What Users Want for My App
-- Implementing a System to Analyze User Reviews for Mobile Application Maintenance Activities

Libin Tan

University of Tampere
Faculty of Natural Sciences
Computer Science / Soft. Development
M.Sc. thesis
Supervisor: Zheying Zhang
November 2017

University of Tampere
Faculty of Natural Sciences
Computer Science / Software Development
Libin Tan: What Users Want for My App -- Implementing a System to Analyze User Reviews for Mobile Application Maintenance Activities
M.Sc. thesis, 83 pages, 20 index and appendix pages
November 2017

User reviews of mobile applications are often provided by users about their satisfaction, complaints or suggestions on the mobile apps. They form valuable and instructive feedback for app developers in the maintenance process to fix bugs, add new features, enhance the user experience of using the app, etc. However, due to the large quantity and various qualities of the user reviews, it is not realistic for developers to manually read every review to elicit useful information for application maintenance. To address this problem, the study proposes a mobile Application Review Analysis system for Maintenance (ARAM). ARAM is able to identify the useful app review information that can contribute to mobile application maintenance, extract keyphrases from review contents, and analyze users' sentiments for the keyphrases. A new method called DoubleRank was proposed for extracting keyphrases from app reviews. By classifying and summarizing the user reviews, ARAM produces a report of users' opinion on the extracted keyphrases and their associated reviews, which can provide useful suggestions for developers to plan maintenance activities. The generated report is accurate when comparing with the results of manual analysis.

Key words and terms: mobile application user review, software maintenance, keyphrase extraction, sentiment analysis, app review classification, TextRank.

Contents

1. Introduction 1
1.1. Research background 1
1.2. Research aims 1
1.3. Thesis structure 2
2. Application maintenance 4
2.1. Types of software maintenance 4
2.2. Sources of change requests that lead to software maintenance 6
2.3. User reviews as a guidance for application maintenance 8
3. User reviews in app store 10
3.1. User reviews in different app stores 10
3.2. Characteristics of app review contents 11
3.3. Studies of app review data 12
3.4. Taxonomy of user review sentences for maintenance purpose 13
3.5. Four categories of user review sentences leading to possible types of maintenance activities 15
3.6. An overview of ARAM 19
4. Keyphrase extraction 21
4.1. Definition of keyphrase and keyphrase extraction 21
4.2. Keyphrase extraction methods 22
4.2.1. Candidate keyphrase identification 22
4.2.2. Supervised methods 23
4.2.3. Unsupervised methods 23
4.3. TextRank method 26
4.3.1. Introduction to PageRank 26
4.3.2. From PageRank to TextRank 27
5. Sentiment analysis 30
5.1. Terms of sentiment analysis 31
5.2. Three levels of sentiment analysis 31
5.3. Sentiment classification methods 32
5.4. Related studies of sentiment analysis using app review data 35
5.5. Stanford CoreNLP Sentiment Analyzer 37
6. Implementation of ARAM 39
6.1. Overall process of the user review analysis system 39
6.2. Collection of data 40
6.3. User review classifier 41
6.4. Keyphrase extractor 43
6.4.1. Term frequency method 43
6.4.2. TextRank method 45

6.4.3. DoubleRank method 47
6.5. Sentiment analyzer 50
6.6. Visulization of results 50
7. Evaluation of ARAM 53
7.1. System performance 53
7.2. Proportion of the informative reviews classified by ARAM 58
7.3. Evaluation of the keyphrase extraction result 60
7.4. Evaluation of the sentiment analysis result 64
8. Results and discussion 67
8.1. Discussion 67
8.2. Limitations and possible improvements 72
9. Couclusions 75

References 77

Appendix 1 : Questionnaire for keyphrase extraction test 84
Appendix 2 : Questionnaire for sentiment analysis test 92
Appendix 3 : Results of the sentiment analysis test 99

1. Introduction

1.1. Research background

Motivated by the ubiquity of smartphones and tablets, abundant and various mobile applications continue to grow and evolve at a tremendous pace. Resulting from the unexpected software defects and the changing requirements, application developers have to maintain and update their applications frequently to keep them robust and trendy.

Most app stores, such as Google Play[1], App Store (iOS)[2] and Microsoft Store[3], not only offer users comprehensive information and convenient download of mobile applications, but also allow users to comment on mobile applications they have downloaded. As a free source of feedback, these user reviews may provide application developers with a wealth of information related to application maintenance and updates, such as bug discovery, enhancements of existing functionalities and even ideas for new features. Previous studies [Pagano and Maalej, 2013; Chen et al., 2014] have showed that nearly one third of app reviews contain useful information including existing shortcomings and feature requests that can directly help developers improve their applications. Consequently, application developers can benefit from user reviews by exploiting useful information for arranging maintenance activities to keep users satisfied with their applications.

However, due to the large quantities and various qualities of the user reviews, it is not realistic for application developers to manually read each user review to gather useful feedback. Thus, different automated approaches have been proposed in literature to classify and extract information contained in user reviews. Nevertheless, only few recent studies [Panichella et al., 2015; Panichella et al., 2016; Sorbo et al., 2016] concentrate on analyzing mobile application user reviews for the purpose of maintenance. More researches and works still need to be done in this area to develop a system that can automatically analyze user reviews for mobile application maintenance, which will benefit developers in planning further software maintenance tasks to meet up-to-date market requirements.

1.2. Research aims

The main research aim of this study is to implement a mobile application user review analysis system for the maintenance purpose. Building such a system requires an answer to the question “How do the analysis of a mobile app's user reviews contribute

1 https://play.google.com/store. Access date: 29.11.2017
2 https://www.apple.com/ios/app-store/. Access date: 29.11.2017
3 https://www.microsoft.com/en-us/store/. Access date: 29.11.2017

to its maintenance activities". To answer this question, three related research questions are proposed in the course of implementing the app review analysis system. They are as follows:

- What kind of app review information can contribute to the maintenance of mobile applications?
- What are the proper techniques to extract keyphrases from app reviews?
- How to analyze the users' sentiments expressed in their reviews towards the extracted keyphrases?

Some app reviews contain useful information for application maintenance while others are uninformative. The app review analysis system should be able to identify the informative user reviews for application maintenance. Thus, the first research question "What kind of app review information can contribute to the mobile application maintenance" is proposed. By solving this question, the system could filter out useless app review information and only process the informative app reviews that can contribute to application maintenance.

The app review analysis system should also have the ability to summarize app reviews for application developers to quickly figure out the key issues discussed in the reviews and further plan corresponding maintenance activities for the application. In this study, the techniques of keyphrase extraction and sentiment analysis are chosen for summarizing the reviews in the proposed system named ARAM (App Review Analyzer for Maintenance). The reasons and details of choosing these techniques will be explained in the thesis. Thus, the second research question is proposed to find the proper keyphrase extraction methods for summarizing app review contents. The third research question is related to analyze the users' sentiments towards the extracted keyphrases.

1.3. Thesis structures

This thesis is composed of nine chapters. The second chapter introduces the background knowledge of application maintenance, which includes the types of maintenance activities and user reviews as a source of requests for application maintenance. The third chapter analyzes the characteristics of user reviews in app stores, presents the taxonomy of app review sentences for the maintenance purpose proposed by Panichella et al. [2015], and maps the informative types of user review sentences namely "information giving", "information seeking", "feature request" and "problem discovery" [Panichella et al., 2015] with twelve types of maintenance activities [Chapin et al., 2001]. The fourth chapter focuses on the technique of keyphrase extraction, which exhibits the keyphrase extraction methods that are widely used in the literature and illustrates the TextRank [Mihalcea and Tarau, 2004] method that is employed in ARAM. The fifth chapter introduces the background and terminology of sentiment

analysis. This chapter also introduces the related studies of sentiment analysis using app review data, and the sentiment analysis tool named Stanford CoreNLP Sentiment Analyzer [Socher et al., 2013]. The sixth chapter demonstrates the overall process and the implementation details of ARAM. The seventh chapter evaluates ARAM by testing its overall performance, data using rate, keyphrase extraction results, and sentiment analysis results. The eighth chapter validates the results produced by ARAM and discusses the limitations and possible improvements of ARAM, followed by the conclusion with the contributions of this study in the ninth chapter.

2. Application maintenance

Software maintenance is “the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment” [IEEE, 1998]. After the software is delivered, the maintenance phase starts until the software is retired. Software maintenance consumes a major share of the entire software life circle cost, which is between 40% and 70% [Hunt et al., 2008]. Software maintenance is a vital phase in software engineering. Not only is it the most expensive phase, but business opportunities are lost if a software cannot be maintained timely and reliably [Bennett and Rajlich, 2000].

This chapter contains three sections. The first section introduces the types of software maintenance based on the taxonomy of maintenance activities proposed by Chapin et al. [2001]. The second section shows that the users are a vital source of requests for change that lead to software maintenance. The third section demonstrates that user reviews in app stores, as a free source of user feedback, contain useful information for application maintenance.

2.1. Types of software maintenance

A widely cited study by Lientz and Swanson [1980] categorized maintenance activities into four types:

- **Corrective maintenance**: maintenance activities that fix discovered defects.
- **Adaptive maintenance**: maintenance activities that modify the software to adapt a changed or changing environment.
- **Perfective maintenance**: maintenance activities that improve performance or meeting new requirements.
- **Preventive maintenance**: maintenance activities that prevent problems before they occur.

However, this categorization was not detailed enough for classifying maintenance activities in an illuminating way for practitioners [Chapin et al., 2001]. Thus, Chapin et al. [2001] extended the prior work by refining software maintenance, and classified maintenance activities into twelve types of four clusters.

As presented in Figure 2.1, maintenance activities were classified into four clusters, and they were “support interface”, “documentation”, “software properties”, and “business rules”, and presented as four rectangles. The classification was based on three criterion decisions, and they were “was software changed”, “was source code changed”, and “was function changed”) [Chapin et al., 2001]. The maintenance activities were further categorized into different maintenance types in each cluster.

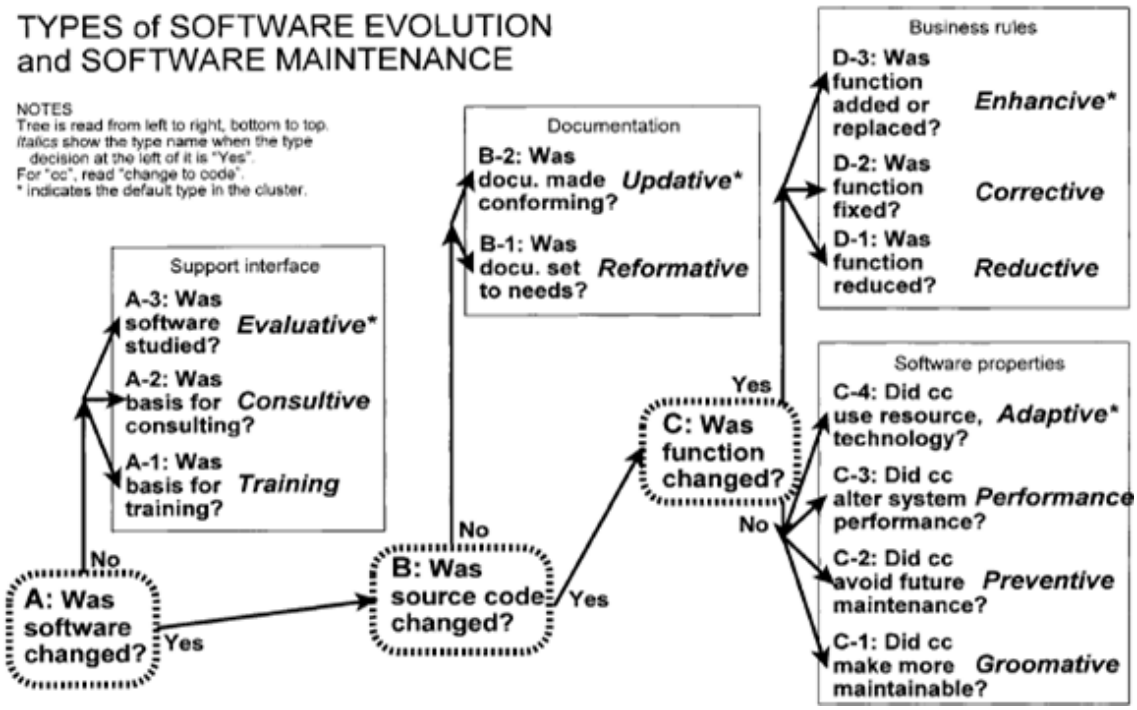


Figure 2.1. Decision tree for classifying maintenance activities into twelve types of four clusters [Chapin et al., 2001].

If the answer to criterion decision A "Was the software changed by the maintenance activity" was "no", which meant the maintenance activity did not change the software, then this activity belonged to the "support interface" cluster. In this cluster, activities were further divided into three different types namely "evaluative", "consultive" and "training". If the activity involved evaluating the software such as auditing and testing, it would be of the "evaluative" type. "Consultive" activities used the software as a basis for consultation. It evolved such activities as answering questions for customers about the software, conferring with customers or managers about the possible changes to the software, and informing stakeholders the estimated cost and time of the maintenance work. If the activity used the software for stakeholder training such as conducting classes for customer personnel, it belonged to the "training" type.

If the maintenance activity resulted in software change, it came to criterion decision B "Was the source code changed by the activity". If the activity did not change the source code, it belonged to the "documentation" cluster, which consisted of two types namely "reformative" and "updative". "Reformative" activities changed the non-code documentation for the stakeholders. It involved activities such as updating a user manual and preparing instruction materials for users. "Updative" activities changed the non-code documentation for describing the implemented system. It contained activities such as updating UML models and test plans.

If the maintenance activity changed the source code, criterion decision C “Did the activity change the user-experienced functionality” was then considered. If the activity did not change the system functionality sensible by the users, it belonged to the “software properties” cluster containing four different types namely “groomative”, “preventive”, “performance” and “adaptive”. The “groomative” type involved source code grooming activities, which could improve the system maintainability or security. It included activities such as updating source code annotation lines, changing personal access authorizations, and altering code readability. “Preventive” activities made efforts to avoid or reduce future maintenance activities such as changing the source code to make the system more extensible for future changes. The “performance” type involved activities that improve software performance such as changing the source code to speed system execution, reduce memory space, and improve system reliability and robustness. “Adaptive” activities changed the technology or resources used by the system. They were activities that updated the system to work on a new platform, in a new operation system or based on a new database.

If the maintenance activity changed the user-experienced functionality, it belonged to the “business rules” cluster, which included three different types namely “reductive”, “corrective” and “enhancive”. “Reductive” activities reduced or restricted the user-experienced functionality such as removing a launched function. “Corrective” activities fix the bugs concerning the system functionality. These activities remove defects caused by coding errors or design shortfalls. The “enhancive” type involved activities that replaced, added to, or extended the user-experienced functionality. It included activities that were significant for software evolution such as adding a new function or changing existing functions to enlarge their scope.

Mobile applications are the kind of software designed to run on mobile devices, and there is no study suggesting mobile application maintenance activities differed significantly from the maintenance activities of other kinds of software. Thus, the taxonomy of maintenance activities proposed by Chapin et al. [2001] can also provide a generic guideline on studying the possible activities in mobile application maintenance, which is supposed to help app developers understand and organize the maintenance activities in a systematic way.

2.2. Sources of change requests that lead to software maintenance

Software maintenance is carried out to meet the requests for software change. A request for change often originates from the users of the software in a form of a bug report or a request for new functionality [Bennett and Rajlich, 2000]. It can also come from the stakeholders inside the company to support the company’s business processes.

Change requests from users are important for software maintenance. Studies by Sundararajan et al. [2017] and Heeager and Rose [2015] present the situations of the

software maintenance caused by the change request from users. Sundararajan et al. [2017] describes that the change requests start with registering the users' requests received at the "help desk". If a request is not resolved at the front desk level, it is assigned to appropriate application support team for action. If software maintenance is necessary, then the corresponding maintenance activities are planned. The case study by Heeager and Rose [2015] shows that the maintenance operation at Aveva, a British multinational software company specializing in technology consulting services for the plant, power and marine industries, is for many years based on a conventional task management system, with maintainers servicing their own customers. By this way, maintenance activities are carried out mostly according to the change requests from the customers at Aveva. Both studies show that the software maintenance activities are carried out due to the change requests from users.

Change requests can also generate from the stakeholders inside the company (e.g., project managers, designers, QA engineers) to support the company's business processes. For example, if a software is implemented on the Windows operation system and proves to be successful in the market place, then the company might decide to update the software to also work on the Mac operating system to make a profit from attracting more users. Another example is that a competing company realises a new function in their software that rapidly becomes popular to attract many users, other companies should also act quickly to update their softwares with new features to prevent their customers from being taken away by the competing company. Additionally, the designers of the software can design new features after the app is releases, and they can be the source of requests for enhancive maintenance. The QA engineers might discover a problem after the software is released, which involves corrective maintenance. Although these change requests from the stakeholders inside the company might not be explicitly proposed by the users, they are still proposed in order to change the software to meet users' potential needs and further create profits for the company. Just as mentioned by Bennett and Rajlich [2000], "the software is being evolved because it is successful in the marketplace; revenue streams are buoyant, user demand is strong, the development atmosphere is vibrant and positive, and the organization is supportive. Return on investment is excellent". Companies would not be willing to spend money in maintaining their softwares if they cannot get return on investment, and user demand is of prime importance to consider when companies maintain their softwares in order to gain benefits.

Change requests and other feedback from users are also important for mobile application maintenance. Inukollu et al. [2014] described the mobile application maintenance as "fixing the issues that were fronted by the mobile users and also involves developing/releasing new features". They also suggested that "developers should integrate user feedback into subsequent versions of the app to remove non-

obvious blocks to sustained usage, so developers need to treat user questions and comments like high-valuable unpaid consulting opinions" to avoid making bad apps [Inukollu et al., 2014]. Thus, user feedback should be valued, understood and seriously treated by developers in application maintenance. Although there are no case studies found concerning how mobile application companies have carried out maintenance activities in practice, the feedback from users has to be an important source of change requests for application maintenance.

2.3. User reviews as a guidance for application maintenance

In classic software engineering, companies spend lots of time and money in gathering user feedback by techniques such as questionnaires, interviews, or focus groups [Iacob et al., 2013]. For the mobile application maintenance, app stores change this by allowing users to submit their feedback by writing reviews. Users can share their experiences, ideas and suggestions towards a mobile application by submitting a review in the app store. These user reviews as a free source of change requests from users can contain valuable information for application maintenance.

#	topic	description	freq.
t1	praise	expresses appreciation	75.36%
t2	helpfulness	scenario the app has proven helpful for	22.45%
t3	feature information	concrete feature or user interface	14.45%
t4	shortcoming	concrete aspect, user is not happy with	13.27%
t5	bug report	bug report or crash report	10.00%
t6	feature request	asks for missing feature	6.91%
t7	other app	reference to other app, e.g. for comparison	3.91%
t8	recommendation	suggests acquisition	3.82%
t9	noise	meaningless information	3.27%
t10	dissuasion	advises against purchase	3.27%
t11	content request	asks for missing content	2.91%
t12	promise	trades a better rating for a specific improvement	2.00%
t13	question	asks how to use specific feature	1.27%
t14	improvement request	requests improvement (e.g. app is slow)	1.18%
t15	dispraise	opposite of praise	1.18%
t16	other feedback	references or answers other feedback	1.09%
t17	howto	explains other users how to use app	0.91%

Figure 2.2. Topics in app user reviews [Pagano and Maalej, 2013].

A previous empirical study by Pagano and Maalej [2013] manually analyzed 528 app reviews (12 reviews from each of 44 apps in Google Play) to investigate the topics discussed in the reviews. Figure 2.2 showed the analysis results of topics covered in these reviews. Although a large proportion of user reviews contained the topic of "praise", which might be of less importance for application maintenance, there were still about one third of user reviews covering the topics of "feature information",

"shortcoming", "bug report", "feature request", "content request" or "improvement request", which might contain useful information for maintenance. Another study by Chen et al. [2014] used a semi-supervised machine learning algorithm based on the combination of Expectation-Maximization and Naive Bayes [Nigam et al., 2000] to train a model for filtering out non-informative reviews, and found that nearly 35.1% of app reviews were informative for developers to improve their apps. Thus, user reviews in app stores form a valuable source for developers to identify change requests and to plan application maintenance activities.

However, user reviews in app store are less structured and in copious amounts [Iacob et al., 2013]. Famous apps such as the social app Facebook received around 4275 reviews in just one single day [Pagano and Maalej, 2013]. Also from the data in App Annie[4], Facebook received average 3136 English reviews each day in Google Play during September, 2017. It is not realistic for application developers to manually read all user reviews to gather useful feedback. Thus, a user review analysis system is needed to crawl, filter, and summarize user reviews in the app store for application maintenance purposes. Build such a system needs the background knowledge of the characteristics of the app review data and other existing studies of the app review data, which are discussed in the next chapter.

4 https://www.appannie.com/apps/google-play/app/com.facebook.katana/reviews/. Access date: 29.11.2017

3. User reviews in app store

This chapter analyzes the characteristics of the app review data and other existing studies of the app review data. It contains six sections. The first section introduces the user reviews in different app stores. The second section analyzes the characteristics of app review contents. The third section exhibits the existing studies and system of analyzing app reviews. The fourth section focuses on the studies of analyzing app reviews for maintenance purpose, and explains the taxonomy proposed by Panichella et al. [2015], which classifies app review sentences into five categories related to application maintenance. The fifth section carries out a systematic mapping between the four informative types of user review sentences [Panichella et al., 2015] with twelve types of maintenance activities [Chapin et al., 2001]. The sixth section introduces the high-level overall architecture of ARAM, which leads to the topics of keyphrase extraction and sentiment analysis employed in ARAM.

3.1. User reviews in different app stores

App stores, which are also called app marketplaces, are digital distribution platforms for mobile applications. All major mobile operating systems, such as Android, iOS and Windows, have their corresponding official app stores that are Google Play, App Store (iOS) and Microsoft Store. Users not only can search and download mobile applications, but are also able to share their thoughts by writing comments for the mobile applications they have downloaded in these app stores.

Figure 3.1. User reviews in Google Play (a), App Store (b) and Microsoft Store (c).

As shown in Figure 3.1, a user review in all three app stores generally comprises the review contents and other meta-data such as the review rate (from one to five stars), the reviewer's username, the review title and the review date. Two exceptions are that the review title is currently not required in Google Play when writing a review, and reviews in Microsoft Store do not show their review dates. There is no study suggesting that user reviews in different app stores vary largely in contents or other respects.

3.2. Characteristics of app review contents

App review contents are quite different from other documents or long length texts used in the text mining field including journal articles, news articles and blogs. App review contents are short, informal, version specified, and of various quality.

App review contents are generally short in length since a large portion of them are submitted from mobile devices on which typing is not easy [Fu et al., 2013]. Vasa et al. [2012] analyzed a large dataset of user reviews (8.7 million reviews from 17,330 apps) from App Store (iOS). They found that user review length is highly skewed with an average of 117 characters, and median at 69 characters. Nearly 75% of the reviews are short enough to fit within a tweet (140 characters). Another similar research by Pagano and Maalej [2013] found that the review data from Google Play had a slightly reduced result concerning review length (Average of 106 characters, median at 61 characters and 76.7% of reviews less than 140 characters). Overall app review contents are much shorter than journal articles, news articles and blogs.

App review contents are also very informal. They tend to have many misspellings, slang terms, abbreviations and incomplete sentences. Hoon et al. [2012] discovered that 17% of the text expressions occurring less than 3 times in user reviews suggests either sentences without spaces or intentional and unintentional spelling mistakes (e.g., lovin, coul, amazzzzzzzin). Observably, emoticons and abbreviations (e.g., lmao, wtf, cuz) are also widely observed in app reviews. These types of informal expressions will definitely pose a tremendous challenge for computer based text analysis.

One app review is always specific to a particular version of the mobile application. A mobile application might have many different versions, and users always commented on the version they were using at the time of submitting the reviews. Therefore, app reviews vary over time due to the evolvement of the app, and generally only recent app reviews for the current version of the app can serve as the useful analysis sources for planning maintenance activities.

The quality of app review contents varies from help advices and innovation ideas to insulting offenses [Pagano and Maalej, 2013]. As discussed in Section 2.3, previous studies [Pagano and Maalej, 2013; Chen et al., 2014] suggested that about one third of user reviews were informative for developers. There are still a large portion of user reviews that do not provide useful information. Users wrote these uninformative

reviews mostly for expressing their sentiments without explaining the reasons such as "Nice app", "I love this app", and "Fire the idiot who designed this stupid app". Thus, an app review analysis system should have the ability to filter out these uninformative reviews.

3.3. Studies of app review data

Several studies [Iacob et al., 2013; Guzman and Maalej, 2014; Chen et al., 2014; Maalej and Nabil, 2015; Gu and Kim, 2015; Panichella et al., 2015; Panichella et al., 2016; Sorbo et al., 2016] have proposed different approaches to analyze app review data.

Iacob et al. [2013] extracted bug information in app reviews. They identified three categories of bugs (major, medium and minor bugs) based on the extent of the malfunctioning caused by the bugs, and further identified a set of linguistic rules associated with each type of bugs. They manually associated each sentence in the training sample with a keyword, identified the contexts in which these keywords were used, and finally conclude the linguistic rules.

Guzman and Maalej [2014] proposed an automated approach to identifying features and user sentiments towards the features in app reviews. They firstly extracted features based on the co-occurrence relations of the words and the term frequencies of the co-occurred words. Then sentiments related to the extracted features were calculated using the SentiStrength [Thelwall et al., 2012]. Finally, they used the topic modelling algorithm namely Latent Dirichlet Allocation (LDA) [Blei et al., 2003] to group features into topics and produce a high-level summary as a final result.

Chen et al. [2014] proposed AR-Miner, a computational framework for app review mining, and it automatically extracts, groups and prioritizes informative reviews. AR-Miner [Chen et al., 2014] firstly extracts informative user reviews using a semi-supervised machine learning algorithm based on the combination of Expectation-Maximization and Naive Bayes [Nigam et al., 2000]. Then two algorithms in topic modelling, i.e., Aspect and Sentiment Unification Model (ASUM) [Jo and Oh, 2011] and LDA [Blei et al., 2003] were compared to group the reviews. Finally, the groups and review instances in each group are ranked by their importance and presented in the report [Chen et al., 2014].

Maalej and Nabil [2015] presented a classifier for analyzing app reviews according to the type of information they include. They classified app reviews into four types namely "bug reports", "feature requests", "user experience" and "ratings", and used the supervised machine learning algorithms to train a classifier based on different attributes including text feature, review metadata (rate and length), and sentiment score.

Gu and Kim [2015] demonstrated SUR-Miner for automatic app review summarization. SUR-Miner [Gu and Kim, 2015] classified reviews into five categories namely "aspect evaluation", "praises", "feature requests", "bug reports" and "others"

based on lexical features and structural features. Then it extracted aspect-opinion pairs, associated a sentiment score to each pair, and finally presented the results of popular aspects and their sentiment trends by two diagrams namely the aspect heat map and the aspect trend map [Gu and Kim, 2015].

Only three studies [Panichella et al., 2015; Panichella et al., 2016; Sorbo et al., 2016] were found in analyzing app reviews for maintenance purpose. These three studies would be introduced in the next section.

3.4. Taxonomy of user review sentences for maintenance purpose

Panichella et al. [2015] proposed a tailor-made taxonomy of app review categories related to application maintenance. They firstly investigated the types of messages exchanged by developers in development mailing lists in Qt Project[5] and Ubuntu[6], and came up with six different categories of developers' communication purposes concerning application maintenance namely "information giving", "information seeking", "feature request", "problem discovery", "opinion asking" and "solution proposal" [Panichella et al., 2015]. Then they performed a systematic mapping between the six categories with 17 topics in app reviews proposed by Pagano and Maalej [2013] (shown in Figure 2.2), and found "opinion asking" and "solution proposal" were not covered by the topics in app reviews. Thus they finally classified user reviews into five categories at the sentence-level granularity:

- **Information giving**: sentences that inform users or developers about an aspect related to the app.
- **Information seeking**: sentences that request information or help from other users or developers.
- **Feature request**: sentences that express ideas, advices or needs for improving or enhancing the application or its functionalities.
- **Problem discovery**: sentences that describe bugs or unexpected behaviors of the application.
- **Other**: sentences that do not belong to any one of the four previous categories and are considered uninformative for maintenance purpose.

For instance, following is an app review of a mobile game (a number is marked before each sentence for easy reference):

"(1) Awesome game! (2) The sound effects for the battle are inspiring. (3) However, it feels terrible when the attack button sometimes does not work. (4) Please release the new weapon in next version. (5) By the way, does anybody know how to unlock the new campaign?"

5 https://www1.qt.io/developers/. Access date: 29.11.2017
6 https://www.ubuntu.com/. Access date: 29.11.2017

Sentence (1) is considered as the “other” type. Sentence (2) belongs to “information giving” since it talks about the “sound effect” aspect of the app. Sentence (3) is of the “problem discovery” type because it describes the bug concerning the attack button. Sentence (4) is seen as “feature request” that expresses needs for the new weapon. Sentence (5) is classified as “information seeking” for requesting information of the new campaign.

Panichella et al. [2016] further presented the ARdoc classifier, a tool that classified user review sentences into these five categories [Panichella et al., 2015] based on the combination of techniques of natural language parsing, text analysis and sentiment analysis. ARdoc could correctly classify user review sentences with high precision (ranging from 84% to 89%), recall (ranging from ranging from 84% to 89%), and F-measure (ranging from ranging from 84% to 89%) [Panichella et al., 2016]. The ARdoc Java API[7] is available for developers to integrate this tool in their own projects.

Sorbo et al. [2016] proposed SURF (Summarizer of User Reviews Feedback), which was a system to analyze app reviews for recommending software changes. Sorbo et al. [2016] proposed a set of 12 topic clusters observed in app reviews (“App”, “GUI”, “Contents”, “Pricing”, “Feature or Functionality”, “Improvement”, “Updates”, “Resources”, “Security”, “Download”, “Model” and “Company”), and implemented a classifier to assign a review sentence to one or more topics. They also integrated ARdoc [Panichella et al., 2016] into SURF to further classify review sentences in each topic cluster. Thus, SURF [Sorbo et al., 2016] performed a two-level clustering for each review sentence and then the review sentences in each topic cluster were ranked and presented in a summary report.

As mentioned in Section 3.2, the quality of app reviews varies, and ARAM should have the ability to filter out the reviews that are uninformative for maintenance purpose. The first research question of this study is also related to find the kinds of informative user reviews that can contribute to application maintenance. The taxonomy presented by Panichella et al. [2015] actually helps to solve this question. The “information giving”, “information seeking”, “feature request” and “problem discovery” review sentences are supposed to be informative for application maintenance while the review sentences of the “other” type are uninformative.

However, the taxonomy presented by Panichella et al. [2015] classifies user review sentences based on the users’ intentions. In other words, the users write the review sentences to give information about an aspect of the app (“information giving”), seek for information concerning the app (“information seeking”), request their needs for improving the app (“feature request”), or describe bugs that they have found (“problem discovery”). Thus the taxonomy is considered from the user’s perspective. As explained

7 http://www.ifi.uzh.ch/en/seal/people/panichella/tools/ARdoc.html. Access date: 29.11.2017

in Section 2.1, the taxonomy of maintenance activities is from the developer's perspective to categorize maintenance activities. To verify whether the four informative types of app review sentences (e.g., "information giving", "information seeking", "feature request" and "problem discovery") based on the taxonomy proposed by Panichella et al. [2015] contain useful information that can contribute to application maintenance, this study maps the four informative types of user review sentences [Panichella et al., 2015] with twelve types of maintenance activities [Chapin et al., 2001] by analyzing the typical types of maintenance activities that are possibly carried out for the four informative types of user review sentences.

3.5. Four categories of user review sentences leading to possible types of maintenance activities

To analyze what typical types of maintenance activities that the four informative categories of user review sentences (e.g., "information giving", "information seeking", "feature request" and "problem discovery") [Panichella et al., 2015] can lead to, the author firstly used the ARdoc classifier [Panichella et al., 2016] to classify the sentences in 549 reviews of the social app Instagram and 963 reviews of the mobile game Pokémon GO published between 20th and 23rd, May 2017. Then the classified sentences were manually analyzed by the author for the possible types of maintenance activities that could be carried out relying on the information in each category of the review sentences. The mapping results in Table 3.1, Table 3.2, Table 3.3 and Table 3.4 included the typical examples of the different possible maintenance activities and types that could be inspired by the four categories of informative review sentences.

Information Giving		
Example	Possible Maintenance Activity	Type
"The algorithm for the timeline is really annoying."	Gather more feedback about the timeline	Evaluative
"It's a really good app, but the new version is not working on my Samsung tablet."	Carry out diagnostic testing concerning this issue.	Evaluative
"People are using fake GPS add-ons and have ruined this game."	Add functions to forbid fake GPS or other plug-ins.	Enhancive
"It stops working sometimes, and it really does need quite a bit of battery, so you can't have it up for long."	Reduce battery consumption of the app.	Performance

“Once the Pokemon is there, it can stay there forever (should not it get tired and needing rest and thus be removed from the gym after couple of hours?).”	Change the corresponding functionality.	Enhancive
"I loved this game before it kept saying no GPS signal even though I have turned it on."	Fix the possible bug	Corrective

Table 3.1. Typical maintenance types for “information giving” review sentences.

For the “information giving” review sentences, they discussed different aspects of an app and could lead to different kinds of maintenance activities as shown in Table 3.1. Evaluative maintenance activities were the most relevant type of activities that are supposed to be considered for the “information giving” review sentences. Developers could find out the possible requirements and bugs in the “information giving” review sentences where the user experiences of different aspects of the app were described, and further gather more information concerning the issues or plan diagnostic tests, which belonged to evaluative maintenance activities. Moreover, it was also noticed that some “information giving” review sentences contained important messages for the app but were not related to maintenance activities. For instance, an “information giving” review sentence of Pokémon GO said “So far no responses from their Twitter support page and no other way to contact them.”, in which the reported issue was not concerned with application maintenance but the customer service. Thus, by analyzing the “information giving” review sentences, developers can discover different issues related to the app that are not only limited in the application maintenance field. In summary, the “information giving” review sentences generally covered rich information concerning different aspects of the app and they were most likely to lead to evaluative maintenance activities.

Information Seeking		
Example	Possible Maintenance Activity	Type
"Does the face filters only available for OS higher than android 5.1?"	Answer questions about the software.	Consultive
"I’m still confusing how to do stuff on the app."	Update the user manual, non-code documentation changed.	Reformative
"Why my Instagram is so slow?"	Change the code to improve execution speed with functionality unchanged.	Performance
“How to find the gym and pokestops?”	Add instructions for new users	Training

“Why is the GPS such a killer on my battery usage?”	Reduce battery consumption of the app.	Performance

Table 3.2. Typical maintenance types for “information seeking” review sentences.

As shown in Table 3.2, the “information seeking” review sentences generally contained users’ questions for information or help, which mostly required consultive maintenance activities to answer these questions. If the problem asked in the “information seeking” review sentences were common among users, developers could also consider updating the user manual (reformative maintenance) or creating some tutorials for users (training maintenance).

Feature Request		
Example	Possible Maintenance Activity	Type
"You also need to allow the option of chronological order as I STILL don't see the point of this algorithmic feed."	Algorithm replacement, add or change functionality.	Enhancive
"I wish you could add a time limit share function for uploaded photos and videos!"	Add component or algorithm implementing functionality.	Enhancive
"I wish we could battle other players not just at gyms and I wish we could use our Pokemon to battle wild ones."	Update battle component.	Enhancive
"I love this game, but I wish it was better at keeping up with your location."	Increase the accuracy of location.	Performance

Table 3.3. Typical maintenance types for “feature request” review sentences.

The “feature request” review sentences generally expressed users’ ideas and suggestions for enhancing the app, which could inspire developers to add new features. As presented in Table 3.3, enhancive maintenance was the main type of maintenance activities that developers would plan after analyzing the “feature request” review sentences.

The “problem discovery” category of review sentences might be the most vital category that developers should pay attention to. These review sentences described bugs and unexpected behaviors of the app encountered by users. Developers should firstly

check the common described bugs in these review sentences, which involved evaluative maintenance activities. Then if the bugs were indeed existed in the system, further corrective maintenance activities or performance maintenance activities were needed to be planned and carried out as shown in Table 3.4.

Problem Discovery		
Example	Possible Maintenance Activity	Type
"It crashes way too much and error reports are displayed after every second!"	Fix the coding error that causes the problem.	Corrective
“It's lagging to watch videos hope it gets fixed.”	Change the execution speed.	Performance
“Fix the app that takes too much space.”	Change the external storage utilization.	Performance
“I also have a problem with the amount of battery it uses, it just kills it, even in battery saver mode."	Reduce battery consumption of the app.	Performance

Table 3.4. Typical maintenance types for “problem discovery” review sentences.

Five types of maintenance activities were not covered in any four categories of informative user review sentences when the total 1512 app reviews were manually analyzed, which included updative activities, reductive activities, adaptive activities, preventive activities, and groomative activities. Although no example was found in the sample reviews, the reductive maintenance activities and the adaptive maintenance activities could still be inspired by the informative user reviews. For example, if a new function released by an app was considered to be racism, users were likely to discuss it in their reviews and thus might lead to reductive maintenance activities by developers to remove corresponding components of this function. “Feature request” or “problem discovery” user review sentences might contain information that request an update to make the app run on the newest Android system or describe the bug that the app could not run on the newest system, which might lead to adaptive maintenance activities by the developers to make the app work on a changed system. However, updative activities, preventive activities, and groomative activities were generally carried out by developers to improve the maintainability and rarely related to and sensible by users [Chapin et al., 2001]. Thus, user reviews generally did not contain information for these three types of maintenance activities.

In conclusion, classified informative user review sentences (e.g., “information giving”, “information seeking”, “feature request” and “problem discovery”) had valuable information for guiding developers to plan most types of maintenance activities. Different categories of user review sentences could lead to different typical types of maintenance activities. With the mapping results discussed above, developers

could have a good understanding and expectation of different kinds of maintenance activities that were related to each category of informative user review sentences. Moreover, the classified results of the 1512 app reviews by ARdoc [Panichella et al., 2016] used in the manual mapping process were mostly correct. Although there were some wrong cases discovered such as classifying some “other” type review sentences as the “information giving” type (e.g., “I just loved the game” and "I am a very rare user so cannot judge"), ARdoc was broadly in line with its high precision (ranging from 84% to 89%) as tested in the study of Panichella et al. [2016].

3.6. An overview of ARAM

Figure 3.2 presented the high-level overall architecture of ARAM. ARAM firstly collected the user reviews of an app. English user reviews in Google Play were used in the implementation and test of ARAM. However, ARAM was also able to process user reviews from other app stores if the reviews were stored in the predefined JSON format described in Section 6.2. Review contents were the main subject that ARAM would process. Review rates were used in the test of sentiment analysis results (in Section 7.4). Other meta-data including reviewers’ usernames and review dates were stored for the presentation purpose of the user reviews in the final report produced by ARAM.

Then, ARAM employed the ARdoc [Panichella et al., 2016] to classify user review sentences into five different categories namely “Information Giving”, “Information Seeking”, “Feature Request”, “Problem Discovery” and “Other”. If all sentences of an app review were classified as “Other”, this review was considered uninformative and was filtered out by ARAM. Other app reviews with at least one sentence classified as “Information Giving”, “Information Seeking”, “Feature Request” or “Problem Discovery” were considered to contain useful information for app maintenance. These reviews were further processed by ARAM in order to produce an instructive report for app developers to plan maintenance activities.

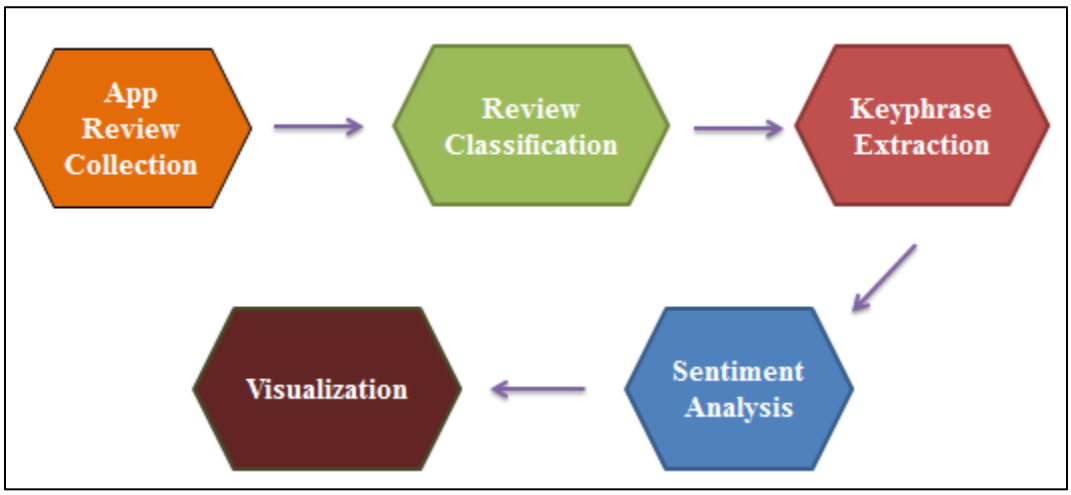


Figure 3.2. An overview of ARAM.

Techniques of keyphrase extraction and sentiment analysis were employed in ARAM to process the informative reviews. Keyphrases extraction could help app developers to quickly figure out the key features and aspects of the app that were discussed in each category of user reviews. For example, if a new version of a map app had a bug that its navigation function could not work, many users might start mentioning the problematic navigation function in their reviews, and these reviews would contain sentences classified as “Problem Discovery”. Thus, the keyphrases related to the navigation function such as “navigation” and “navigation function” would be extracted and listed in the category of “Problem Discovery” by ARAM. Therefore, developers could notice that something was wrong with the navigation function and it might need corrective maintenance. Moreover, if many users required a new voice navigation function for the map app, it was supposed that the keyphrase “voice navigation” would appear in the “Feature Request” category of the report produced by ARAM. Hence developers could notice the users’ requests and might carry out enhancive maintenance to add the voice navigation function in the next version.

Sentiment analysis was utilized to analyze users’ sentiments towards the extracted keyphrases, which mostly represented for the features and aspects of the app. Therefore, developers could figure out which features or aspects of the app were considered worst by users and respond quickly to maintain the corresponding parts. Sentiment analysis could also help developers to examine whether a new feature was loved or hated by the users. For instance, if a map app released a new voice navigation function, many users might start talking about their user experience of this new feature. Therefore, the keyphrase “voice navigation” would appear in the “Information Giving” category, and users’ sentiments towards the keyphrase could reflect whether they liked or hated this new feature.

Therefore, by employing keyphrase extraction and sentiment analysis, ARAM could produce a concise and informative summary of useful app reviews to guide developers for maintenance activities. The techniques of keyphrase extraction and sentiment analysis were introduced separately in the following two chapters.

4. Keyphrase extraction

Classified app reviews are considered to contain useful information for mobile application maintenance. They need to be summarized in an effective way for developers to quickly figure out what these reviews are mainly talking about. It appears natural to use the keyphrase to summarize the review contents.

When using search engines, people do not have to enter a complete sentence but only a couple of keyphrases to be able to find the related contents that they want. Scientific writing often includes a list of keywords to give readers an idea about the topics. In a textbook, keywords are also utilized as the titles of each chapter and section, which serve as good summaries and indexes of the corresponding contents. Hence appropriate keywords are able to summarize the text in a highly concise way, which can save people lots of time and energy to grasp the key points. Therefore, it would be of great benefit to developers if the app review analysis system could have the ability to automatically extract keywords as a succinct summary of the classified app reviews. This involves the topic of keyphrase extraction.

This chapter focuses on the technique of keyphrase extraction. It contains three sections. The first section introduces the basic concepts of keyphrase and keyphrase extraction. The second section describes the general process of keyphrase extraction, which includes the supervised methods and the unsupervised methods that are widely used in the literature. The third section illustrates the TextRank algorithm, which is a graph-based ranking method implemented in ARAM for the keyphrase extraction task.

4.1. Definition of keyphrase and keyphrase extraction

The keyphrases of a given document, which are also called keywords or key terms, are a small group of representative words or phrases that can capture the primary topics of a document [Turney, 2000]. There is a difference between the term “keyphrase” and “keyword” in the literature. “Keyword” always refers to a single word, while “keyphrase” emphasizes the ability to find the multi-word phrase. In this study, the term “keyphrase” is used to infer both single words and multi-word phrases. Furthermore, the ideal keyphrases of app review contents are the words and phrases that can represent the most relevant features or aspects of the app that are discussed in the reviews. Appropriate keyphrases can help readers to quickly understand the key points expressed in a large quantity of textual material. They can serve as a concise summary of the review contents.

Keyphrases can be assigned manually. For example, most academic theses require their authors to summarize a list of keyphrases added after the abstract. Some journal articles, news articles and blogs are also labeled with keyphrases by the editors. However, many documents and web contents do not have keyphrases assigned by their authors, and manually assigning keyphrases can be very time-consuming. Thus, the

need for automatic process of extracting the keyphrases, which is called (automatic) keyphrase extraction, has spawned many studies in the last two decades. Keyphrase extraction is a vital task in the study field of Natural Language Processing (NLP), Text Mining, and Information Retrieval, and can be used for text summarization, automatic indexing, query refinement, documents classification, etc [Siddiqi and Sharan, 2015].

From early studies of keyphrase extraction, it has been conventional to only consider noun phrases as keyphrases [Turney, 2000; Hulth, 2003; Mihalcea and Tarau, 2004]. This convention is based on the fact that when people manually assign keyphrases, the vast majority are also proved to be nouns or noun phrases with adjectives [Hulth, 2003]. This fact remains unchanged for now when inspecting the manually assigned keyphrases by the authors of academic theses, nearly all of these keyphrases are noun phrases. There are many approaches for keyphrase extraction, which are discussed in the following sections.

4.2. Keyphrase extraction methods

As presented in Figure 4.1, the keyphrase extraction task generally processes in two steps: (1) identifying a list of words and phrases from the document as the candidate keyphrases; (2) selecting correct keyphrases from the candidate keyphrases using supervised methods or unsupervised methods [Hasan and Ng, 2014].

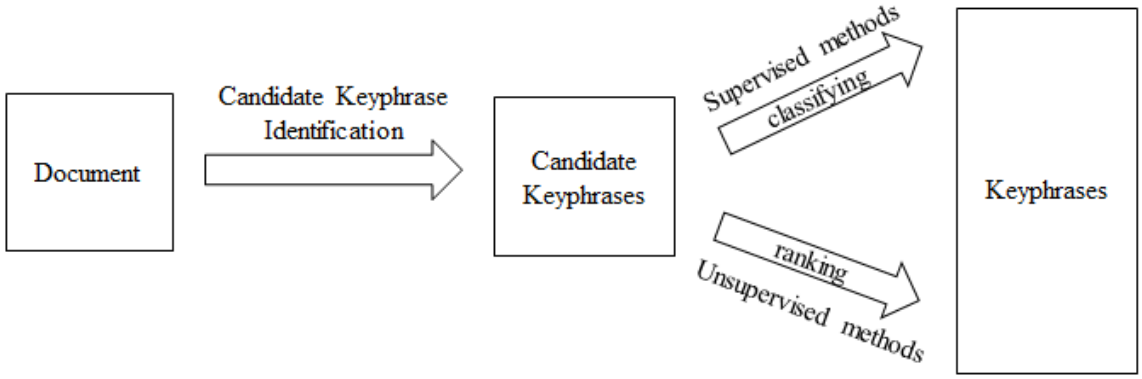


Figure 4.1. General process of keyphrase extraction.

4.2.1. Candidate keyphrase identification

Selecting a list of candidate keyphrases from a document generally involves three steps of Natural Language Processing:

The first step is to select words and phrases with certain part-of-speech (POS) tags (e.g., nouns, verbs, adjectives, etc). As noted before, keyphrases are always proved to be noun phrases [Turney, 2000; Hulth, 2003; Mihalcea and Tarau, 2004]. Therefore, adjectives and nouns, or noun phrases are selected in this step.

Secondly, the stop words are removed from the candidate keyphrases. The stop words are uninformative words that appear frequently in texts such as “the”, “you” and “is”. Candidate keyphrases should not contain any of these stop words.

Thirdly, the candidate keyphrases are lemmatized to avoid repetition. For example, the lemmatization forms of “image”, “images” and “imaging” are the same, and they should be counted as one candidate keyphrase using the lemmatization form “image”. Stemming is another method to convert a word to its common base form. The difference between stemming and lemmatization is that the stemming method always cut the end of the word to remove derivational affixes, while lemmatization method only remove the inflectional ending after understanding the POS of the word and return its base or dictionary form [Jivani, 2011]. For instance, the stemming form of “saw” is “s” or “saw” (some stemmers consider “aw” as a derivational affix while the others do not). The lemmatization form of “saw” will be “see” if it is used as a verb or “saw” if it is used as a noun. Thus, although lemmatization generally takes more time, it can produce more accurate results than truncated and ambiguous stems [Korenius et al., 2004].

After identifying the candidate keyphrases, supervised methods or unsupervised methods are used to determine whether a candidate keyphrase is correct or not.

4.2.2. Supervised methods

The supervised methods consider keyphrase extraction as a binary classification problem, where a model is trained on annotated data to determine whether a candidate keyphrase is a correct keyphrase or not [Hasan and Ng, 2014]. Different classifiers have been used in training the model such as Naive Bayes [Frank et al., 1999], Decision Trees [Turney, 2000; Ercan and Cicekli, 2007], and Neural Networks [Wang et al., 2006; Sarkar et al., 2010].

The disadvantage of the supervised methods is that they generally require a large amount of annotated data to train the model, which means that many documents with known keyphrases are needed. The trained models also show bias towards the domain on which they are trained [Hasan and Ng, 2010]. Currently no annotated dataset is available for the keyphrase extraction task of app reviews. There are only few app review studies using keyphrase extraction [Oh et al., 2013; Vu et al., 2015], and all of them employ unsupervised methods. Unsupervised methods remove the need for training data. Considering the huge amount of time and energy needed for annotating the train data, ARAM also used unsupervised methods rather than supervised methods.

4.2.3. Unsupervised methods

The unsupervised methods formulate the keyphrase extraction as a ranking problem. Candidate keyphrases with higher rank values are selected as keyphrases. They can be

divided into the statistics methods, the graph-based ranking methods and the topic-based clustering methods.

For the statistics methods, the most straightforward one is to rank the candidate keyphrases by their term frequencies. This is based on the fact that keyphrases generally appear more frequently than other phrases in the document. Many keyphrase extraction studies [Tonella et al., 2003; Zhang et al., 2007; Lee and Kim, 2008; Liu et al., 2009a] use term frequency-inverse document frequency (tf-idf) instead of term frequency to reduce the influence of the words that appear more frequently in the corpus of websites [Tonella et al., 2003; Zhang et al., 2007], news [Lee and Kim, 2008] and meeting transcripts [Liu et al., 2009a]. However, for analyzing app reviews, the words that appear in many reviews probably imply important issues concerning the app and should not be penalized. For example, in Feb 2015, Facebook Messenger released a new version with a bug that caused high CPU usage and thus drained the battery quickly, which leaded to the majority of reviews discussing the "battery draining" topic and its related keyphrases [Vu et al., 2015]. Using tf-idf in this case would largely reduce the rank values of these keyphrases such as "battery" and "energy" since they appeared in many reviews, which was definitely not a good choice because these keyphrases were the most vital ones that developers should pay attention to. Thus, term frequency is supposed to work better than tf-idf in the keyphrase extraction task of app reviews.

The graph-based ranking methods determine the importance of a candidate by its relatedness to other candidates. These methods assume that a candidate is important if it is related to (1) many other candidates and (2) candidates that are important [Hasan and Ng, 2014]. The relatedness between two candidates can be computed by the co-occurrence relation [Mihalcea and Tarau, 2004] or semantic relatedness [Wan et al., 2007]. A graph is generated with each node (vertice) corresponding to a candidate keyphrase, and edges between two related nodes. Then the rank value of each node is calculated recursively based on the rank values of its adjacent nodes connected by edges [Hasan and Ng, 2014]. TextRank [Mihalcea and Tarau, 2004] is one of the most famous graph-based ranking methods for keyphrase extraction. There are also some variants of TextRank such as ExpandRank [Wan and Xiao, 2008] and CiteTextRank [Gollapalli and Caragea, 2014].

For the topic-based clustering methods, the basic idea is to group candidate keyphrases into different clusters, where each cluster corresponds to a topic covered in the document. The candidate phrases that are close to the centroid of a cluster are selected as keyphrases. This method is called KeyCluster [Liu et al., 2009b], and it ensures that the result list of keyphrases can cover all topics of the document [Liu et al., 2009b]. Besides, there is another method called CommunityCluster [Grineva et al., 2009], and it is to rank clusters based on their density and informativeness. All candidate phrases in the top clusters are selected as keyphrases. This assumes that if a

candidate phrase is related to an important topic of the document, it should be extracted as a keyphrase [Grineva et al., 2009].

Considering the studies of app reviews, only two studies [Oh et al., 2013; Vu et al., 2015] are found using the technique of keyphrase extraction, and both of them extract keyphrases based on term frequency. Oh et al. [2013] aimed to find a set of issue keywords for building a model to determine whether a review was informative or not. They used term frequency to generate one set of keywords of the informative reviews named $Set_{Issue\_freq,}$ and another set of keywords appeared in the uninformative reviews named $Set_{Non\text{-}Issue\_freq}$. The final keywords were the words in the Set_{Issue_freq} but not in the $Set_{Non\text{-}Issue\_freq}$. In their study, only stemming was applied for preprocessing the review contents, and they did not have the process of identifying candidate keywords by using POS tagging, lemmatization and stop word removal. As a result, the generated keywords were very ambiguous (e.g., “clos”, “const”, “xper”, “otherw”, “confus”, “pls”, “7”, etc).

Vu et al. [2015] developed a keyword-based framework for review analysis. They chose the nouns and verbs as candidate keywords, and the candidate keywords were ranked based on their term frequency in negative reviews and positive reviews. The candidate keywords that appeared more often in negative reviews than in positive reviews had the higher rank value, which was based on the assumption that keywords frequently appearing in negative reviews could help identify the problems of an app. Furthermore, they did the work of keyword clustering and expanding after keyword extraction. However, their framework was not able to extract multi-word keyphrases. The ability to extract also phrases as results is important. For example, many reviews of the social app Instagram talked about its powerful “face filter” function. The result would be very confusing if the keyphrase “face filter” was separated as two keywords “face” and “filter”.

ARAM planned to use the statistics method term frequency and the graph-based ranking method TextRank for the keyphrase extraction task. The reason for choosing these two methods was that they were relatively easy to implement and had been proved to work well in practice. However, to the best of the author’s knowledge, the graph-based ranking method TextRank had not been used in analyzing short informal data such as app reviews in previous studies. Therefore, it remained to be seen which method could perform better in the keyphrase extraction task of app reviews.

Since the idea of term frequency is easy to understand, the tricky part of using term frequency is the preprocessing phase for candidate keyphrase identification and the detailed implementation is presented in section 6.4. The Textrank method is introduced in the next section.

4.3. TextRank method

TextRank [Mihalcea and Tarau, 2004] is a graph-based ranking method for keyphrase extraction and it originates from PageRank [Page et al., 1999], a famous algorithm used by Google Search to rank websites in their search engine results. The idea of TextRank could be easily understood with the background knowledge of PageRank. Therefore, PageRank is firstly introduced in the next subsection.

4.3.1. Introduction to PageRank

In order to rank web pages by their importance, PageRank works by considering the link structure of the web [Page et al., 1999]. A web page has a high rank if there are many other web pages linking to it, or some web pages that have high ranks linking to it. This is based on the assumption that an important web page is received many links from other web pages or a link from another important web page. Formally, the simplified algorithm defines the rank value of a web page *u* as follows [Page et al., 1999]:

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)} \qquad \text{(Formula 4.1)}$$

The rank value of the web page *u* ($PR(u)$) equals to the sum of the rank value of each web page *v* ($PR(v)$) contained in the set B_u (the set that contains all web pages linking to *u*) divided by its $L(v)$, which is the number of links from *v*.

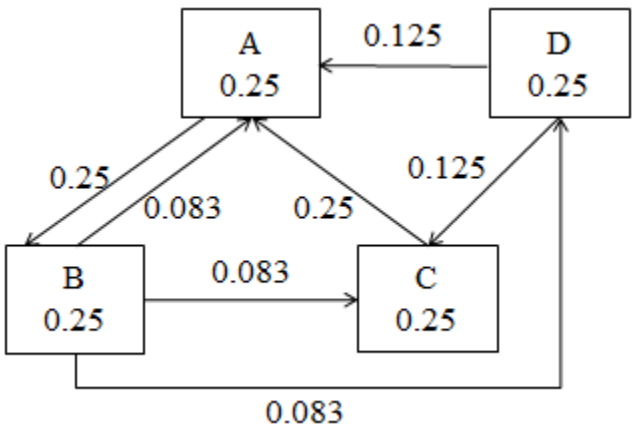


Figure 4.2. An example of PageRank calculation in the first iteration.

For example, as shown in Figure 4.2, there are four web pages (A, B, C, and D) in the whole web. The initial rank value for each web page is equal to one divided by the number of web pages, which is 0.25 in this example. For the web page A, all other web pages have a link to it. Since the web page B has three links to all other web pages, it will give out one third of its current rank value to A. Similarly, A will also receive the rank value of C and half of the rank value of D. Thus, the rank value of A is calculated by the following formula:

$$PR(A) = \frac{PR(B)}{3} + \frac{PR(C)}{1} + \frac{PR(D)}{2} \qquad \text{(Formula 4.2)}$$

Similarly,

$$PR(B) = \frac{PR(A)}{1}$$
$$PR(C) = \frac{PR(B)}{3} + \frac{PR(D)}{2}$$
$$PR(D) = \frac{PR(B)}{3} \qquad \text{(Formulae 4.3)}$$

PageRank works like a voting. In every iteration, each web page gives out all of its votes (current rank value) evenly to all the web pages that it is linking to, and receives the votes from other web pages that are linking to it. As shown in Table 4.1, the rank values of the web pages are calculated using the above formulae for several iterations. The rank values gradually become stable when the iterations go on. As a result, the web pages A and B rank higher than C and D.

Iteration	1	2	3	4	5	6	7	8	9
A	0.458	0.333	0.319	0.382	0.350	0.343	0.361	0.353	0.350
B	0.25	0.458	0.333	0.319	0.382	0.350	0.343	0.361	0.353
C	0.208	0.125	0.194	0.188	0.162	0.181	0.180	0.172	0.177
D	0.083	0.083	0.153	0.111	0.106	0.130	0.117	0.114	0.120

Table 4.1. Rank values of the four web pages during nine iterations.

Moreover, if a web page has no other pages linking to it, its rank value will always be 0 after the first iteration. To avoid this, the damping factor *d* is added to the PageRank, and changes Formula 4.1 into Formula 4.4:

$$PR(u) = (1 - d) + d \sum_{v \in B_u} \frac{PR(v)}{L(v)} \qquad \text{(Formula 4.4)}$$

The damping factor *d* is between 0 and 1. It originates from the idea that if a web surfer who starts on a random web page has a *d* likelihood of choosing a random link from the page he/she is currently visiting, and (1-*d*) likelihood of getting bored and requesting another random page in the whole web [Brin and Page, 2012]. It is a common practice to set the damping factor *d* to 0.85 [Brin and Page, 2012; Mihalcea and Tarau, 2004]. Therefore, if a web page has no other pages linking to it, it still has a (1- *d*) likelihood of being visited. Formula 4.4 is also the key formula used in TextRank.

4.3.2. From PageRank to TextRank

TextRank is an adaption of PageRank for ranking phrases instead of web pages. In PageRank, the graph is built using web pages as vertices and their link relations as edges. The problem is how to present the text into a graph in TextRank. For the keyphrase extraction task, the idea of TextRank is to select candidate keywords as vertices and using their co-occurrence relations to form the edges [Mihalcea and Tarau,

2004]. Selecting nouns and adjectives as candidate keywords to form the vertices could achieve the best performance [Mihalcea and Tarau, 2004]. Then two vertices are connected if their corresponding candidate keywords co-occur within a window of maximum *N* words (the distance between two words is less than *N*). In PageRank, the links among web pages have directions and thus form a directed graph such as Figure 4.2. In TextRank, Mihalcea and Tarau [2004] also considered building a directed graph by the word order in the text. A candidate keyword will point to the following (or previous) candidate keywords in the text that co-occur within a window of maximum *N* words. They also used the undirected graph, which was equal to the idea that two keywords will point to each other if they meet the co-occurrence constraint. Their experiments showed that undirected graphs outperformed directed graphs, and the best result was achieved when the co-occurrence window *N* was set to 2.

To take a concrete example, following is a piece of user review of a map application:

> *"It is an awesome app to help me reach destination. The navigation function always works well in finding short routes. However, the voice navigation was too chatty and once led me to a wrong destination."*

To form a graph, firstly the candidate keywords are found by identifying nouns and adjectives, removing the stop words, and lemmatizing. Then the list of candidate keywords that has no duplications ("awesome", "app", "destination", "navigation", "function", "short", "route", "voice", "chatty", "wrong") forms the vertices of the undirected graph. Next, if two keywords co-occur within a window of maximum 2 words, an edge is added between them. Finally, the generated graph of the review text is shown in Figure 4.3.

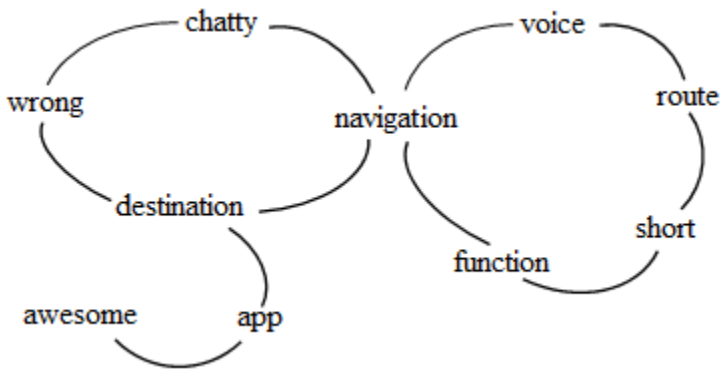


Figure 4.3. The undirected graph built by TextRank from a sample review text.

After the graph is built, the rank value for each vertex is set to 1, and then the same formula used in PageRank (Formula 4.4) is run on the graph for several iterations (usually 20–30 iterations) until it converges [Mihalcea and Tarau, 2004]. Then the top *T* vertices with highest rank values are selected as keywords for post-processing to form

the keyphrases. Mihalcea and Tarau [2004] set the T to a third of the total vertices number. Next, in the post-processing phase, these selected keywords are marked in the original text, and if there are adjacent keywords in the original text, these adjacent keywords are collapsed into a multi-word keyphrase. For instance, if “navigation”, “destination” and “voice” are calculated as the top T keywords in the last example, then the final keyphrases will be “destination” and “voice navigation”, since “voice” and “navigation” are once adjacent in the original text.

TextRank has been proved to work well in the keyphrase extraction task of journal papers’ abstracts with the F-measure score of 36.2% [Mihalcea and Tarau, 2004]. However, considering app reviews as a kind of short informal texts are very different from the abstracts, some necessary adjustments are made in the implementation of TextRank in ARAM, which are explained in the Section 6.4.

5. Sentiment analysis

Most of people tend to consider and be influenced by others' opinions before making decisions. Consumers like to seek out other existing users' sentiments towards a product before purchasing it. When people shop online, negative reviews received by a product can eliminate their thoughts of buying it; Movies that have a good reputation among people who have watched them will attract more new audience; When a user needs to download a mobile application for some purposes such as booking a hotel, chances are ten to one that he will choose from the hotel booking mobile applications that are rated high by other users instead of those with low scores. The rating scores or the amounts of positive and negative reviews are a direct reflection of users' sentiments.

Users' sentiments represented by their rating scores are vital to a product's future. Previous studies [Chevalier and Mayzlin, 2006; Flanagin et al., 2014] have demonstrated that how positively a product is rated by customers can influence the sales of the product. Chevalier and Mayzlin [2006] prove that an improvement in a book's review scores increase its sales. Flanagin et al. [2014] also argue that user ratings reflect perceived product quality and in return influence customers' purchase intention. For mobile applications, Harman et al. [2012] identify a strong positive correlation between the average customer rating score and the rank of app downloads by analyzing 32,108 apps in the Blackberry app store [8]. Therefore, mobile application development companies should care about users' sentiments expressed in the app store reviews towards their products and services, since the ratings and reviews on app stores can motivate or discourage users to download an app [Khalid et al., 2015]. Hence application developers should find out and respond quickly the worst aspects of the application from the users' perspective, and thus satisfy users' needs to harvest higher rating scores in return. However, it is not realistic for developers to manually go through large quantities of reviews to summarize users' sentiments towards different aspects of the application. Therefore, techniques to automatically identify users' sentiments presented in their reviews towards different aspects of the apps are needed.

This chapter contains five sections. The first section introduces the basic concepts related to sentiment analysis. The second section illustrates the document-based, sentence-based and aspect-based sentiment analysis. The third section gives an overall presentation of the sentiment classification methods. The fourth section exhibits the related studies of sentiment analysis using app review data. The fifth section introduces the Stanford CoreNLP Sentiment Analyzer [Socher et al., 2013], a powerful sentiment analysis tool that is utilized in building ARAM.

8 https://appworld.blackberry.com/webstore/. Access date: 29.11.2017

5.1. Terms of sentiment analysis

Sentiment analysis, also known as opinion mining, is the research field that analyzes people's subjective judgements (sentiments, opinions, evaluations and attitudes) towards entities and their aspects [Liu, 2012]. The basic task of the sentiment analysis is to identify the sentiment polarity of a given text, which means to determine the author's attitude is positive, neutral or negative towards the subject.

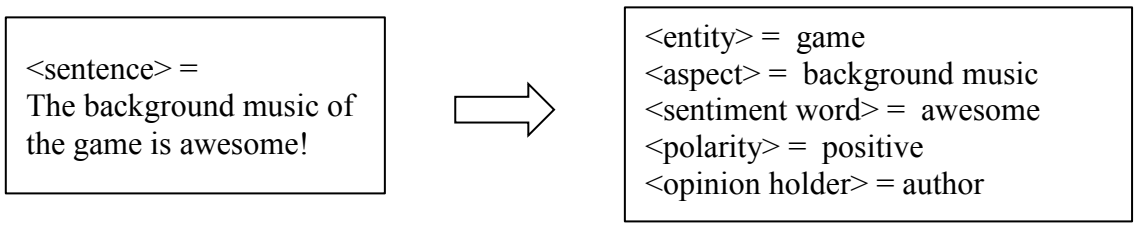


Figure 5.1. An example of sentiment analysis.

As shown in Figure 5.1, a simple review sentence "*The background music of the game is awesome!*" is analyzed to introduce the basic terms of sentiment analysis. An entity is the subject on which an opinion is expressed. It can be an item, individual, organization, service, topic, or event [Liu, 2012]. In the example, the entity is the "game" on which the review sentence is commenting. An aspect is a part, attribute, or feature of an entity [Liu, 2012], e.g., the camera of a smartphone, the plot of a movie, or the food of a restaurant. In the example, the aspect is the "background music" of the game. In a review, a user can express the opinions on different aspects of a mobile application, such as the graphics, sound effects, price, security, updates, or even its development company. The sentiment polarity or orientation (positive, neutral or negative) on the aspect "background music" of the entity "game" is positive in the example, which is deduced by the positive sentiment word "awesome". The sentiment word is a word that conveys positive or negative sentiment polarity [Qiu et al., 2009]. The opinion holder is the author of the review.

5.2. Three levels of sentiment analysis

According to different levels of analysis, sentiment analysis can generally be divided into document-based, sentence-based, and aspect-based sentiment analysis. Document-based, sentence-based, and aspect-based sentiment analysis correspondingly analyzes the sentiment polarity of a whole document, a single sentence, and the aspects of a given text [Liu, 2012]. A user review of a social app is used as an example. The reviewer rates the app four stars (from one to five) and the review content is as follows (a number is marked before each sentence for easy reference):

> *"(1) I downloaded this app last week. (2) It was a great app. (3) I liked the convenient way to chat with friends using the voice input instead of typing. (4) The friend recommendation system was also fabulous, which helped me*

find my old friends and meet lots of interesting new people. (5) However, the frequent updates were quite annoying. (6) My girlfriend was mad with me as I spent too much time chatting with my friends."

Firstly, it can be noticed that the sentiment polarity of sentence (1) is neutral. It does not express any opinions but simply states a fact. Sentences (2), (3) and (4) express positive sentiments, while sentences (5) and (6) show negative sentiments expressed by two different opinion holders "I" and "my girlfriend". To classify sentiment expressed in each sentence is called the sentence-based sentiment analysis [Liu, 2012]. Then, the sentence (2) is an overall assessment of the entity "app", and the sentiments expressed in sentences (3), (4) and (5) are towards different aspects "voice input", "friend recommendation system" and "frequent updates" of the entity "app". It is easy to figure out that the author's sentiments towards "voice input" and "friend recommendation system" are positive while the sentiment towards "frequent updates" is negative. To discover the aspects and determine the sentiment polarity of each aspect is called the aspect-based sentiment analysis [Liu, 2012]. Finally, it can be evaluated that the sentiment polarity of the whole review content is positive, which can also be supported by the high rate score of the review. Document-based sentiment analysis analyzes the sentiment polarity of the whole document, which usually provides an overall opinion on an entity [Liu, 2012].

In this study, the goal is to analyze the sentiment polarities of the keyphrases extracted in the reviews and calculate the numbers of positive, negative and neutral reviews concerning each keyphrase. Thus, the mobile application developers can firstly focus on the keyphrases that receive most negative reviews and take quick actions to find the causes from the reviews. To analyze the sentiments for the keyphrases is counted as an aspect-based sentiment analysis task.

5.3. Sentiment classification methods

As stated earlier, the basic task of the sentiment analysis is to identify the sentiment polarity of a given text, which is called sentiment classification [Pang et al., 2002]. It can be done by three different types of approaches: the machine learning approach, the lexicon-based approach and the hybrid approach [Maynard and Funk, 2011; Medhat et al., 2014].

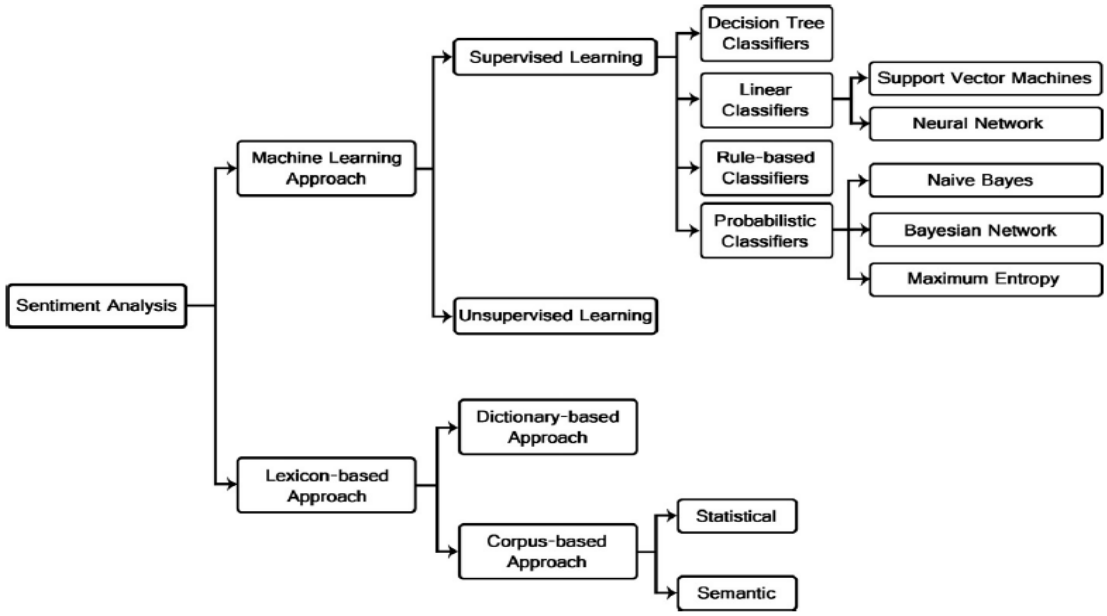


Figure 5.2. Sentiment classification methods [Medhat et al., 2014].

The machine learning approach considers the sentiment analysis task as a text classification problem, which classifies text based on its sentiment polarity into three different categories: positive, negative and neutral. It always relies on the supervised learning methods, where a large quantity of labeled texts of their sentiment polarities is utilized to train classifiers. The unsupervised learning methods are used when it is challenging to gain the labeled train data.

The lexicon-based approach is to judge the sentiment polarity of a text based on the sentiment words that it contains. It relies on a sentiment lexicon, which means a list of positive and negative words manually created or automatically generated. Then the sentiment polarity of the text can be decided on the ratio of the positive to negative sentiment words [Maynard and Funk, 2011]. In the previous example of the user review of a social app in Section 5.2, the whole review content is considered to be positive because more positive words such as “great”, “liked”, “fabulous” and “helped” can be found than the negative words in the review content. To use the lexicon-based approach, the key task is to generate the sentiment lexicon. Except manually creating the sentiment lexicon, which is very time consuming, there are two automated approaches collecting a sentiment lexicon, and they are the dictionary-based approach and the corpus-based approach.

The basic idea of the dictionary-based approach is to firstly manually collect a small set of sentiment words as the seed list. Then, these sentiment words are searched in a dictionary such as WordNet [Miller, 1995], a popular lexical database for the English language, for their synonyms and antonyms. For a positive (or negative) word, its synonyms are considered as positive (or negative), while its antonyms are seen as negative (or positive). All these new found synonyms and antonyms are added to the

seed list and then the next iteration starts. When no new synonyms and antonyms are found, the iteration process stops, and the sentiment lexicon is finally obtained. The major disadvantage of the dictionary-based approach is that the generated lexicon is not domain specific [Medhat et al., 2014]. To explain that some sentiment words usually have different sentiment polarities in different domains, a good example is the word "unpredictable". It always has a negative polarity in the health domain and aerospace filed, e.g., "the new flu virus is evolving in an unpredictable way" and "the landing point of the rocket is unpredictable", but is most likely to be positive in the movie domain, e.g., "the plot of this movie is unpredictable" [Ofek et al., 2013]. The SentiWordNet [Baccianella et al., 2010] is one of the most commonly used sentiment lexicons generated based on this dictionary-based approach. As shown in Figure 5.3., the adjective "unpredictable" always has a positive sentiment score of 0 and a negative sentiment score of 0.625 or 0.25 based on its different meanings in the SentiWordNet v3.0.0, which means it will be calculated as negative regardless of the domain in which it appears.

#	POS	ID	PosScore	NegScore	SynsetTerms	Gloss
a		01842001	0	0.625	unpredictable#1	not capable of being foretold
a		00739789	0	0.25	unpredictable#2	unknown in advance; "an unpredictable (or indeterminable) future"
a		00594146	0	0.25	unpredictable#3 irregular#2	not occurring at expected times

Figure 5.3. The word "unpredictable" in the SentiWordNet.

The corpus-based approach helps to generate the domain specific sentiment lexicon [Medhat et al., 2014]. It also starts with a small set of sentiment words as a seed list and the list is expanded by finding other sentiment words in a large document corpus based on syntactic or co-occurrence patterns [Medhat et al., 2014]. Hatzivassiloglou and McKeown [1997] did the first study using the corpus-based approach. They started with a labelled seed list of adjectives (657 positive and 679 negative adjectives) and used the conjunctions between adjectives to determine other adjectives' sentiment polarities. For most connectives such as "and", "or", "either-or" and "neither-nor", the conjoined adjectives were seen of the same sentiment polarity. The only exception was for the connective "but", which connected two adjectives of the opposite sentiment polarities. The weakness of this method using conjunction rules was obvious that it could not extract unpaired adjectives. Qiu et al. [2009] proposed a more complex method called double propagation, which parsed words in a sentence by their dependency relations and used part-of-speech (POS) information to extract the related features and sentiment words that matched the predefined dependency rules. After finding the related features and sentiment words, Qiu et al. [2009] defined the rules to estimate the sentiment polarity of a sentiment word based on two observations: "same polarity for same feature in a review" and "same polarity for same sentiment word in a domain corpus".

Therefore, features were added as a bridge to connect sentiment words that are related (modify the same feature) but not literally connected by connectives, which was proved to be an effective method with a high F-score [Qiu et al., 2009]. To the best of the author's knowledge, currently there is no domain specific sentiment lexicon available based on the app review data as a corpus, which might be caused by the fact that the app store data mining is a new study field inspired by the proliferation of mobile application business.

5.4. Related studies of sentiment analysis using app review data

Along with the explosive growth of social media in recent years, sentiment analysis has been broadly applied to analyze movie reviews, product reviews, news, blogs and twitter. The twitter sentiment, for instance, was proved as a valid indicator of political elections [Tumasjan et al., 2010]. Bollen et al. [2011] used the twitter sentiment to predict the daily up and down changes in the stock market. There are also online applications available to track public opinions towards a topic from Twitter. An example can be seen in Figure 5.4, the Sentiment Viz[9] was used to find public opinions about the topic "gay marriage" on Twitter. It was an online application to search for tweets containing the keyword and to analyze their sentiments. Recent tweets containing the keyphrase "gay marriage" were shown as the dots positioned by the estimated sentiments of their texts. Unpleasant (negative) tweets were drawn as blue dots on the left side, while pleasant (positive) tweets were shown as green dots on the right side.

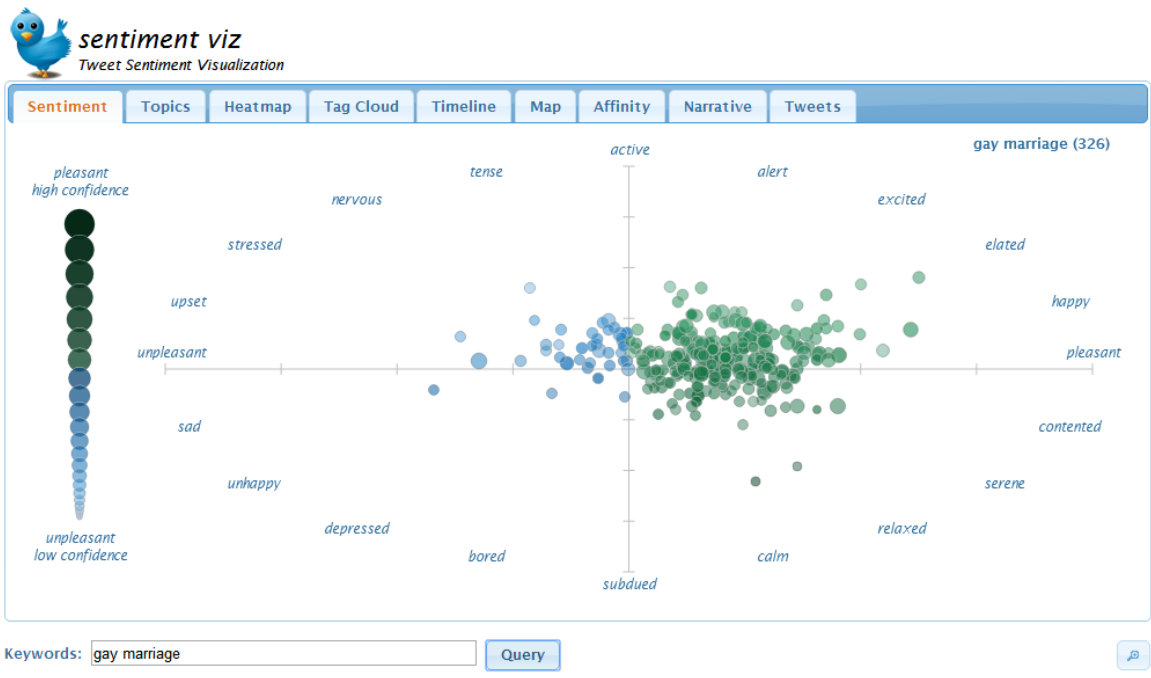


Figure 5.4. The online sentiment analysis application for Twitter.

9 https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app. Access date: 29.11.2017

Sentiment analysis of app review data is a relatively new domain and only a few studies can be found. Guzman and Maalej [2014] performed the aspect-based sentiment analysis of apps' review in Google Play and App Store (iOS). They extracted features (aspects) from the user reviews using the collocation finding algorithm provided by the NLTK toolkit [Loper and Bird, 2002], a natural language processing tool in Python. Then, they utilized the SentiStrength [Thelwall et al., 2012], a lexicon-based sentiment analysis tool for short informal English text, to calculate the sentiment score of the review sentences from which the features are extracted. Finally, they considered the sentiment score of a feature to be the sentiment score of the sentence from which it is extracted.

Maalej and Nabil [2015] and Panichella et al. [2015] both performed sentiment analysis on the sentence level and used the analyzed sentiment score or polarity as an attribute of the sentence for the follow-up text classification tasks. Maalej and Nabil [2015] also used the SentiStrength for analyzing the sentiment polarities of the review sentences, while Panichella et al. [2015] implemented the supervised learning method by training the Naive Bayes classifier using a set of 2090 manually labelled review sentences. Both studies found that adding the sentiment analysis result as an attribute improved the accuracy of the trained classifier.

Gu and Kim [2015] carried out the aspect-based sentiment analysis. They built a pattern-based parser to extract aspect-opinion pairs from the semantic dependency graph (SDG) of a review sentence. They also used the Deeply Moving, which was an early version of the Stanford CoreNLP Sentiment Analyzer [Socher et al., 2013], to analyze the sentiment score of each review sentence and adjusted the sentiment score using the review rate. Their evaluation of the sentiment analysis step on reviews of 17 popular Android apps from Google Play achieved a good result with a high average F1-scores of 0.80.

In this study, the goal is to analyze the sentiments of the extracted keyphrases. The sentiment polarity of the keyphrase will be considered the same as the sentiment polarity of the review sentence from which the keyphrase is extracted. Although this assumption was also used in [Guzman and Maalej, 2014] and [Gu and Kim, 2015], it was not precise. For the review sentence "I really love the outstanding graphics of the app, but the sound effects is not satisfying.", the sentiment polarity of the whole sentence is positive. However, the author's sentiment towards the two keyphrases "graphics" and "sound effects" are not both positive. Nevertheless, by manually analyzing 100 review sentences randomly selected from the 1512 app reviews of Instagram and Pokémon GO used in the mapping process in Section 3.5, 72 review sentences only described one aspect of the app, and the sentiment polarity of the sentence could represent the sentiment polarity of the corresponding aspect described in the sentence. Thus, this assumption was correct in most cases. Since it is also the third

research question of this study to find a proper way to analyze the users' sentiments expressed in their reviews towards the extracted keyphrases, the accuracy of the sentiment analysis results based on the assumption that represents the sentiment polarity of the keyphrase by the sentiment polarity of the review sentence from which it is extracted will be further tested. For analyzing the sentiment polarity of a review sentence, the Stanford CoreNLP Sentiment Analyzer [Socher et al., 2013] is utilized in ARAM and will be discussed in next section.

5.5. Stanford CoreNLP Sentiment Analyzer

Stanford CoreNLP [Manning et al., 2014] is a Java annotation pipeline framework which is a comprehensive Natural Language Processing toolkit that can perform most of the common core NLP tasks such as tokenization, POS tagging, name entity recognition (NER), dependency parsing, and sentiment analysis. The sentiment analysis tool and other annotators that support it in Stanford CoreNLP, which is called the Stanford CoreNLP Sentiment Analyzer, is a deep-learning model for sentiment analysis based on the Stanford Sentiment Treebank and the Recursive Neural Tensor Network [Socher et al., 2013]. The Stanford Sentiment Treebank is the first corpus with fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences extracted from movie reviews of the famous film review aggregator website Rotten Tomatoes [Socher et al., 2013]. The Recursive Neural Tensor Network (RNTN) is a powerful model to predict the compositional semantic effects with high accuracy [Socher et al., 2013]. It parses phrases into a tree of word vectors and then uses a tensor-based composition function to calculate the vectors in a bottom up fashion [Socher et al., 2013].

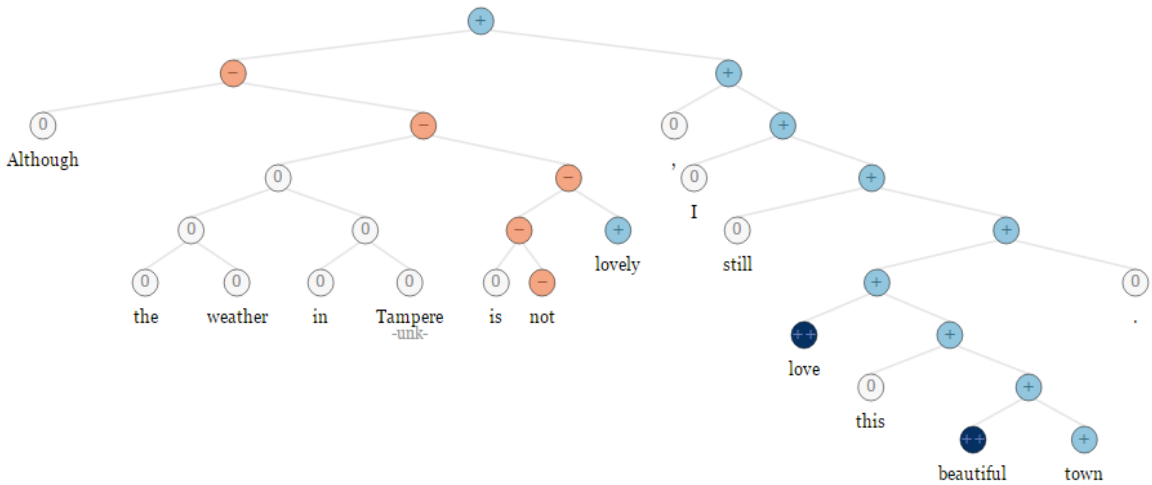


Figure 5.5. An example of RNTN predicting sentiment classes

Figure 5.5 shows the parsing result of the sentence "*Although the weather in Tampere is not lovely, I still love this beautiful town.*" using RNTN. From bottom to top,

every node of the parse tree is calculated for its sentiment polarities from very negative to very positive (--, -, 0, +, ++) [Socher et al., 2013]. The sentiment polarity of the root node is the sentiment polarity of the whole sentence, which is correctly calculated as positive in the example. The advantage of using RNTN is that it can accurately handle the negations (71.4% accurate for handling negated positive and 81.8% accurate for handling negated negative [Socher et al., 2013]) and clauses, which can also be supported in the above example for correctly handling the negation "is not lovely" and the although-clause.

The reason to choose the Stanford CoreNLP Sentiment Analyzer is that it has reached a very high accuracy of 85.4% in binary classification of positive/negative for single sentence and 80.7% accuracy on fine-grained sentiment prediction (from very negative to very positive) across all phrases [Socher et al., 2013]. Although its model is trained and tested using the movie review data, it is still promising to perform well in analyzing mobile application user review data. The reason is that the app review data and the movie review data are both informal review texts that share many characteristics in common. This study will test whether the Stanford CoreNLP Sentiment Analyzer can also perform effectively in the sentence-based sentiment analysis of app review data.

6. Implementation of ARAM

This chapter demonstrates the implementation details of ARAM. It contains six sections. The first section gives an overall introduction of the design and process of ARAM. The second section introduces the data collector and the JSON format used to store app review data. The third section presents the user review classifier that employs ARdoc [Panichella et al., 2016] to classify app review sentences and filter out uninformative reviews. The fourth section exhibits three different implementations of the keyphrase extractor, and they are relatively based on the term frequency method, the TextRank method [Mihalcea and Tarau, 2004], and the DoubleRank method that combines TextRank with term frequency. In the fifth section, the implementation of the sentiment analysis of the extracted keyphrase is explained followed by the visualization of the analysis results of the app reviews produced by ARAM in the sixth section.

6.1. Overall process of the user review analysis system

With the purpose of extracting useful information concerning application maintenance from app reviews and producing a report that concluded the analysis result for application developers, ARAM was designed to implement three main tasks. The first task was to classify user review sentences based on the taxonomy proposed by Panichella et al. [2015] and filter out uninformative reviews. Secondly, the top ten keyphrases of each informative category of review sentences ("information giving", "information seeking", "feature request", and "problem discovery") were extracted to represent the primary topics contained in each category of informative review sentences. Thirdly, users' sentiments on each extracted keyphrase were analyzed and presented in the report, so developers would be able to pay more attention to the keyphrases with more negative votes, find the causes of users' negative attitudes towards the keyphrases, and take action to maintain the corresponding aspects of the app if necessary.

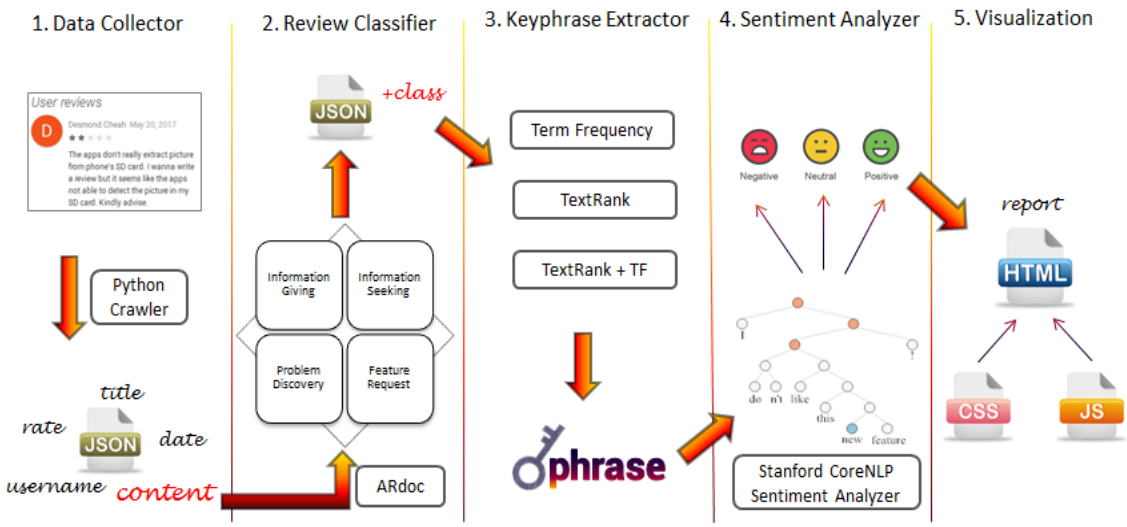


Figure 6.1. Overview of the system process.

To perform these three tasks, ARAM was composed of five components as shown in Figure 6.1. The data collector was coded with Python 2.7.10, while the other four components of ARAM including the user review classifier, the keyphrase extractor, the sentiment analyzer, and the visualization component were implemented by Java SE 1.8. The design of ARAM was not complicated, and it straightforwardly followed the process of collecting app review data, classifying user review sentences using ARdoc [Panichella et al., 2016], extracting top ten keyphrases, analyzing users' sentiments towards the extracted keyphrases, and visualizing the results. Apart from the data collector, each part carried out a further analysis based on the output of the previous part until the final report was produced. The implementation details of the five components of ARAM would be discussed separately in the next five sections.

6.2. Collection of data

A data collector, which was a web crawler implemented in Python, was used to crawler the user review data of a selected app in Google Play. The reviews were crawled in chronological order, and the number of reviews to crawl could be set to a desired value. The input of the data collector was the app id (e.g., "com.facebook.katana", "com.instagram.android", "com.nianticlabs.pokemongo"). The id of an app could be found on the URL of the app home page in Google Play as shown in Figure 6.2.

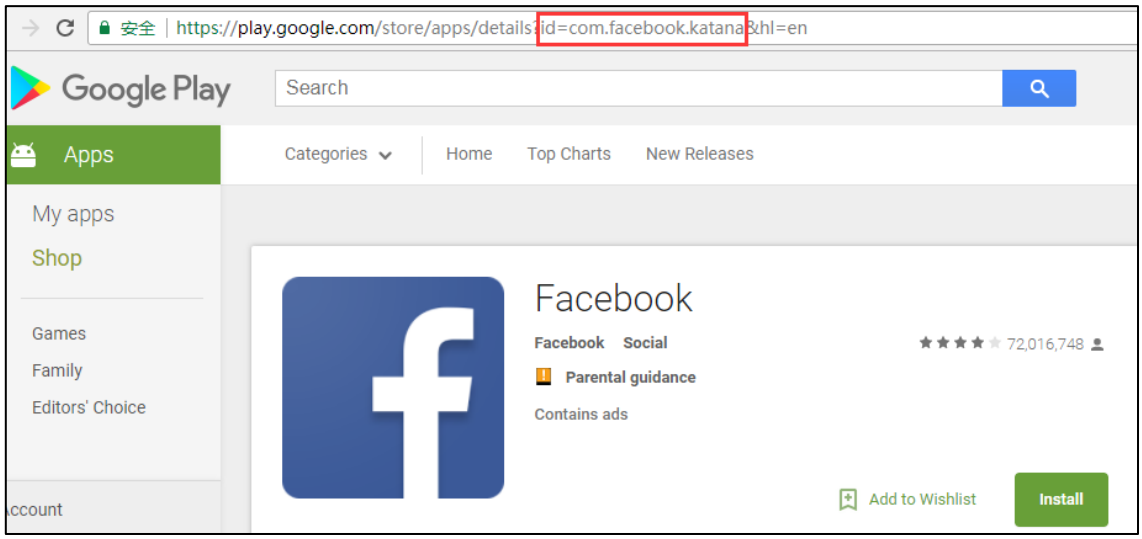


Figure 6.2. The Facebook app home page in Google Play.

```
"com.facebook.katana": [
    {
        "review_date": "May 11, 2017",
        "review_contents": "Can we have the side chat bar back instead of it going to the messenger app",
        "review_title": "",
        "review_rates": "5",
        "author_name": "Natalie Daleki"
    },
    {
        "review_date": "May 11, 2017",
        "review_contents": "That app is very good and also its very easy to use",
        "review_title": "",
        "review_rates": "5",
        "author_name": ""
    },
    {
        "review_date": "May 11, 2017",
        "review_contents": "As I do love Facebook I'm very upset that I dont get to chose whether or no I wanna see
        "review_title": "",
        "review_rates": "1",
        "author_name": "Ni West"
    },
```


Figure 6.3. Raw JSON data of user reviews.

The output was a JSON file storing the crawled user reviews of the app. As shown in Figure 6.3, for each user review, its review data, review contents, review title, review rate and author's username were stored in a piece of JSON data. Additionally, as mentioned in Section 3.1, user reviews in Google Play currently do not have review titles. When the data collector was firstly implemented in January 2017, some reviews in Google Play were observed to have review titles, and thus the data collector also collected the review title. The review titles would always be an empty value since reviews in Google Play do not have titles nowadays. However, it had no impact on the system since only the review contents were analyzed by ARAM.

6.3. User review classifier

The user review classifier firstly filtered out reviews whose contents were less than four words. The reason was that after manually reading total 1,200 user reviews of six different apps (200 reviews of each app including Facebook, Instagram, Clash Royale, Pokémon GO, Maps - Navigation & Transit, and TripAdvisor Hotels Restaurants) collected in 11th, May 2017, it was found that the reviews whose contents were less than four words were mostly uninformative for maintenance. Most of them were either generic praises or complaints such as "Cool app", "Definitely 5 stars!" and "Nothing good", or totally meaningless words or incomplete sentences. Although these reviews would mostly be correctly classified as the "other" type by ARdoc [Panichella et al., 2016], directly filtering out these uninformative short reviews before the classification work could increase efficiency.

```
{
    "author_name": "Corey Campbell",
    "review_class": [
        {"INFORMATION GIVING": "But I'm not okay with you constantly updating it and this will be the 3rd update I have
        {"PROBLEM DISCOVERY": "Now thanks to the updates, it crashes constantly now!"}
    ],
    "review_date": "May 10, 2017",
    "review_rates": "3",
    "review_contents": "I love this app. But I'm not okay with you constantly updating it and this will be the 3rd updat
    "review_title": ""
},
{
    "author_name": "",
    "review_class": [{"INFORMATION SEEKING": "where is the sd card transfer option?"}],
    "review_date": "May 10, 2017",
    "review_rates": "3",
    "review_contents": "where is the sd card transfer option?",
    "review_title": ""
},
```


Figure 6.4. Processed JSON data of user reviews with classification results.

After filtering out the reviews whose contents are less than four words, the user review classifier further employed ARdoc [Panichella et al., 2016] to classify the user reviews at the sentence-level granularity into five categories namely “information giving”, “information seeking”, “feature request”, “problem discovery” and “other” based on the taxonomy proposed by Panichella et al. [2015]. The input of the user review classifier was the JSON file produced by the data collector. As presented in Figure 6.4, the output was another JSON file, in which the classification results were stored by adding a property named the “review_class”. The value of the property “review_class” was a JSON array, and it stored the review sentences that were classified as one of the four informative types including “information giving”, “information seeking”, “feature request” and “problem discovery”. For instance, the user review *“I love this app. But I'm not okay with you constantly updating it and this will be the 3rd update I have had this week. Now thanks to the updates, it crashes constantly now!”* contained three sentences. The first sentence *“I love this app.”* was classified as “other”, the second sentence *“But I'm not okay with you constantly updating it and this will be the 3rd update I have had this week.”* was categorized as “information giving”, and the third sentence *“Now thanks to the updates, it crashes constantly now!”* was tagged as “problem discovery” by ARdoc [Panichella et al., 2016]. Then the second sentence of the “information giving” type and the third sentence of the “problem discovery” category were stored as two JSON objects in the value array of the property “review_class” as shown in Figure 6.4. All classified sentences that were not of the “other” type were stored in the array as the values of JSON objects and their corresponding types as the keys. In this way, the classified results of review sentences were stored for further analysis.

The reason to produce a JSON file as output was to separate the implementation of the user review classifier from the following three components of ARAM including the keyphrase extractor, the sentiment analyzer, and the visualization component. Two reasons for this separation were: firstly, the user review classifier often took some time to process reviews (the performance of the user review classifier was tested in Section

7.1). If the user review classifier was combined with the following three components without a JSON output, then any crashes in the following three components would result in restarting the whole computation from beginning and waste time on the user review classifier performing the same classification work again. Secondly, the user review classifier employed ARdoc [Panichella et al., 2016], and ARdoc relied on the old version 3.4.1 (published on 27th, August 2014) of Stanford CoreNLP[10]. In the following three components, the new version 3.7.0 (published on 31th, October 2016) of Stanford CoreNLP was utilized for performing NLP tasks including POS tagging, lemmatization and parsing. Since the ARdoc [Panichella et al., 2016] was not open source, its code could not be updated to adopt the Stanford CoreNLP 3.7.0, which was used in the keyphrase extractor and the sentiment analyzer of ARAM. To avoid possible incompatibility problems, the implementation of the user review classifier was also needed to be separated from the following three components of ARAM.

6.4. Keyphrase extractor

Based on the background knowledge of keyphrase extraction in Chapter 4, the keyphrase extractor implemented two different algorithms including the statistics method term frequency, and the graph-based ranking method TextRank [Mihalcea and Tarau, 2004]. Moreover, the author also proposed a new method named DoubleRank that combined TextRank with term frequency. The input of the keyphrase extractor was the JSON file produced by the user review classifier. The output would be the top ten keyphrases of the user review sentences in each category of informative review sentences (e.g., "information giving", "information seeking", "feature request" and "problem discovery").

6.4.1. Term frequency method

As mentioned before in Subsection 4.2.3, the tricky part of the term frequency method was the preprocessing phase for candidate keyphrase identification. To identify candidate noun phrases, Stanford Parser [Chen and Manning, 2014], an annotator integrated in Stanford CoreNLP [Manning et al., 2014] to analyze the grammatical structure of sentences, was utilized to extract noun phrases from informative review sentences. For example, for the "information giving" review sentence *"The app keeps closing randomly while I'm trying to watch a video"*, the Stanford Parser [Chen and Manning, 2014] could generate a parse tree of the review sentence as shown in Figure 6.5. The structure of the sentence was parsed and tagged with syntactic tags (e.g., "S" for "simple declarative clause", "NP" for "noun phrase", "VP" for "verb phrase", "ADVP" for "adverb phrase", etc) and POS tags (e.g., "DT" for "determiner", "NN" for "singular noun", "VBZ" for "3rd person singular present verb", etc) [Marcus et al.,

10 https://stanfordnlp.github.io/CoreNLP/history.html. Access date: 29.11.2017

1993]. The matching pattern *"NP !<< NP & !<< VP"* was used to extract the noun phrases that were not dominating another noun phrase or verbal phrase from the parse tree. Thus, for the example sentence in Figure 6.5, three noun phrases "The app", "I" and "a video" would be extracted. Although a parse tree was powerful to extract grammatically relevant words as phrases, which was a necessary step of the term frequency method for keyphrase extraction, the time to generate a parse tree was much longer than POS tagging. This would be shown when comparing the performance of the term frequency method and the TextRank method in Section 7.1.

```
The app keeps closing randomly while I'm trying to watch a video.
(ROOT
  (S
    (NP (DT The) (NN app))
    (VP (VBZ keeps)
      (S
        (VP (VBG closing)
          (ADVP (RB randomly))
          (SBAR (IN while)
            (S
              (NP (PRP I))
              (VP (VBP 'm)
                (VP (VBG trying)
                  (S
                    (VP (TO to)
                      (VP (VB watch)
                        (NP (DT a) (NN video))))))))))))
```

Figure 6.5. A parse tree of a review sentence.

Then, the extracted noun phrases were converted to lowercase, and the stop words in the phrases were removed. A default English stop words list[11] containing 174 stop words was used, and some informal abbreviations such as "bc", "plz" and "cuz" encountered when manually reading the app reviews were also appended to the list. Thus, the three extracted noun phrases "The app", "I" and "a video" would turn into two noun phrases "app" and "video". Finally, the noun phrases were lemmatized using the Stanford CoreNLP Toolkit [Manning et al., 2014] and formed the "candidate phrases". The "candidate phrases" contained both nouns and noun phrases.

The term frequencies of the candidate phrases extracted from each informative category of user review sentences were calculated. If reduplicative candidate phrases were extracted from one review's same category of the informative review sentences ("information giving", "information seeking", "feature request" or "problem discovery"), the candidate phrases would only be counted once. For example, if a review had three "information giving" review sentences that all included the noun phrase "user interface", or one "information giving" review sentence where the "user

11 https://www.ranks.nl/stopwords. Access date: 29.11.2017

interface" appeared twice, the candidate noun phrase "user interface" would only be counted once in the "information giving" category. Moreover, if a review had one "information seeking" sentence and one "problem discovery" sentence that both contained the noun phrase "user interface", the candidate noun phrase "user interface" would be counted once in both the "information seeking" category and the "problem discovery" category. The purpose of this process was to keep the principle of "one review one vote". Therefore, a candidate phrase in one review would only be counted once even though it might appear multi-times, and hence enhance the keyphrase results' representativeness of the topics that were covered in most reviews.

For each informative category of user review sentences, the term frequency of every candidate keyphrase was counted, and the ten keyphrases with highest term frequencies were selected as the final result. Figure 6.6 showed the top ten keyphrases result of 4449 informative reviews of the social app Instagram analyzed by the term frequency method. The top ten keyphrases of each informative category were sorted by their frequencies (the numbers after the symbol "=>").

Information Giving	Information Seeking	Feature Request	Problem Discovery
app ==> 240	app ==> 22	feature ==> 23	problem ==> 163
instagram ==> 202	face filter ==> 21	app ==> 22	app ==> 155
video ==> 162	filter ==> 12	instagram ==> 14	face filter ==> 118
update ==> 129	update ==> 11	filter ==> 12	update ==> 88
photo ==> 107	phone ==> 10	photo ==> 11	instagram ==> 87
account ==> 101	instagram ==> 9	phone ==> 7	video ==> 85
face filter ==> 93	post ==> 8	chronological order ==> 7	issue ==> 65
filter ==> 86	photo ==> 8	update ==> 7	phone ==> 54
post ==>86	story ==> 8	post ==> 7	filter ==> 52
story ==> 77	android ==> 7	face filter ==> 5	account ==> 45

Figure 6.6. The keyphrase result of the term frequency method.

6.4.2. TextRank method

Four sets of the review sentences namely Set_{ig}, Set_{is}, Set_{fr}, and Set_{pd} were formed from the JSON file produced by the user review classifier, and they separately contained all the "information giving" sentences, the "information seeking" sentences, the "feature request" sentences and the "problem discovery" sentences of the reviews. The size of a set was defined as the number of reviews that contained the corresponding informative review sentences. For example, if the Set_{ig} contained 105 information giving sentences that were from 74 user reviews, the size of Set_{ig} was the number of reviews which was 74. The set size was used as a threshold in the post-processing phase.

The preprocessing of the review sentences in each set also applied the lowercase conversion, the stop words removal and the word lemmatization. However, instead of parsing to extract noun phrases as candidate phrases, which was a very time-consuming process, only POS tagging was needed to extract nouns and adjectives as candidate words. The Stanford Part-Of-Speech Tagger [Toutanova et al., 2003] was utilized for automatic POS tagging.

After the preprocessing, for each review sentence set (Set_{ig}, Set_{is}, Set_{fr}, and Set_{pd}), all the candidate words (nouns and adjectives) were added to an undirected graph, and an edge was added between two words if they co-occurred within a window of two words. Then the initial rank value of each word was set to 1, and the Formula 4.4 introduced in the Section 4.3 (the damping factor d was set to 0.85 in the implementation) was run on the graph for 20 iterations. After 20 iterations of calculation, all the candidate words obtained their final score, and the top one third of the candidate words with highest scores were formed the keywords set.

Then, for the post-processing phase to form the keyphrases, the method proposed by Mihalcea and Tarau [2004] was quite straightforward: if there existed adjacent keywords in the original text, they would be collapsed into a multi-word keyphrase. Unlike the journal papers' abstracts analyzed in [Mihalcea and Tarau, 2004], the app review contents were informal, and the keyphrase results would contain many unexpected phrases if directly applying this post-processing method. Thus, a threshold was set that only the adjacent keywords had appeared at least T times instead of just one time would be collapsed into the final keyphrase. T was a positive integer and equaled the larger value of V_1 and V_2: V_1 was set to one fortieth of the set size (the size of Set_{ig}, Set_{is}, Set_{fr}, or Set_{pd}); V_2 was set to one third of the lowest term frequency of the keywords forming the keyphrase. For instance, if the size of Set_{ig} was 480 reviews, its keywords set included the nouns "user" and "interface", and the "user" and the "interface" separately appeared 83 times and 68 times in the original text, then V_1 would be 24 (one twentieth of 480) and V_2 would be 22 (the integer part of 68 divided by 3). Hence the combination of the two words ("user interface" or "interface user") needed to appear at least 24 times in the original text to be collapsed into the keyphrase (which would be represented by the one combination with the highest frequency, which was supposed to be "user interface" in this example). The value V_1 was to guarantee the combined keyphrases appeared enough times in the reviews, while the value V_2 was to ensure that only the candidate words frequently appearing together could form the keyphrases. The threshold could also be set to other suitable values. However, after testing different thresholds, using the combination of V_1 and V_2 as the threshold produced the best results.

The rank value of a multi-words keyphrase would be the average value of its combined words' rank values multiplied by 1.2 to promote the multi-words keyphrase.

Finally, the final keyphrase set with both single-word and multi-words keyphrases was formed, and the top ten keyphrases with highest rank values were selected as the result. Figure 6.7 showed the top ten keyphrases result of 4449 reviews of the social app Instagram analyzed by the TextRank method. The top ten keyphrases of each informative category were sorted by their TextRank values (the numbers after the symbol "=>").

Information Giving	Information Seeking	Feature Request	Problem Discovery
app ==> 28.6415	app ==> 7.4039	app ==> 9.9604	problem ==> 19.493
instagram ==> 23.3549	face filter ==> 6.7477	new feature ==> 6.6838	video ==> 16.8881
video ==> 20.8997	instagram ==> 4.2418	photo ==> 4.3761	app ==> 15.5727
photo ==> 17.2388	post ==> 3.9739	post ==> 4.192	instagram ==> 15.3623
account ==> 14.5209	story ==> 3.5403	face filter ==> 4.167	fix ==> 10.8826
post ==> 14.1897	android ==> 3.5185	instagram ==> 4.0384	face filter ==> 10.456
face filter ==> 12.7488	phone ==> 3.0261	account ==> 3.4722	issue ==> 10.161
picture ==> 12.6378	time ==> 2.9104	way ==> 2.7734	new ==> 9.2831
new ==> 10.8956	photo ==> 2.8492	video ==> 2.4762	time ==> 9.066
phone ==> 10.6103	new ==> 2.787	ig ==> 2.442	phone ==> 9.0357

Figure 6.7. The keyphrase result of the TextRank method.

6.4.3. DoubleRank method

A self-designed algorithm named DoubleRank that combined TextRank with term frequency was proposed. To keep the performance advantage of using POS tagging instead of parsing, DoubleRank mainly adopted the process of the TextRank method presented in the previous subsection, and it simplified the post-processing phase by adding term frequency as a factor in calculating the rank value.

DoubleRank firstly followed the same approach of the TextRank method to forming the keywords set that contained the top one third of words with highest TextRank scores. Then, for the post-processing phase to form the keyphrases, if there existed adjacent keywords in the original text, they would be combined into a multi-word keyphrase. The TextRank value of a combined keyphrase was set to the average value of its combined words' TextRank values. In the previous subsection, the TextRank method would "collapse" the keywords into the keyphrases if the thresholds were met, which meant the keywords that had combined into multi-word keyphrases would be discarded. However, the DoubleRank method would still store these

keywords as candidates. For example, if the nouns “face” and “filter” were in the keywords set and the noun phrase “face filter” had appeared once in the original text, then “face”, “filter” and “face filter” would all be candidates and their DoubleRank values would be calculated later to decide whether they could become top ten keyphrases.

All the words in the keywords set and the phrases they formed in the post-processing phase formed the candidate set *V*. For a given word or phrase *k* in the set, its DoubleRank value DR_k was defined as follows:

$$DR_k = \alpha * \frac{TF_k}{\sum_{i \in V} TF_i} * \beta + (1 - \alpha) * \frac{TR_k}{\sum_{i \in V} TR_i} \qquad \text{(Formula 6.1)}$$

TF_k was the term frequency of *k*, and TR_k was the TextRank value of *k*. $\sum_{i \in V} TF_i$ was the sum of the term frequencies of words and phrases in the set *V*. $\sum_{i \in V} TR_i$ was the sum of the TextRank values of words and phrases in the set *V*. α was a parameter for adjusting the weights between the term frequency and the TextRank value. It was set to 0.15 in the implementation. β was a parameter for promoting multi-word keyphrases. If *k* was a phrase, β was set to 8.0; if *k* was a single word, β was set to 1.0.

Finally, words and phrases were sorted in descending order of their DoubleRank values for selecting the top ten keyphrases. The word or phrase with the highest DoubleRank value was directly selected as the first keyphrase in the result set named Set_{res}. Then, for the word or phrase with the second highest DoubleRank value, if it did not contain any duplicated words appeared in the Set_{res}, it would be added into the Set_{res}; if it contained a duplicated word appeared in the Set_{res}, it would be discarded, and the word or phrase with the third highest DoubleRank value would be considered. The process went on until Set_{res} obtained ten elements. For example, if “app” and “face filter” had already been selected in the Set_{res}, then the words including “face” and “filter”, and the phrases such as “good app”, “useless app”, “cute face”, “bad filter” would not be added into the Set_{res} even though they had the highest DoubleRank values among the remaining words and phrases that were “competing” to be added into Set_{res}. In this way, the words or phrases containing a duplicated word would be represented by the only one word or phrase with the highest DoubleRank value among them.

Following was an example to explain the reason of setting parameters α and β. Considering the phrase “face filter”, if either one of the two words “face” and “filter” had a larger DoubleRank value than “face filter”, and was selected into the Set_{res}, then “face filter” would have no chance to be selected into the Set_{res}. The desired situation was that the “face filter” was added into the Set_{res} and further prevented its combined words “face” and “filter” from adding into the Set_{res}. Thus, the phrase “face filter” should have a larger DoubleRank value than “face” and “filter”. The DoubleRank value was relied on the TextRank value and the term frequency. While the TextRank value of a phrase was set to the average value of its combined words’ TextRank values in the

post-processing phase, the term frequency was the main reason why a phrase's DoubleRank value was generally smaller than the DoubleRank values of its combined words if the parameters α and β were not introduced in calculating the DoubleRank value. Theoretically, the term frequencies of the words must be no less than the phrases containing them. Actually, the term frequencies of words were much larger than the phrases containing them. By analyzing 4449 reviews of the social app Instagram, it was found that the term frequency of the phrase "face filter" was 863, while the "face" and the "filter" separately appeared 1112 times and 1532 times. Thus, the parameter α was set to 0.15 to reduce the weight of the term frequency in DoubleRank, and β was set for making up the relatively low term frequencies of phrases. β was set to 8.0 for calculating the DoubleRank values for phrases. This set of β could highly promote the phrases that appeared many times in the original text, while it had little influence on the DoubleRank values of the phrases with low term frequencies.

Information Giving	Information Seeking	Feature Request	Problem Discovery
face filter ==> 1.7282	face filter ==> 7.2777	face filter ==> 2.6519	face filter ==> 3.5824
app ==> 0.7544	app ==> 2.367	feature ==> 2.3434	problem ==> 0.982
instagram ==> 0.6239	instagram ==> 1.3803	app ==> 2.3069	app ==> 0.898
video ==> 0.5744	post ==> 1.1724	chronological order ==> 1.4081	video ==> 0.7615
refresh feed ==> 0.4636	story ==> 1.0907	instagram ==> 1.0071	instagram ==> 0.7212
photo ==> 0.3846	android ==> 1.0452	photo ==> 0.9849	fix ==> 0.5359
post ==> 0.3363	phone ==> 1.0193	post ==> 0.9772	issue ==> 0.4603
account ==> 0.3281	photo ==> 0.9198	social media ==> 0.8383	phone ==> 0.407
story ==> 0.2886	time ==> 0.8216	account ==> 0.7494	samsung galaxy ==> 0.3769
latest update ==> 0.2778	last update ==> 0.7566	live video==> 0.6082	account ==> 0.3726

Figure 6.8. The keyphrase result of the DoubleRank method.

Figure 6.8 showed the top ten keyphrases result of 4449 reviews of the social app Instagram analyzed by the DoubleRank method. The top ten keyphrases of each informative category were sorted by their DoubleRank values multiplied by 100 (the numbers after the symbol "=>"). The keyphrases result of the DoubleRank method contained the keyphrases such as "samsung galaxy", "refresh feed" and "latest update" that were not discovered by the previous two methods.

6.5. Sentiment analyzer

The sentiment analyzer employed the Stanford CoreNLP Sentiment Analyzer [Socher et al., 2013] to analyze the sentiments towards the keyphrases produced by the keyphrase extractor. The keyphrase extractor not only extracted the keyphrases as the result, but also stored the information of the review sentences from which each keyphrase was extracted. The review sentences from which a keyphrase was extracted were called the original review sentences for the keyphrase.

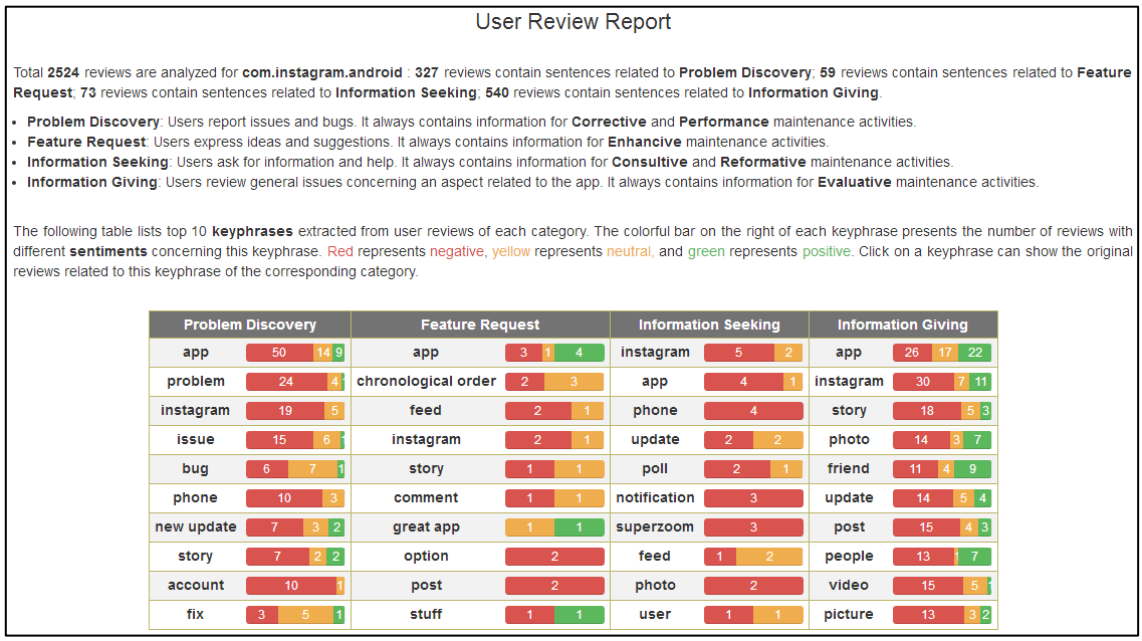
The sentiments for each keyphrase's original review sentences were calculated by the Stanford CoreNLP Sentiment Analyzer to represent the sentiment towards the keyphrase. The Stanford CoreNLP Sentiment Analyzer calculated the sentiments for each review sentence into five categories including "very negative", "negative", "neutral", "positive", and "very positive". For each keyphrase, the sentiment analyzer added the numbers of its original review sentences that were "very negative" and "negative" as the number of negative votes for the keyphrase. Similarly, the numbers of the "very positive" original review sentences and the "positive" original review sentences were added as the number of positive votes for the keyphrase. The number of the "neutral" original review sentences was seen as the number of neutral votes for the keyphrase. In this manner, the numbers of positive, neutral, and negative votes for each keyphrase were obtained and stored as the final result of the sentiment analyzer.

6.6. Visualization of results

The final output of ARAM was a HTML report on the local machine. As shown in Figure 6.9, combined with a predefined CSS file, the HTML report presented the analysis results for the input reviews including the numbers of reviews containing each category of informative review sentences (e.g., "information giving", "information seeking", "feature request" and "problem discovery"), the top ten keyphrases in each category of informative review sentences, and the numbers of positive, negative and neutral votes for each keyphrase.

The keyphrases were listed for each category of informative review sentences from top to bottom in the order of discending term frequencies or rank values. On the right side of each keyphrase, the colorful sentiment bar showed the numbers of negative, neutral and positive votes for the corresponding keyphrase. The ratios of negative, neutral and positive votes for each keyphrase were also visualized by the widths of three different colors in the sentiment bar (red for negative votes, yellow for neutral votes, and green for positive votes). Thus, app developers could pay more attention to the keyphrases whose votes were mostly negative with largest widths of the red color in their sentiment bars. Additionally, a brief introduction of the four categories of informative review sentences was also listed in the report, combined with a brief

conclusion of the typical maintenance types concerning each category of informative review sentences analyzed in Section 3.5.



User Review Report

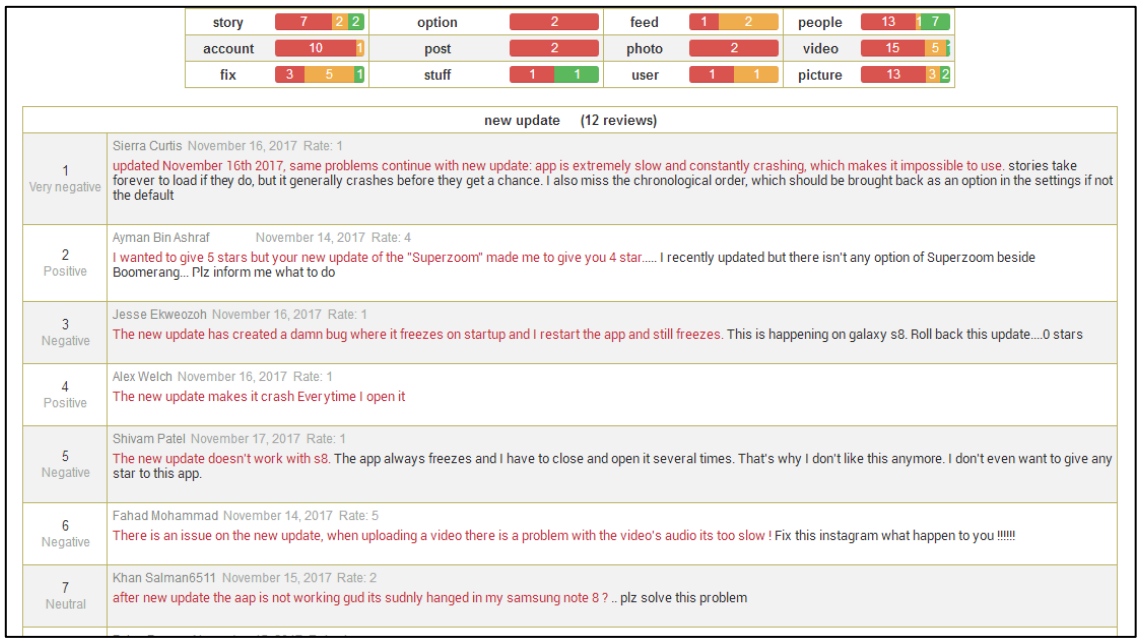
Total **2524** reviews are analyzed for **com.instagram.android** : **327** reviews contain sentences related to **Problem Discovery**; **59** reviews contain sentences related to **Feature Request**; **73** reviews contain sentences related to **Information Seeking**; **540** reviews contain sentences related to **Information Giving**.

- **Problem Discovery**: Users report issues and bugs. It always contains information for **Corrective** and **Performance** maintenance activities.
- **Feature Request**: Users express ideas and suggestions. It always contains information for **Enhancive** maintenance activities.
- **Information Seeking**: Users ask for information and help. It always contains information for **Consultive** and **Reformative** maintenance activities.
- **Information Giving**: Users review general issues concerning an aspect related to the app. It always contains information for **Evaluative** maintenance activities.

The following table lists top 10 **keyphrases** extracted from user reviews of each category. The colorful bar on the right of each keyphrase presents the number of reviews with different **sentiments** concerning this keyphrase. Red represents negative, yellow represents neutral, and green represents positive. Click on a keyphrase can show the original reviews related to this keyphrase of the corresponding category.

Problem Discovery		Feature Request		Information Seeking		Information Giving	
app	50 14 9	app	3 1 4	instagram	5 2	app	26 17 22
problem	24 4	chronological order	2 3	app	4 1	instagram	30 7 11
instagram	19 5	feed	2 1	phone	4	story	18 5 3
issue	15 6	instagram	2 1	update	2 2	photo	14 3 7
bug	6 7 1	story	1 1	poll	2 1	friend	11 4 9
phone	10 3	comment	1 1	notification	3	update	14 5 4
new update	7 3 2	great app	1 1	superzoom	3	post	15 4 3
story	7 2 2	option	2	feed	1 2	people	13 7
account	10 1	post	2	photo	2	video	15 5
fix	3 5 1	stuff	1 1	user	1 1	picture	13 3 2

Figure 6.9. A sample HTML report produced by ARAM.



story	7 2 2	option	2	feed	1 2	people	13 7
account	10 1	post	2	photo	2	video	15 5
fix	3 5 1	stuff	1 1	user	1 1	picture	13 3 2

new update (12 reviews)

1 Very negative	Sierra Curtis November 16, 2017 Rate: 1 updated November 16th 2017, same problems continue with new update: app is extremely slow and constantly crashing, which makes it impossible to use. stories take forever to load if they do, but it generally crashes before they get a chance. I also miss the chronological order, which should be brought back as an option in the settings if not the default
2 Positive	Ayman Bin Ashraf November 14, 2017 Rate: 4 I wanted to give 5 stars but your new update of the "Superzoom" made me to give you 4 star..... I recently updated but there isn't any option of Superzoom beside Boomerang... Plz inform me what to do
3 Negative	Jesse Ekweozoh November 16, 2017 Rate: 1 The new update has created a damn bug where it freezes on startup and I restart the app and still freezes. This is happening on galaxy s8. Roll back this update....0 stars
4 Positive	Alex Welch November 16, 2017 Rate: 1 The new update makes it crash Everytime I open it
5 Negative	Shivam Patel November 17, 2017 Rate: 1 The new update doesn't work with s8. The app always freezes and I have to close and open it several times. That's why I don't like this anymore. I don't even want to give any star to this app.
6 Negative	Fahad Mohammad November 14, 2017 Rate: 5 There is an issue on the new update, when uploading a video there is a problem with the video's audio its too slow ! Fix this instagram what happen to you !!!!!!
7 Neutral	Khan Salman6511 November 15, 2017 Rate: 2 after new update the aap is not working gud its sudnly hanged in my samsung note 8 ? .. plz solve this problem

Figure 6.10. Retrieving the original reviews containing a keyphrase in the report.

Moreover, the original review sentences for each keyphrase and the corresponding app reviews (including the review contents and other meta-data) were also stored in the HTML report. A predefined JavaScript file was coded for the users to interact with the HTML report to retrieve the app reviews related to a selected keyphrase. As shown in Figure 6.10, when users clicked on a keyphrase in the table, its original reviews related

to the keyphrase were presented. For each original review, its review contents and other meta-data including the reviewer's username, the review date and the review rate were shown in a cell of the table, and the original review sentence for the keyphrase in the review was highlighted in red. The sentiment polarity of the keyphrase (the original review sentence) in each review was also shown under the order number. Therefore, benefit from the retrieval function, app developers could check the original reviews related to the keyphrases that they were interested in to dig out more information for planning maintenance activities.

7. Evaluation of ARAM

This chapter contains four sections. The first section tests the performance of each component of ARAM including the user review classifier, the keyphrase extractor, and the overall performance. The second section analyzes the proportion of the informative reviews in Google Play according to the data using rate of ARAM. The third section carries out a keyphrase extraction test for comparing the automatic generated keyphrases by the keyphrase extractor to the human-generated keyphrases. In the fourth section, a sentiment analysis test was carried out to test how well the sentiment polarities of the keyphrases calculated by the sentiment analyzer of ARAM could agree with the human judgments.

7.1. System performance

A system performance test was carried out for testing the execution speed of the major components of ARAM. The test was run on a laptop computer. The testing environment was presented in Table 7.1.

CPU	Intel® Core™ i5-3230M @ 2.60GHz (2 cores)
memory	4 GB
hard disk	500 GB, 7200 RPM
graphics card	Nvidia GeForce GT 740M
operation system	Windows 7 SP1 64-bit
runtime environment	Java 1.8.0_77

Table 7.1. The testing environment.

Total 21,500 pieces of reviews of the social app Facebook in Google Play published during 8th and 15th November, 2017 were crawled to form the test dataset. These reviews were randomly assigned into one of the six small test datasets of 500, 1000, 2000, 4000, 6000 and 8000 reviews. The performance was tested by running ARAM to process the reviews in each small test dataset, and the processing times of the user review classifier, the processing times of three different implementations of the keyphrase extractor including the term frequency method, the TextRank method and the DoubleRank method, and the total processing times of the three components of ARAM including the keyphrase extractor, the sentiment analyzer, and the visualization component were recorded.

Table 7.2 presented the processing times of the user review classifier. The processing of the user review classifier started from reading the JSON file that contained the user reviews of the test dataset and ended by producing another JSON file that included the classification results.

dataset	1	2	3	4	5	6	average
number of reviews	500	1000	2000	4000	6000	8000	-
total processing time (second)	321.9	582.9	1559.1	2419.1	3344.2	5091.0	-
average time (second / per review)	0.644	0.583	0.780	0.605	0.557	0.636	0.634

Table 7.2. The processing times of the user review classifier.

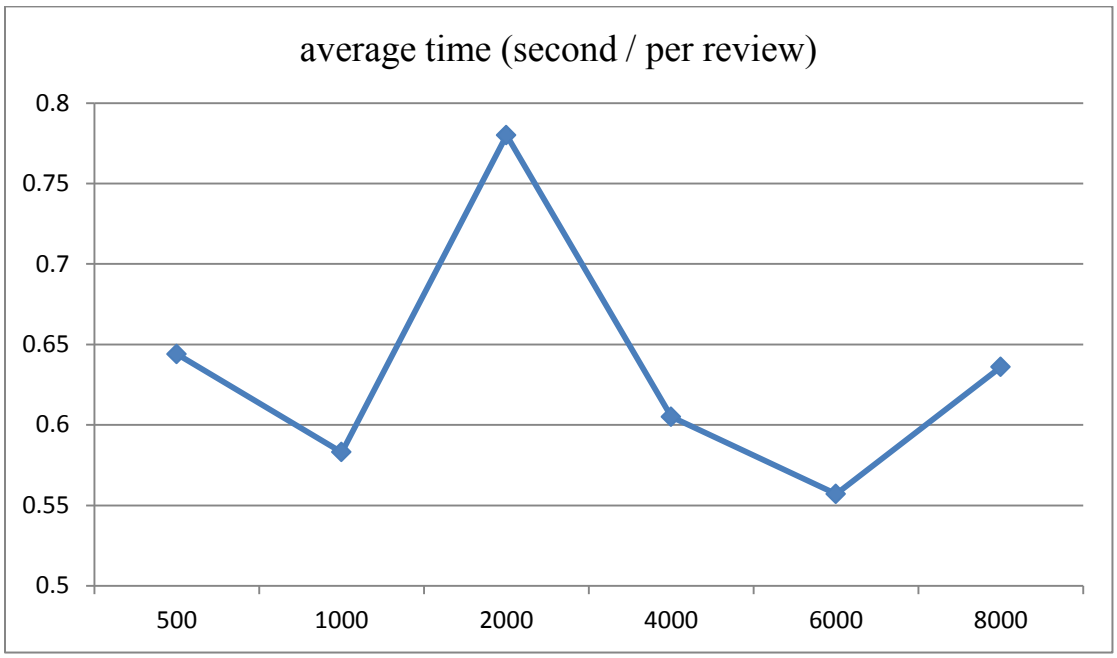


Figure 7.1. The average times of the user review classifier to process one review.

Figure 7.1 showed the diagram of the average times of the user review classifier to process one review. The average time of processing one review by the user review classifier was between 0.55s and 0.78s.

As explained in Section 6.3, the reviews whose contents were less than four words would be filtered out by the user review classifier. Thus, the six output JSON files obtained by running the user review classifier to process the six test datasets of 500, 1000, 2000, 4000, 6000 and 8000 reviews correspondingly contained 234, 470, 1101, 1908, 2748 and 4050 reviews remained. These six output JSON files were used for the tests of the following components of ARAM.

The author further tested the performances of the three different implementations of the keyphrase extractor including the term frequency method, the TextRank method, the DoubleRank method. The processing of the keyphrase extractor started from

reading the JSON file produced by the user review classifier and ended by producing the top ten keyphrase results of each type of informative reviews (e.g., “information giving”, “information seeking”, “feature request” and “problem discovery”). Table 7.3 presented the processing times of the three different implementations of the keyphrase extractor.

dataset		1	2	3	4	5	6	average
number of reviews		234	470	1101	1908	2748	4050	-
total processing time (second)	Term Frequency	55.8	74.2	203.5	312.9	412.0	670.6	-
	TextRank	3.8	3.9	6.9	15.1	23.5	53.2	-
	DoubleRank	3.5	3.9	7.8	13.6	23.8	49.5	-
average time (second / per review)	Term Frequency	0.238	0.158	0.185	0.164	0.150	0.166	0.177
	TextRank	0.016	0.008	0.006	0.008	0.009	0.013	0.010
	DoubleRank	0.015	0.008	0.007	0.007	0.009	0.012	0.010

Table 7.3. The processing times of the three different keyphrase extraction methods.

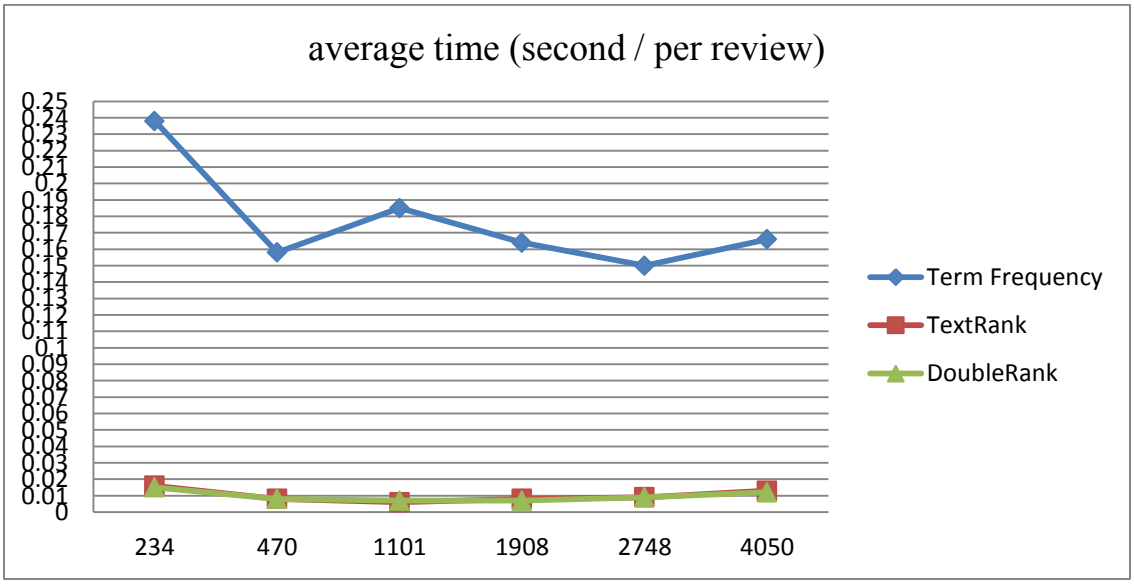


Figure 7.2. The average times of the keyphrase extractor to process one review.

Figure 7.2 showed the average times of the three different keyphrase extraction methods to process one review. The TextRank method and the DoubleRank method only needed around 0.01s to process one review, while the term frequency method took between 0.15s and 0.24s to process one review. The execution speeds of the TextRank method and the DoubleRank method vastly outperformed the term frequency method. It was mainly caused by the fact that the sentence parsing used in the term frequency method took much longer than the POS tagging used in the TextRank method and the DoubleRank method.

The author finally tested the total processing time of the three components of ARAM including the keyphrase extractor based on the term frequency method, the sentiment analyzer, and the visualization component. The processing of these three components started from reading the JSON file produced by the user review classifier and ended by producing the summary report. Currently only the keyphrase extractor implemented by the term frequency method had been integrated into ARAM due to the time constraints of this study. The keyphrase extractor based on the TextRank method and the DoubleRank method would be integrated into ARAM in the future work. Table 7.4 presented the test result of the total processing times of the three components of ARAM.

dataset	1	2	3	4	5	6	average
number of reviews	234	470	1101	1908	2748	4050	-
total processing time (second)	176.6	240.1	651.4	1106.4	1490.3	2586.9	-
average time (second / per review)	0.755	0.511	0.592	0.580	0.542	0.639	0.603

Table 7.4. The total processing times of the three components including the keyphrase extractor based on the term frequency method, the sentiment analyzer, and the visualization component.

Figure 7.3 showed the average times of the three components of ARAM including the keyphrase extractor based on the term frequency method, the sentiment analyzer, and the visualization component to process one review. The average time of processing one review by the three components was between 0.51s and 0.76s.

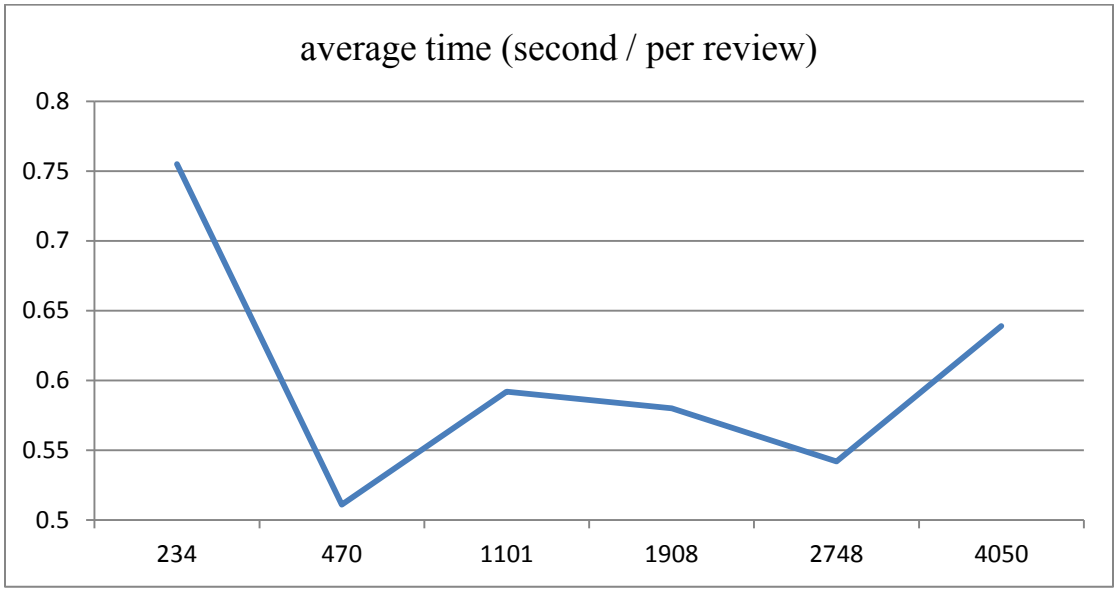


Figure 7.3. The average times of three components to process a review.

The total processing time of ARAM was considered the sum of the processing time of the user review classifier and the processing time of the following three components of ARAM including the keyphrase extractor, the sentiment analyzer, and the visualization component. Thus, combined the test results in Table 7.2 with Table 7.4, the total processing time of ARAM was shown in Table 7.5.

dataset	1	2	3	4	5	6	average
number of reviews	500	1000	2000	4000	6000	8000	-
total processing time (second)	498.5	823.0	2210.5	3525.5	4834.5	7677.9	-
average time (second / per review)	0.997	0.823	1.105	0.881	0.806	0.960	0.929

Table 7.5. The total processing time of ARAM.

It could be seen in Table 7.5 that ARAM took averagely one second to process a review, which meant that it could process about 1800 reviews in half an hour. The performance could be better if the keyphrase extractor based on the TextRank method or the DoubleRank method was employed in ARAM. Overall, the performance speed was acceptable considering the sophisticated processing of the app reviews by ARAM.

7.2. Proportion of the informative reviews classified by ARAM

As mentioned in Section 2.3, Chen et al. [2014] trained a model for filtering out non-informative reviews, and found that nearly 35.1% of app reviews were classified as informative reviews for developers to improve their apps. This study also tested the proportion of the informative reviews in Google Play for the application maintenance according to the classification results of the user review classifier of ARAM.

The user review classifier of ARAM firstly filtered out the reviews whose contents were less than four words, and these reviews were called as the "discarded" reviews. Then, if all sentences of one review were classified as the "other" type by the user review classifier, the review belonged to the "other" review and was considered uninformative and useless by ARAM. Hence the proportion of the informative reviews was defined as follows:

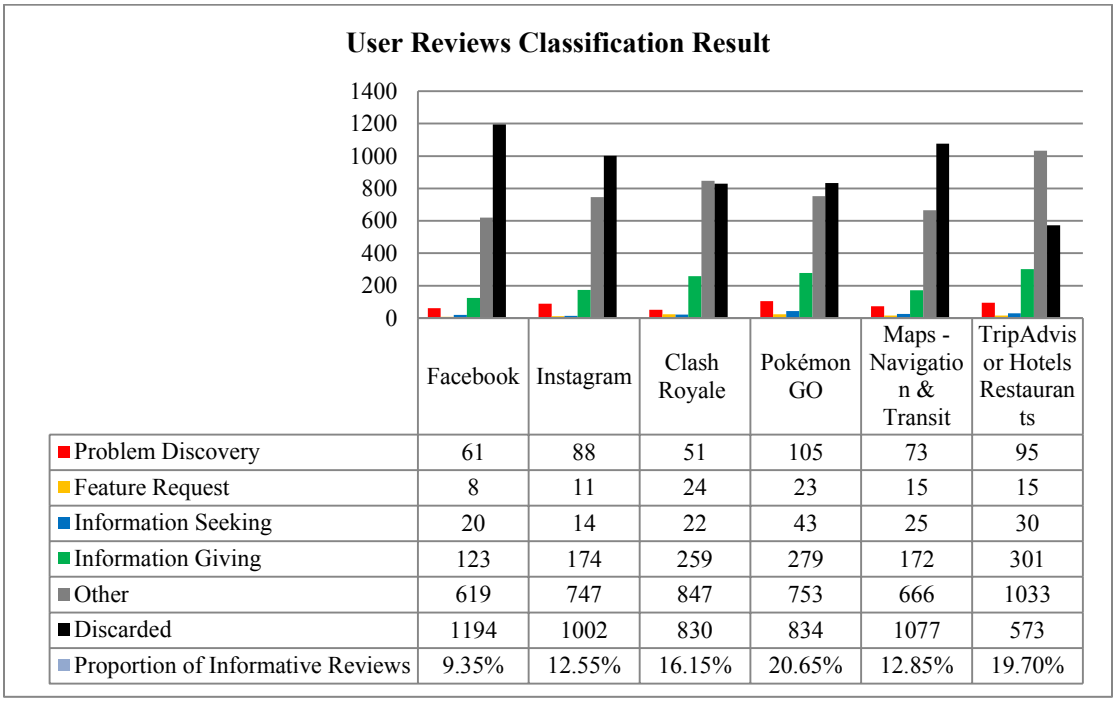
$$Ratio_{informative} = 1 - \frac{Num_{other} + Num_{discarded}}{Num_{total}} \qquad \text{(Formula 7.1)}$$

The number of uninformative reviews was the number of the "other" reviews (Num_{other}) added with the number of the "discarded" reviews ($Num_{discarded}$). The proportion of informative reviews equaled one minus the number of uninformative reviews divided by the number of total reviews (Num_{total}).

The user reviews of six Android mobile applications were collected by the data collector for analysis. The six apps were Facebook (social app), Instagram (social app), Clash Royale (strategy game), Pokémon GO (adventure game), Maps - Navigation & Transit (travel and local app) and TripAdvisor Hotels Restaurants (travel and local app). These mobile applications were mainly chosen because they covered three different application categories including social app, game, and travel app. Users of different categories of apps might belong to different age groups, have different requirements for the apps, and write their reviews in different styles. The reviews of different apps might also contain diverse vocabularies and describe different features. Thus, different apps were selected to evaluate whether the informative rates of app reviews for ARAM would vary for different kinds of apps. Additionally, these six apps were popular applications that had large datasets of user reviews available for analysis.

Two thousand reviews for each app including Facebook, Instagram, Clash Royale, Pokémon GO, Maps - Navigation & Transit, and TripAdvisor Hotels Restaurants were crawled on May 11th, 2017. Figure 7.4 presented the classification results of these reviews by the user review classifier of ARAM. The numbers in the "Problem Discovery" row were the numbers of reviews that contained at least one sentence classified as the "problem discovery" type. Similarly, for the 2000 reviews of the app Facebook, 8 reviews contained at least one sentence classified as the "feature request" type, 20 reviews contained at least one sentence classified as the "information seeking" type, and 123 reviews contained at least one sentence classified as the "information

giving" type. The numbers in the "Other" row and the "Discarded" row were the numbers of the "other" reviews and the "discarded" reviews respectively. The numbers in the "Proportion of Informative Reviews" row were the proportion of the informative reviews calculated by Formula 7.1.



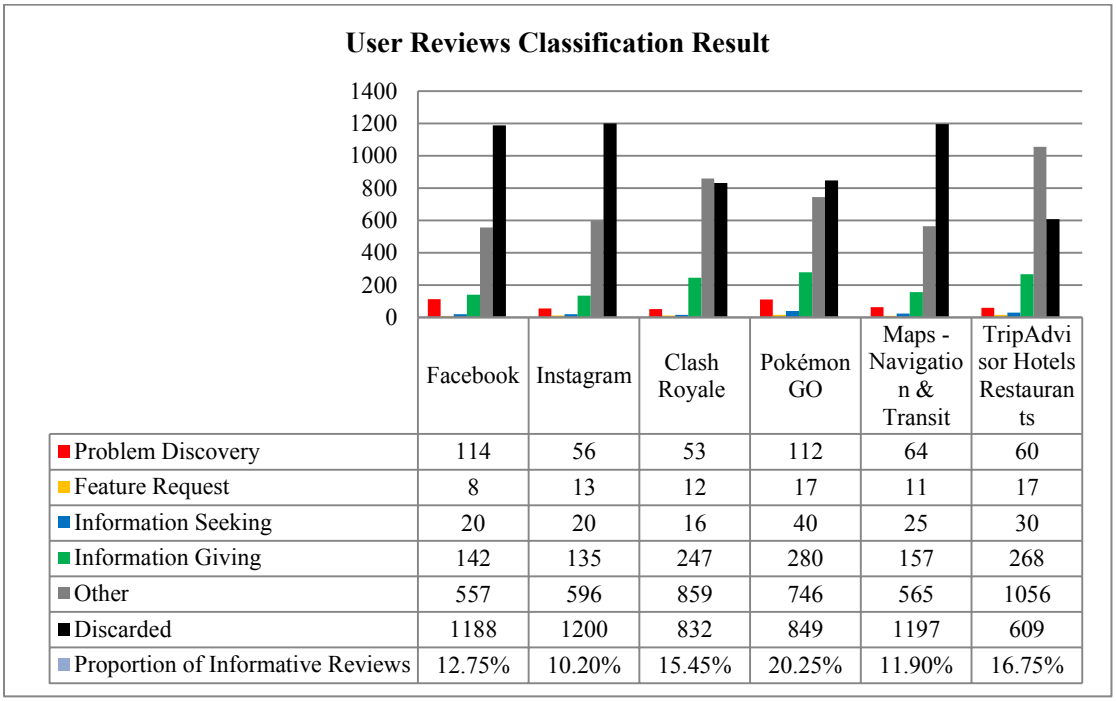
	Facebook	Instagram	Clash Royale	Pokémon GO	Maps - Navigation & Transit	TripAdvisor Hotels Restaurants
Problem Discovery	61	88	51	105	73	95
Feature Request	8	11	24	23	15	15
Information Seeking	20	14	22	43	25	30
Information Giving	123	174	259	279	172	301
Other	619	747	847	753	666	1033
Discarded	1194	1002	830	834	1077	573
Proportion of Informative Reviews	9.35%	12.55%	16.15%	20.65%	12.85%	19.70%

Figure 7.4. Classification results of the reviews crawled on May 11th, 2017.

The average proportion of the informative reviews of the six apps was 15.21%. The proportion of the informative reviews of the social apps Facebook and Instagram were lower than the other four apps. It could be noticed that more than half of their reviews were the "discarded" reviews whose contents were less than four words.

Figure 7.5 showed another test based on the dataset of two thousand reviews for each of the six apps crawled on November 1st, 2017, which was nearly five months later after the first test.

The results of the second test based on the reviews crawled on November 1st, 2017 were consistent with the results of the first test. The average proportion of the informative reviews of the six apps was 14.55% according to the second test.



	Facebook	Instagram	Clash Royale	Pokémon GO	Maps - Navigation & Transit	TripAdvisor Hotels Restaurants
Problem Discovery	114	56	53	112	64	60
Feature Request	8	13	12	17	11	17
Information Seeking	20	20	16	40	25	30
Information Giving	142	135	247	280	157	268
Other	557	596	859	746	565	1056
Discarded	1188	1200	832	849	1197	609
Proportion of Informative Reviews	12.75%	10.20%	15.45%	20.25%	11.90%	16.75%

Figure 7.5. Classification results of the reviews crawled on November 1st, 2017.

Both tests showed that the proportion of the informative reviews classified by ARAM was around 15%. It meant that if 1000 reviews were processed by ARAM, only about 150 reviews could pass the user review classifier, and be further analyzed by the keyphrase extractor and the sentiment analyzer. The keyphrase extractor was supposed to process a large number of reviews for text summarization. The results produced by the keyphrase extractor would be reasonably unacceptable when the number of reviews is not enough. Thus, ARAM was not suitable for analyzing mobile applications that did not receive many reviews in the app store.

7.3. Evaluation of the keyphrase extraction result

As illustrated in Section 6.4, three different methods including the term frequency method, the TextRank method and the DoubleRank method were implemented in the keyphrase extractor of ARAM. In order to test which methods are effective to extract the keyphrases from the app review contents, the author carried out a keyphrase extraction test by designing a questionnaire (see Appendix 1). The questionnaire included 50 reviews of the social app Facebook crawled from Google Play, and the testers were required to manually extract the top ten keyphrases of the reviews. Three testers including two graduate students and one researcher in the software development field took the test, and the 50 reviews in the questionnaire were also processed by the keyphrase extractor using the term frequency method, the TextRank method and the DoubleRank method. Table 7.6 showed the top ten keyphrases analyzed by the testers and the keyphrase extractor.

Term Frequency	TextRank	Double Rank	Tester 1	Tester 2	Tester 3
app	useless app	app	latest updates	recent update	recent update terrible
update	video	news feed	news feed	news feed	crashes
video	notification	new post	video	notification	new useless feature
news feed	news feed	video	new post	video	recommended friends
post	facebook	last update	notification	auto updates	new feeds not loading
facebook	post	notification	update features	posts	videos
phone	friend	facebook	auto update	bug	notification tab
notification	phone	friend	button	friend tab	auto updates
feed	latest update	phone	bugs	back button	draining my battery
bug	recent update	back button	comment section	camera	privacy setting

Table 7.6. Top ten keyphrases extracted by the keyphrase extractor and three testers.

The number of matches between the machine-generated phrases and the human-generated phrases was employed to measure the three different keyphrase extraction methods. As recommended in [Turney, 2000], the precision was defined as the number of matches divided by the number of generated keyphrases. Formally,

$$Precision = Count_{match}/Count_{total} \quad \text{(Formula 7.2)}$$

To calculate the precision of one keyphrase extraction method, the top ten keyphrases extracted by this method formed the set *V*. The three lists of the keyphrases extracted by the testers were firstly lemmatized and named as S_1, S_2 and S_3 respectively. The precision of the keyphrases set *V* based on each human-generated keyphrase list S_1, S_2 and S_3 would be calculated using Formual 7.2. The $Count_{total}$ would be 10, and the $Count_{match}$ would be the sums of the match scores of the ten keyphrases in the set *V*. If the keyphrase *k* in the set *V* exactly matched another keyphrase in the human-generated keyphrase list *S*, the match score would be 1; if the keyphrase *k* was a noun phrase and

only a noun matches another keyphrase in the list *S*, the match score would be 0.7; in other cases, the match score would be 0. Formally,

$$Count_{match} = \sum_{k \in V} MatchScore_k$$

$$MatchScore_k = \begin{cases} 1\text{ , if lematized form exactly matches} \\ 0.7\text{, if a noun matches} \\ 0\text{ , if no nouns matches} \end{cases} \quad \text{(Formulae 7.3)}$$

For instance, as shown in Table 7.6, “new post”, “back button” and “friend” were included in the top ten keyphrases results of the DoubleRank method. To calculate the precision of the results of the DoubleRank method based on the keyphrase list S_1 extracted by the tester 1, the match score for the keyphrase “new post” was 1, since “new post” was also in the keyphrase list S_1. The match score for the keyphrase “back button” was 0.7, since only “button” was found in in the keyphrase list S_1 and the noun matched. The match score for the keyphrase “friend” was 0, because there was no keyphrase containing the noun “friend” in the list S_1.

Table 7.7 showed the precisions of the three keyphrase extraction methods including the term frequency method, the TextRank method and the DoubleRank method.

Precision	Tester 1	Tester 2	Tester 3	Average
Term Frequency	0.61	0.64	0.38	0.543
TextRank	0.54	0.64	0.45	0.543
DoubleRank	0.57	0.61	0.38	0.52

Table 7.7. Precisions of the keyphrase extraction methods.

The agreements between each pair of the results produced by three testers were also calculated using the same criterion of calculating the precision, and the results were listed in Table 7.8.

Agreement	Tester 1	Tester 2	Tester 3	Average
Tester 1	-	0.71	0.48	0.595
Tester 2	0.71	-	0.48	0.595
Tester 3	0.48	0.48	-	0.48

Table 7.8. Agreements of the keyphrase results between testers.

It could be found that the precisions of these three automatic keyphrase extraction methods including the term frequency method, the TextRank method, and the DoubleRank method did not vary greatly, and the precisions were also very close to the

agreements between testers' manual extraction results. Although it might be caused by the test dataset of 50 reviews was too small to reveal the differences among the keyphrase results of the three methods and three testers, it showed that these three automatic keyphrase extraction methods could produce the keyphrase results that were close to human's results. Additionally, although the precision of the DoubleRank method was slightly lower than the precisions of the other two methods, the DoubleRank method was better at extracting multi-word keyphrases than the other two methods. It could be seen by that the keyphrase results of the DoubleRank method included "new post" and "back button" that were not extracted as phrases by the other two methods. Moreover, it could be noticed that the keyphrase results of the three keyphrase extraction methods all include the keyphrase "app" (or "useless app") and the keyphrase "facebook", while the three testers' results did not contain them. "App" and "facebook" were the words mentioned a lot by users in their reviews to refer to the Facebook mobile application as a whole. These two words inevitably had high term frequencies or rank values to be selected into the top ten keyphrases by the automatic keyphrase extraction methods. Testers tended to extract keyphrases that could represent the specific aspects of the app and did not consider the "app" as an aspect. This was one of the major differences found between the results produced by the automatic keyphrase extraction methods and the testers. What's more, it could be seen that the results produced by the Tester 3 included many informative adjectives, and gave keyphrases the ability to imply the issues related to an aspect such as "recent update terrible" and "new useless feature". However, the three automatic keyphrase extraction methods could not produce such "informative" keyphrases. Thus, the retrieval function based on the produced keyphrases as shown in Section 6.6 was added in the report to compensate for this disadvantage. Although the keyphrases extracted by the three automatic keyphrase extraction methods could reveal the important aspects of the app discussed by users such as "news feed", "video" and "post", developers still had no idea what issues were related to these aspects by only reviewing these extracted keyphrases. Thus, the keyphrases were used as indexes for retrieving the reviews from which they were extracted, and developers could conveniently study the reviews concerning the keyphrases that they were interested in.

It was also found from the test that the average time for testers to extract the top ten keyphrases from 50 reviews were 32 minutes (24 minutes for Tester 1, 46 minutes for Tester 2, and 25 minutes for Tester 3). That was the reason why the test dataset was designed to be small because a large number of reviews would cost testers much times to extract the top ten keyphrases and could threaten the accuracy of the test. Additionally, three testers believed that their extracted top ten keyphrases can represent 70% - 80% of the information contained in the reviews, which proved the ability of keyphrase extraction as an effective approach for text summarization. Moreover, three

testers all claimed that they considered the term frequency as the main criteria for extracting the keyphrases. Tester 1 and Tester 2 extracted most frequently mentioned aspects of the app as keyphrases, while Tester 3 focused more on extracting "the frequent complaints".

In conclusion, according to the test results of a small set of 50 reviews, the term frequency method, the TextRank method, and the DoubleRank method all performed well in the keyphrase extraction task of app review contents. A potential drawback that the keyphrases produced by these three methods rarely contained noun phrases with "informative" adjectives was also remedied by the retrieval function based on the automatic extracted keyphrases. Thus, the keyphrases results produced by the keyphrase extractor of ARAM could not only represent the most relevant aspects of the app that were discussed in the reviews, but also act as indexes for developers to easily retrieve the reviews concerning the keyphrases.

7.4. Evaluation of the sentiment analysis result

As explained in Section 6.5, the sentiment analyzer of ARAM represented the sentiment for a keyphrase by the sentiment for the sentence from which the keyphrase was extracted. A sentiment analysis test was carried out to test whether this approach was accurate. A questionnaire (see Appendix 2) included 50 reviews of the adventure game app "Pokémon GO" crawled from Google Play. The 50 reviews were divided into two sets. The first set of 25 reviews all contained the keyphrase "pokemon", and the second set of 25 reviews all contained the keyphrase "update". The testers were required to judge the author's sentiment for the keyphrase in each review was positive, negative or neutral. Four testers including three graduate students and one researcher in the software development field took the test, and the 50 reviews in the questionnaire were also processed by the sentiment analyzer to calculate the sentiment polarities of the keyphrase in each review. Moreover, the review rate (from one to five stars) of each review was also used to represent the sentiments for the keyphrase in the review as a control group in the test. The review rates could provide the information about users' overall sentiments towards the app [Gu and Kim, 2015], and they could be seen as a document-level sentiment of the review contents. The result of using the document-level sentiment showed by the review rates to represent the sentiment towards the keyphrase was used as a baseline in the test. If the review rate was three stars, the user's sentiment towards the keyphrase in this review was considered as neutral; if the review rate was one star or two stars, the user's sentiment towards the keyphrase was seen as negative; if the review rate was four stars or five stars, the user's sentiment towards the keyphrase was regarded as positive.

Appendix 3 listed the test results of the sentiment polarities (positive, negative, and neutral) of the keyphrase in each review judged respectively by the four testers, the

sentiment analyzer, and the review rates. The accuracy of a sentiment analysis system depended on how well it agreed with human judgments[12]. Thus, the agreement was defined as the number of matched sentiment polarities divided by the number of the total test reviews. Formally,

$$Agreement = Count_{match}/Count_{total} \quad \text{(Formula 7.4)}$$

Table 7.9 and Table 7.10 showed the inter-rater agreements of the sentiment polarities for the first test set of 25 reviews containing the keyphrase "pokemon" and the second test set of 25 reviews containing the keyphrase "update". The average inter-rater agreement was 73.3% and 80% for the two sets.

Agreement	Tester 1	Tester 2	Tester 3	Tester 4
Tester 1	-	0.68	0.68	0.8
Tester 2	0.68	-	0.84	0.72
Tester 3	0.68	0.84	-	0.68
Tester 4	0.8	0.72	0.68	-
Average	0.72	0.747	0.733	0.733

Table 7.9. Inter-rater agreements of the sentiment polarities for the first test set concerning the keyphrase "pokemon".

Agreement	Tester 1	Tester 2	Tester 3	Tester 4
Tester 1	-	0.8	0.68	0.88
Tester 2	0.8	-	0.8	0.84
Tester 3	0.68	0.8	-	0.8
Tester 4	0.88	0.84	0.8	-
Average	0.787	0.813	0.76	0.84

Table 7.10. Inter-rater agreements of the sentiment polarities for the second test set concerning the keyphrase "update".

As presented in Table 7.11 and Table 7.12. The average agreement between the results of the Sentiment Analyzer and the four testers reached 65% and 67% for the two test sets. The average agreement between the results of the review rates and the four testers were 52% and 57% for the two sets.

12 https://en.wikipedia.org/wiki/Sentiment_analysis. Access date: 29.11.2017

Agreement	Tester 1	Tester 2	Tester 3	Tester 4	Average
Sentiment Analyzer	0.64	0.64	0.68	0.64	0.65
Review rate	0.48	0.52	0.64	0.44	0.52

Table 7.11. The agreements between the results of the Sentiment Analyzer and the review rates, and the results of the four testers for the first test set.

Agreement	Tester 1	Tester 2	Tester 3	Tester 4	Average
Sentiment Analyzer	0.68	0.64	0.64	0.72	0.67
Review rate	0.6	0.52	0.6	0.56	0.57

Table 7.12. The agreements between the results of the Sentiment Analyzer and the review rates, and the results of the four testers for the second test set.

The test results showed the average inter-rater agreement was 76.65% for the aspect-based keyphrase sentiment analysis task for the keyphrase “pokemon” and the keyphrase “update”. Thus, the sentiment analyzer that achieved the average agreement of 66% with the raters in calculating the sentiment polarities could be considered to perform nearly as well as humans. The sentiment analyzer also outperformed using the review rates to represent the sentiments for keyphrases, which had the average agreement of 54.5% with the raters.

In summary, the Sentiment Analyzer that employed the Stanford CoreNLP Sentiment Analyzer [Socher et al., 2013] performed well in analyzing the app review data. The results were also accurate in most cases to represent the sentiment for a keyphrase by the sentiment for the sentence from which the keyphrase was extracted.

8. Results and discussion

8.1. Discussion

To discuss how users of ARAM could benefit from its result, the author manually analyzed a summary report produced by ARAM. The summary report was generated by running ARAM to analyze 4480 reviews of the social app Instagram published during 15th and 21st, May 2017 in Google Play. Instagram had no new release during 15th and 21st, May 2017, and all these reviews should comment on the version 10.21.0 of Instagram. Additionally, these reviews were not crawled by the "newest" order but by the order of "helpfulness" provided by Google Play. As shown in Figure 8.1, Google Play provided three different sorting methods of the user reviews (e.g., "newest", "rating", and "helpfulness"). The data collector of ARAM collected user reviews by the "newest" chronological order by default. However, it could also be set to collect the reviews by the order of "helpfulness" or "rating". The mechanism how Google Play calculated the "helpfulness" of one review was unclear. However, among the 4480 reviews crawled by the "helpfulness" order, only 31 reviews' contents were less than four words and discarded. Thus, the ARAM's data using rate (the ratio of informative reviews) of the reviews crawled by the "helpfulness" order was much higher than the ARAM's data using rate of the reviews crawled by the "newest" order. Therefore, collecting user reviews by the "helpfulness" as the data source could be an effective way to improve the efficiency of ARAM. However, the review dates of the reviews crawled by the "helpfulness" order should be checked to make sure that all the reviews had commented on the same version of the app.

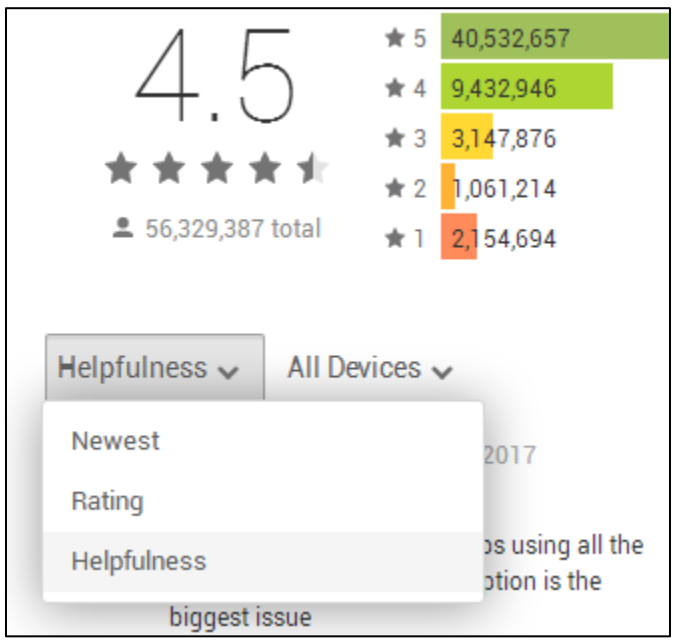
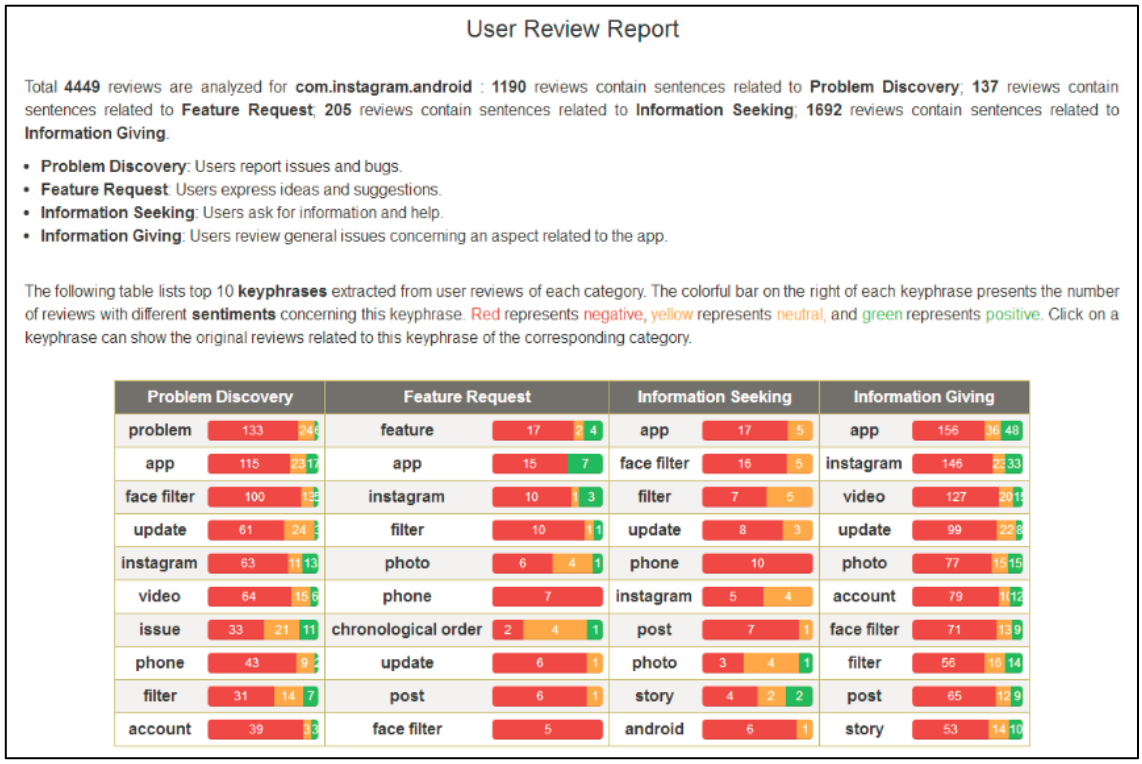


Figure 8.1. Three different sorting methods of user reviews provided by Google Play.

Figure 8.2 presented the summary report of the 4449 reviews whose contents were no less than four words. The author would select one keyphrase in each category of the informative reviews (e.g., “information giving”, “information seeking”, “feature request” and “problem discovery”) to illustrate how application developers (the main target user of ARAM) could benefit from the summary report by exploiting the useful information for arranging maintenance activities.



User Review Report

Total **4449** reviews are analyzed for **com.instagram.android** : **1190** reviews contain sentences related to **Problem Discovery**; **137** reviews contain sentences related to **Feature Request**; **205** reviews contain sentences related to **Information Seeking**; **1692** reviews contain sentences related to **Information Giving**.

- **Problem Discovery**: Users report issues and bugs.
- **Feature Request**: Users express ideas and suggestions.
- **Information Seeking**: Users ask for information and help.
- **Information Giving**: Users review general issues concerning an aspect related to the app.

The following table lists top 10 **keyphrases** extracted from user reviews of each category. The colorful bar on the right of each keyphrase presents the number of reviews with different **sentiments** concerning this keyphrase. Red represents negative, yellow represents neutral, and green represents positive. Click on a keyphrase can show the original reviews related to this keyphrase of the corresponding category.

Problem Discovery		Feature Request		Information Seeking		Information Giving	
problem	133 24 6	feature	17 2 4	app	17 5	app	156 36 48
app	115 23 17	app	15 7	face filter	16 5	instagram	146 23 33
face filter	100 13 3	instagram	10 1 3	filter	7 5	video	127 20 11
update	61 24 3	filter	10 1 1	update	8 3	update	99 22 8
instagram	63 11 13	photo	6 4 1	phone	10	photo	77 15 15
video	64 15 6	phone	7	instagram	5 4	account	79 10 12
issue	33 21 11	chronological order	2 4 1	post	7 1	face filter	71 13 9
phone	43 9 2	update	6 1	photo	3 4 1	filter	56 16 14
filter	31 14 7	post	6 1	story	4 2 2	post	65 12 9
account	39 9 3	face filter	5	android	6 1	story	53 14 10

Figure 8.2. The summary report of the Instagram reviews produced by ARAM.

The keyphrase “face filter” was mentioned frequently in the reviews classified as the “problem discovery” category and its sentiment polarities were mostly negative. By clicking the keyphrase “face filter” in the “Problem Discovery” column of the table, the 118 reviews from which the keyphrase “face filter” were extracted were presented as shown in Figure 8.3. After manually reading the 118 reviews, it was found that 117 reviews described the issues and bugs that the face filter could not work on their devices. The only exception was one review complaining that Instagram did not fix the issues concerning (Samsung Galaxy) S7 phones by writing “Instead of fixing issues for a lot of S7 users they copy Snapchat with face filters because that's more important”. Additionally, 56 reviews mentioned that the users had updated or reinstalled the app but the new “face filter” function still could not work on their devices (e.g., *“I am unable to use the face filters, the option doesn't even show and I just updated the app”*, *“Also I updated the app and the face filter ain't working just worked for 1 day!”*, and *“I have the new update which is with face filters however they are not coming up or not*

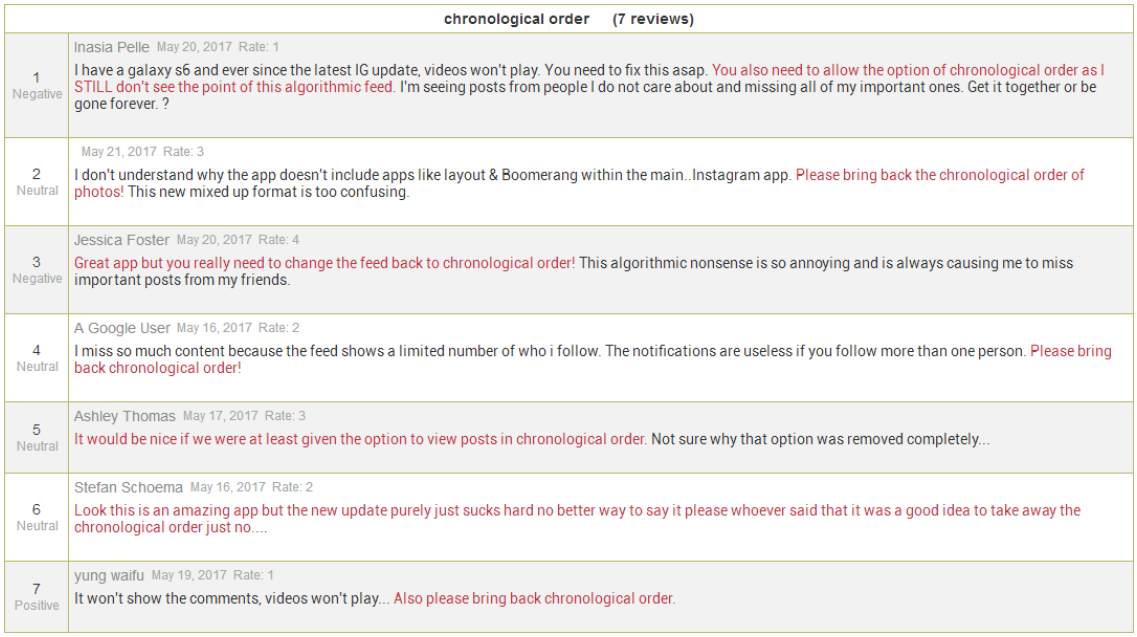
working properly please fix"). Some reviews also mentioned the devices on which the face filter could not work, and the devices included the Samsung Galaxy phones (S8+, J1, E5, A7, Express 3, and J5), the Redmi phone and the Huawei Y360.

face filter (118 reviews)	
1 Negative	Joe-Free trenard May 19, 2017 Rate: 2 App WAS running great but I have issues after this latest update on my galaxy s8+. First I cannot play any videos. When a video appears on my timeline, all I see is the loading image and nothing ever plays, that is really annoying to sit and watch. Also would like to see my timeline in chronological order like I can on facebook, your parent company.. lastly you launched new face filters that also is not working or appearing on my device.. you guys are slowly ruining the app
2 Very negative	May 18, 2017 Rate: 2 Hey Instagram. I just wondering why the face filter is still not working after updated for a few and i'm even reinstalling it for few times and still not pop up at the corner. A nice app but spoilt a little. I hope Instagram could find a way and fix this issue ASAP for a better rate. I guess not only me and my friends hv this problem,maybe other users outside there might hv the same problem. So fix it okayy.
3 Negative	yo girl May 19, 2017 Rate: 5 THE best thing everrrr... But my face filter doesn't work,still.. I tried updating the app then also tried switching off my phone.. Nothing worked??
4 Negative	May 19, 2017 Rate: 3 The live video and face filters DO NOT WORK! I can't update it either. It won't let me. ;-; I have tried multiple times but they do not show up anywhere! I have a Samsung Galaxy J1 and live video/ face filters don't work at all! +-+ PLEASE FIX!
5 Negative	Anandana Manglani May 19, 2017 Rate: 3 I'm not able to use the face filters.... I got the message saying that it's available but can't see any option for face filters.... Other than that everything's good... Can I please know how to get the face filters
6 Negative	Rasyadan Alghazian May 18, 2017 Rate: 1 I already updated it, I use the face filter once and it didnt showing again on my instagram camera.. and also i can't see people (even mine own profile) old pic, its limited until about 15 photos on every profiles . Thats suck! ???
7 Negative	prudence murorua May 16, 2017 Rate: 3 I am unable to use the face filters,the option doesn't even show and I just updated the app.....please fix it
8 Negative	May 19, 2017 Rate: 1 I did updated It but The new face filter doesn't work..

Figure 8.3. A part of the reviews that contain the keyphrase "face fileter" in the "problem discovery" category.

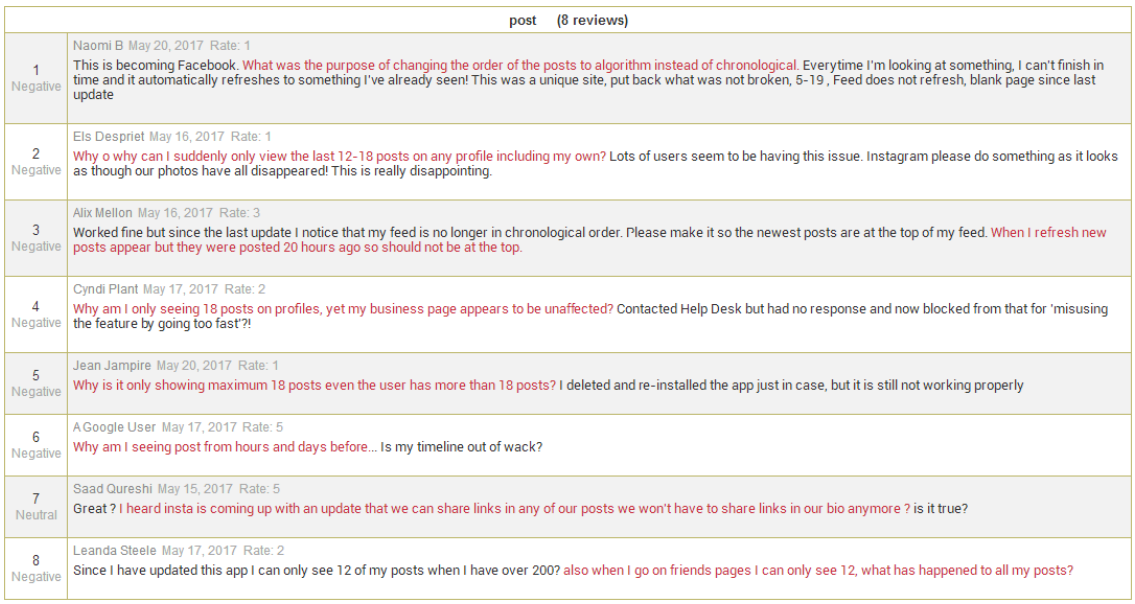
Thus, the evaluative maintenance activities might be planned to test whether the face filter function could work properly after installing the new updates on most devices. Then it should be further tested whether the face filter function could work on the devices on which the users had trouble in using the face filter function. If the test found that the face filter function could not work on some devices or even most devices, further corrective maintenance activities should be planned and carried out.

The keyphrase "chronological order" was selected to analyze in the "feature request" category. As presented in Figure 8.4, 7 reviews in the "feature request" category mentioned the "chronological order". The users expressed their requirement in the reviews to have the option to view photos / posts by the chronological order. Thus, developers might consider adding different options to sort the photos / posts and allow users to choose their favorite sorting methods. This could involve the enhancive maintenance activities.



chronological order (7 reviews)	
1 Negative	Inasia Pelle May 20, 2017 Rate: 1 I have a galaxy s6 and ever since the latest IG update, videos won't play. You need to fix this asap. You also need to allow the option of chronological order as I STILL don't see the point of this algorithmic feed. I'm seeing posts from people I do not care about and missing all of my important ones. Get it together or be gone forever. ?
2 Neutral	May 21, 2017 Rate: 3 I don't understand why the app doesn't include apps like layout & Boomerang within the main..Instagram app. Please bring back the chronological order of photos! This new mixed up format is too confusing.
3 Negative	Jessica Foster May 20, 2017 Rate: 4 Great app but you really need to change the feed back to chronological order! This algorithmic nonsense is so annoying and is always causing me to miss important posts from my friends.
4 Neutral	A Google User May 16, 2017 Rate: 2 I miss so much content because the feed shows a limited number of who i follow. The notifications are useless if you follow more than one person. Please bring back chronological order!
5 Neutral	Ashley Thomas May 17, 2017 Rate: 3 It would be nice if we were at least given the option to view posts in chronological order. Not sure why that option was removed completely...
6 Neutral	Stefan Schoema May 16, 2017 Rate: 2 Look this is an amazing app but the new update purely just sucks hard no better way to say it please whoever said that it was a good idea to take away the chronological order just no....
7 Positive	yung waifu May 19, 2017 Rate: 1 It won't show the comments, videos won't play... Also please bring back chronological order.

Figure 8.4. Reviews that contain the keyphrase "chronological order" in the "feature request" category.

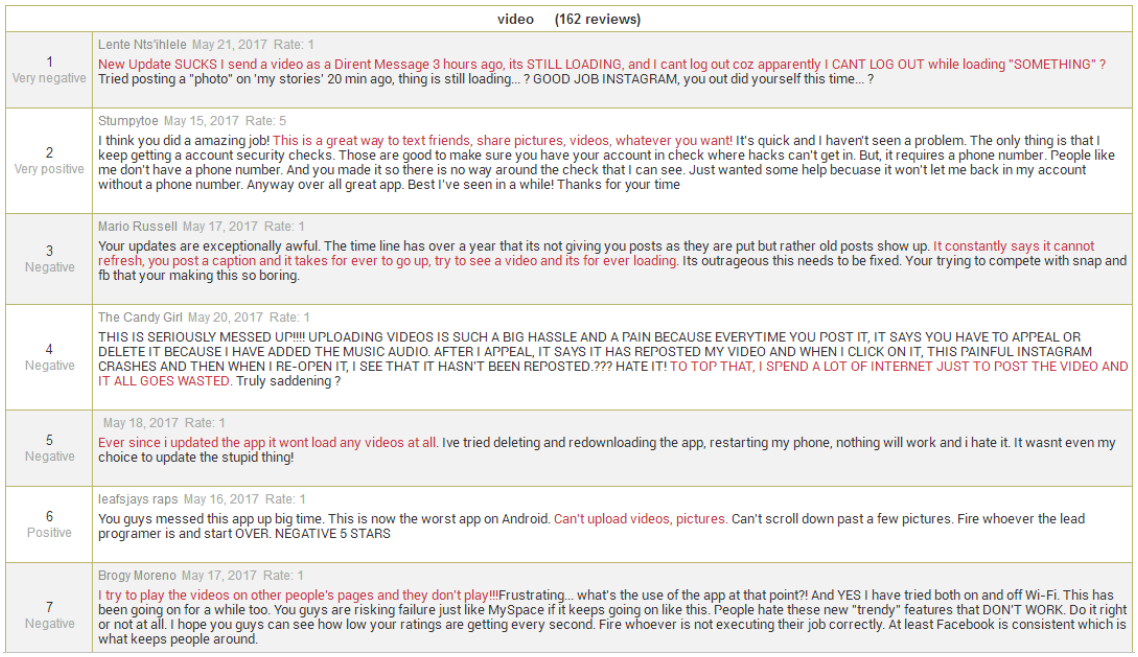


post (8 reviews)	
1 Negative	Naomi B May 20, 2017 Rate: 1 This is becoming Facebook. What was the purpose of changing the order of the posts to algorithm instead of chronological. Everytime I'm looking at something, I can't finish in time and it automatically refreshes to something I've already seen! This was a unique site, put back what was not broken, 5-19 , Feed does not refresh, blank page since last update
2 Negative	Els Despriet May 16, 2017 Rate: 1 Why o why can I suddenly only view the last 12-18 posts on any profile including my own? Lots of users seem to be having this issue. Instagram please do something as it looks as though our photos have all disappeared! This is really disappointing.
3 Negative	Alix Mellon May 16, 2017 Rate: 3 Worked fine but since the last update I notice that my feed is no longer in chronological order. Please make it so the newest posts are at the top of my feed. When I refresh new posts appear but they were posted 20 hours ago so should not be at the top.
4 Negative	Cyndi Plant May 17, 2017 Rate: 2 Why am I only seeing 18 posts on profiles, yet my business page appears to be unaffected? Contacted Help Desk but had no response and now blocked from that for 'misusing the feature by going too fast'?!
5 Negative	Jean Jampire May 20, 2017 Rate: 1 Why is it only showing maximum 18 posts even the user has more than 18 posts? I deleted and re-installed the app just in case, but it is still not working properly
6 Negative	A Google User May 17, 2017 Rate: 5 Why am I seeing post from hours and days before... Is my timeline out of wack?
7 Neutral	Saad Qureshi May 15, 2017 Rate: 5 Great ? I heard insta is coming up with an update that we can share links in any of our posts we won't have to share links in our bio anymore ? is it true?
8 Negative	Leanda Steele May 17, 2017 Rate: 2 Since I have updated this app I can only see 12 of my posts when I have over 200? also when I go on friends pages I can only see 12, what has happened to all my posts?

Figure 8.5. Reviews that contain the keyphrase "post" in the "information seeking" category.

The keyphrase "post" was chosen to analyze in the "information seeking" category. There were 8 reviews related to the keyphrase "post" in the "information seeking" category as shown in Figure 8.5. The sentence *"What was the purpose of changing the order of the posts to algorithm instead of chronological"* in the first review was complaining the new sorting order of the posts. The issues mentioned in the third review (*"When I refresh new posts appear but they were posted 20 hours ago so should not be at the top"*) and the sixth review (*"Why am I seeing post from hours and days*

before") were actually also caused by that the chronological order of the posts used in the old versions had been changed into the new sorting order. The users did not notice this change and thought that something might went wrong that the posts were not in the chronological order. Thus, the customer service should notify this change to the users, which could be counted as the consultive maintenance activity. Moreover, for the issues mentioned in review 2, 4, 5 and 8, these reviews all described a possible bug that the users could only view 12 or 18 posts in the profiles, which required the evaluative maintenance activities for a diagnostic test and possible corrective maintenance activities to fix the bug.



video (162 reviews)	
1 Very negative	Lente Nts'ihlele May 21, 2017 Rate: 1 New Update SUCKS I send a video as a Dirent Message 3 hours ago, its STILL LOADING, and I cant log out coz apparently I CANT LOG OUT while loading "SOMETHING" ? Tried posting a "photo" on 'my stories' 20 min ago, thing is still loading... ? GOOD JOB INSTAGRAM, you out did yourself this time... ?
2 Very positive	Stumpytoe May 15, 2017 Rate: 5 I think you did a amazing job! This is a great way to text friends, share pictures, videos, whatever you want! It's quick and I haven't seen a problem. The only thing is that I keep getting a account security checks. Those are good to make sure you have your account in check where hacks can't get in. But, it requires a phone number. People like me don't have a phone number. And you made it so there is no way around the check that I can see. Just wanted some help becuase it won't let me back in my account without a phone number. Anyway over all great app. Best I've seen in a while! Thanks for your time
3 Negative	Mario Russell May 17, 2017 Rate: 1 Your updates are exceptionally awful. The time line has over a year that its not giving you posts as they are put but rather old posts show up. It constantly says it cannot refresh, you post a caption and it takes for ever to go up, try to see a video and its for ever loading. Its outrageous this needs to be fixed. Your trying to compete with snap and fb that your making this so boring.
4 Negative	The Candy Girl May 20, 2017 Rate: 1 THIS IS SERIOUSLY MESSED UP!!!!! UPLOADING VIDEOS IS SUCH A BIG HASSLE AND A PAIN BECAUSE EVERYTIME YOU POST IT, IT SAYS YOU HAVE TO APPEAL OR DELETE IT BECAUSE I HAVE ADDED THE MUSIC AUDIO. AFTER I APPEAL, IT SAYS IT HAS REPOSTED MY VIDEO AND WHEN I CLICK ON IT, THIS PAINFUL INSTAGRAM CRASHES AND THEN WHEN I RE-OPEN IT, I SEE THAT IT HASN'T BEEN REPOSTED.??? HATE IT! TO TOP THAT, I SPEND A LOT OF INTERNET JUST TO POST THE VIDEO AND IT ALL GOES WASTED. Truly saddening ?
5 Negative	May 18, 2017 Rate: 1 Ever since i updated the app it wont load any videos at all. Ive tried deleting and redownloading the app, restarting my phone, nothing will work and i hate it. It wasnt even my choice to update the stupid thing!
6 Positive	leafsjays raps May 16, 2017 Rate: 1 You guys messed this app up big time. This is now the worst app on Android. Can't upload videos, pictures. Can't scroll down past a few pictures. Fire whoever the lead programer is and start OVER. NEGATIVE 5 STARS
7 Negative	Brogy Moreno May 17, 2017 Rate: 1 I try to play the videos on other people's pages and they don't play!!!Frustrating... what's the use of the app at that point?! And YES I have tried both on and off Wi-Fi. This has been going on for a while too. You guys are risking failure just like MySpace if it keeps going on like this. People hate these new "trendy" features that DON'T WORK. Do it right or not at all. I hope you guys can see how low your ratings are getting every second. Fire whoever is not executing their job correctly. At least Facebook is consistent which is what keeps people around.

Figure 8.6. A part of the reviews that contain the keyphrase "video" in the "information giving" category..

The keyphrase "video" was selected in the "information giving" category. After manually went through the 162 reviews as shown in Figure 8.6, it was found that all these reviews described the reviewers' experiences and issues concerning the "video" of Instagram. The mostly discussed issues in these reviews were that the videos could not be loaded (e.g., *"Ever since I updated the app it won't load any videos at all."*, *"Massive issues loading videos, some fail to load completely."*, and *"I can't load videos to watch them even though I'm on the Wi-Fi at home."*) or uploaded (e.g., *"I cannot upload the video, even it full video or trimmed video"*, *"Having tried everything from compressing videos, Instagram modifying apps and even a new phone I still cannot upload a video."*, and *"Can't upload any videos after last update...seriously Instagram."*). Some reviews also complained that loading or uploading the videos was slow (e.g., *"Videos take FOREVER to load and now I can't see all the pictures I have*

posted", and *"I can't post videos I want to share, it either takes a long time or never posts, and the videos are 1 minute long"*), or the videos crashed during play (e.g., *"I recorded the whole video correctly and you instagram KEEP CRASHING on the part when my idol is laughing??"*, and *"Videos will play shortly and then the refresh arrow will pop up and the video stops."*). Thus, these reviews revealed different issues related to the "video" of Instagram, and reading these reviews could actually be counted as an evaluative maintenance activity to gather data concerning a specific aspect of the app. Moreover, the issues found by reading these reviews should also be further tested, which also belonged to the evaluative maintenance activities.

In conclusion, the report produced by ARAM contained the informative reviews for the application maintenance, and these reviews were classified into four different categories (e.g., "information giving", "information seeking", "feature request", and "problem discovery") that contained different types of information that could lead to different categories of maintenance activities. The top ten keyphrases in each category of informative reviews could not only summarize the review contents, but also provide indexing for users to easily retrieve the reviews concerning the keyphrases. If the user reviews were like the scattered pages of a rough draft provided by an unprofessional writer, then ARAM was the editor who reviewed these pages carefully and bound the informative ones into a delicate book with a catalog. The author only took around half an hour to finish reading the whole report, and could have a comprehensive understanding of the different issues, problems, suggestions and ideas provided by the reviewers, which inspired the author to figure out different possible maintenance activities to improve the app. It would take much more time and energy to directly read the 4480 reviews to gain the same information. Moreover, as developers of the app, they were more familiar with the app, and could be more sensitive to the useful information contained in the reviews that might be ignored by the author. Thus, developers could even identify more issues by reading the report produced by ARAM to plan maintenance activities. Therefore, the author would argue that the results produced by ARAM could benefit developers in understanding their app from the users' perspective and plan maintenance activities to meet users' requirements.

8.2. Limitations and possible improvements

The main defect of ARAM was its slow performance speed. As tested in Section 7.1, although the performance speed of ARAM was still acceptable (nearly one second to process a review), it would take a long time to process a large number of reviews. ARAM was actually designed to help developers to filter and summarize large quantities of app reviews. Thus, the performance speed of ARAM needed to be improved.

To improve the performance speed of ARAM, firstly the keyphrase extractor based on the TextRank method or the DoubleRank method needed to be integrated into ARAM to replace the current keyphrase extractor based on the term frequency method. It was originally designed and implemented in the first version of ARAM that users could choose one of these three methods used in the keyphrase extractor based on the strategy pattern[13]. However, the keyphrase extractor of the first version of ARAM only extracted the top ten keyphrases and did not store the reviews related to each keyphrase for the retrieval function. The retrieval function was added in the second version of ARAM when the author noticed that the top ten keyphrases extracted by the keyphrase extractor could not provide detailed information of the issues related to each keyphrase that generally represented an aspect of the app. Thus, the author spent much time in changing the data structure and implementing the retrieval function in the keyphrase extractor based on the term frequency method. Due to the time constraints of this study, currently the retrieval function was not implemented in the keyphrase extractor based on the TextRank method and the DoubleRank method, and thus they were not integrated into ARAM. As shown in Section 7.1, the performance speeds of the TextRank method and the DoubleRank method vastly outperformed the term frequency method in the implementation of the keyphrase extractor. Thus, by integrating the keyphrase extractor based on the TextRank method or the DoubleRank method into ARAM in the future work would improve the performance speed of ARAM. Additionally, the results of the sentiment analyzer were not informative. The sentiment analyzer was designed to sort the keyphrases by the ratios of negative reviews they received, and thus the developers could focus more on the keyphrases that receive most negative reviews. However, as shown in Figure 8.2, the sentiment polarities of the most keyphrases were mostly negative. Thus, developers might still need to check all the keyphrases, and the results of the sentiment analyzer were not really helpful. Therefore, the sentiment analyzer could be considered to be removed from ARAM to improve the performance speed. Moreover, the code could also be optimized in the future version of ARAM to improve the performance.

Another limitation of ARAM was that it was not suitable for analyzing apps that only received a small number of reviews. As mentioned in Section 7.2, the proportion of the informative reviews classified by ARAM was low (around 15%), and the keyphrase extractor that only processed the informative reviews required enough inputs to produce an acceptable result. Generally, ARAM could not generate a nice report when the number of the input reviews was less than a thousand. However, for the applications that only received hundreds of reviews during one version, developers could directly read all the reviews instead of using a tool to summarize the reviews in

13 https://en.wikipedia.org/wiki/Strategy_pattern . Access date: 29.11.2017

practice. It was also possible to only utilize the user review classifier of ARAM to filter out the informative reviews for manual inspection. Normally, if there were more than two thousand reviews as input, ARAM could produce a reasonably good summary report.

9. Conclusions

This thesis proposed a mobile application user review analysis system for the maintenance purpose named ARAM. ARAM was a comprehensive system that started from collecting app reviews from Google Play and ended by producing a summary report for the users. It was composed of five components including the data collector, the user review classifier, the keyphrase extractor, the sentiment analyzer, and the visualization component.

The user review classifier employed ARdoc [Panichella et al., 2016] to classify user review sentences based on the taxonomy presented by Panichella et al. [2015]. To verify the information contained in the four types of informative review sentences (e.g., "information giving", "information seeking", "feature request" and "problem discovery") could contribute to planning application maintenance activities, the author carried out a systematic mapping between the four informative types of user review sentences [Panichella et al., 2015] with twelve types of maintenance activities [Chapin et al., 2001] in Section 3.5. The analysis and discussion of the four types of informative review sentences and their implication for maintenance activities form the rationale of the ARAM implementation, and is an important contribution in this study.

The keyphrase extractor was implemented using three different algorithms including the statistics method term frequency, the graph-based ranking method TextRank [Mihalcea and Tarau, 2004], and a self-designed algorithm named DoubleRank that combined TextRank with term frequency. TextRank [Mihalcea and Tarau, 2004] had rarely been used in analyzing short informal texts, and this study made some adjustments of the post-processing phase of the TextRank algorithm. The test in Section 7.3 showed that the adjusted TextRank method and the DoubleRank method performed well in the keyphrase extraction task of application user reviews.

The Stanford CoreNLP Sentiment Analyzer [Socher et al., 2013] was employed in the sentiment analyzer to perform sentence-based sentiment analysis. The test in Section 7.4 validated the assumption that for analyzing the sentiments towards the keyphrases extracted from the app review contents, it was accurate to represent the sentiment for a keyphrase by the sentiment for the sentence from which the keyphrase was extracted.

The visualization component would produce a summary report on the local machine after finishing analyzing the reviews. The author manually went through a summary report generated by running ARAM to analyze the 4480 reviews of the social app Instagram in Section 8.1. It was shown that the report produced by ARAM was concise and informative, and could benefit developers in quickly figuring out the key issues discussed in the reviews and further planning corresponding maintenance activities.

In conclusion, ARAM was the first system that applied the techniques of keyphrase extraction and sentiment analysis to analyze the mobile application user reviews for maintenance purpose. It was a promising tool for helping developers to easily gain useful information from app reviews for planning application maintenance activities.

References

[Baccianella et al., 2010] Baccianella S, Esuli A, Sebastiani F. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining[C]//LREC. 2010, 10: 2200-2204.

[Bennett and Rajlich, 2000] Bennett K H, Rajlich V T. Software maintenance and evolution: a roadmap[C]//Proceedings of the Conference on the Future of Software Engineering. ACM, 2000: 73-87.

[Blei et al., 2003] Blei D M, Ng A Y, Jordan M I. Latent dirichlet allocation[J]. Journal of machine Learning research, 2003, 3(Jan): 993-1022.

[Bollen et al., 2011] Bollen J, Mao H, Zeng X. Twitter mood predicts the stock market[J]. Journal of computational science, 2011, 2(1): 1-8.

[Brin and Page, 2012] Brin S, Page L. Reprint of: The anatomy of a large-scale hypertextual web search engine[J]. Computer networks, 2012, 56(18): 3825-3833.

[Chapin et al., 2001] Chapin N, Hale J E, Khan K M, et al. Types of software evolution and software maintenance[J]. Journal of Software: Evolution and Process, 2001, 13(1): 3-30.

[Chen and Manning, 2014] Chen D, Manning C D. A Fast and Accurate Dependency Parser using Neural Networks[C]//EMNLP. 2014: 740-750.

[Chen et al., 2014] Chen N, Lin J, Hoi S C H, et al. AR-miner: mining informative reviews for developers from mobile app marketplace[C]//Proceedings of the 36th International Conference on Software Engineering. ACM, 2014: 767-778.

[Chevalier and Mayzlin, 2006] Chevalier J A, Mayzlin D. The effect of word of mouth on sales: Online book reviews[J]. Journal of marketing research, 2006, 43(3): 345-354.

[Ercan and Cicekli, 2007] Ercan G, Cicekli I. Using lexical chains for keyword extraction[J]. Information Processing & Management, 2007, 43(6): 1705-1714.

[Flanagin et al., 2014] Flanagin A J, Metzger M J, Pure R, et al. Mitigating risk in ecommerce transactions: perceptions of information credibility and the role of user-generated ratings in product quality and purchase intention[J]. Electronic Commerce Research, 2014, 14(1): 1-23.

[Frank et al., 1999] Frank E, Paynter G W, Witten I H, et al. Domain-specific keyphrase extraction[C]//16th International Joint Conference on Artificial Intelligence (IJCAI 99). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, 2: 668-673.

[Fu et al., 2013] Fu B, Lin J, Li L, et al. Why people hate your app: Making sense of user feedback in a mobile app store[C]//Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013: 1276-1284.

[Gollapalli and Caragea, 2014] Gollapalli S D, Caragea C. Extracting Keyphrases from Research Papers Using Citation Networks[C]//AAAI. 2014: 1629-1635.

[Grineva et al., 2009] Grineva M, Grinev M, Lizorkin D. Extracting key terms from noisy and multitheme documents[C]//Proceedings of the 18th international conference on World wide web. ACM, 2009: 661-670.

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

[Guzman and Maalej, 2014] Guzman E, Maalej W. How do users like this feature? a fine grained sentiment analysis of app reviews[C]//Requirements Engineering Conference (RE), 2014 IEEE 22nd International. IEEE, 2014: 153-162.

[Harman et al., 2012] Harman M, Jia Y, Zhang Y. App store mining and analysis: MSR for app stores[C]//Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on. IEEE, 2012: 108-111.

[Hasan and Ng, 2010] Hasan K S, Ng V. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art[C]//Proceedings of the 23rd International Conference on Computational Linguistics: Posters. Association for Computational Linguistics, 2010: 365-373.

[Hasan and Ng, 2014] Hasan K S, Ng V. Automatic Keyphrase Extraction: A Survey of the State of the Art[C]//ACL (1). 2014: 1262-1273.

[Hatzivassiloglou and McKeown, 1997] Hatzivassiloglou V, McKeown K R. Predicting the semantic orientation of adjectives[C]//Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 1997: 174-181.

[Heeager and Rose, 2015] Heeager L T, Rose J. Optimising agile development practices for the maintenance operation: nine heuristics[J]. Empirical Software Engineering, 2015, 20(6): 1762-1784.

[Hoon et al., 2012] Hoon L, Vasa R, Schneider J G, et al. A preliminary analysis of vocabulary in mobile app user reviews[C]//Proceedings of the 24th Australian Computer-Human Interaction Conference. ACM, 2012: 245-248.

[Hulth, 2003] Hulth A. Improved automatic keyword extraction given more linguistic knowledge[C]//Proceedings of the 2003 conference on Empirical methods in natural language processing. Association for Computational Linguistics, 2003: 216-223.

[Hunt et al., 2008] Hunt B, Turner B, McRitchie K. Software maintenance implications on cost and schedule[C]//aerospace conference, 2008 IEEE. IEEE, 2008: 1-6.

[Iacob et al., 2013] Iacob C, Harrison R, Faily S. Online reviews as first class artifacts in mobile app development[C]//International Conference on Mobile Computing, Applications, and Services. Springer, Cham, 2013: 47-53.

[IEEE, 1998] Software Engineering Standards Committee. IEEE Standard for Software Maintenance[J]. IEEE Std, 1998: 1219-1998.

[Inukollu et al., 2014] Inukollu V N, Keshamoni D D, Kang T, et al. Factors influencing quality of mobile apps: Role of mobile app development life cycle[J]. arXiv preprint arXiv:1410.4537, 2014.

[Jivani, 2011] Jivani A G. A comparative study of stemming algorithms[J]. Int. J. Comp. Tech. Appl, 2011, 2(6): 1930-1938.

[Jo and Oh, 2011] Jo Y, Oh A H. Aspect and sentiment unification model for online review analysis[C]//Proceedings of the fourth ACM international conference on Web search and data mining. ACM, 2011: 815-824.

[Khalid et al., 2015] Khalid M, Asif M, Shehzaib U. Towards improving the quality of mobile app reviews[J]. International Journal of Information Technology and Computer Science (IJITCS), 2015, 7(10): 35.

[Korenius et al., 2004] Korenius T, Laurikkala J, Järvelin K, et al. Stemming and lemmatization in the clustering of finnish text documents[C]//Proceedings of the thirteenth ACM international conference on Information and knowledge management. ACM, 2004: 625-633.

[Lee and Kim, 2008] Lee S, Kim H. News keyword extraction for topic tracking[C]//Networked Computing and Advanced Information Management, 2008. NCM'08. Fourth International Conference on. IEEE, 2008, 2: 554-559.

[Lientz and Swanson, 1980] Lientz B P, Swanson E B. Software maintenance management[J]. 1980.

[Liu et al., 2009a] Liu F, Pennell D, Liu F, et al. Unsupervised approaches for automatic keyword extraction using meeting transcripts[C]//Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics. Association for Computational Linguistics, 2009: 620-628.

[Liu et al., 2009b] Liu Z, Li P, Zheng Y, et al. Clustering to find exemplar terms for keyphrase extraction[C]//Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1. Association for Computational Linguistics, 2009: 257-266.

[Liu, 2012] Liu B. Sentiment analysis and opinion mining[J]. Synthesis lectures on human language technologies, 2012, 5(1): 1-167.

[Loper and Bird, 2002] Loper E, Bird S. NLTK: The natural language toolkit[C]//Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1. Association for Computational Linguistics, 2002: 63-70.

[Maalej and Nabil, 2015] Maalej W, Nabil H. Bug report, feature request, or simply praise? on automatically classifying app reviews[C]//Requirements Engineering Conference (RE), 2015 IEEE 23rd International. IEEE, 2015: 116-125.

[Manning et al., 2014] Manning C D, Surdeanu M, Bauer J, et al. The stanford corenlp natural language processing toolkit[C]//ACL (System Demonstrations). 2014: 55-60.

[Marcus et al., 1993] Marcus M P, Marcinkiewicz M A, Santorini B. Building a large annotated corpus of English: The Penn Treebank[J]. Computational linguistics, 1993, 19(2): 313-330.

[Maynard and Funk, 2011] Maynard D, Funk A. Automatic detection of political opinions in tweets[C]//Extended Semantic Web Conference. Springer, Berlin, Heidelberg, 2011: 88-99.

[Medhat et al., 2014] Medhat W, Hassan A, Korashy H. Sentiment analysis algorithms and applications: A survey[J]. Ain Shams Engineering Journal, 2014, 5(4): 1093-1113.

[Mihalcea and Tarau, 2004] Mihalcea R, Tarau P. TextRank: Bringing Order into Text[C]//EMNLP. 2004, 4: 404-411.

[Miller, 1995] Miller G A. WordNet: a lexical database for English[J]. Communications of the ACM, 1995, 38(11): 39-41.

[Nigam et al., 2000] Nigam K, McCallum A K, Thrun S, et al. Text classification from labeled and unlabeled documents using EM[J]. Machine learning, 2000, 39(2): 103-134.

[Ofek et al., 2013] Ofek N, Caragea C, Rokach L, et al. Improving sentiment analysis in an online cancer survivor community using dynamic sentiment lexicon[C]//Social Intelligence and Technology (SOCIETY), 2013 International Conference on. IEEE, 2013: 109-113.

[Oh et al., 2013] Oh J, Kim D, Lee U, et al. Facilitating developer-user interactions with mobile app review digests[C]//CHI'13 Extended Abstracts on Human Factors in Computing Systems. ACM, 2013: 1809-1814.

[Pagano and Maalej, 2013] Pagano D, Maalej W. User feedback in the appstore: An empirical study[C]//Requirements Engineering Conference (RE), 2013 21st IEEE International. IEEE, 2013: 125-134.

[Page et al., 1999] Page L, Brin S, Motwani R, et al. The PageRank citation ranking: Bringing order to the web[R]. Stanford InfoLab, 1999.

[Pang et al., 2002] Pang B, Lee L, Vaithyanathan S. Thumbs up?: sentiment classification using machine learning techniques[C]//Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002: 79-86.

[Panichella et al., 2015] Panichella S, Di Sorbo A, Guzman E, et al. How can i improve my app? classifying user reviews for software maintenance and evolution[C]//Software maintenance and evolution (ICSME), 2015 IEEE international conference on. IEEE, 2015: 281-290.

[Panichella et al., 2016] Panichella S, Di Sorbo A, Guzman E, et al. Ardoc: App reviews development oriented classifier[C]//Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016: 1023-1027.

[Qiu et al., 2009] Qiu G, Liu B, Bu J, et al. Expanding domain sentiment lexicon through double propagation[C]//IJCAI. 2009, 9: 1199-1204.

[Sarkar et al., 2010] Sarkar K, Nasipuri M, Ghose S. A new approach to keyphrase extraction using neural networks[J]. arXiv preprint arXiv:1004.3274, 2010.

[Siddiqi and Sharan, 2015] Siddiqi S, Sharan A. Keyword and keyphrase extraction techniques: a literature review[J]. International Journal of Computer Applications, 2015, 109(2).

[Socher et al., 2013] Socher R, Perelygin A, Wu J, et al. Recursive deep models for semantic compositionality over a sentiment treebank[C]//Proceedings of the 2013 conference on empirical methods in natural language processing. 2013: 1631-1642.

[Sorbo et al., 2016] Di Sorbo A, Panichella S, Alexandru C V, et al. What would users change in my app? summarizing app reviews for recommending software changes[C]//Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016: 499-510.

[Sundararajan et al., 2017] Sundararajan S, Bhasi M, Pramod K V. Managing Software Risks in Maintenance Projects, from a Vendor Perspective: A Case Study in Global Software Development[J]. International Journal of Information Technology Project Management (IJITPM), 2017, 8(1): 35-54.

[Thelwall et al., 2012] Thelwall M, Buckley K, Paltoglou G. Sentiment strength detection for the social web[J]. Journal of the Association for Information Science and Technology, 2012, 63(1): 163-173.

[Tonella et al., 2003] Tonella P, Ricca F, Pianta E, et al. Using keyword extraction for web site clustering[C]//Web Site Evolution, 2003. Theme: Architecture. Proceedings. Fifth IEEE International Workshop on. IEEE, 2003: 41-48.

[Toutanova et al., 2003] Toutanova K, Klein D, Manning C D, et al. Feature-rich part-of-speech tagging with a cyclic dependency network[C]//Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics, 2003: 173-180.

[Tumasjan et al., 2010] Tumasjan A, Sprenger T O, Sandner P G, et al. Predicting elections with twitter: What 140 characters reveal about political sentiment[J]. Icwsm, 2010, 10(1): 178-185.

[Turney, 2000] Turney P D. Learning algorithms for keyphrase extraction[J]. Information retrieval, 2000, 2(4): 303-336.

[Vasa et al., 2012] Vasa R, Hoon L, Mouzakis K, et al. A preliminary analysis of mobile app user reviews[C]//Proceedings of the 24th Australian Computer-Human Interaction Conference. ACM, 2012: 241-244.

[Vu et al., 2015] Vu P M, Nguyen T T, Pham H V, et al. Mining user opinions in mobile app reviews: A keyword-based approach (t)[C]//Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on. IEEE, 2015: 749-759.

[Wan and Xiao, 2008] Wan X, Xiao J. Single Document Keyphrase Extraction Using Neighborhood Knowledge[C]//AAAI. 2008, 8: 855-860.

[Wan et al., 2007] Wan X, Yang J, Xiao J. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction[C]//ACL. 2007, 7: 552-559.

[Wang et al., 2006] Wang J, Peng H, Hu J. Automatic keyphrases extraction from document using neural network[J]. Advances in Machine Learning and Cybernetics, 2006: 633-641.

[Zhang et al., 2007] Zhang Y, Milios E, Zincir-Heywood N. A comparative study on key phrase extraction methods in automatic web site summarization[J]. Journal of Digital Information Management, 2007, 5(5): 323.

Appendix 1

Questionnaire for keyphrase extraction test

Designer: Libin Tan

Thanks a lot for participating in the keyphrase extraction test! The purpose of the test is to obtain manually extracted keyphrase from 50 selected user reviews as the test data to compare with the output of my system. Following are 50 user reviews of the social app Facebook crawled from Google Play (review dates are between May 1st and May 5th, 2017). All of them are classified as useful reviews that contain helpful information for app maintenance and updates. Your task is to read all the reviews and conclude top 10 keyphrases from them.

Following are something to notice before the test:
(1) Please record the time that you spend on the test when you start analyzing the reviews. Notice that this is not an easy test and might take some times.

(2) The keyphrase means the word or phrase that can capture the primary topics and represent the most relevant information contained in the reviews. A typical keyphrase can be a noun such as "button", or a noun phrase such as "social app" (noun + noun) and "nearest hotel location" (adjective + noun).

(3) For the test reason, please only use nouns or noun phrases as your final result. It can contain adjective if you think the adjective is informational and necessary. Moreover, the keyphrase must have appeared at least once in the original texts of the 50 reviews (please don't use your own word that the original text does not have). You can consider the phrases of the same meaning as the same when you choose the keyphrase. Moreover, it is normal that the review texts have spelling and grammar mistakes.

(4) There is no specified criterion for choosing the final top 10 keyphrases from the 50 reviews (Different algorithm also have different criteria such as the appearing times, phrase locations and so on. As human beings, we can even depend on our solid instinct to figure out the keyphrases based on our rich experience of reading). Please choose the top 10 keyphrases and ranked them by the descending order of the "key" extent (importance) in your own opinion.

(5) The result of the test will only be used for my research test purpose. There are two answer pages at the end. You can draft anywhere you want except the answer pages. The order of the 50 reviews is totally random and does not indicate their importance. If

you are ready to start, please go to next page to start reading the 50 reviews. Please don’t forget to start recording the time. Thank you!

(reviews page)
1. Most recent update is terrible. I can see notifications but I cannot view the post that the notification is for.

2. I was hoping the latest update would fix the issue of the app not loading new stories. But now it doesn't load anything at all. You'd think a company with the resources that Facebook has would be able to develop a decent app at this point. Uninstalling until you can prove that you're capable of fixing things.

3. Any plans on releasing a Night Mode setting? Twitter and Reddit have one. It's nice, especially at night. :)

4. Update, new phone, year further still no improvement, instant crash now. Old review : Huawei Ascend P7, Android 5.1.1 keeps crashing, less crashes when i turn off auto play videos.

5. "Unfortunately facebook needs to be restarted" every 15-25mins. Also, it bounces me off some pages, profiles jus so back to my news feed..slow loading like really? Just feel to uninstall soon

6. The new update means I can't even open my app anymore. Even without trying to open the app it reminds me every 30 seconds that it had crashed. Not to mention the horrific content that they allow to be posted, even when I report gruesome violent content they tell me they don't have a problem with it. Facebook just gets worse.

7. I can't see my friends online or not? Please fix it for me ,thanks in advance

8. Every update is worse than the last. *edit* I read the reviews now and not a single good one. Useless app, how is it not possible to view the posts from newest to oldest? It keeps going to the top of the news feed when you go back from a post or link, if you search in a group that has for sale posts it only brings up current sale no sold items or 'discussions' often the notifications don't come through. It's just all round the worst app out there. Possibly better to have a worm on your phone than this app.

9. Facebook not working . Please deduct problem.and saluve

10. I gave it 3 stars because why do you feel the need to update like every day or every 2 days. Then it takes forever to install seriously STOP WITH THE CONSTANT UPDATES EVERY TWO OR THREE DAYS!!!! Can't update it without it force closing. Fix it!!! Now since my phone updated it takes forever to load and so I logged out and it won't let me log back in. Keeps saying try again later. It won t load or it takes forever to load. I have to shut my phone off and power it back on just to get it to load. Fix it already and stop with the crappy updates

11. So my phone told me yesterday that I needed to update for reliability for improvements and speed. This is the biggest joke. My app is slower than ever, and it freezes up. I'm so aggravated

12. I can't use this app to download videos..I have to go through the browser in order to download videos.

13. Great app, but the newest update made the Like button on comments highlighted instead of a number. This makes it hard to read on a mobile device and I have to squint to make sure it is highlighted. Other than that it's still great.

14. It can't play videos. How can i solve this?

15. News Feed is STILL NOT loading. My Newsfeed is BLANK again This morning. Get someone who knows what their doing to fix this annoying problem. Please fix....

16. Hate, hate, HATE the new "camera"! It's on an inconvenient spot and when I do accidently hit it, it freezes my entire phone. Please remove from top left corner!

17. facebook continues to ruin its own mobile app with update features that nobody has asked for, nobody needs and that are annoyingly cluttering up every section of the app. the newest addition is the news page being split after 2-3 new items by some contacts suggested for adding - a total useless action during every day usage of the news feed. this, and many other details, makes checking facebook a tedious, unfriendly experience and i personally feel cheated and let down by it. unfortunately, facebooks crushing dominance on the social media market is unbroken, so they continue to act like market monopolists who dont give a monkeys wet fart about any user experience. at all. as long as they can sell ads.

18. Why not you show my post to all friends

19. Ever since the last update, the comment section is very touch sensitive. Simply scrolling through comments will bring up the "copy text" list menu completely unwarranted. It makes viewing comments extremely tedious. Please fix.

20. Since this last update photos and videos don't load on the news feed. It's definitely not a speed issue as I just tested and I'm getting 35 Mbps, but all I get on the feed is spinning circles. The media never loads.

21. I hate it when it updates! Thank you fb, but every time you update, it messes up. I can't seem to open it right now bc the app crashes when I open it.

22. Its so irritating having recommended friends pop up right in the middle of my notifications. Why is that necessary? Notifications are of importance. I don't need all these recommendations about who to add and j can't get rid of them from that section.

23. After recent update I can't tap on a notification as it won't let me so I have to close the app sometimes for it to work I'm on a galaxy s8 plus so it's not my phone

24. It doesn't work. Crashes on startup making it a useless app, how can a multibillion ? business not be able to build a reliable app?

25. After the latest update 03May17, the app is now continuously crashing and is unusable! Please fix the bug and I will fix my rating.

26. Fix your video playback. The app keeps closing randomly while I'm trying to watch a video. As well as live videos getting me stuck watching it as I'm trying to back out, and it will also set my orientation to landscape even though I haven't flipped it. It's a bit of a mess. Also in my opinion, you still need to remove the your day feature. It's useless this isn't Snapchat and clearly trying to pay attention to new useless features is causing you to forget about taking care of the app itself.

27. Updated and STILL not loading my feed. I have to access FB via browser for me to see my feed...why do i have this app if it doesnt work...?

28. Two "updates" in a row that do not work. News feed not loading. Does anyone from Facebook even read these comments? Freaking idiots.

29. Your Facebook team works hard to keep us Well informed. Great to use with other apps regarding games, email, social! Enjoying the new added features! Such as the

videos (those special occasions Are thoughtful from All of you. My favorite, is the Little blue 'Like' balls that drop into a bin! So ? ?) that your team puts together. Nice you Personalize these for each of us Facebook users! ? I Love seeing these! They are Amazing! ? Thank you for putting these mini videos out on our walls to share, if we like too! This tells us that you Do care, for all of us! ~

30. Hate when it just automatically updates itself. Why don't you ask first before updating. Really frustrating.

31. Fix the app!!!! Keeps crashing after the update i open the app then the app keeps freezing and crashing so tell me for what good is the updates you keep doing all day (yeah this app updates a lot during the day) for nothing just to see the app closings on its own!!!

32. New update has bug! auto closes 3 posts into feed. Fix to get 5 stars back! I use a Droid Turbo.

33. Bad update. Making the notification and friend tab the same is not a good idea. I don't want to see friend recommendations in my notification tab.

34. I have used this app for a few months and everything seems to work as expected. However now I can only see new posts in my feed. Once I have viewed the new posts and refresh my feed the posts disappear. It's like the app can't get past posts from the sever. If I log into facebook using Chrome everything comes up normal and I can see the posts I have previously viewed. I have checked everywhere for a setting but can't find one. Sadly I am only using this app for the notifications then I open facebook in chrome. I have filed a help ticket and have yet to hear anything. I can only hope that some future update will fix it.

35. The videos are not displaying properly and there seems to be some issues with the alignment part. I am using MI note 4 model.

36. Every time I think this app, that updates every other day, couldn't get worse, it surprises and runs crappier than before the update. I go on pages and it's showing me stuff from months ago but not the most recent posts. Stop updating the app all the time and fix the bugs before you update. Ridiculous!

37. It drains up too much battery when on background. What happened??

38. Jfc you guys made it so the back button on android not only scrolls back up to the top but also refreshes your entire news feed. This is by far one of the worst features to happen to this app yet. If you ever decide to actually fix anything on this app get rid of this feature. It's not how android is supposed to work.

39. Unable to open details like birthday of friends and other details. The app keeps on crashing the moment i try to access / see friends details. Kindly fix the same

40. Auto updates? Why is this app allowed to override my phone settings and auto update when my phone is set NOT to allow this? Stop taking the piss out of us all. You've lost the final 2 stars because of it

41. Getting really hating the current version. I'll be reading something on my news feed and suddenly it will refresh for no reason. Then I won't be able to find the post i was reading. Seriously annoying!

42. Please fix the problem with video player... It just stops working after the last update... Done times the aspect ratio is not good... Sometimes the video starts coming

43. The new feed should be in order of what was last posted, not what was last commented on. The privacy settings don't work. I put everything on private or friends only but I can still see what I don't want other people to see when I check from a different fb that is not connected to my profile at all.

44. Rubbih, many bugs still. Watch a video and cannot return to the main page feeds. Have to quit the app then go back to FB. Please fix.

45. Videos constantly crash or do not load. Screen layout is getting worse.

46. Awful app. Forces auto play on videos which isn't good for 500MB data limits. The app should ask this in the setup process. No messenger included it tries to force download of the separate messenger app. I see repeated posts in my feed at times and I'm defiantly scrolling down. Back button refreshes news feed. I like to pull down to refresh only. No option to default to most recent news. Top stories is default.

47. I love this app because it is something different than other apps this app helps to stay connected with people and many more facilities are given in the app

48. Well! I have been using fb for the last 10 years, but i with the passage of time i made myself used to and up to date with the upgradation and introduction of new features, but recent update is really not good! In which friend requests and notifications are joint together in notification panel, when i try to view profile of new request it goes to request panel and then get mix up with other pending requests rather than showing in order of date.. kindly separate this option... thanks

49. Latest update again intermittently fails to load notifications. That's four updates this has been broken now! Rubbish!! Also, whenever I click on a photo and try to enlarge it the app freezes. I have to force-close it and open it again, losing where I was into the bargain. Further to these problems, even when closed the app constantly works away draining my battery in the background....why?? What are you doing exactly, spying? Yet another app I wish that I hadn't updated :-/

50. Love the app, just wish it wouldn't auto refresh when I'm watching something or reading a post. Very frustrating to have to find that post again and then it auto refreshes again.

(All 50 reviews are done! You might need to go back and rank the top 10 keyphrases that you have recognized.)

(answer pages)
1. Your name:

2. Your result of the top 10 keyphrases (written in descending order of importance):

1 (Most important)	
2	
3	
4	
5	
6	
7	

8	
9	
10 (Not most important)	

3. Time you used for finding out all the 10 keyphrases : __ min __ s

4. What are your criteria for choosing the 10 keyphrases?

5. Score from 0 to 10, to what extent do you think your selected 10 keyphrases can represent the information contained in the 50 reviews? (0 means the keyphrases can represent nearly nothing, 1 means they can represent about 10% of the information and so on, while 10 means they can represent nearly all the information.)

6. You have manually extracted 10 keyphrases from the 50 reviews. Do you think it is possible for you to manually go through 500 reviews to extract top 10 keyphrases? Select one from the following answers:
(1) Sure, it's a snap. (2) Yes, although it might be a challenging task.
(3) Not sure, but it is possible to have a try. (4) Not really, it is too challenging.
(5) Are you kidding me? It is impossible!

Appendix 2

Questionnaire for sentiment analysis test

Designer: Libin Tan

Thanks a lot for participating in the aspect sentiment test! It might take about 15 minutes to finish the test. In this test, there are 50 user reviews of the adventure game app "Pokémon GO" crawled from Google Play, which are divided into two sets. The first set of 25 reviews all contains the keyphrase (aspect) "pokemon", and the second set all contains the keyphrase "update". Your task is to judge the author's sentiment of each review towards the keyphrase is positive, negative or neutral. Notice that you need only to analyze the sentiment towards the keyphrase but not the sentiment of the whole review. Please put a tick mark (√) on the corresponding underline of each review on the last two answer pages.

A simple example:
01. I hate this game because it costs too much time, but I can't stop catching these cute Pokemons!!

We know that the sentiment on "Pokemon" is positive in this review, so tick the corresponding place on the answer page:
01. Positive _√_ Negative ___ Neutral ___

Now you can start reading the 50 reviews, and the keyphrase in each review is marked in red color. You only need to judge the sentiment towards the keyphrase. You can mark the result on the answer pages every time you finish analyzing one review.

(reviews page)
Set one: keyphrase "pokemon"

01. I love the game and I also love pokemon I real life aswell. So I love the game as I said and I love the pokemon and make it easier to find pikachus please I'm desperate to find one seriously.

02. Why can't I catch Pokemon's during this event, it started well but now I can't catch anything. ,Help please.

03. Please make the amount of candies smaller to evolve your pokemon. I needed to evolve my Eevee and it was 25 and I had 24.

04. I wish you could trade and actually fight pokemon except for just catching them.Overall it's amazing.

05. I hate it because it won't let me get the pokemon on the same space some one tell me how to get the pokemon!!!!!!!!!!

06. I go to my grandma village in summer vacation and I can't find any pokestop or any pokemon in that place. the area is nizampatnam,andhra Pradesh,Hyderabad,telangana,india

07. Cant even catch a pokemon now it just stops working when i try to catch a pokemon..... Sad i used to love this game

08. They are telling me to go the gym when I'm level 5, but when I'm level 5, they are telling me to go where the gym is. Also, wwhen I'm near a pokespot, I open my tablet and they are telling me that there is no internet connection. Besides, I find the same pokemon in the same place evreyday. How do they expect me to play it like this?

09. I think we should encounter Pokemon in a new way. I think we should fight to encounter Pokemon .

10. Can you plz sort the bugs out as I carnt see my pokemon when I catch them plz I am 23 and yer still love my pokemon since I was a kid ??

11. Good, but not everyone has the funds to travel the world to catch Pokemon. Would be great if you made them all accessible on the states. Thanks

12. I had to delete and reinstall the game and now the Pokemon are invisible. and now my character is out of control

13. I just love Pokemon as a kid and now this game came out it just brings out my child hood memories only better because I have to go out and look for them to get a little of realness of the game just love this game

14. It needs much better improvement because there are still several problems with this game including how the Pokemon don't show up when I check on or use them in a gym

battle making me have to wait until something pops up and I'm also incredibly sick of dealing with encounting such useless Pokemon that aren't specific enough and the egg groups still needs to be fixed really badly because they're still some Pokemon out there that are very difficult for us to find in the wild such as Snubbull and Houndour. But instead it gives us Pokemon that are very common and that really annoys us still like Staryu, the Nidorans, and those loathsome Goldeens that I couldn't stand to waste 2 kilometers over. That problem really needs to be fixed first or give us much better events to participate in if we ever want to enjoy playing this yet again.

15. I Wish you have to fight each pokemon you find to weaken them like in the shoe and games...

16. The game is so slow i cant do anything i wasted an incesne waiting for pokemon to pop up all there is, is a ball of light that will disappear

17. I can't catch the Pokemon they get away every time when throw the ball.

18. Not know how to catch Pokemon first time

19. You need to make pokemon around water inevitable. It's stupid to travel to a lake to do d nothing.

20. HELP I'M STUCK. I tried to catch the pokemon on the tutorial and I was on top of them. When i touched them they wouldnt move. Plz fix

21. I can hear the Pokemon spawning around me but they are not visible and I'm unable to click on any Pokemon

22. Amazing game, highly addictive, pokestops, walking eggs and ur buddy, as well as catching all those pokemon. Bringing adults and their children together to bond over a common intrest for a blast from the past =). Does have some bugs, but deff looking forward to the updates to come =)

23. How about some updates like Pokemon coming to us like using an insence but an automated insence according to player's stats on everyday run.... I have been playing this game day and night yet I fail to catch any rare Pokemons. And in here My place (Siliguri and Kolkata) very less pokestop are available and very far. Yet somehow I manage to fill my bag And yet I fail to catch good Pokemons. It's been 3/4 months I have been playing I have reached lvl 24 yet I see only pidgey, rattata, speraow, mukrow,

sentret ,etc very common Pokemon. All my lil good Pokemons are hatched from egg, else all those I have caught are either have very low cp or poor IVs. It feels like all those hardwork have gone in vain. And there are players who cheats can't you do something about it. Those players have best of defense pokemon and Cp yet I fail to see a single player or anybody near gym. Cause I play day and night I have been to gyms at 2/3 am and I am the only one walking yet I see battle is going on and there's no one present. How about gym battles on the basis of players lvl from 1 to 40... I guess that's the max lvl one could reach. They can have options whom to battle according to palyers lvl. 24 vs 24 if one wishes to go higher they can. I just mean fair play and everyone should be Happy. But I guess doing this you won't earn. All I want to say is there are 2 types one who could pay and have it all. And one who cannot and do all on his own. And that's me so why not we have some advantages who really works hard and are true to Game. Otherwise Pokemon go is now my among the best Games I have ever palyed and wanted to play. Hope you have given a thought about the hard working palyers.

24. You know all this talk about spoofers is pissing me off i personally know a past spoofer and had a legitimate reason for it...He lived in the middle of no where which spawned the same pokemon every day nothing new. When you go to big populated cities you catch pokemon that are super rare but they catch them everyday...How is this game supposed to be fair if the type of pokemon and amount depends on how many spray on tan blownout hair people you live by

25.I love this game. I just wish they would hurry and add all the rare pokemon like mew and mewtwo and the legendary birds or if they have make it more often to find them. And ones like Charizard i have yet to find one. Also i wish they would make the game to where we can trade pokemon with people and also battle other people 1v1 or 2v2 etc. Also could you guys put some pokestops up at truck stops??? I am a truck driver and i like playing this game but i run out of balls and have to wait until i go home to get more. I live on the road and truckstops are all i got for weeks.... thanks.

Set two: keyphrase “update”

26. I can't login using mobile data after i update....login problem.

27. I think you niantic should make an update on the voice of the Pokémons... Only Pikachu has its original voice from the TV series

28. I HATE THE UPDATES. I WAS DOING GOOD AND NOW MY PHONE ISN'T COMPATIBLEEEE. PLZ FIX THIS.

29. I absolutely love this game more than any other game hands down. However the updates help, but sometimes make things worse. When I catch a Pokemon sometimes it tells me I didnt.. Never had this problem before this most recent update. Also gps is getting better. But why one week straight I caught enough Onix to get 39 candies and that was 3 months ago and I go to the same spot every day and dont find a single one now.. Also the evolvements are getting shorted on CP some double.. Others only barely move. Still love the game, but have a few bugs to fix.

30. I love these updates each time it drains more battery than last time ... better start taking 2 battery packs instead of 1 extra

31. PLEASE give us the option to delay updates!!!!! Twice now a forced update cost me a gym.

32. Since the update I can't play it because it refuses to find my GPS.

33. Still waiting !!!!!Can't get any pokemon to show up on game HELP FIX IT ill give 5 stars but I can't play Does not need update Already updated Also just spent money on game Love this game but HATE IT RIGHT NOW HELP FIX IT HELP FIX IT FIX IT I LIKE PLAYING BUT HAVEN'T BEEN ABLE TO PLAY FOR ABOUT 2 WEEKS FIX IT

34. This is such a fun game but there needs to be trading and breeding and also more updates love all of the events and we need more of them thx Niantic for this game good job ??

35. This update sucks....i have to wait nearly 30 mins everytime I login.....it often shows unable to authenicate.....please fix it....

36. It's a great game but it's not getting 5 stars cause since the update when i load the game all the Pokémon in my area flash on the screen then disappear this only seams to happen where i live cause when i go anywhere else it works perfectly.

37. Thank you for the updates. Please keep adding features such as trading system and player vs player battle.

38. I think you should add a update where we can battle to other pokemon players that are near us please

39. Since a long time waited for an update but can't play the game after update as GPS signal not found

40. Would you fix the connectivity issues with the Go Plus! With each update it gets worse. I have to discover to proper sequencing of unpairing/ pairing/ restarting the game/ rebooting my phone... it must rival those at nuclear missile silos by now.

41. Update mended it all, can play again, the game is great fun and addicting but I still haven't found any new pokemon for a while

42. Unfinished game released. Updates are very slow. Can't breed eggs or get candies from my buddy, while i lock my mobile phone. Items are hard to get without real money. Arena fights while the GPS isn't fully stable are very bad/bugged. IV can#t be shown without 3rd party apps. Niantic does everthing to make the life hard for 3rd party developers and the community, while not really doing anything against cheaters. .....

43. THE UPDATE THAT KILLED POKEMON GO! The reason for this is simple, it says that it can't log in, and it's not just my account because I've tested it on different accounts on different devices and they all sat the same message, "Failed to authenticate to server, please try agein later"

44. I love this game but it is hard to get 7 days in a row when the updates frequently cause the game to crash.

45. Great for my ADHD, until everything slowed down and login failed repeatedly, or game crashed several times a day. Hoping this update fixes those new prolems.

46. Worst update ever for a game it's absolutely irritating I've lost my interest completely on this game me and my friends are gonna drop playing this game If this update is not fixed soon once again thank you for making me forget a good game by a worst update fy!

47. Update as of 4/9/17 this game is trash now. Location no longer works and the game is unplayable. I uninstalled this game and I will NEVER come back. Niantic is a trash developer.

48. Still no trading. Still no battling. It's impossible to hunt for pokemom you want or need. Niantic doesn't support tracking features. Gyms are horrible experiences. No status effects on moves, boring gameplay and a huge time sink. Everything in this game is a random gamble(spawns, moves, items, and even evolutions!). You never know if it's worth it to actually play. Its too bad.. Pokemon GO could have been the greatest Pokémon game ever :(I play everyday hoping it gets better. The problem is everytime Niantic releases an event or an update that doesn't address these issues, it proves that the real issues are of no concern to them.

49. The updates are getting better, but the inability to spin pokestops above 20mph is just stupid. I can understand not allowing radar to work, but spinning stops should still be allowed.

50. It would be nice if updates were tested and debugged BEFORE being forced onto everyone.

Appendix 3

Results of the sentiment analysis test

P: Positive N: Negative O: Neutral

Set one: keyphrase “pokemon”

Review ID	Sentiment Analyzer	Review Rate	Tester 1	Tester 2	Tester 3	Tester 4
01	P	P	P	P	P	P
02	N	P	O	N	N	O
03	O	P	O	N	O	O
04	N	P	O	P	P	O
05	N	N	N	N	N	N
06	N	O	O	N	N	O
07	O	N	N	N	N	N
08	N	O	N	N	N	N
09	O	P	O	O	O	O
10	O	P	O	N	N	N
11	N	P	O	O	P	O
12	N	P	N	N	N	N
13	P	P	P	P	P	P
14	N	N	N	O	N	N
15	O	O	O	O	O	O
16	N	N	O	N	N	N
17	N	N	N	N	N	O
18	O	N	O	O	N	O
19	O	O	N	N	N	N
20	P	N	N	N	N	O
21	N	N	N	N	N	N
22	P	P	P	P	P	P
23	N	P	N	N	N	N
24	N	N	N	N	N	N
25	N	P	O	P	P	N

Set two: keyphrase “update”

Review ID	Sentiment Analyzer	Review Rate	Tester 1	Tester 2	Tester 3	Tester 4
26	N	O	O	N	N	N
27	N	P	O	N	N	O
28	O	N	N	N	N	N
29	N	P	N	N	P	N
30	N	O	N	P	N	N
31	N	O	N	N	N	N
32	N	N	N	N	N	N
33	N	N	N	N	N	N
34	P	P	O	O	P	P
35	O	N	N	N	N	N
36	O	P	N	N	P	N
37	N	P	P	P	P	P
38	O	P	O	O	O	O
39	O	N	N	N	N	N
40	N	O	N	N	N	N
41	P	P	P	P	P	P
42	N	N	N	N	N	N
43	N	O	N	N	N	N
44	N	N	N	N	P	N
45	O	P	O	N	N	O
46	N	N	N	N	N	N
47	N	N	N	N	N	N
48	N	N	N	N	N	N
49	N	N	N	P	P	P
50	N	N	N	N	N	N