Expert Systems With Applications 113 (2018) 186–199

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Review

Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools

Mohammadali Tavakoli, Liping Zhao*, Atefeh Heydari, Goran Nenadić

School of Computer Science, University of Manchester, Manchester, United Kingdom

ARTICLE INFO

Article history:
Received 31 August 2017
Revised 15 March 2018
Accepted 30 May 2018
Available online 9 June 2018

Keywords:
Mobile application review
App review
App review mining, app development
Intelligent app review mining tools
Intelligent app review mining techniques

ABSTRACT

Mobile application (app) websites such as Google Play and AppStore allow users to review their downloaded apps. Such reviews can be useful for app users, as they may help users make an informed decision; such reviews can also be potentially useful for app developers, if they contain valuable information concerning user needs and requirements. However, in order to unleash the value of app reviews for mobile app development, intelligent mining tools that can help discern relevant reviews from irrelevant ones must be provided. This paper surveys the state of the art in the development of such tools and techniques behind them. To gain insight into the maturity of the current support mining tools, the paper will also find out what app development information these tools have discovered and what challenges they are facing. The results of this survey can inform the development of more effective and intelligent app review mining techniques and tools.

1. Introduction

Web-based software application distribution platforms have become increasingly popular among the Internet users. According to statistica.com, the number of mobile application (app) downloads from Google Play and App Store has increased from 17 billion in 2013 to 80 billion in 2016. These platforms also allow users to share their opinions on their downloaded applications (apps). From the user perspective, such opinions can influence their decision on purchasing or choice of a particular app (Heydari, ali Tavakoli, Salim, & Heydari, 2015). From the app provider perspective, positive reviews can attract more customers and bring financial gains. Similarly, negative reviews often cause sales loss. In addition, user reviews or opinions may also contain information relevant to app development and improvement, such as bug reports, user experience, and user requirements.

However, not all the user reviews are relevant and useful for app development. In order to unleash the value of app reviews for app development, intelligent mining tools that can help discern relevant reviews from irrelevant ones must be provided. In recent years, a variety of such techniques have been proposed, ranging from sentiment analysis (Fernández-Gavilanes, Álvarez-López, Juncal-Martínez, Costa-Montenegro, & González-Castaño, 2016), spam detection (Heydari, Tavakoli, & Salim, 2016; Savage, Zhang, Yu, Chou, & Wang, 2015), to more general mining techniques (Castelli, Manzoni, Vanneschi, & Popovič, 2017). Yet, there is a lack of systematic understanding of these techniques in the context of mining mobile app reviews and their support tools. So far, we have found only two surveys that assess App Store mining techniques. The first survey (Genc-Nayebi & Abran, 2016) provides an analysis of general data-mining techniques for spam detection, opinion mining, review evaluation, and feature extraction. The second survey (Martin, Sarro, Jia, Zhang, & Harman, 2016) is concerned with App Store analysis, such as API analysis, feature analysis, app review analysis, and so on. None of these surveys cover the work on specific mining techniques or tools for app reviews.

To address this gap, this paper surveys the state of the art in the development of mobile app review mining techniques and tools. To

* Correspondence author at: Schoolof Computer Science, University of Manchester, Manchester, M13 9PL, United Kingdom.
E-mail addresses: mohammadali.tavakoli@manchester.ac.uk (M. Tavakoli), liping.zhao@manchester.ac.uk (L. Zhao), atefeh.heydari@manchester.ac.uk (A. Heydari), gnenadic@manchester.ac.uk (G. Nenadić).

https://doi.org/10.1016/j.eswa.2018.05.037

gain insight into the maturity of the current support mining tools, the paper will also find out what app development information these tools have discovered and what challenges they are facing. The results of this survey can inform the development of more effective and intelligent app review mining techniques and tools.

The rest of the paper is organized as follows: Section 2 describes our survey methodology. Section 3 presents and analyses the survey results. Section 4 discusses some open issues and challenges facing the development of app review mining tools. Finally, Section 5 discusses the validity threats to this review and Section 6 concludes the review.

2. Survey methodology

The methodology for conducting our survey is systematic literature review (Kitchenham, Charters, Budgen, Brereton, & Turner, 2007). To avoid confusion, we have used the term "survey", instead of "review", to refer to this paper, as the subject matter of the paper is app reviews. Based on the systematic literature review (SLR) guidelines provided by Kitchenham et al. (2007), in what follows, we describe the steps in our survey process.

2.1. Research questions

As stated in Section 1, the main goal of this paper is to survey the state of the art in the development of mobile app review mining techniques and tools. Specifically, the survey will cover the primary studies that report the development of mobile app review mining techniques and tools. In addition, this paper will also find out what specific app development topics the reported tools are used to discover, as this finding will help us evaluate the maturity of the current mobile app mining tools. In line with these goals, we have formulated the following three research questions and will use them to drive our survey:

- R1: What mobile app review mining techniques have been reported in the literature?
- R2: What software tools have been developed to support these techniques?
- R3: What mobile app development topics are used to illustrate the reported techniques and tools? Which topics are used most?

2.2. Identifying relevant primary studies

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

We developed our search string by selecting keywords from the studies that we had already reviewed in the domain. Then, we identified and applied alternatives and synonyms for each term and linked them all by the use of AND/OR Boolean expressions to cover more search results. In order to perform a widespread search, formulating a comprehensive query is indispensable. Thus, we optimized and refined our preliminary search string during multiple iterations as mulling over the revealed results and skimming retrieved relevant studies helped us to manipulate our search string effectively with more appropriate keywords. We excluded keywords whose inclusion was not advantageous or replaced them with more apt ones. Our search terms are presented in Table 1 and the finalized search term is '(mobile OR android OR IOS) AND (app OR application) AND (feedback OR review OR opinion OR comment) AND (bug report OR feature request OR complain OR requirement OR issue OR expectation) AND (analysis OR process OR mining OR extract OR discover) AND (developer OR development team OR requirements team OR software vendor) OR (requirement engineering OR RE OR requirements elicitation OR requirements analysis)'.

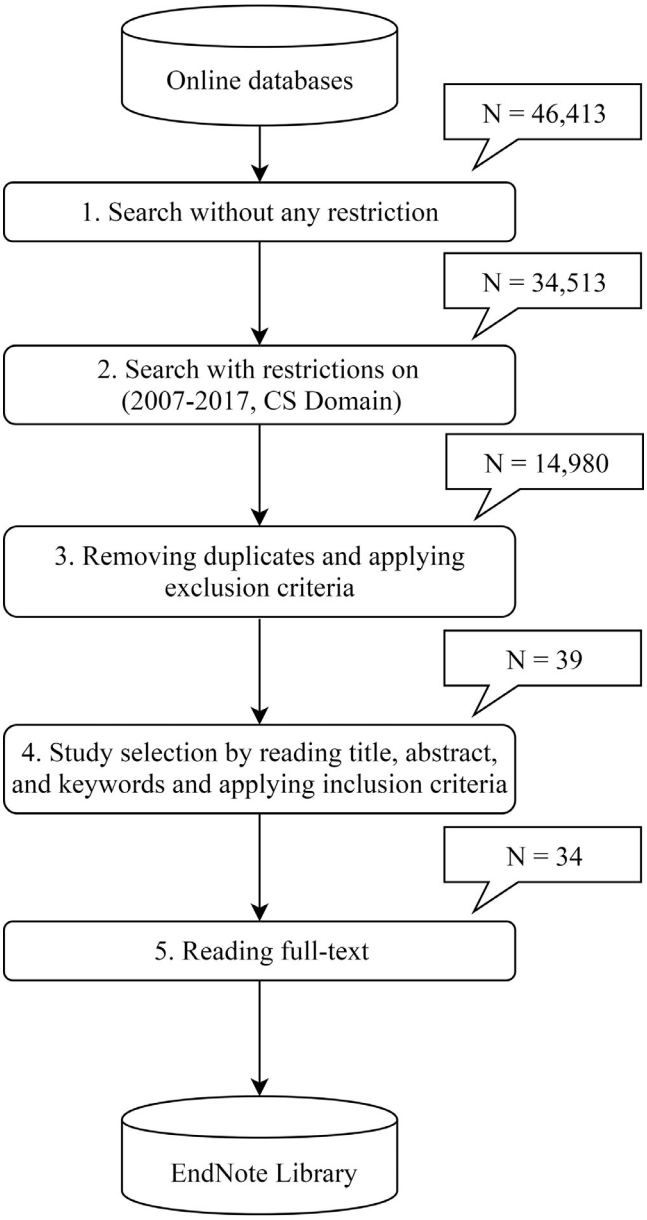

Fig. 1. Phases in the selection process. N denotes the number of papers.

After constructing the search string, we used it to search the following databases: ScienceDirect, IEEEXplore, ACM, GoogleScholar, Scopus, and SpringerLink. The search returned a total of 46,413 results. We found that the first related paper was published on 2011 and the last one on 2017. Table 2 summarizes the search results.

2.3. Selecting relevant studies

Based on the search results, we have used the following inclusion and exclusion criteria to select the relevant primary studies. The selection phases are shown in Fig. 1.

Inclusion criteria:

Table 1
Types and components of the search strings.

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

Table 2
Search results (2011–2017).

Database	Search results
Science Direct	8880
Scopus	4513
ACM	1777
SpringerLink	18,465
IEEEXplore	3528
Google Scholar	9250
Total	46,413

I1. Studies reporting app reviews related to application development were included.
I2. In addition to I1, studies reporting detailed empirical research methods, such as case studies, surveys, experiments, and ethnographical studies, were included.
I3. If more than one paper reports the same study, only the latest or fullest paper was included.

Exclusion criteria:

E1. White and grey literature (i.e. research outputs that are not peer reviewed, such as reports, online documents, and working papers) was excluded.
E2. Abstract papers with no full-text available were excluded.
E3. Short papers with less than four pages were excluded.

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

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

At the end of Step 3, a total of 34 primary studies were selected as relevant to our survey, the full text of which was imported into an Endnote library for data extraction. The references of these studies are listed in Appendix A.

Fig. 2 shows the distribution of the selected 34 studies published from 2011 to 2017. The maximum number of papers was published in 2015, whereas 2011 only had one paper. It is worth mentioning that the small number of the papers found in 2017 is due to the search period, which covers the first two months of the year. Of the 34 selected papers, 27 of the selected papers (77%) are conference papers and 8 of them (23%) are journal articles. Appendix B summarizes the number of the papers published in each channel.

2.4. Extracting and synthesizing data

In this step, the required data were extracted from each of the 34 primary studies. A predefined data extraction form (see Table 3) was used to record the data for each study. Two types of data were extracted: the data required for answering the research questions and the data required for displaying the bibliographic information of the study. The extracted data were stored in an Excel file for further process and analysis.

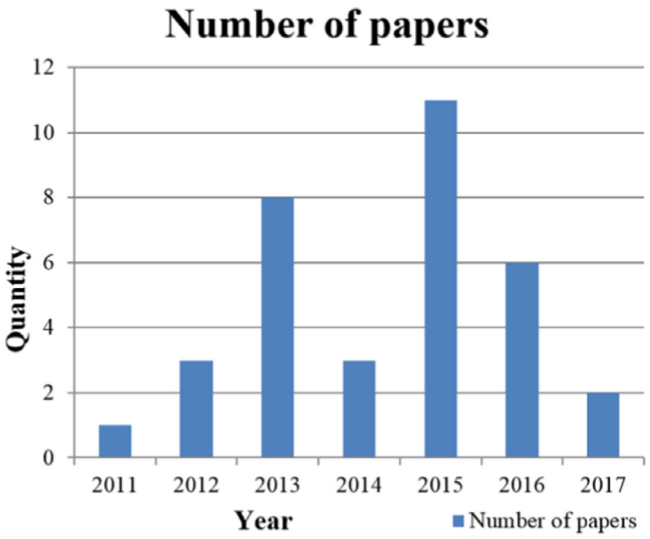

Fig. 2. Distribution of the selected 34 primary studies from 2011 to 2017.

Table 3
The data extraction form.

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

The extracted data were synthesized using the constant comparison method (CCM) (Boeije, 2002). The steps involved in data synthesis were:

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

3. Survey results

This section analyses the review results and answers our research questions.

3.1. Mobile app review mining techniques (RQ1)

RQ1: What mobile app review mining techniques have been reported in the literature?

The mining techniques identified by our survey are of three types: supervised machine learning, natural language processing (NLP) and feature extraction. This section presents each type of technique in detail.

3.1.1. Supervised machine learning techniques

Supervised machine learning techniques had been found in 14 (40%) of the selected studies. These techniques have been used to classify mobile app reviews.

Platzer (2011) used motivational models to find usage motives that are addressed in the text of user reviews. To attach these motives to the reviews, they applied SVM and NB algorithms. Oh, Kim, Lee, Lee, and Song (2013) believed that uninformative reviews should be filtered out to help developers overcoming the overload of feedback. First, they defined three categories for reviews (i.e. functional bug, functional demand, and non-functional request) and manually classified 2800 reviews into the categories to train their filtering model. Identification of categories was done by app developers who analysed the review contents manually, though details of the process, such as number of experts, number of analysed reviews, and selection criteria is not provided. Then, they extracted keywords of each review using Latent Dirichlet Allocation (LDA) and heuristic methods and applied SVM algorithms to classify reviews into informative and uninformative. The authors argued that the performance of LDA in identifying keywords is weak due to shortness of reviews in length.

In order to identify the reasons why users dislike apps, Fu et al. (2013) developed a system, named WisCom, analysing reviews in three different levels. Firstly, at the review level, they discovered reviews with inconsistent ratings by applying a regularized regression model. This was to understand individual reviews and the words used in those. Secondly, at the app level, they applied topic modelling (LDA) techniques to discover specific problems of each app behind the users' complaints about that. Finally, in entire app market level, they analysed top 10 complaints of each app to find similar complaints and the most critical aspects of apps leading to identifying global trends in the market. Their focus was only on negative reviews as they hypothesized that those could be directly used to enhance the quality of applications. However, researchers observed that feature requests that are important feedback for developers mostly appeared in positive reviews (Iacob & Harrison, 2013; Iacob, Veerappa, & Harrison, 2013).

Chen, Lin, Hoi, Xiao, and Zhang (2014) developed a computational framework for App Review Mining, namely AR-Miner which filters useless reviews and uses topic modelling techniques to group informative reviews based on the topics discussed in them, and a review ranking scheme to prioritize them with respect to the developers' needs. AR-Miner uses Expectation Maximization for Naive Bayes (EMNB), a semi-supervised algorithm in machine learning to classify reviews to informative and uninformative. In next phase, it uses LDA for grouping, and creates a ranked group of reviews based on their rating, fluctuation in time of reviews, and volume of reviews reporting a similar issue/request. The authors compared the performance of two algorithms in topic modelling, i.e., Latent Dirichlet Allocation (LDA) and Aspect and Sentiment Unification Model (ASUM) for grouping informative reviews based on their contents. In classifying user feedback into informative and non-informative, they did not justify the superiority of Expectation Maximization for Naive Bayes (EMNB) to other existing algorithms. Testing the performance of other classifiers and comparing the outcomes was a good way for justification of superiority of their approach. Moreover, there are several important aspects neglected in their approach. Firstly, sentiment of the review relates to its content and could be used to leverage the performance of the tool (Guzman & Maalej, 2014). Secondly, although they employed time stamp, text, and rating of a review to do their task, important features such as title of review and meta-data features are neglected. Finally, they removed stop-words, stemmed, and applied other pre-processing techniques on raw crawled reviews. State-of-the-art approaches (McIlroy, Ali, Khalid, & Hassan, 2016) show that these pre-processing tasks increase the chance of losing helpful content. Additionally, they discriminated informative and uninformative reviews based on their own understanding of "informative" and reviewing few numbers of online forums, while to identify what really is important for mobile app developers to extract from user reviews, their needs and requirements should be studied comprehensively.

AR-MINER was used further by Palomba et al. (2015) to investigate how addressing user feedback by developers influences the overall rank of the app. Their proposed tool, named CRISTAL (Crowdsourcing Revlews to SupporT App evoLution), collects reviews posted for previous release of a certain application and tracks its rating. Then it extracts informative reviews using the AR-MINER and checks whether the comments are applied in next release. Finally, it checks the effect of them on rating after the last release.

Maalej and Nabil (2015) designed and applied different probabilistic techniques and heuristics to automatically classifying reviews into four basic types: Bug reports, feature requests, User experiences, and rating. They generated a list of keywords by string matching, bag of words, sentiment scores, NLP pre-processed text, review rating and length, to be used for classification task. Then, they applied Naive Bayes, Decision Trees, and MaxEnt to compare the performance of binary to multiclass classifiers in classification of user feedback into the predefined basic types. The authors extended their approach in Maalej, Kurtanović, Nabil, and Stanik (2016) by adding bigram and its combinations to utilized classification techniques, and by improving preprocessing phases and classification scripts. They argued that by the use of meta-data combined with text classification and natural language pre-processing of the text, the classification precision rises significantly.

Guzman, El-Haliby, and Bruegge (2015), relied on categories found in Pagano and Maalej (2013) to form their taxonomy with 7 categories relevant for software evolution. The authors then used the taxonomy to investigate the performance of various machine learning techniques (i.e. Naive Bayes, Support Vector Machines (SVMs), Logistic Regression and Neural Networks) in classification of reviews. The set of features they used includes number of upper/lower case characters, length, positive/negative sentiment, and rating. However, many other features could be found in user feedback to be used for classification purposes in order to enhance the throughput of the model (Tavakoli, Heydari, Ismail, & Salim, 2015).

Panichella et al. (2015) argued that among the 17 topics identified by Pagano and Maalej (2013), only 8 of them were relevant to software maintenance and evolution tasks. They proposed a method to identify constructive feedback for software maintenance and evolution tasks. The authors hypothesized that understanding the intention in a review has an important role in extracting useful information for developers. And to understand the intention of a review, they used sentences structure, sentiment of a review, and text features contained in a review. Thus, they formed a taxonomy (i.e. Information Giving, Information Seeking, Feature

Request, and Problem Discovery) by manually reviewing a number of reviews. Then they extracted a set of features by the use of NLP, text analysis, and sentiment analysis techniques to train a classifier and finally classified reviews according to the taxonomy. They compared performance of different machine learning techniques, namely, the standard probabilistic naive Bayes classifier, Logistic Regression, Support Vector Machines, J48, and the alternating decision tree (ADTree) and reported that J48 performed well. The authors identified and grouped review sentences into 6 categories. They compared these categories with topics identified by Pagano and Maalej and found that all 17 topics match with 4 out of their 6 identified categories.

Panichella et al. (2016) improved the approach in Panichella et al. (2015) by proposing ARdoc (App Reviews Development Oriented Classifier), a tool that automatically classifies useful sentences in user reviews according to a taxonomy designed by Panichella et al. (2015) to model developers' information needs when performing software maintenance and evolution tasks. After dividing the review into sentences, ARdoc extract lexicon, grammatical structure, and sentiment of each sentence to be used by a machine-learning algorithm (J48) for classification purposes.

The SURF (Summarizer of User Reviews Feedback) was proposed in Di Sorbo et al. (2016), which is a tool for categorization and summarization of reviews. It splits a review into sentences and performs the summarization task in three phases. Firstly, it employs the approach proposed in their previous study (Panichella et al., 2015) and their identified sentence categories to classify sentences into user intentions and assigns one of the intentions to each review sentence. Secondly, it employs sets of keywords to build an NLP classifier and automatically assign one or more concept (topic) to each sentence in a review. Specifically, the authors manually analysed each sentence in 438 reviews selected as a training set to discover topics discussed by reviewers resulting in identification of 12 topics. They manually assigned keywords to each sentence and finally, for each topic, created a finite set of keywords and enriched it with WordNet synonyms. Finally, for summarization purposes, they relied on their observations (i.e. bug reports and feature requests are more important, sentences discussing certain aspects of the app are needed, longer reviews are more informative, and frequently discussed features are needed) and assigned a score to each sentence for each observation. The tool categorizes sentences according to their topics and intention categories and generates summaries as structured HTML. However, their approach suffers from lack of a comprehensive research on what really is needed by app developers as these points are observed only by authors and a software developer. Moreover, they assigned different relevance scores to each category of intentions to score the first observation without studying the impact of each score.

To overcome the problem of processing colloquial terminologies used by users in their informal language which causes complex classification models and overfitting problems, (Jha & Mahmoud, 2017b) proposed a FrameNet tagging based approach to classify reviews based on the notion of semantic role labelling (SRL). The aim of using SRL is to obtain a higher level of abstraction from sentences. SRL classifies the words used in a sentence into semantic classes describing an event along with its participants. In their classification task using Naive Bayes (NB) and Support Vector Machines (SVM), the authors used frames generated from each review, rather than each word. However, their target classes are limited to bug reports and feature requests indicating that other types of valuable information are ignored in their approach. The authors released their classifier as a tool in Jha and Mahmoud (2017a).

In another study targeting only one-star and two-star reviews, McIlroy et al. (2016) studied the extent of multi-labelled user reviews (reviews raising more than one issue type) and proposed an approach to automatically labelling multi-labelled user reviews. They defined 13 types of issues and labelled a number of reviews manually to form their gold standard dataset. For labelling task, they transformed the problem of multi labelling into single labelling and used a classifier for each label and combined their results. They used several different classifiers e.g., support vector machines (SVM), decision tree (J48) and Naive Bayes (NB) as well as several different multi-labelling approaches e.g., Binary Relevance (BR) [it does not leverage the correlations between labels], Classifier Chains (Palomba et al., 2015) [it does leverage the correlations between labels], and Pruned Sets with threshold extension (PSt). They defined a threshold to assign each label to a review. Finally, they used 10-fold cross-validation to evaluate results. In preprocessing phase, they removed numbers and special characters, but not stop words, expanded abbreviations, filtered words occurring less than three times in dataset, stemmed words, and removed reviews consisting of three words or less. However, observations exhibit that reviews with less than three words report bugs and issues as well (e.g.: "poor camera", "save button sucks", and "can't upload picture"). Moreover, they used (TF-IDF) as a mean to increase the weight of words that occur frequently in a single user review and to decrease the weight of words that occur frequently in many user reviews. Although it helps to devalue ordinary words, this way of weighting words might demote issues repeated and discussed between several users.

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

Table 4 summarizes these techniques.

Table 4
Supervised machine learning techniques found in 14 studies.

Algorithm	Brief description	Used in study
SVM		S1, S9, S18, S26, S27, S31, S32, S33, S34
NB		S1, S13, S18, S21, S23, S26, S27, S29, S31, S32, S33, S34
NN		S18
J48		S26, S27, S31, S32
Regression Models		S5, S18, S26, S32
Decision Tree		S21, S26, S27, S29, S32
LDA		S5, S9, S13, S23
MaxEnt		S21, S29

Table 5
NLP techniques found in six studies.

Techniques	Brief description	Used in study
Linguistic rules		S11
NLP: n-gram and PMI analyses		S20
definition of lexical-POS patterns		S22
Definition of regular expressions, TF-IDF		S25
Term Vector Model, TF-IDF		S24
Aspect and Sentiment Unification Model, Topic Modelling		S10

3.1.2. NLP techniques

NLP techniques are found in six (17%) of the selected studies for pre-processing or extracting relevant information from mobile app reviews (Table 5).

Iacob and Harrison (2013) studied identification of proportion of feature requests among users' feedback. They developed a prototype to automatically mine online reviews of mobile apps and retrieve feature requests. It crawls reviews by the use of a data crawler, extract feature requests by 237 predefined linguistic rules, summarize extracted feature requests by ranking them based on their frequency and length, and visualize the results. The authors crawled 3279 reviews from Google play to manually read them and formulate the linguistic rules to identify feature requests. To identify common topics across the feature requests, they then applied the LDA model.

To study the enhancement requests of Android OS by users, Lee, Licorish, MacDonell, Patel, and Savarimuthu (2015) applied NLP techniques (i.e. n-gram and PMI analyses) to identify the most pressing requests, though their focus was on android OS which reviews vary from the ones generated for applications. Their mere focus on text and ignoring the available meta-data of feedback is an obvious drawback of their approach.

Moghaddam (2015) defined some lexical-POS patterns (8 for improvements and 5 for defects) to implicit reviews containing improvement/defects. Then she used these reviews as positive cases to train her distance learning classifier. After using SVM to extract sentences containing defects/improvements, the author applied LDA to cluster similar sentences and score the importance of founded topics. However, her effort on finding patterns to capture defects/improvements was not sufficient as in her approach, problem, issue, and bug report is defined as defects and improvements include modification, upgrade, enhancement request, but only 13 patterns are defined to extract them among user reviews. Thus, many forms of explanations used by users to report defects/improvements are most probably missed by her approach. These patterns are used to label reviews and it causes inaccuracy in the results. Moreover, the proposed approach provides developers with more categorized sets of reviews, but they still need to expend too much effort and sources to explore these sets for the definition of defects/improvements which is too general in terms of categories fallen in these two groups.

Yang and Liang (2015) proposed an approach to extract user reviews from AppStore, extract requirements information from them, and classify them into functional and non-functional (NFR) requirements. In their approach, requirement engineers manually identify and classify a certain number of user reviews as NFRs or FRs. Then, TF-IDF technique extracts keywords from these reviews to be used for automatic classification of reviews by the use of predefined regular expressions. However, human judgement is required to check and select the keywords which makes their approach labour-intensive. Their corpus consists of only 1000 reviews which is not appropriate to obtain a stable evaluation. Moreover, the authors annotated the data and extracted keywords by themselves. And they defined the regular expressions too which was, most probably, based on their observations over the data. Thus, performance of their approach severely depends on their preferences and evaluation accuracy is under question.

Vu, Nguyen, Pham, and Nguyen (2015) developed MARK (Mining and Analysing Reviews by Keywords), a semi-automated framework assisting developers in searching reviews for certain features. It uses keyword-based approaches to search for relevant reviews. Several NLP techniques are used for keywords extraction, ranking keywords, and categorizing them based on their semantic similarity. MARK uses the standard Vector Space Model to query its review database and fetch results with respect to the keyword set.

Galvis Carreno and Winbladh (2013) used topic modelling and IE techniques to discover the topics from reviews that can be used to change and/or create new requirements for a future release of software. They used Aspect and Sentiment Unification Model (ASUM) for extraction of topics. The authors focused only on requirement changes, while other kinds of valuable information for developers are neglected in their approach. Moreover, the authors manually classified reviews to build their gold standard dataset and acknowledged that this way of procuring training data is error-prone as the expertise of authors in the domain is under question.

Table 6
Feature extraction techniques found in three studies.

Technique	Brief description	Study ID
Pattern-based parsing		S16
Collocation finding algorithm		S14, S17

3.1.3. Feature extraction techniques

Finally, only three studies (0.9%) used feature extraction techniques for mobile app reviews (Table 6).

Gu and Kim (2015) applied sentiment patterns to parse review sentences and elicit app features. Their proposed tool (SURMiner), firstly, splits each review into sentences and applies Max Entropy to classify reviews into five categories, namely *aspect evaluation, bug reports, feature requests, praise* and *others*. Then, it identifies aspects and their associated opinions from sentences fallen in the category of *aspect evaluation* by designing a pattern-based parsing method. To design the method, the authors manually analysed 2000 sentences from reviews labelled as *aspect evaluation* and identified 26 templates. The tool, then, analyses the sentiment of each sentence and assigns a rating to that. It, finally, mines frequent items for all aspect words and clusters aspect-opinion pairs with common frequent items to summarize the output. Their focus is merely on aspect evaluation sentences in reviews. Thus, majority of topics such as feature requests and bug reports are ignored in their study. The main aim of their study was to extract application aspects and their associated opinions by designing a pattern-based parsing method. However, definition of patterns for unclear aspects is error-prone as there is uncertainty in the aspects discussed in the user reviews. Besides, from app to app aspects differ significantly which demands dramatic updates in predefined patterns.

Guzman and Maalej (2014) extracted features from the text of reviews by the use of collocation finding algorithm. The algorithm finds collection of words frequently occurring often (e.g. "battery life" and "screen resolution"). Then they applied SentiStrength, an automated sentiment analysis techniques (Thelwall, Buckley, & Paltoglou, 2012) designed to tackle with short and low quality texts, to extract opinions and sentiments associated with each feature. SentiStrength divides the input text into sentences and assigns a positive value along with a negative value to each sentence. Finally, they grouped related features and aggregated their sentiments using topic modelling techniques (Blei, Ng, & Jordan, 2003). Their aim was to automatically identify application features and their associated sentiments mentioned in user reviews. Guzman, Aly and Bruegge (2015) extended their previous approach (Guzman & Maalej, 2014) and studied the identification of conflicting opinions. They developed DIVERSE, to identify diverse user opinions concerning different applications. It groups reviews by their mentioned features and sentiment. Similar to their previous approach, the authors used collocation finding algorithm to extract features from user reviews and a lexical sentiment analysis tool to find opinions and experiences concerning the features. Then, they used a greedy algorithm to retrieve a set of diverse feature-sentiments.

There are several issues affecting the accuracy of their method. Firstly, a feature in their approach is a collection of two words occurred in more than three reviews including different orders, syn-

Table 7
The studies in Section 3.1.4.

Study ID	Technique	No. of review samples
S6	Manual analysis	556
S7	Manual analysis	3279
S12	Manual analysis	528
S8	Manual analysis	6390
S15	Manual analysis	6390
S19	Manual analysis	6390
S2	Manual analysis	32,108
S3	Manual analysis, regex	8,701,198
S4	Manual analysis, regex	8,701,198

onyms, and misspells. Limiting the feature to be consisted of two keywords prevents the approach to comprehensively cover various types of features mentioned in user feedback. Secondly, according to definition of a feature, their approach ignores features appearing in less than three reviews. These features might be very important for development team. finally, the lexical sentiment analysis scores each sentence in the review. Then it assigns this score to the feature mentioned in the sentence. However, in many cases, the overall sentiment of a review is negative, but the user is admiring a feature in that sentence. In the other word, negative words may be used in the favour of a feature (Sohail, Siddiqui, & Ali, 2016). For example, in the review: ⟨the UI is nice, but I don't like the background⟩, their approach assigns the neutral score of (0) to the whole review and, correspondingly, to each feature.

3.1.4. Manual analysis

The majority of selected studies in this group (6 out of 9) have used manual analysis of user feedback to identify topics discussed (Table 7). In this section, each of the studies are explained in detail.

Ha and Wagner (2013) manually analysed 556 reviews from 59 different applications of Google Play to find what users are talking about. They identified 18 categories of topics discussed in the reviews after multiple irritations. To understanding what recurring issues users like to report in their reviews, Iacob et al. (2013) manually analysed 3279 Google play reviews. The authors defined a coding scheme to capture the recurring issues. Two coders annotated 125 randomly selected reviews resulting in identification of 9 classes of codes, namely positive, negative, comparative, price related, request for requirements, issue reporting, usability, customer support, and versioning. Then they divided each review into significant snippets of text to assign them an associate refined code. Their approach is vague, as they have not described how the whole 3279 reviews had been processed.

In an exploratory study, to demonstrate how user reviews are useful for developers, Pagano and Maalej (2013) investigated how and when users provide feedback, how to identify and classify topics in reviews, and how to integrate user feedback into requirements and software engineering infrastructures (more focus was on the impact of reviews on rating and on community of users). The authors applied a descriptive statistic to investigate usage of feedback. They, then, manually analysed a random sample of reviews (528 reviews) to explore and assign topics to each review. They grouped their 17 observed topics into four themes, namely community, rating, requirements, and user experience. In another preliminary contribution in the field (Khalid, 2013), Khalid manually analysed 6390 one-star and two-star reviews for 20 iOS apps in order to aid developers by listing the most frequent complaints. They discovered 12 types of issue and complaint about iOS apps in user feedback and repeated their approach in Khalid, Shihab, Nagappan, and Hassan (2014). In 2015, they extend their approach by proposing a review system enabling users to see more visualized ratings, reply to comments, sort reviews by like/dislike, and categorize the reviews (Khalid, Asif, & Shehzaib, 2015).

All of the abovementioned studies in this section tried to identify topics discussed in feedback by manually analysing them. This way of topic finding is accurate as annotation reports of the studies show that results of different human judgments on a same dataset are close to each other. However, the studies could not be considered as approaches facilitating developers in analysing reviews as too much cost and effort is required to approach them. The identified topics, on the other hand, could be used in other proposed methods to more accurately classify feedback contents, though domain dependency is a barrier that should be taken under consideration properly. Moreover, reviews containing syntactic, sarcasm, and ironic contents are excluded from their scope.

Apart from identification of topics in user feedback, manual analysis had been used to study the role of star rating and relations between various aspects of applications including volume of download, length of reviews, amount of sale, and price. By manually analysing users' reviews on BlackBerry apps, Harman, Jia, and Zhang (2012) confirmed that the rating given by user has a significant impact on number of downloads. To better understand what users communicate in their reviews, Hoon, Vasa, Schneider, and Mouzakis (2012) analysed 8.7 million reviews from 17,330 apps and categorized keywords appearing frequently in each star rating as they hypothesized that it can inform and focus development efforts. The authors determined the distribution of word and character counts per star rating respectively applying a regular expression to extract words from the review entities, and monitored which star rating the appearance of the extracted keywords pertains to. However, they did not apply appropriate pre-processing techniques (e.g. stemming, removing of stop words, and spell checking) to normalize input reviews. Moreover, their focus was only on single words resulting in missing multi-word expressions. The authors used their dataset in another approach (Vasa, Hoon, Mouzakis, & Noguchi, 2012) to discover possible relations between rating of a review and its content. They argued that reviews with lower ratings include more useful feedback, and that the depth of feedback in certain categories is significantly higher than for other. In both of their approaches, the authors did not discuss details and settings of the analysis. Obviously, beside manual analysis and observations they have used some automatic processing techniques that are not mentioned in the studies.

3.2. Support tools for mining mobile app reviews (RQ2)

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

Nine support tools were found in the selected studies, a summary of which is presented in Table 8. In what follows, we provide an overview of these tools.

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

WisCom. This tool analyses user feedback at three levels of detail, involving discovering inconsistencies in reviews, identifying reasons why users like or dislike a given app, and identifying general user preferences and concerns over different types of apps. Firstly, a regularized regression model is used for discovering inconsistencies and detect ratings that do not match the actual text

Table 8
Support tools for mining mobile app reviews.

Tool name	Description	Underlying technique	Unit of analysis	Study ID
MARA	Based on the frequency and length of the terms in a sentence, the tool uses 273 syntactical rules to identify, extract and summarize feature requests.	Syntactical rules, LDA	Sentences	S11
WisCom	The tool identifies inconsistent feedback using a regression model, uncovers main causes of complaints for each app using LDA, and aggregates top complaints of apps in a category to identify market trends	Regression models LDA Aspect and Sentiment Unification Model	Multi-level	S5
AR-miner	In order to filter out useless reviews. The tool uses topic modelling techniques to group informative reviews and a review ranking scheme to prioritize them.	Expectation Maximization for Naive Bayes, LDA	Sentences	S13
CRISTAL	The tool detects traceability links between incoming app reviews and source code changes likely addressing them. It uses such links to analyse the impact of crowd reviews on the development process.	AR-MINER Information Retrieval heuristics	Whole Documents	S23
DIVERSE	For detection of conflicting opinions, the tool responds to developer's query by grouping reviews by their mentioned features and sentiment.	Collocation finding algorithm, Lexical sentiment analysis tool, Greedy algorithm	Whole Documents	S17
SURMiner	To summarize reviews, the tool classifies reviews and extracting aspects using a pattern-based parser.	Sentiment patterns, Max Entropy	Sentences	S16
MARK	The semi-automated tool assists developers in searching reviews for certain features.	keyword-based approaches, NLP techniques, Vector Space Model	Whole Documents	S24
SURF	Combining topic extraction and intention classification techniques, the tool categorizes and summarizes user reviews.	NLP classifier, WordNet synonyms	Sentences	S27
ARdoc	The tool automatically classifies useful sentences in user reviews according to a taxonomy.	NLP, Sentiment analysis, Text analysis, J48	Sentences	S32

of the feedback. Secondly, feedback comments of individual apps are aggregated and LDA algorithm is applied to discover why users dislike these apps. The algorithm was trained using words that receive negative weight in the regression model. Finally, to identify outstanding complaints in each category of apps, most common complaints from negative feedback comments of each app identified in the previous step are aggregated on categories, summarized and displayed to the user.

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

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

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

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

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

SURF. This tool summarizes user reviews to assist developers in managing huge amount of user reviews. The tool relies on a conceptual model for capturing user needs useful for developers per-

forming maintenance and evolution tasks. Then it uses summarisation techniques for summarizing thousands of reviews and generating an interactive agenda of recommended software changes. The tool is equipped with a dictionary of topics and uses the NLP classifier to automatically assign a sentence in a review to one or more topics. To suggest the specific kinds of maintenance tasks developers have to accomplish, it also classify intentions in a review using intent classifier (Panichella et al., 2015). Based on a certain scoring mechanism, the tool then generates the summaries as structured HTML.

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

3.3. Mobile app development topics discovered in the selected studies (RQ3)

RQ3: What mobile app development topics are used to illustrate the reported techniques and tools? Which topics are used most?

This SLR has found 10 groups of topics discovered by the literature and each group is further elaborate in a number of specific topics. Table 9 summarizes all identified topics. We have grouped common topics across different studies and separated them with solid lines in the table.

To answer the second part of the RQ1: 'Which topics are hot?', we can easily refer to Table 9 and count number of studies that have discovered a topic. Obviously, the topic with most number of studies referred to, would be the hottest one. The most frequent category of topics discovered in the literature is 'bug reports' with 19 studies referring to it by different terms. Only 5 studies have mentioned the exact term of 'bug report' for the topic. The second frequent one is 'feature requests' discovered by 16 studies and mentioned by the exact term of 'feature request' in 7 papers. The statistics show that the main concerns of application users are reporting bugs and shortcomings and requesting for new features to be added to the application. Fig. 3 shows the distribution of topics discovered in 16 of selected studies.

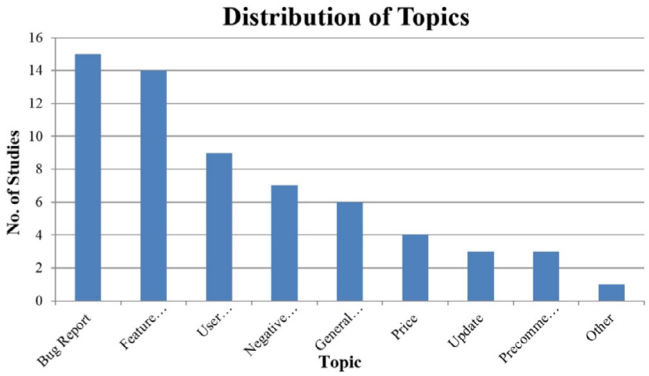

Fig. 3. Distribution of mobile app development topics.

4. Open issues

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

1) *Huge volume of user feedback:* The volume of user-generated feedback is huge. Extracting insights from massive amount of reviews is a labour-intensive and challenging task for researchers. Quantity of reviews generated for an application usually exceeds a human capacity to manually analyse them and extract useful information. It makes difficulties in identifying recurring trends and issues across reviews. Moreover, processing such an amount of data is time consuming. In case of application updates and revealing new versions, the time commitment involved in extracting required changes for next release is crucial (Galvis Carreno & Winbladh, 2013). Thus, more novel approaches for effectively and efficiently processing massive amounts of feedback by taking relations between them into account are on demand.
2) *Unstructured data:* Unstructured nature of user feedback is one of the main challenges in automatically processing them. App users often write their reviews in unconventional manners making automate interpretation as difficult as possible (Iacob & Harrison, 2013). They tend to express their reviews using informal language, which often includes colloquial terminologies. Moreover, they regularly neglect grammar and punctuation rules and use eccentric syntactic entities, and ironic and sarcasm sentences in their reviews (Pagano & Maalej, 2013). One key challenge to app developers is dealing with such unstructured short pieces of text. To overcome these problems, technical text mining and NLP approaches should be integrated into proposed models and accompany them in order to deal with these types of user generated text.
3) *Annotating data:* Data annotation is required in supervised approaches for training the machine and for evaluation purposes. It is a time and cost consuming task usually accompanied with mistakes and errors, though it is necessary to obtain better results in supervised and semi supervised approaches. As it is discussed in Section 3.1, in majority of cases, studies have provided different approaches demanding certain types of annotation resulting in impossibility of creation of a gold standard dataset to be used for evaluation of various approaches. Apart from difficulties of manual annotation of data, accuracy of the annotated data is under question as instead of domain experts, article authors or non-experts are employed to do it in majority of cases. Thus, more efforts and novel approaches are required to alleviate the problem of data annotation.
4) *Data pre-processing:* With the use of some certain preprocessing tasks, researchers try to prepare the data to be used as the input of their method. Stop-word removal techniques are

Table 9
Mobile app development topics discovered in the 34 selected studies.

Topic category	Specific topic	Study ID	No. of studies per category
1. General comment	• Entirely reviewing the app, Works/Doesn't work	S6	6
	• Positive	S7	
	• Helpfulness, Praise	S12	
	• Rating	S21	
	• Feature strength, General Praise	S18	
	• Descriptions	S13	
2. Negative comments	• Works/Doesn't work - Uninstalled	S6	7
	• Negative	S7	
	• Shortcoming, Dispraise, Dissuasion	S12	
	• General complaint	S13, S18	
	• Compatibility, App Crashing, Network Problem, Interface Design, Privacy and Ethical, Response Time, Uninteresting Content, Resource Heavy	S8, S31	
3. App Comparison	• Comparison with other apps	S6	3
	• Comparative	S7	
	• Other app	S12	
4. Bug Report	• Feature/Functionality	S6	15
	• Issue reporting	S7	
	• Content request, Feature, Bug report	S12	
	• Functional bug	S9, S31	
	• Performance flaw	S13	
	• Functional error, Feature Removal	S8, S13	
	• Bug report	S18, S21, S22, S33, S34	
	• Solution proposal	S26, S32	
	• Problem discovery	S26, S27, S32	
	• Feature shortcoming	S18	
5. Feature Request	• Aesthetics	S6	14
	• Request for requirements	S7	
	• Improvement request	S12, S22	
	• Promise better rate for improvement	S12	
	• Functional demand, Non-Functional request	S9	
	• Feature Request	S8, S11, S12, S21, S26, S27, S32	
	• User Request	S18	
	• User Requirements	S33, S34	
6. User Experiences	• Tips (installation/usage)	S6	9
	• Usability	S7	
	• Question (How to use)	S12, S13	
	• User experiences	S21	
	• Information Seeking, Information Giving	S26, S27, S32	
	• Opinion asking	S26, S32	
	• User scenario	S18	
7. Updates	• Updates (comparing to previous version)	S6	3
	• Update issues –	S31	
	• Versioning –	S7	
8. Price	• Money (worth the money)	S6	4
	• Price related	S7	
	• Hidden Cost	S8	
	• Additional Cost	S31	
9. Recommendation	• Recommending the app	S6	3
	• Customer support	S7	
	• Recommendation	S12	
10. Other	• Additional Program needed, Number and content of ads, Company, Just downloaded, Not used yet, Device model, Permissions, Preinstalled app, Consumption of resources	S6	1

an example of these tasks. Various proposed methods include these tasks to perform more efficiently in terms of time and computation. However, a group of approaches have argued that using some of the preprocessing tasks causes missing valuable information in user feedback (Jha & Mahmoud, 2017b). Therefore, more comprehensive and analytical research is required to investigate the impact of applying each pre-processing task on performance and accuracy of the techniques.

5) *Data interpretation:* While various approaches and techniques for extraction of actionable information from user feedback have been proposed by researchers, their interpretation of actionable information is, to a certain extent, different from developers'. Previous research (Begel & Zimmermann, 2014) has tried to discover what developers are highly interested in, and what the designers' viewpoint is when mulling the reviews over (Liu, Jin, Ji, Harding, & Fung, 2013; Qi, Zhang, Jeon, &

Zhou, 2016). Although some of these factors (i.e. functionality, and app features) are used by Guzman, El-Haliby et al. (2015), and informativity of reviews from developers' perspective was investigated by reading some relevant forum discussions by Chen et al. (2014), many proposed approaches have made the data analysis from the authors' point of view. Inconsistency between researchers' and developers' interpretation results in development of inefficient methods and techniques. More precise study of what exactly needs to be extracted from user reviews is required to make proposed approaches more efficient.

5. Validity threats to this SLR

The primary threats to validity of results from this SLR are concerned with comprehensiveness and coverage of the relevant studies. Firstly, the search only covers publications that were published before the end of February 2017. We conducted the review in 2017. We, therefore, used 2017 as an upper bound on the search term. Various other relevant studies will have been published since March 2017 that we have not included in this review.

A further search-related limitation of the review is that we might have missed some papers in identification of relevant studies. The completeness of the search is dependent upon the search criteria used and the scope of the search, and is also influenced by the limitations of the search engines used (Brereton, Kitchenham, Budgen, Turner, & Khalil, 2007). We used a set of well-known references to validate the search string and made necessary amendments before undertaking the review. However, four papers were identified by snowballing that were indexed by the digital libraries but were not found with the search terms used in the review.

Aiming to cope with construct validity which is related to generalization of the result to the concept or theory behind the study execution (Claes et al., 2000), we defined and applied several synonyms for main constructs in this SLR: "requirements engineering", "feedback analysis", and "software evolution".

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

If the identified literature is not externally valid, neither is the synthesis of its content (Gasparic & Janes, 2016). To alleviate this threat, we formed our search process after multiple trial searches and compromise of the authors. Unqualified papers were excluded as well by the application of our exclusion criteria.

6. Conclusion

This paper reports our research effort aimed at systematically reviewing and analysing application feedback processing practices toward assisting developers with extraction of actionable information and insights. The review was conducted by the guidelines provided in Kitchenham et al. (2007). In total, 34 relevant studies were identified and analysed. Firstly, we categorized and discussed studies based on underlying techniques. Then, we investigated available supporting tools for feedback mining. After that, we described topics discovered by the selected studies and identified the most popular one. Finally, we discussed some challenges and open problems in feedback mining, which require further research. The findings from this review provide several implications for researchers, requirements engineers and tool developers to gain a better understanding of the available studies and tools and their suitability for different contexts resulting in significant improvements in development of intelligent app review mining techniques and tools.

Our results showed that strict quality control on discovered topics in user feedback did not seem sufficient to ensure that the feedback is a rich source of RE related useful information. This fact affects investments of application vendors on user feedback mining. Moreover, we discovered that the accuracy and effectiveness of proposed techniques varies when applied on user reviews of different types of applications indicating that the terminology and app features used in user reviews is impressed by the domain of the app. Developers and requirements engineers, therefore, should take the domain dependency issues into account while selecting the target techniques for mining reviews of their apps.

The results of this study also show that recently published papers have studied extraction techniques comparing to earlier ones focusing on discovering discussed topics. By considering such a dissemination, domain dependency of approaches, and probable newly emerged topics, researchers and practitioners might want to get prepared for encountering newfangled topics in the reviews.

There are several future research directions derived from insights provided by this SLR. Firstly, although many researchers have proposed approaches in user feedback mining, the usefulness and value of user feedback for RE experts is still under question as there is, to our knowledge, no solid research on investigation of quality and usefulness of user feedback from RE experts' viewpoint. So, in-depth exploratory studies are needed to address this shortcoming. Secondly, we discovered that there is no user feedback mining tool addressing all the challenges listed in Section 4. Development of such a tool could assist RE experts in mining user feedbacks efficiently. Thirdly, there are other sources of user feedbacks on mobile apps available for users and developers. Experimenting current techniques on them, enriches the data that will result in more interesting findings. Finally, proposing techniques for automatically transforming extracted user needs into requirement specifications and modelling would be a great complementary component of current requirements extraction tools.

Acknowledgements

We wish to thank the reviewers for their constructive comments, which have helped improve the paper. Ali and Atefeh are grateful for the President Doctoral Scholarship of the University of Manchester, which enable them to carry out their PhD study.

Appendix A. The 34 selected primary studies

Study ID	Citation	Title	Venue
S1	Platzer (2011)	Opportunities of automated motive-based user review analysis in the context of mobile app acceptance	Central European Conference on Information and Intelligent Systems
S2	Harman et al. (2012)	App store mining and analysis: MSR for app stores	Proceedings of the 9th IEEE Working Conference on Mining Software Repositories
S3	Hoon et al. (2012)	A preliminary analysis of vocabulary in mobile app user reviews	Proceedings of the 24th Australian Computer-Human Interaction Conference
S4	Vasa et al. (2012)	A preliminary analysis of mobile app user reviews	Proceedings of the 24th Australian Computer-Human Interaction Conference
S5	Fu et al. (2013)	Why people hate your app: Making sense of user feedback in a mobile app store	Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining
S6	Ha and Wagner (2013)	Do android users write about electric sheep? examining consumer reviews in google play	Consumer Communications and Networking Conference
S7	Iacob et al. (2013)	What are you complaining about?: a study of online reviews of mobile applications	Proceedings of the 27th International BCS Human Computer Interaction Conference
S8	Khalid (2013)	On identifying user complaints of iOS apps	35th International Conference on Software Engineering
S9	Oh et al. (2013)	Facilitating developer-user interactions with mobile app review digests	CHI'13 Extended Abstracts on Human Factors in Computing Systems
S10	Galvis Carreno and Winbladh (2013)	Analysis of user comments: an approach for software requirements evolution	Proceedings of the International Conference on Software Engineering
S11	Iacob and Harrison (2013)	Retrieving and analyzing mobile apps feature requests from online reviews	10th IEEE Working Conference on Mining Software Repositories
S12	Pagano and Maalej (2013)	User feedback in the appstore: An empirical study	21st IEEE international requirements engineering conference
S13	Chen et al. (2014)	AR-miner: mining informative reviews for developers from mobile app marketplace	Proceedings of the 36th ACM International Conference on Software Engineering
S14	Guzman and Maalej (2014)	How do users like this feature? a fine grained sentiment analysis of app reviews	22nd IEEE international requirements engineering conference
S15	Khalid et al. (2014)	What do mobile app users complain about?	IEEE Software
S16	Gu and Kim (2015)	What Parts of Your Apps are Loved by Users?	30th IEEE/ACM International Conference on Automated Software Engineering
S17	Guzman, Aly et al. (2015)	Retrieving diverse opinions from app reviews.	Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on, IEEE.
S18	Guzman, El-Haliby et al. (2015)	Ensemble methods for app review classification: An approach for software evolution	Automated Software Engineering
S19	Khalid et al. (2015)	Towards improving the quality of mobile app reviews	International Journal of Information Technology and Computer Science
S20	Lee et al. (2015)	They'll Know It When They See It: Analyzing Post-Release Feedback from the Android Community	In Proc 21st Amer Conf. Info. Sys.-AMCIS. Puerto Rico
S21	Maalej and Nabil (2015)	Bug report, feature request, or simply praise? on automatically classifying app reviews	23rd IEEE international requirements engineering conference
S22	Moghaddam (2015)	Beyond sentiment analysis: mining defects and improvements from customer feedback	European Conference on Information Retrieval, Springer.
S23	Palomba et al. (2015)	User reviews matter! tracking crowdsourced reviews to support evolution of successful apps	IEEE International Conference on Software Maintenance and Evolution
S24	Vu et al. (2015)	Mining User Opinions in Mobile App Reviews: A Keyword-Based Approach	30th IEEE/ACM International Conference on Automated Software Engineering
S25	Yang and Liang (2015)	Identification and Classification of Requirements from App User Reviews	ACM International conference on Software engineering and knowledge engineering
S26	Panichella et al. (2015)	How can i improve my app? classifying user reviews for software maintenance and evolution	IEEE International Conference on Software Maintenance and Evolution

(*continued on next page*)

Study ID	Citation	Title	Venue
S27	Di Sorbo et al. (2016)	What would users change in my app? summarizing app reviews for recommending software changes	Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering
S28	Genc-Nayebi and Abran (2016)	A Systematic Literature Review: Opinion Mining Studies from Mobile App Store User Reviews	Journal of Systems and Software
S29	Maalej et al. (2016)	On the automatic classification of app reviews	Journal of Requirements Engineering
S30	Martin et al. (2016)	A survey of app store analysis for software engineering	IEEE Transactions on Software Engineering
S31	McIlroy et al. (2016)	Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews	Empirical Software Engineering
S32	Panichella et al. (2016)	ARdoc: app reviews development oriented classifier.	Proceedings of 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering
S33	Jha and Mahmoud (2017b)	Mining User Requirements from Application Store Reviews Using Frame Semantics	International Working Conference on Requirements Engineering: Foundation for Software Quality
S34	Jha and Mahmoud (2017a)	MARC: A Mobile Application Review Classifier	International Working Conference on Requirements Engineering: Foundation for Software Quality

Appendix B. Publication channels of the 34 selected primary studies

Publication channel	Type	No. studies	Study ID
International Conference on Software Engineering (ICSE)	Conference	3	S8, S10, S13
IEEE international requirements engineering conference (RE)	Conference	3	S12, S14, S21
IEEE/ACM International Conference on Automated Software Engineering (ASE)	Conference	3	S16, S18, S24
IEEE Working Conference on Mining Software Repositories	Conference	2	S2, S11
Australian Computer-Human Interaction Conference (CHI)	Conference	2	S3, S4
IEEE International Conference on Software Maintenance and Evolution (CSME)	Conference	2	S23, S26
ACM SIGSOFT International Symposium on Foundations of Software Engineering	Conference	2	S27, S32
International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)	Conference	2	S33, S34
The Journal of Society for e-Business Studies	Journal	1	S1
ACM SIGKDD international conference on Knowledge discovery and data mining	Conference	1	S5
Conference on Consumer Communications and Networking	Conference	1	S6
International BCS Human Computer Interaction Conference	Conference	1	S7
Human Factors in Computing Systems	Journal	1	S9
IEEE Software	Journal	1	S15
ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)	Conference	1	S17
International Journal of Information Technology and Computer Science	Conference	1	S19
American Conference on Information Systems (AMCIS)	Conference	1	S20
European Conference on Information Retrieval	Conference	1	S22
ACM International conference on Software engineering and knowledge engineering	Conference	1	S25
Journal of Systems and Software (JSS)	Journal	1	S28
Journal of Requirements Engineering (REJ)	Journal	1	S29
IEEE Transactions on Software Engineering (TSE)	Journal	1	S30
Empirical Software Engineering (ESE)	Journal	1	S31

References

Begel, A., & Zimmermann, T. (2014). Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th international conference on software engineering* (pp. 12–23). ACM.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research, 3*, 993–1022.

Boeije, H. (2002). A purposeful approach to the constant comparative method in the analysis of qualitative interviews. *Quality & Quantity, 36*, 391–409.

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software, 80*, 571–583.

Castelli, M., Manzoni, L., Vanneschi, L., & Popovič, A. (2017). An expert system for extracting knowledge from customers' reviews: The case of Amazon. com, Inc. *Expert Systems with Applications, 84*, 117–126.

Chen, N., Lin, J., Hoi, S. C. H., Xiao, X., & Zhang, B. (2014). AR-miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th international conference on software engineering* (pp. 767–778). ACM.

Claes, W., Per, R., Martin, H., Magnus, C., Björn, R., & Wesslén, A. (2000). *Experimentation in software engineering: An introduction Online Available:*, http://books.google.com/books.

Di Sorbo, A., Panichella, S., Alexandru, C. V., Shimagaki, J., Visaggio, C. A., Canfora, G., et al. (2016). What would users change in my app? Summarizing app reviews for recommending software changes. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering* (pp. 499–510). ACM.

Fernández-Gavilanes, M., Álvarez-López, T., Juncal-Martínez, J., Costa-Montenegro, E., & González-Castaño, F. J. (2016). Unsupervised method for sentiment analysis in online texts. *Expert Systems with Applications, 58*, 57–75.

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., & Sadeh, N. (2013). Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1276–1284). ACM.

Galvis Carreno, L. V., & Winbladh, K. (2013). Analysis of user comments: An approach for software requirements evolution. In *Proceedings of the international conference on software engineering* (pp. 582–591). IEEE Press.

Gasparic, M., & Janes, A. (2016). What recommendation systems for software engineering recommend: A systematic literature review. *Journal of Systems and Software, 113*, 101–113.

Genc-Nayebi, N., & Abran, A. (2016). A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software, 125*, 207–219.

Gu, X., & Kim, S. (2015). "What Parts of Your Apps are Loved by Users?" (T). In *Automated software engineering (ASE), 2015 30th IEEE/ACM international conference on* (pp. 760–770). IEEE.

Guzman, E., Aly, O., & Bruegge, B. (2015). Retrieving diverse opinions from app reviews. In *Empirical software engineering and measurement (ESEM), 2015 ACM/IEEE international symposium on* (pp. 1–10). IEEE.

Guzman, E., El-Haliby, M., & Bruegge, B. (2015). Ensemble methods for app review classification: An approach for software evolution (N). In *Automated software*

engineering (ASE), 2015 30th IEEE/ACM international conference on (pp. 771–776). IEEE.
Guzman, E., & Maalej, W. (2014). How do users like this feature? A fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)* (pp. 153–162). IEEE.
Ha, E., & Wagner, D. (2013). Do android users write about electric sheep? Examining consumer reviews in google play. In *Consumer communications and networking conference (CCNC), 2013 IEEE* (pp. 149–157). IEEE.
Harman, M., Jia, Y., & Zhang, Y. (2012). App store mining and analysis: MSR for app stores. In *Proceedings of the 9th IEEE working conference on mining software repositories* (pp. 108–111). IEEE Press.
Heydari, A., ali Tavakoli, M., Salim, N., & Heydari, Z. (2015). Detection of review spam: A survey. *Expert Systems with Applications, 42*, 3634–3642.
Heydari, A., Tavakoli, M., & Salim, N. (2016). Detection of fake opinions using time series. *Expert Systems with Applications, 58*, 83–92.
Hoon, L., Vasa, R., Schneider, J.-G., & Mouzakis, K. (2012). A preliminary analysis of vocabulary in mobile app user reviews. In *Proceedings of the 24th Australian computer-human interaction conference* (pp. 245–248). ACM.
Iacob, C., & Harrison, R. (2013). Retrieving and analyzing mobile apps feature requests from online reviews. In *Mining software repositories (MSR), 10th IEEE working conference on* (pp. 41–44). IEEE.
Iacob, C., Veerappa, V., & Harrison, R. (2013). What are you complaining about?: A study of online reviews of mobile applications. In *Proceedings of the 27th international BCS human computer interaction conference* (p. 29). British Computer Society.
Jha, N., & Mahmoud, A. (2017a). MARC: A mobile application review classifier. *International working conference on requirements engineering: Foundation for software quality*.
Jha, N., & Mahmoud, A. (2017b). Mining user requirements from application store reviews using frame semantics. In *International working conference on requirements engineering: Foundation for software quality* (pp. 273–287). Springer.
Khalid, H. (2013). On identifying user complaints of iOS apps. In *2013 35th international conference on software engineering (ICSE)* (pp. 1474–1476). IEEE.
Khalid, H., Shihab, E., Nagappan, M., & Hassan, A. E. (2014). What do mobile app users complain about. *IEEE Software, 32*, 70–77.
Khalid, M., Asif, M., & Shehzaib, U. (2015). Towards improving the quality of mobile app reviews. *International Journal of Information Technology and Computer Science (IJITCS), 7*, 35.
Kitchenham, B., Charters, S., Budgen, D., Brereton, P., & Turner, M. (2007). *Guidelines for performing systematic literature reviews in software engineering Technical report, Ver. 2.3 EBSE Technical report. EBSE*.
Lee, C. W., Licorish, S., MacDonell, S., Patel, P., & Savarimuthu, T. (2015). They'll know it when they see it: Analyzing post-release feedback from the android community. In *Proc 21st Americas conference on information systems-AMCIS. Puerto Rico: AISeL, 2015: Vol. 11* (p. 1).
Liu, Y., Jin, J., Ji, P., Harding, J. A., & Fung, R. Y. (2013). Identifying helpful online reviews: A product designer's perspective. *Computer-Aided Design, 45*, 180–194.
Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. (2016). On the automatic classification of app reviews. *Requirements Engineering, 21*, 311–331.
Maalej, W., & Nabil, H. (2015). Bug report, feature request, or simply praise? On automatically classifying app reviews. In *2015 IEEE 23rd international requirements engineering conference (RE)* (pp. 116–125). IEEE.
Martin, W., Sarro, F., Jia, Y., Zhang, Y., & Harman, M. (2016). A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering, 43*, 817–847.
McIlroy, S., Ali, N., Khalid, H., & Hassan, A. E. (2016). Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering, 21*, 1067–1106.
Moghaddam, S. (2015). Beyond sentiment analysis: Mining defects and improvements from customer feedback. In *European conference on information retrieval* (pp. 400–410). Springer.
Oh, J., Kim, D., Lee, U., Lee, J.-G., & Song, J. (2013). Facilitating developer-user interactions with mobile app review digests. In *CHI'13 extended abstracts on human factors in computing systems* (pp. 1809–1814). ACM.
Pagano, D., & Maalej, W. (2013). User feedback in the appstore: An empirical study. In *21st IEEE international requirements engineering conference (RE)* (pp. 125–134). IEEE.
Palomba, F., Linares-Vásquez, M., Bavota, G., Oliveto, R., Di Penta, M., Poshyvanyk, D., et al. (2015). User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In *Software maintenance and evolution (ICSME), 2015 IEEE international conference on* (pp. 291–300). IEEE.
Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. (2015). How can i improve my app? Classifying user reviews for software maintenance and evolution. In *Software maintenance and evolution (ICSME), IEEE international conference on* (pp. 281–290). IEEE.
Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. (2016). ARdoc: App reviews development oriented classifier. In *Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering* (pp. 1023–1027). ACM.
Platzer, E. (2011). Opportunities of automated motive-based user review analysis in the context of mobile app acceptance. *CECIIS-2011*.
Qi, J., Zhang, Z., Jeon, S., & Zhou, Y. (2016). Mining customer requirements from online reviews: A product improvement perspective. *Information & Management, 53*, 951–963.
Savage, D., Zhang, X., Yu, X., Chou, P., & Wang, Q. (2015). Detection of opinion spam based on anomalous rating deviation. *Expert Systems with Applications, 42*, 8650–8657.
Sohail, S. S., Siddiqui, J., & Ali, R. (2016). Feature extraction and analysis of online reviews for the recommendation of books using opinion mining technique. *Perspectives in Science, 8*, 754–756.
Tavakoli, M., Heydari, A., Ismail, Z., & Salim, N. (2015). A Framework for review spam detection research. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 10*, 67–71.
Thelwall, M., Buckley, K., & Paltoglou, G. (2012). Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology, 63*, 163–173.
Vasa, R., Hoon, L., Mouzakis, K., & Noguchi, A. (2012). A preliminary analysis of mobile app user reviews. In *Proceedings of the 24th Australian computer-human interaction conference* (pp. 241–244). ACM.
Vu, P. M., Nguyen, T. T., Pham, H. V., & Nguyen, T. T. (2015). Mining user opinions in mobile app reviews: A keyword-based approach (T). In *Automated software engineering (ASE), 2015 30th IEEE/ACM international conference on* (pp. 749–759). IEEE.
Yang, H., & Liang, P. (2015). Identification and classification of requirements from app user reviews. In *ACM international conference on software engineering and knowledge engineering (SEKE)* (pp. 7–12).