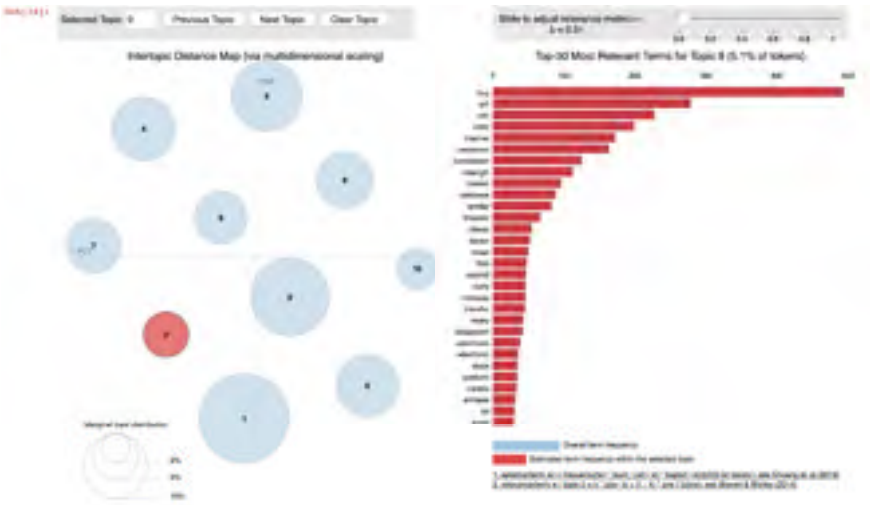


Fig. 7 Usability topics visualized and interpreted with PyLDAvis

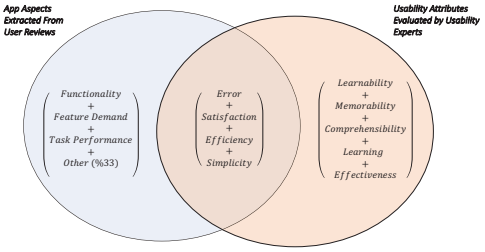


Fig. 8 Usability Topics Venn Diagram

4.3 Threat to Research Validity

The validity of our study is discussed based on two types of threats: construct and external validity. Since our data was extracted from Apple app stores, we relied on reviewers and app store regulators for the reliability of our raw data. As an example, customers may, for various reasons, inappropriately leave deceptive reviews and ratings that do not reflect their true opinions and app store regulators may not have detected or removed these deceptive reviews from their app stores. Therefore, inaccuracies and imprecision in the data may have affected some of our conclusions.

In addition, as the apps subject to this study were selected by usability experts in a prior evaluation, we need to address another type of construct validity due to possible biased selection of the apps, i.e. sampling bias [26]. Even though our app

dataset covers a high degree of diversity in application type, domain and size, we cannot claim that our results generalize beyond the usability subject studied.

4.4 Further Research

In future work, we will investigate app store the deceptive review detection problem to address the construct validity issue. When deceptive reviews are removed from the total app review corpus, the impact of inaccuracies and imprecision in the data to our conclusions will likely diminish. Furthermore, we will expand our guided LDA model to identify other app aspects such as request for requirements, issue reporting, etc. from the review text.

5 CONCLUSION

Usability has become a significant quality dimension to determine the success of mobile applications in app stores, while user ratings and reviews are user-driven feedback that may help improve software quality and address missing application dimensions. However, it is difficult for users to read all the reviews and reach an informed decision due to the ever growing volume of textual review data. In addition, usability is often an implicit aspect well hidden within the review text, and while both experts and end users are effective in revealing different usability perspectives, extracting usability features from the review text requires new approaches and validation.

The first objective of this empirical study was to extract usability related information from user reviews. Hence, we used two approaches: one was usability term frequency analysis, another was guided topic modelling to extract usability information from app store user reviews. Next, in order to accomplish our second research objective, the results were compared with expert evaluations of the same artifacts from a previous research study. We provided only partial information rather than high-level supervision for the topic modelling task in this study. Lexical priors (seed words) were incorporated to guide the model in a certain direction, namely usability. From our empirical analysis, it was found that seed words can improve word clustering performance significantly.

Our empirical research reveals that 55% amount of user reviews addresses functionality, bug/error and feature request related issues and praises, while only 12% of user reviews are directly associated with usability aspects of the app. Also, user usability evaluation highly matches with expert-based evaluation, particularly in terms of identifying existing usability features. In addition, expert evaluations revealed more general interface design problems, while user reviews identified more bugs, functional requirements and obstacles to task performance. We believe that our approach can be useful for supporting developers, U&UX designers and researchers to better understand user experience and opinion on mobile application usability aspects and finally improve software quality.

References

1. T. Grossman, G. Fitzmaurice, R. Attar, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (ACM, New York, NY, USA, 2009), CHI '09, pp. 649–658. DOI 10.1145/1518701.1518803. URL http://doi.acm.org/10.1145/1518701.1518803
2. L. Hasan, A. Morris, S. Probets, Behaviour & Information Technology **31**(7), 707 (2012). DOI 10.1080/0144929X.2011.596996. URL https://doi.org/10.1080/0144929X.2011.596996
3. P.Y. Yen, S. Bakken, AMIA Annual Symposium proceedings / AMIA Symposium. AMIA Symposium **2009**, 714 (2009)
4. D. Pagano, W. Maalej, in *RE* (IEEE Computer Society, 2013), pp. 125–134. URL http://dblp.uni-trier.de/db/conf/re/re2013.html#PaganoM13
5. N. Genc-Nayebi, A. Abran, J. Syst. Softw. **125**(C), 207 (2017). DOI 10.1016/j.jss.2016.11.027. URL https://doi.org/10.1016/j.jss.2016.11.027
6. R. Harrison, D. Flood, D. Duce, Journal of Interaction Science **1**(1), 1 (2013). DOI 10.1186/2194-0827-1-1. URL https://doi.org/10.1186/2194-0827-1-1
7. J. Chang, S. Gerrish, C. Wang, J.L. Boyd-graber, D.M. Blei, in *Advances in Neural Information Processing Systems 22*, ed. by Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, A. Culotta (Curran Associates, Inc., 2009), pp. 288–296. URL http://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.pdf
8. D. Andrzejewski, X. Zhu, M. Craven, in *Proceedings of the 26th Annual International Conference on Machine Learning* (ACM, New York, NY, USA, 2009), ICML '09, pp. 25–32. DOI 10.1145/1553374.1553378. URL http://doi.acm.org/10.1145/1553374.1553378
9. A. Mukherjee, B. Liu, in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1* (Association for Computational Linguistics, Stroudsburg, PA, USA, 2012), ACL '12, pp. 339–348. URL http://dl.acm.org/citation.cfm?id=2390524.2390572
10. J. Jagarlamudi, H. Daumé, III, R. Udupa, in *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics* (Association for Computational Linguistics, Stroudsburg, PA, USA, 2012), EACL '12, pp. 204–213. URL http://dl.acm.org/citation.cfm?id=2380816.2380844
11. N. Genc-Nayebi, A. Abran, in *Proceedings of t28th International Workshop on Software Measurement (IWSM) and the 13th International Conference on Software Process and Product Measurement (MENSURA), IWSM-Mensura 2018, Beijing, China, September 18-20, 2018* (2018), pp. 64–76. DOI 10.1145/3143434.3143461. URL http://doi.acm.org/10.1145/3143434.3143461
12. ISO, *ISO 9241-210:2010 - Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*, 1st edn. (ISO, 2010). URL http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=52075
13. B. Liu, *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies (Morgan & Claypool Publishers, 2012). DOI 10.2200/S00416ED1V01Y201204HLT016. URL https://doi.org/10.2200/S00416ED1V01Y201204HLT016
14. D. Zhang, B. Adipat, International Journal of Human–Computer Interaction **18**(3), 293 (2005). DOI 10.1207/s15327590ijhc1803_3. URL https://doi.org/10.1207/s15327590ijhc1803_3
15. A. Holzinger, Communications of the ACM **48**, 71 (2005). DOI 10.1145/1039539.1039541
16. F. Nayebi, iOS application user rating prediction using usability evaluation and machine learning. PhD Thesis, École de technologie supérieure (2015)
17. S. Hedegaard, J.G. Simonsen, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (ACM, New York, NY, USA, 2013), CHI '13, pp. 2089–2098. DOI 10.1145/2470654.2481286. URL http://doi.acm.org/10.1145/2470654.2481286
18. C. Iacob, R. Harrison, in *Proceedings of the 10th Working Conference on Mining Software Repositories* (IEEE Press, Piscataway, NJ, USA, 2013), MSR '13, pp. 41–44. URL http://dl.acm.org/citation.cfm?id=2487085.2487094
19. E. Bakiu, E. Guzman, in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)* (2017), pp. 182–187. DOI 10.1109/REW.2017.76
20. M.F. Porter, (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997), pp. 313–316. URL http://dl.acm.org/citation.cfm?id=275537.275705

21. D.M. Blei, A.Y. Ng, M.I. Jordan, J. Lafferty, Journal of Machine Learning Research **3**, 2003 (2003)
22. T.L. Griffiths, M. Steyvers, Proceedings of the National Academy of Sciences **101**(suppl 1), 5228 (2004). DOI 10.1073/pnas.0307752101. URL http://www.pnas.org/content/101/suppl_1/5228
23. H.M. Wallach, I. Murray, R. Salakhutdinov, D. Mimno, in *Proceedings of the 26th Annual International Conference on Machine Learning* (ACM, New York, NY, USA, 2009), ICML '09, pp. 1105–1112. DOI 10.1145/1553374.1553515. URL http://doi.acm.org/10.1145/1553374.1553515
24. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, J. Mach. Learn. Res. **12**, 2825 (2011). URL http://dl.acm.org/citation.cfm?id=1953048.2078195
25. C. Sievert, K. Shirley, in *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces* (Association for Computational Linguistics, 2014), pp. 63–70. DOI 10.3115/v1/W14-3110. URL http://aclweb.org/anthology/W14-3110
26. W. Martin, M. Harman, Y. Jia, F. Sarro, Y. Zhang, in *Proceedings of the 12th Working Conference on Mining Software Repositories* (IEEE Press, Piscataway, NJ, USA, 2015), MSR '15, pp. 123–133. URL http://dl.acm.org/citation.cfm?id=2820518.2820535
27. p. 1–5 (2009)
28. L. Hasan, A. Morris, S. Probets, Behaviour & Information Technology **31**(7), 707 (2012). DOI 10.1080/0144929X.2011.596996. URL https://doi.org/10.1080/0144929X.2011.596996

6 Appendix I

Table 3: Seed words for guided LDA model

Topic	Usability Attribute	Seed Words
0	Simplicity	['easy', 'simple', 'function', 'save', 'control', 'focus', 'extra', 'content', 'balance']
1	User control-navigation	['view', 'back', 'button', 'control', 'state', 'marker', 'cancel', 'navigation', 'exist']
2	Understandability	['screen', 'understand', 'content', 'tap', 'area', 'control', 'manual', 'consistent', 'purpose', 'navigate', 'hierarchy']
3	Linguistic Clarity	['word', 'clear', 'phrase', 'abbreviation']
4	Ease of user input data	['easy', 'list', 'format', 'information', 'text', 'field', 'table', 'view', 'data']
5	Collaboration and Connectedness	['location', 'share', 'progress', 'opinion', 'score', 'status']
6	Settings	['setting', 'configuration']
7	Branding	['brand', 'logo']
8	Searching	['list', 'search', 'data', 'local', 'sort', 'result', 'filter']
9	Application description	['error', 'description', 'spell', 'capital', 'letter', 'grammar']
10	User interface structure	['accurate', 'user', 'interface', 'ui', 'text', 'size', 'tap', 'area', 'order', 'content', 'title', 'label', 'orientation', 'shortcut']
11	User Interface consistency	['user', 'interface', 'ui', 'control', 'custom', 'color', 'graphic', 'style', 'control', 'button', 'icon', 'consistent', 'gesture']
12	Physicality and Realism	['form', 'background', 'control', 'shape', 'silhouette']
13	Aesthetic integrity	['design', 'ratio', 'color', 'clear', 'clarity', 'display', 'quality', 'icon', 'view', 'aesthetic']
14	Subtle Animation	['builtin', 'animate', 'interaction']
15	Gestures	['task', 'complex', 'gesture']
16	Rapidity	['fast', 'navigate', 'rapid', 'quick']
17	Help	['documentation', 'current', 'task', 'context', 'help']
18	Error correction and prevention	['bug', 'error', 'fix']
19	In App Purchases	['purchase', 'app', 'store']
20	Missing Functionalities	['missing', 'feature', 'function']

Table 4: List of Mobile App Feature Extraction Studies

Usability Attributes	Criteria	Aspect Words	Aspect Words after Stemming
Simplicity	A.1 App uses visual weight and balance to show users the relative importance of onscreen elements. (Pareto Guideline) A.2 The UI is appropriate for the user's task and skill level, makes it easy to focus on the main task by elevating important content or functionality. A.3 App does not ask user to save when it is not necessary A.4 Always an obvious and safe way to exit a modal task is provided, to reassure users that their work is safe when they dismiss a modal view. A.5 The number and prominence of controls is minimized, in order to decrease their weight in the UI. (Hick-Hyman Law)	simple (A.1), balance (A.1), <easy to focus> (A.2), content (A.2), functionality (A.2), save (A.3), closed (A.4), <without saving> (A.4), <extra control> (A.5, Negative)	simpl(A.1), balanc(A.1), easy (A.2), content (A.2), <easy to focu> (A.2), function (A.2), save (A.3), close(A.4), sav (A.4), <extra control> (A.5, Negative)
User control-navigation	B.1 The path of navigation is predictable, and markers, such as the back button, are provided to inform users where they are and how to retrace their steps. B.2 The user can leave an unwanted state via clearly marked cancel/exit points and without having to embark on an extended UI interaction. B.3 The user knows where he is in the app, how he got there, and where he can go via a navigation controller stack and accurate view names. B.4 A back button or gesture returns the app to a previous view without loss of data.	control (B.1), navigation (B.1), markers (B.1), <back button> (B.1), <unwanted state>, Negative, (B.2), cancel (B.2), exist (B.2), navigation controller (B.3), <view> (B.4)	control (B.1), navig (B.1), marker (B.1), <back button> (B.1), <unwanted st, Negative> (B.2), cancel (B.2), exist (B.2), navigation control (B.3), <view> (B.4)

Understandability	C.1 The number of controls from which the user must choose is minimized. C.2 The app's purpose and usage area can be readily understood from the start. C.3 The user doesn't need to use workarounds or manuals. C.4 All the views are displayed consistently, so that users can apply knowledge gained in one part of the app to the system as a whole. C.5 The app is consistent with the usage paradigms of built-in apps, with the same screen navigation hierarchy, content listing style, and mode switching capability using the tab bar.	understand (C.1, D.1), <number of controls>(C.1, Negative), purpose (C.2), usage (C.2), workaround (C.3, Negative), manual (C.3, Negative), consistent (C.4), screen (C.5), navigation (C.5), hierarchy (C.5), <content listing> (C.5), <mode switch> (C.5), <tab bar> (C.5)	understand (C.1, D.1), <number of control>(C.1, Negative), purpos (C.2), usag (C.2), workaround (C.3, Negative), manual (C.3, Negative), consist (C.4), screen (C.5), navig (C.5), hierarchi (C.5), <content list> (C.5), <mode switch> (C.5), <tab bar> (C.5)
Linguistic Clarity	D.1 Understandable terminology is used in app, that is, words and phrases that are appropriate for the targeted user groups, in all text-based communications. D.2 Abbreviations and acronyms are not used in an app, unless they are straightforward and easily understood.	clear (D.1) word (D.1), phrase (D.1), abbreviations (D.2, Negative), acronyms (D.2, Negative)	clear (D.1) word (D.1), phrase (D.1), abbrevi (D.2, Negative), acronym (D.2, Negative)

Ease of user input data	E.1 The requested user input is balanced with what the app offers the user in return, providing as much information or functionality as possible for each piece of information entered by the user. E.2 Making choices is easy for the user, e.g. by providing a table view or a list picker component instead of a text field. E.3 Required fields are made clear to the user via visual indicators. E.4 The app validates the information that the user enters data forms, informing him if it is not in an acceptable format. E.5 Information from the device is obtained when it makes sense to do so, so that users aren't obliged to provide information that is easily accessible by the app. E.6 The app supports undo and redo.	easy (E.1, R.1), input (E.1), information (E.1, E.6, Negative), <text field> (E.2, Negative), <table view> (E.2), list (E.2), field (E.3), <data format> (E.4), <input validation> (E.4), format (E.4)	easi (E.1, R.1), input (E.1), inform (E.1, E.6, Negative), <text field> (E.2, Negative), <table view> (E.2), list (E.2), field (E.3), <data format> (E.4), <input valid> (E.4), format (E.4)
Collaboration and Connectedness	F.1 Users are able to easily share information that is important to them, like their location, opinions, and high game scores, when it is appropriate. F.2 The app keeps the user informed about the send/receive status of content via a progress indicator.	share (F.1), location (F.1), opinion (F.1), <game score> (F.1), status (F.2), progress (F.2)	share (F.1), locat (F.1), opinion (F.1), <game scor> (F.1), statu (F.2), progress (F.2)
Settings	G.1 Settings about preferred app behaviours and information that users rarely want to change in the app included only when it is appropriate to do so. G.2 Users can easily set their preferred behaviours by using the configuration options in the app.	setting (G.1), configuration (G.1)	set (G.1), configur(G.1)

Branding	H.1 Brand colors or images presented appropriately in a subtle and understated way for greatest effect.	brand (H.1)	brand (H.1)
Searching	I.1 Local data is live-filtered, so that the app can display results more quickly, narrowing them as the user continues to type. I.2 Remote data is filtered while the user types when possible, informing him that he can opt out if the response time is likely to delay the results by more than a second or two. I.3 Search bars displayed above lists, or lists have index. I.4 Search function featured as a distinct mode if it is a primary function in the app, and search tabs provided only in special circumstances. I.5 Placeholder content and partial results are displayed as they become available to give users prompt access. I.6 Scope bar is provided if the data sort naturally into different categories, as this allows users to specify locations or rules in a search, or to filter objects by specific criteria.	search (I.1), <local data> (I.1), results (I.1), <remote data> (I.2), filter (I.2), response time (I.2), bar (I.3), list (I.3), index (I.3), ¡prompt access¿ (I.5), <score bar> (I.6), sort (I.6), criteria (I.7)	search (I.1), <local data> (I.1), result (I.1), <remote data> (I.2), filter (I.2), response tim (I.2), bar (I.3), list (I.3), index (I.3), <prompt access> (I.5), <score bar> (I.6), sort (I.6), criteria (I.7)
Application description	J.1 There is not any spelling, grammatical, and punctuation errors, to avoid creating a negative impression of an app's quality. J.2 All-capital-letter words kept to a minimum, as they can make text very difficult to read.	description (J.1), spelling (J.1), grammar (J.1), punctuation (J.1), error (J.1), <capital letter> (J.2, Negative)	descript (J.1), spell(J.1), grammar (J.1), punctuat (J.1), error (J.1), <capital lett> (J.2, Negative)

User interface structure	K.1 Information is conveyed in a condensed, headline-type style, so that users can absorb it quickly and easily. K.2 The most frequently used (usually higher level) information are placed near the top, and in the following order: from general to specific, and from high level to low level. K.3 Shortcuts have been developed for the most frequently used parts of the app. K.4 Labels and titles are consistent throughout the app, and accurately define the tasks to be performed in the app. K.5 Focus on the primary content is maintained in all orientations, so that users feel they have control over the app and the content they care about. K.6 Tappable elements in an app have a target area of about 44 x 44 points, as this size is important for ease of use. (Fitts Law) K.7 Tappable and untappable areas of the app are clearly recognizable. K.8 App responses to text size changes properly.	<user interface> (K.1, L.1), order (K.2), <high level> (K.2), <low level> (K.2), shortcut (K.3), label (K.4), title (K.4), accurate (K.4), <primary content> (K.5), orientation (K.5), tap (K.6), area (K.6), <tappable area> (K.6, K.7), <untappable area> (K.7), <text size> (K.8)	<user interfac> (K.1, L.1, L.4), order (K.2), <high level> (K.2), <low level> (K.2), shortcut (K.3), label (K.4), titl (K.4), accur (K.4), <primary cont> (K.5), orient (K.5), tap (K.6), area (K.6), <tappable area> (K.6, K.7), <untappable area> (K.7), <text siz> (K.8)

User Interface consistency	L.1 Short labels or well-understood symbols are given to controls, so that users know what they are doing at a glance. L.2 Standard controls and gestures are appropriately and consistently used, so that they behave the way the user expects them to. L.3 The appearance of a controls that perform standard actions is not changed radically, as users will spend time discovering how to use them and wonder what, if anything, this control does that the standard one does not. L.4 The UI conforms to the user's expectations, in that it meets the predictable contextual needs of the user and respects commonly accepted conventions. L.5 Standard buttons and icons did not used to mean something else. L.6 UI controls are customized and they are integrated with app's graphical style, and can be discovered and understood without being conspicuous. L.7 App avoids using the same color in both interactive and noninteractive elements	consistent (L.1), label (L.1), <short label> (L.1), symbol (L.1), <standard control> (L.2), gesture (L.2), control (L.3), <standard button> (L.5), icon (L.5), <UI control> (L.6), custom (L.6), style (L.6), <graphical style> (L.6), <same color> (L.7), interactive (L.7), noninteractive (L.7)	consist (L.1), label (L.1), <short label> (L.1), symbol (L.1), <standard control> (L.2), gestur (L.2), control (L.3), <standard button> (L.5), icon (L.5), <UI control> (L.6), custom (L.6), style (L.6), <graphical styl> (L.6), ¡same color¿ (L.7), interact (L.7), noninteract (L.7)

Physicality and Realism	M.1 Similar UI controls are grouped close to each other. Similarity occurs when objects look similar to one another, and can be perceived as part of a group or pattern. M.2 Related UI controls placed close to eachother. Proximity occurs when elements are placed close together, and can be perceived as belonging to a group. M.3 Figures (forms, silhouettes, and shapes) are differentiated from background (the surrounding area).	real (M.1), <similar control> (M.1), forms (M.3, Negative), silhouette (M.3, Negative), shape (M.3, Negative), background (M.3)	real (M.1), <similar control> (M.1), form (M.3, Negative), silhouett (M.3, Negative), shape (M.3, Negative), background (M.3)
Aesthetic integrity	N.1 The look of high-quality or precious materials are replicated and materials look realistic and valuable. N.2 Relevant metaphors representing real-life objects are used when needed to help the user understand, and learn, the task. N.3 Color coding is used for clarity where appropriate. N.4 The number of colors is limited to 3-4. N.5 Beautiful, high-resolution artwork and icons have designed and used in application in accordance with Fibonacci sequence. N.6 Views are designed in compliance with the rule of thirds and golden ratio guidelines hence UI controls are placed in proper positions. N.7 App supports retina display	Aesthetic (N.1), high-quality (N.1), realistic (N.1), valuable (N.1), <color coding> (N.3), clarity (N.3), color (N.4), high-resolution (N.5), artwork (N.5), icon (N.5), design (N.5), views (N.6), <golden ratio> (N.6), ratio (N.6), <retina display> (N.7)	Aesthet (N.1), high-quality (N.1), realist (N.1), valuable (N.1), <color cod> (N.3), clarity (N.3), color (N.4), high-resolution (N.5), artwork (N.5), icon (N.5), design (N.5), view (N.6), <golden ratio> (N.6), ratio (N.6), <retina display> (N.7)

Subtle Animation	O.1 App makes custom animation consistent with built-in animation when it is appropriate. O.2 Animations are used consistently throughout the app, so that users can rely on the experience it gives them. O.3 Uses animation and interactivity to engage users and help them learn by doing	animation (O.1, O2), <built-in animation> (O.1), interaction (O.3), gesture (P.1), complex (P.2, Negative), task (P.2)	anim (O.1, O2), <built-in anim> (O.1), interact (O.3), gestur (P.1), complex (P.2, Negative), task (P.2)
Gestures	P.1 The actions associated with the standard gestures that users know are not changed. P.2 Complex gestures, or less common ones like swipe or pinch open, are applied as shortcuts to expedite a task, not as the only way to perform a task.	gesture (P.1), complex (P.2), task (P.2)	gesture (P.1), complex (P.2), task (P.2)
Rapidity	Q.1 A launch image is displayed which closely resembles the first screen of the app, to decrease the app's perceived launch time. Q.2 Displaying an About window or a splash screen is avoided, to ensure that users are not prevented from using the app immediately. Q.3 The login requirement is delayed for as long as possible, to enable users to navigate through much of the app and access some of its functionality without logging in. Q.4 App restores its state when restartes, so that users don't have to remember how they had reached it in the first place. Q.5 App avoids asking people to supply setup information Q.6 App is fast and responsive to touch events	rapid (Q.1), <launch time> (Q.2), about (Q.2), <splash screen> (Q.2), login (Q.3), navigate (Q.3), restore (Q.4), set-up (Q.5), fast (Q.6), responsive (Q.6)	rapid (Q.1), <launch tim> (Q.2), about (Q.2), <splash screen> (Q.2), login (Q.3), navig (Q.3), restor (Q.4), set-up (Q.5), fast (Q.6), respons (Q.6)

Help	R.1 The app provides easily accessible help to users when needed. R.2 The Help documentation is properly prepared, and is both appropriate and informative. R.3 The user can easily move between Help and the current task. R.4 Help is context-based, and addresses all the necessary contexts.	help (R.1), documentation (R.2), <current task> (R.3), necessary (R.4), context (R.4)	help (R.1), document (R.2), <current task> (R.3), necessari (R.4), context (R.4)
Error correction and prevention	S.1 Specific bug fixes that customers have been waiting for are specified in the description of a new version of an app.	error (S.1), bug (S.1, Negative), <bug fix> (S.1), description (S.1)	error (S.1), bug (S.1, Negative), <bug fix> (S.1), descript (S.1)
In App Purchases	T.1 App offers in App Purchases (Achat integre) T.2 In app purchases affects the usability	<app purchase> (T.1)	<app purchas> (T.1)
Missing Functionalities	V.1 Are there any features missing which make app less usable?	<missing feature> (V.1)	< featur> (V.1)

Table 5: Evaluation Results for 30 Apps and Top-5 Explicit and Top-5 Implicit Usability Features

Usability Feature	App ID	App Category	User Evalutation Score	Expert Evaluation Score	Match
K4	8	Utility	Positive	5	TRUE
	17	Utility	Positive	4	TRUE
	22	Utility	Positive	4	TRUE
	30	Health	Positive	5	TRUE
	33	Finance	**Negative**	**5**	**FALSE**
	38	Utility	Positive	5	TRUE
	40	Weather	Positive	5	TRUE
	42	Business	Positive	5	TRUE
	73	Health	Positive	5	TRUE
	95	References	Positive	4	TRUE
E1	8	Utility	**Positive**	**3**	**FALSE**
	23	Business	Positive	5	TRUE
	37	References	Neutral	3	TRUE
	40	Weather	Positive	5	TRUE
	48	Business	Positive	5	TRUE
	65	Food&Drink	Positive	5	TRUE
	73	Health	Positive	5	TRUE
	75	Health	Positive	5	TRUE
	84	Sport	Positive	4	TRUE
	95	References	**Positive**	**3**	**FALSE**
I1	12	Productivity	Positive	3	TRUE
	37	References	**Positive**	**5**	**FALSE**
	48	Business	**Neutral**	**3**	**FALSE**
	64	Food&Drink	Positive	5	TRUE
	65	Food&Drink	**Positive**	**5**	**FALSE**
	75	Health	**Positive**	**5**	**FALSE**
	78	Music	Positive	5	TRUE
	91	References	Positive	5	TRUE
	93	References	Positive	4	TRUE
	95	References	Positive	3	TRUE
V1	11	Productivity	Positive	4	TRUE
	12	Productivity	Positive	4	TRUE
	40	Weather	Positive	4	TRUE
	47	Business	Positive	4	TRUE
	48	Business	**Negative**	**4**	**FALSE**
	65	Food&Drink	Positive	4	TRUE
	73	Health	Positive	4	TRUE
	75	Health	Positive	4	TRUE
	78	Music	Positive	4	TRUE
	87	Photo	**Positive**	**2**	**FALSE**
N5	11	Productivity	**Positive**	**2**	**FALSE**
	17	Utility	Positive	5	TRUE
	40	Weather	**Neutral**	**3**	**FALSE**
	48	Business	**Negative**	**4**	**FALSE**

	65	Food&Drink	Positive	4	TRUE
	73	Health	Positive	4	TRUE
	75	Health	Positive	5	TRUE
	78	Music	Negative	3	TRUE
	87	Photo	Positive	4	TRUE
	95	References	**Positive**	**3**	**FALSE**
K1	11	Productivity	Positive	5	TRUE
	17	Utility	Positive	5	TRUE
	39	Weather	Positive	5	TRUE
	40	Weather	Positive	5	TRUE
	48	Business	Positive	5	TRUE
	65	Food&Drink	**Negative**	**5**	**FALSE**
	73	Health	Positive	5	TRUE
	75	Health	Positive	5	TRUE
	87	Music	Positive	4	TRUE
	95	Photo	Positive	4	TRUE
C5	8	Utility	Positive	5	TRUE
	37	References	**Negative**	**5**	**FALSE**
	39	Weather	Positive	5	TRUE
	40	Weather	Positive	4	TRUE
	47	Business	**Negative**	**5**	**FALSE**
	48	Business	**Negative**	**5**	**FALSE**
	49	Business	**Negative**	**4**	**FALSE**
	75	Health	Positive	5	TRUE
	87	Photo	Positive	4	TRUE
	95	References	**Positive**	**3**	**FALSE**
S1	37	References	**Negative**	**3**	**FALSE**
	47	Business	**Negative**	**4**	**FALSE**
	48	Business	**Negative**	**5**	**FALSE**
	49	Business	**Negative**	**5**	**FALSE**
	56	Productivity	**Neutral**	**4**	**FALSE**
	73	Health	Positive	5	TRUE
	78	Music	Positive	4	TRUE
	84	Music	**Negative**	**4**	**FALSE**
	87	Photo	**Neutral**	**4**	**FALSE**
	95	References	Positive	5	TRUE
L6	8	Utility	Positive	4	TRUE
	17	Utility	**Neutral**	**4**	**FALSE**
	39	Weather	Positive	5	TRUE
	40	Weather	Positive	4	TRUE
	48	Business	**Negative**	**5**	**FALSE**
	49	Business	Positive	5	TRUE
	73	Health	Positive	5	TRUE
	75	Health	Positive	5	TRUE
	87	Photo	Positive	4	TRUE
	88	Photo	Positive	4	TRUE
N3	12	Productivity	Positive	5	TRUE
	39	Weather	Positive	5	TRUE

	40	Weather	Positive	5	TRUE
	47	Business	**Negative**	**5**	**FALSE**
	48	Business	**Positive**	**5**	**FALSE**
	49	Business	Positive	4	TRUE
	78	Music	**Negative**	**4**	**FALSE**
	83	Music	Positive	4	TRUE
	87	Photo	Positive	4	TRUE
	88	Photo	**Positive**	**4**	**FALSE**

Table 6: Evaluation Results for Top and Lowest Rated 25 Apps and Top-10 Explicit Usability Terms

Usability Feature	**App ID**	**App Category**	**User Evalutation Score**	**Expert Evaluation Score**	**Match**
K4	21	Productivity	**Positive**	**1**	**FALSE**
	37	Utility	Positive	5	TRUE
	48	Business	**Negative**	**5**	**FALSE**
E1	39	Business	**Positive**	**1**	**FALSE**
	19	Business	Positive	5	TRUE
I1	8	Utility	**Positive**	**1**	**FALSE**
	10	Business	**Negative**	**5**	**FALSE**
V1	26	Business	**Positive**	**2**	**FALSE**
	9	Business	Positive	5	TRUE
N5	23	Business	**Positive**	**1**	**FALSE**
	80	Business	Positive	5	TRUE
K1	25	Business	**Positive**	**1**	**FALSE**
	33	Business	Positive	5	TRUE
	91	Business	Positive	5	TRUE
C5	21	Utility	**Positive**	**2**	**FALSE**
	12	Business	Positive	5	TRUE
	56	Business	Positive	5	TRUE
S1	1	Utility	Negative	1	TRUE
	61	Business	Positive	5	TRUE
	95	Business	Positive	5	TRUE
L6	41	Utility	**Positive**	**2**	**FALSE**
	2	Business	Positive	5	TRUE
	54	Business	Positive	5	TRUE
N3	5	Utility	Negative	1	TRUE
	80	Business	Positive	5	TRUE

APPENDIX III

LIST OF CITATIONS BY PUBLICATIONS

This chapter lists 45 publications that cited author's publications, as of April 2019.

1. A Systematic Literature Review: Opinion Mining Studies from Mobile App Store User Reviews

[1] Fontao, A., Ekwoge, O. M., Santos, R., & Dias-Neto, A. C. (2017, May). Facing up the primary emotions in Mobile Software Ecosystems from Developer Experience. 2nd Workshop on Social, Human, and Economic Aspects of Software, 5-11.

[2] Fontao, A., Lima, F., Ábia, B., dos Santos, R. P., & Dias-Neto, A. C. (2017, September). Hearing the Voice of Developers in Mobile Software Ecosystems. In Proceedings of the 31st Brazilian Symposium on Software Engineering, 4-13.

[3] Licorish, S. A., Savarimuthu, B. T. R., & Keertipati, S. (2017, June). Attributes that Predict which Features to Fix: Lessons for App Store Mining. 21st International Conference on Evaluation and Assessment in Software Engineering, 108-117.

[4] Lin, D., Bezemer, C. P., Zou, Y., & Hassan, A. E. (2018). An empirical study of game reviews on the Steam platform. Empirical Software Engineering, 1-38.

[5] Fontao, A., Ábia, B., Wiese, I., Estacio, B., Quinta, M., dos Santos, R. P., & Dias-Neto, A. C. (2018). Supporting governance of mobile application developers from mining and analyzing technical questions in stack overflow. Journal of Software Engineering Research and Development, 6(1), 8.

[6] Ribeiro, M. I. C., & Dias-Neto, A. C. (2017, May). Company health in mobile software ecosystem (MSECO): research perspectives and challenges. IEEEACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS), 74-75.

[7] Yu, L., Chen, J., Zhou, H., Luo, X., & Liu, K. (2018, June). Localizing Function Errors in Mobile Apps with User Reviews. 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 418-429.

[8] Lin, D. (2019). How can game developers leverage data from online distribution platforms? A case study of the Steam platform (Doctoral dissertation).

[9] Liu, Y., Liu, L., Liu, H., & Li, S. (2019). Information Recommendation Based on Domain Knowledge in App Descriptions for Improving the Quality of Requirements. IEEE Access, 7, 9501-9514.

[10] Dalpiaz, F., & Parente, M. (2019, March). RE-SWOT: From User Feedback to Requirements via Competitor Analysis. In International Working Conference on Requirements Engineering: Foundation for Software Quality, 55-70.

[11] Hassan, S., Bezemer, C. P., & Hassan, A. E. (2018). Studying Bad Updates of Top Free-to-Download Apps in the Google Play Store. IEEE Transactions on Software Engineering.

[12] Garcia Parente, M. (2018). Using NLP and Information Visualization to analyze app reviews (Master's thesis).

[13] Bukhsh, F. A., Arachchige, J. J., & Malik, F. (2018, December). Analyzing excessive user feedback: A big data challenge. In 2018 International Conference on Frontiers of Information Technology (FIT), 206-211.

[14] Zhang, P., Ge, Y., & Lee, H. M. (2018). E-WOM's Impact on App Development. Journal of Computer Information Systems, 1-10.

[15] Prasetyo, B. E., Putri, D. G. P., & Pamungkas, E. W. Aspect Extraction using Informative Data from Mobile App Data Review. International Journal of Computer Applications, 975, 8887.

[16] Mayordomo-Martinez, D., Carrillo-de-Gea, J. M., Garcia-Mateos, G., Garcia-Berna, J. A., Fernandez-Aleman, J. L., Rosero-Lopez, S.,& García-Hernandez, M. (2019). Sustainable Accessibility: A Mobile App for Helping People with Disabilities to Search Accessible Shops. International journal of environmental research and public health, 16(4), 620.

[17] de Lima Fontao, A., dos Santos, R. P., & Dias-Neto, A. C. (2018). Exploiting Repositories in Mobile Software Ecosystems from a Governance Perspective. Information Systems Frontiers, 1-19.

[18] Li, H., Zhang, T., & Wang, Z. Bug or Not Bug? Labeling Issue Reports via User Reviews for Mobile Apps.

[19] Scoccia, G. L., Ruberto, S., Malavolta, I., Autili, M., & Inverardi, P. (2018, May). An investigation into Android run-time permissions from the end users' perspective. 5th International Conference on Mobile Software Engineering and Systems, 45-55.

[20] Bakar, N. H., Kasirun, Z. M., Salleh, N., & Halim, A. H. (2017). Crowdsource Requirements Engineering: Using Online Reviews as Input to Software Features Clustering. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 9(3-3), 141-146.

[21] Dabbous, F. (2017). App download decision from the perspective of Transaction Costs influence on App revenue model.

[22] Akbarabadi, M., & Hosseini, M. (2018). Predicting the helpfulness of online customer reviews: The role of title features. International Journal of Market Research, 1470785318819979.

[23] Genc-Nayebi, N. & Abran, A. (2018). A Measurement Design for the Comparison of Expert Usability Evaluation and Mobile App User Reviews. 28th International Workshop on Software Measurement (IWSM) and the 13th International Conference on Software Process and Product Measurement (MENSURA), IWSM-Mensura 2018, Beijing, China, September 18-20, 64-76.

[24] Srisopha, K., Behnamghader, P., & Boehm, B. (2019). Do users talk about the software in my product? Analyzing user reviews on IoT products. arXiv preprint arXiv:1901.09474.

[25] Zhang, J., Wang, Y., & Xie, T. (2019). Software feature refinement prioritization based on online user review mining. Information and Software Technology, 108, 30-34.

[26] Cunliffe, D. (2019). The market for Welsh language mobile applications–A developers' perspective. Telematics and Informatics, 36, 12-26.

[27] Tavakoli, M., Zhao, L., Heydari, A., & Nenadic, G. (2018). Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools. Expert Systems with Applications.

[28] Ahmad, M., Aftab, S., Bashir, M. S., & Hameed, N. (2018) Sentiment Analysis using SVM: A Systematic Literature Review. International Journal of Advanced Computer Science and Applications, 9.

[29] Pandey, M., Litoriya, R., & Pandey, P. (2018). An ISM Approach for Modeling the Issues and Factors of Mobile App Development. International Journal of Software Engineering and Knowledge Engineering, 28(07), 937-953.

[30] Munoz, S., Araque, O., Llamas, A. F., & Iglesias, C. A. (2018, August). A Cognitive Agent for Mining Bugs Reports, Feature Suggestions and Sentiment in a Mobile Application Store. 4th International Conference on Big Data Innovations and Applications (Innovate-Data), 17-24.

[31] Buchan, J., Bano, M., Zowghi, D., & Volabouth, P. (2018, November). Semi-Automated Extraction of New Requirements from Online Reviews for Software Product Evolution. 25th Australasian Software Engineering Conference (ASWEC), 31-40.

[32] Pedersen, P. E., & Nysveen, H. (2018). A Brief Systematic Review of Mobile App Markets Research: User, Developer and Platform Perspectives. Developer and Platform Perspectives.

[33] AlSubaihin, A., Sarro, F., Black, S., Capra, L., & Harman, M. (2019). App Store Effects on Software Engineering Practices. IEEE Transactions on Software Engineering.

[34] Volabouth, P. (2017). Semi-automated Extraction of New Product Features from Online Reviews to Support Software Product Evolution (Doctoral dissertation, Auckland University of Technology).

[35] Srisopha, K., & Alfayez, R. (2018, May). Software quality through the eyes of the end-user and static analysis tools: a study on Android OSS applications. 1st International Workshop on Software Qualities and Their Dependencies, 1-4.

[36] Kurtanovic, Z. (2018). Mining and Analyzing User Rationale in Software Engineering (Doctoral dissertation, University of Hamburg)

[37] Jisha, R. C., Krishnan, R., & Vikraman, V. (2018, September). Mobile Applications Recommendation Based on User Ratings and Permissions. International Conference on Advances in Computing, Communications and Informatics (ICACCI), 1000-1005.

[38] Ferreira, T., Fernandes, J., Rivero, L., Viana, D., & Santos, R. (2018) Quando os Desenvolvedores Desabafam: Analise de Sentimentos sobre os Comentarios em Ecossistemas de Software de Duas Game Engines.

[39] How SNS activities and knowledge-generating activities affect application purchasing. Internet Electronic Commerce Research, 17 (6), 1-18.

[40] Hendrawan, R. A., Aristio, A. P., & Destrianto, P. R. (2017). Aagregasi toko aplikasi pada piranti bergerak untuk memudahkan pencarian aplikasi multi platform.

[41] Parra-Merono, M. C., Padilla-Piernas, J. M., & Beltran-Bueno, M. A. (2018). Estudio comparativo del comportamiento de queja de los usuarios de la aplicación hotelera Accor para iOS. International Journal of Information Systems and Tourism (IJIST), 3(2), 53-65.

[42] Sirinya La-Ten, & Seri Chomchum. (2018). Mobile application design according to Bloom's new adaptation concept. Research Methodology and Cognitive Science, 15(2), 1-11.

[43] Wulfert, T., Betzing, J. H., Becker, J., Eliciting Customer Preferences for Shopping Companion Apps: A Service Quality Approach. 14. Internationale Tagung Wirtschaftsinformatik (WI 2019), 1220-1235.

[44] Rizun, M., Strzelecki, A. (2019). Knowledge Graph Development for App Store Data Modelling. Computing Research Repository (CoRR), abs/1903.07182.

2. An Empirical Study on the Comparison of Expert Usability Evaluation and Mobile App User Reviews

Genc-Nayebi, N. & Abran, A. (2019). An Empirical Study on the Comparison of Expert Usability Evaluation and Mobile App User Reviews. Empirical Software Engineering. (Submission:EMSE-D-18-00342)

BIBLIOGRAPHY

Ahn, A. (2016). Keeping it real: Improving reviews and ratings in Google Play. Retrieved from https://android-developers.googleblog.com/2016/11/keeping-it-real-improving-reviews-and-ratings-in-google-play.html.

Andrzejewski, D., Zhu, X. & Craven, M. (2009). Incorporating Domain Knowledge into Topic Modeling via Dirichlet Forest Priors. *26th Annual International Conference on Machine Learning*, 382(26), 25-32.

Aue, A. & Gamon, M. (2005). Customizing Sentiment Classifiers to New Domains: a Case Study. *International Conference on Recent Advances in Natural Language Processing*, 1(3.1), 2-1.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics). *Springer-Verlag*, 215.

Blei, D. M., Ng, A. Y., Jordan, M. I. & Lafferty, J. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 993–1022.

Blitzer, J., Dredze, M. & Pereira, F. (2007). Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. *Association for Computational Linguistics*, 187–205.

Brody, S. & Elhadad, N. (2010). An Unsupervised Aspect-sentiment Model for Online Reviews. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (HLT '10), 804–812.

Cambria, E., Schuller, B., Xia, Y. & Havasi, C. (2013). New Avenues in Opinion Mining and Sentiment Analysis. *IEEE Intelligent Systems*, 28(2), 15–21. doi: 10.1109/MIS.2013.30.

Chandy, R. & Gu, H. (2012). Identifying Spam in the iOS App Store. *2Nd Joint WICOW/AIRWeb Workshop on Web Quality*, (WebQuality '12), 56–59. doi: 10.1145/2184305.2184317.

Chang, C.-C. & Lin, C.-J. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 27:1–27:27. doi: 10.1145/1961189.1961199.

Chen, M. & Liu, X. (2011). Predicting Popularity of Online Distributed Applications: ITunes App Store Case Analysis. *iConference 2011*, pp. 661–663. doi: 10.1145/1940761.1940859.

Chen, N., Lin, J., Hoi, S. C. H., Xiao, X. & Zhang, B. (2014). AR-miner: Mining Informative Reviews for Developers from Mobile App Marketplace. *36th International Conference on Software Engineering*, (ICSE 2014), 767–778. doi: 10.1145/2568225.2568263.

Cheung, C. M., Lee, M. K. & Rabjohn, N. (2008). The impact of electronic word-of-mouth: The adoption of online opinions in online customer communities. *Internet Research*, 18(3), 229–247.

Chiu, J. P. C. & Nichols, E. (2015). Named Entity Recognition with Bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4, 357-370. doi: 10.1162/tacl_a_00104.

Clover, J. (2014). Apple Cracking Down on Fake App Store Reviews. Retrieved from https://www.macrumors.com/2014/06/13/apple-fake-app-store-reviews/.

Collobert, R. & Weston, J. (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *25th International Conference on Machine Learning*, (ICML '08), 160–167. doi: 10.1145/1390156.1390177.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, 2493–2537.

Cosma, A. C., Itu, V.-V., Suciu, D. A., Dinsoreanu, M. & Potolea, R. (2014). Overcoming the domain barrier in opinion extraction. *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 289–296. doi: 10.1109/ICCP.2014.6937011.

Dellarocas, C. (2000). Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior. *2Nd ACM Conference on Electronic Commerce*, (EC '00), 150–157. doi: 10.1145/352871.352889.

Ding, X., Liu, B. & Yu, P. S. (2008). A Holistic Lexicon-based Approach to Opinion Mining. *2008 International Conference on Web Search and Data Mining*, (WSDM '08), 231–240. doi: 10.1145/1341531.1341561.

Fernández-Gavilanes, M., Álvarez López, T., Juncal-Martínez, J., Costa-Montenegro, E. & González-Castaño, F. J. (2016). Unsupervised method for sentiment analysis in online texts. *Expert Systems with Applications*, 58, 57–75. doi: https://doi.org/10.1016/j.eswa.2016.03.031.

Freund, Y. & Schapire, R. E. (1995). A Decision-theoretic Generalization of On-line Learning and an Application to Boosting. *Second European Conference on Computational Learning Theory*, (EuroCOLT '95), 23–37.

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J. & Sadeh, N. (2013). Why People Hate Your App: Making Sense of User Feedback in a Mobile App Store. *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (KDD '13), 1276–1284. doi: 10.1145/2487575.2488202.

Gabrilovich, E. & Markovitch, S. (2007). Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. *20th International Joint Conference on Artifical Intelligence*, (IJCAI'07), 1606–1611.

Galvis Carreño, L. V. & Winbladh, K. (2013). Analysis of User Comments: An Approach for Software Requirements Evolution. *2013 International Conference on Software Engineering*, pp. 582–591.

Ganapathibhotla, M. & Liu, B. (2008). Mining Opinions in Comparative Sentences. *22nd International Conference on Computational Linguistics*, 1, 241–248.

Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. *34th International Conference on Machine Learning*, 70, 1243–1252.

Genc-Nayebi, N. & Abran, A. (2017). A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software*, 125, 207–219.

Genc-Nayebi, N. & Abran, A. (2018). A Measurement Design for the Comparison of Expert Usability Evaluation and Mobile App User Reviews. *28th International Workshop on Software Measurement (IWSM) and the 13th International Conference on Software Process and Product Measurement (MENSURA), IWSM-Mensura 2018, Beijing, China, September 18-20, 2018*, 64–76. doi: 10.1145/3143434.3143461.

Ghose, A. & Ipeirotis, P. G. (2011). Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. *Knowledge and Data Engineering, IEEE Transactions on*, 23(10), 1498–1512.

Gilks, W., Richardson, S. & Spiegelhalter, D. (1996). *Markov Chain Monte Carlo in Practice*. Taylor & Francis.

Glorot, X. & Bengio, Y. (2010, 13–15 May). Understanding the difficulty of training deep feedforward neural networks. *Thirteenth International Conference on Artificial Intelligence and Statistics*, 9, 249–256.

Google. (2015). Mobile App Marketing Insights: How Consumers Really Find and Use Your Apps. Retrieved from https://www.thinkwithgoogle.com/consumer-insights/mobile-app-marketing-insights/.

Graves, A., Mohamed, A. & Hinton, G. E. (2013). Speech Recognition with Deep Recurrent Neural Networks. *Computing Research Repository (CoRR)*, abs/1303.5778.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R. & Schmidhuber, J. (2015). LSTM: A Search Space Odyssey. *Computing Research Repository (CoRR)*, abs/1503.04069.

Groen, E. C., Kopczyńska, S., Hauer, M. P., Krafft, T. D. & Doerr, J. (2017). Users — The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality

Aspects in Online Reviews. *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 80-89. doi: 10.1109/RE.2017.73.

Gu, X. & Kim, S. (2015). "What Parts of Your Apps are Loved by Users?" (T). *30th IEEE/ACM International Conference on Automated Software Engineering*, 760–770. doi: 10.1109/ASE.2015.57.

Guzman, E. & Maalej, W. (2014). How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 153–162.

Gómez, M., Rouvoy, R., Monperrus, M. & Seinturier, L. (2015). A Recommender System of Buggy App Checkers for App Store Moderators. *Second ACM International Conference on Mobile Software Engineering and Systems*, 1–11.

Ha, E. & Wagner, D. (2013). Do Android users write about electric sheep? Examining consumer reviews in Google Play. *2013 IEEE Consumer Communications and Networking Conference (CCNC)*, 149–157. doi: 10.1109/CCNC.2013.6488439.

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

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.

Hoon, L., Vasa, R., Schneider, J.-G. & Mouzakis, K. (2012). A Preliminary Analysis of Vocabulary in Mobile App User Reviews. *24th Australian Computer-Human Interaction Conference*, pp. 245–248. doi: 10.1145/2414536.2414578.

Hosmer Jr, D. W., Lemeshow, S. & Sturdivant, R. X. (2013). *Applied logistic regression*. John Wiley & Sons.

Hu, M. & Liu, B. (2004). Mining and Summarizing Customer Reviews. *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (KDD '04), 168–177. doi: 10.1145/1014052.1014073.

Hu, X. & Liu, H. (2012). Text analytics in social media. *Mining Text Data*, 385–414.

Huang, Z., Xu, W. & Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. *Computing Research Repository (CoRR)*, abs/1508.01991.

Iacob, C. & Harrison, R. (2013). Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews. *10th Working Conference on Mining Software Repositories*, 41–44.

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

Jindal, N. & Liu, B. (2008). Opinion Spam and Analysis. *2008 International Conference on Web Search and Data Mining*, (WSDM '08), 219–230. doi: 10.1145/1341531.1341560.

Johnson, R. & Zhang, T. (2015). Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. *Advances in Neural Information Processing Systems 28*, 919–927.

Jozefowicz, R., Zaremba, W. & Sutskever, I. (2015). An Empirical Exploration of Recurrent Network Architectures. *32nd International Conference on International Conference on Machine Learning*, 37, 2342–2350.

Kalchbrenner, N., Grefenstette, E. & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. *52nd Annual Meeting of the Association for Computational Linguistics*, 1, 655–665.

Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. In *EBSE Technical Report EBSE-2007-012007*.

Khalid, H., Shihab, E., Nagappan, M. & Hassan, A. (2015). What Do Mobile App Users Complain About? *IEEE Software*, 32(3), 70–77. doi: 10.1109/MS.2014.50.

Khalid, H. (2013). On Identifying User Complaints of iOS Apps. *2013 International Conference on Software Engineering*, 1474–1476.

Kim, S.-M., Pantel, P., Chklovski, T. & Pennacchiotti, M. (2006). Automatically assessing review helpfulness. *2006 Conference on empirical methods in natural language processing*, 423–430.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1746–1751. doi: 10.3115/v1/D14-1181.

Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *Computing Research Repository (CoRR)*, abs/1412.6980.

Kitchenham, B. (2004). *Procedures for Performing Systematic Reviews*. Department of Computer Science, Keele University, UK.

Korfiatis, N., García-Bariocanal, E. & Sánchez-Alonso, S. (2012). Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. *Electronic Commerce Research and Applications*, 11(3), 205–217.

Lai, C. L., Xu, K. Q., Lau, R. Y. K., Li, Y. & Jing, L. (2010). Toward a Language Modeling Approach for Consumer Review Spam Detection. *2010 IEEE 7th International Conference on E-Business Engineering*, 1 – 8. doi: 10.1109/ICEBE.2010.47.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. & Dyer, C. (2016). Neural Architectures for Named Entity Recognition. *Computing Research Repository (CoRR)*, abs/1603.01360. Retrieved from http://arxiv.org/abs/1603.01360.

Li, F., Huang, M., Yang, Y. & Zhu, X. (2011). Learning to Identify Review Spam. *Twenty-Second International Joint Conference on Artificial Intelligence*, 3, 2488–2493. doi: 10.5591/978-1-57735-516-8/IJCAI11-414.

Li, L., Qin, B., Ren, W. & Liu, T. (2017). Document representation and feature combination for deceptive spam review detection. *Neurocomputing*, 254, 33 - 41. doi: https://doi.org/10.1016/j.neucom.2016.10.080.

Lim, E.-P., Nguyen, V.-A., Jindal, N., Liu, B. & Lauw, H. W. (2010). Detecting Product Review Spammers Using Rating Behaviors. *19th ACM International Conference on Information and Knowledge Management*, (CIKM '10), 939–948. doi: 10.1145/1871437.1871557.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers. doi: 10.2200/S00416ED1V01Y201204HLT016.

Liu, J., Cao, Y., Lin, C.-Y., Huang, Y. & Zhou, M. (2007). Low-Quality Product Review Detection in Opinion Summarization. *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 334–342. Poster paper.

Liu, P., Joty, S. R. & Meng, H. M. (2015). Fine-grained Opinion Mining with Recurrent Neural Networks and Word Embeddings. *Empirical Methods in Natural Language Processing*, 1433-1443.

Liu, Y., Huang, X., An, A. & Yu, X. (2008). Modeling and Predicting the Helpfulness of Online Reviews. *Eighth IEEE International Conference on Data Mining*, 443–452. doi: 10.1109/ICDM.2008.94.

Loper, E. & Bird, S. (2002). NLTK: The Natural Language Toolkit. *ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 63–70. doi: 10.3115/1118108.1118117.

Lu, M. & Liang, P. (2017). Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. *21st International Conference on Evaluation and Assessment in Software Engineering*, 344–353. doi: 10.1145/3084226.3084241.

Luca, M. & Zervas, G. (2016). Fake it till you make it: Reputation, competition, and Yelp review fraud. *Management Science*, 62(12), 3412–3427.

Ma, X. & Hovy, E. (2016). End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *54th Annual Meeting of the Association for Computational Linguistics*, 1, 1064–1074. doi: 10.18653/v1/P16-1101.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 1, 142–150.

Martin, W., Harman, M., Jia, Y., Sarro, F. & Zhang, Y. (2015). The App Sampling Problem for App Store Mining. *12th Working Conference on Mining Software Repositories*, 123–133.

McIlroy, S., Shang, W., Ali, N. & Hassan, A. (2015a). Is It Worth Responding to Reviews? A Case Study of the Top Free Apps in the Google Play Store. *IEEE Software*, 99, 1-1. doi: 10.1109/MS.2015.149.

McIlroy, S., Ali, N., Khalid, H. & E. Hassan, A. (2015b). Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering*, 1–40. doi: 10.1007/s10664-015-9375-7.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013). Distributed Representations of Words and Phrases and Their Compositionality. *26th International Conference on Neural Information Processing Systems*, 2, 3111–3119.

Mimno, D., Wallach, H. M., Talley, E., Leenders, M. & McCallum, A. (2011). Optimizing Semantic Coherence in Topic Models. *Conference on Empirical Methods in Natural Language Processing*, 262–272.

Moghaddam, S., Jamali, M. & Ester, M. (2012). ETF: Extended Tensor Factorization Model for Personalizing Prediction of Review Helpfulness. *Fifth ACM International Conference on Web Search and Data Mining*, 163–172. doi: 10.1145/2124295.2124316.

Mojica Ruiz, I., Nagappan, M., Adams, B., Berger, T., Dienst, S. & Hassan, A. (2015). An Examination of the Current Rating System used in Mobile App Stores. *IEEE Software*, 1–1. doi: 10.1109/MS.2015.56.

Mudambi, S. M. & Schuff, D. (2010). What Makes a Helpful Online Review? A Study of Customer Reviews on Amazon.Com. *Management Information Systems Quarterly*, 34(1), 185–200.

Mukherjee, A. & Liu, B. (2010). Improving Gender Classification of Blog Authors. *2010 Conference on Empirical Methods in Natural Language Processing*, 207–217.

Newman, D., Karimi, S. & Cavedon, L. (2009). External evaluation of topic models. *2009 Australasian Document Computing Symposium (ADCS)*, 11–18.

Oh, J., Kim, D., Lee, U., Lee, J. & Song, J. (2013). Facilitating developer-user interactions with mobile app review digests. *2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI*, 1809–1814. doi: 10.1145/2468356.2468681.

Ott, M., Choi, Y., Cardie, C. & Hancock, J. T. (2011). Finding deceptive opinion spam by any stretch of the imagination. *49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 1, 309–319.

Pagano, D. & Maalej, W. (2013). User feedback in the appstore: An empirical study. *2013 21st IEEE International Requirements Engineering Conference (RE)*, 125-134. doi: 10.1109/RE.2013.6636712.

Palomba, F., Salza, P., Ciurumelea, A., Panichella, S., Gall, H., Ferrucci, F. & Lucia, A. D. (2017). Recommending and Localizing Change Requests for Mobile Apps Based on User Reviews. *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 106-117. doi: 10.1109/ICSE.2017.18.

Pan, S. J., Ni, X., Sun, J.-T., Yang, Q. & Chen, Z. (2010). Cross-domain Sentiment Classification via Spectral Feature Alignment. *Proceedings of the 19th International Conference on World Wide Web*, (WWW '10), 751–760. doi: 10.1145/1772690.1772767.

Pan, Y. & Zhang, J. Q. (2011). Born Unequal: A Study of the Helpfulness of User-Generated Product Reviews. *Journal of Retailing*, 87(4), 598–612. doi: http://dx.doi.org/10.1016/j.jretai.2011.05.002.

Pang, B. & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1-2), 1–135. doi: 10.1561/1500000011.

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C., Canfora, G. & Gall, H. (2015). How can i improve my app? Classifying user reviews for software maintenance and evolution. *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 281–290. doi: 10.1109/ICSM.2015.7332474.

Park, D. H., Liu, M., Zhai, C. & Wang, H. (2015). Leveraging User Reviews to Improve Accuracy for Mobile App Retrieval. *38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 533–542. doi: 10.1145/2766462.2767759.

Pennington, J., Socher, R. & Manning, C. (2014). Glove: Global Vectors for Word Representation. *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. doi: 10.3115/v1/D14-1162.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. & Zettlemoyer, L. (2018). Deep contextualized word representations. *Computing Research Repository (CoRR)*, abs/1802.05365.

Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques* (ed. 1st). Springer Publishing Company, Incorporated.

Pontiki, M., Galanis, D., Pavlopoulos, J., Papageorgiou, H., Androutsopoulos, I. & Manandhar, S. (2014). SemEval-2014 Task 4: Aspect Based Sentiment Analysis. *8th International Workshop on Semantic Evaluation (SemEval 2014)*, 27–35.

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S. & Androutsopoulos, I. (2015). SemEval-2015 Task 12: Aspect Based Sentiment Analysis. *9th International Workshop on Semantic Evaluation (SemEval 2015)*, 486–495. doi: 10.18653/v1/S15-2082.

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., Clercq, O. D., Hoste, V., Apidianaki, M., Tannier, X., Loukachevitch, N. V., Kotelnikov, E. V., Bel, N., Zafra, S. M. J. & Eryigit, G. (2016). SemEval-2016 Task 5: Aspect Based Sentiment Analysis. *10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016*, 19–30.

Poria, S., Cambria, E. & Gelbukh, A. (2016). Aspect Extraction for Opinion Mining with a Deep Convolutional Neural Network. *Knowledge-Based Systems*, 108(C), 42–49. doi: 10.1016/j.knosys.2016.06.009.

Powers, D. M. W. (2011). Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1), 37–63.

Ren, Y. & Ji, D. (2017). Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, 385-386, 213–224. doi: https://doi.org/10.1016/j.ins.2017.01.015.

Rush, A. M., Chopra, S. & Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. *Computing Research Repository (CoRR)*, abs/1509.00685. Retrieved from http://arxiv.org/abs/1509.00685.

Sandulescu, V. & Ester, M. (2015). Detecting Singleton Review Spammers Using Semantic Similarity. *24th International Conference on World Wide Web*, (WWW '15 Companion), 971–976. doi: 10.1145/2740908.2742570.

Sänger, M., Leser, U., Kemmerer, S., Adolphs, P. & Klinger, R. (2016). SCARE — The Sentiment Corpus of App Reviews with Fine-grained Annotations in German. *Tenth International Conference on Language Resources and Evaluation (LREC)*.

Scaffidi, C., Bierhoff, K., Chang, E., Felker, M., Ng, H. & Jin, C. (2007). Red Opal: Product-feature Scoring from Reviews. *8th ACM Conference on Electronic Commerce*, (EC '07), 182–191. doi: 10.1145/1250910.1250938.

Schouten, K. & Frasincar, F. (2016). Survey on Aspect-Level Sentiment Analysis. *IEEE Transactions on Knowledge and Data Engineering*, 28(3), 813–830. doi: 10.1109/TKDE.2015.2485209.

Schuster, M. & Paliwal, K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11), 2673–2681. doi: 10.1109/78.650093.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1), 1929–1958. Retrieved from http://dl.acm.org/citation.cfm?id=2627435.2670313.

Statista. (2018). Number of apps available in leading app stores as of 1st quarter 2018. Retrieved from https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

Sun, H., Morales, A. & Yan, X. (2013). Synthetic Review Spamming and Defense. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (KDD '13), 1088–1096. doi: 10.1145/2487575.2487688.

Tang, H., Tan, S. & Cheng, X. (2009). A Survey on Sentiment Detection of Reviews. *Expert Systems with Applications*, 36(7), 10760–10773. doi: 10.1016/j.eswa.2009.02.063.

Titov, I. & McDonald, R. (2008). Modeling Online Reviews with Multi-grain Topic Models. *17th International Conference on World Wide Web*, 111–120. doi: 10.1145/1367497.1367513.

Toh, Z. & Wang, W. (2014). DLIREC: Aspect Term Extraction and Term Polarity Classification System. *8th International Workshop on Semantic Evaluation, SemEval@COLING 2014*, 235–240.

Tsytsarau, M. & Palpanas, T. (2012). Survey on Mining Subjective Data on the Web. *Data Min. Knowl. Discov.*, 24(3), 478–514. doi: 10.1007/s10618-011-0238-6.

Ureña-López, A., Buenaga, M. & Gómez, M. (2001). Integrating Linguistic Resources in TC through WSD. *Computers and the Humanities*, 35(2), 215-230.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Berlin, Heidelberg: Springer-Verlag.

Vasa, R., Hoon, L., Mouzakis, K. & Noguchi, A. (2012). A Preliminary Analysis of Mobile App User Reviews. *24th Australian Computer-Human Interaction Conference*, 241–244. doi: 10.1145/2414536.2414577.

Villarroel, L., Bavota, G., Russo, B., Oliveto, R. & Penta, M. D. (2016). Release Planning of Mobile Apps Based on User Reviews. *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 14-24. doi: 10.1145/2884781.2884818.

Vu, P. M., Nguyen, T. T., Pham, H. V. & Nguyen, T. T. (2015). Mining User Opinions in Mobile App Reviews: A Keyword-based Approach. *Computing Research Repository (CoRR)*, abs/1505.04657.

Wang, G., Xie, S., Liu, B. & Yu, P. S. (2011). Review Graph Based Online Store Review Spammer Detection. *2011 IEEE 11th International Conference on Data Mining*, 1242–1247. doi: 10.1109/ICDM.2011.124.

Wang, Y., Huang, M., zhu, x. & Zhao, L. (2016). Attention-based LSTM for Aspect-level Sentiment Classification. *2016 Conference on Empirical Methods in Natural Language Processing*, 606–615. doi: 10.18653/v1/D16-1058.

Wano, M. & Iio, J. (2014). Relationship between Reviews at App Store and the Categories for Software. *2014 17th International Conference on Network-Based Information Systems (NBiS)*, 580–583. doi: 10.1109/NBiS.2014.51.

Wohlin, C. (2014). Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. *18th International Conference on Evaluation and Assessment in Software Engineering*, 38:1–38:10. doi: 10.1145/2601248.2601268.

Xu, H., Liu, B., Shu, L. & Yu, P. S. (2018). Double Embeddings and CNN-based Sequence Labeling for Aspect Extraction. *56th Annual Meeting of the Association for Computational Linguistics*, 2, 592–598.

Yang, H., Callan, J. & Si, L. (2006). Knowledge Transfer and Opinion Detection in the TREC 2006 Blog Track. *Fifteenth Text REtrieval Conference, TREC 2006*.

Yin, W. & Schütze, H. (2015). Convolutional Neural Network for Paraphrase Identification. *2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 901-911.

Yin, W., Kann, K., Yu, M. & Schütze, H. (2017). Comparative Study of CNN and RNN for Natural Language Processing. *Computing Research Repository (CoRR)*, abs/1702.01923.

Young, T., Hazarika, D., Poria, S. & Cambriax, E. (2017). Recent Trends in Deep Learning Based Natural Language Processing. *Computing Research Repository (CoRR)*, abs/1708.02709.

Zhang, H. (2004). The Optimality of Naive Bayes. *Seventeenth International Florida Artificial Intelligence Research Society Conference*, 562–567.

Zhang, X., Zhao, J. & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. *28th International Conference on Neural Information Processing Systems*, 1, 649–657.