Empirical Software Engineering (2022) 27: 43
https://doi.org/10.1007/s10664-021-10065-7

Analysing app reviews for software engineering: a systematic literature review

Jacek Dąbrowski[1,2] · **Emmanuel Letier**[1] · **Anna Perini**[2] · **Angelo Susi**[2]

Accepted: 5 October 2021 / Published online: 20 January 2022

Abstract

App reviews found in app stores can provide critically valuable information to help software engineers understand user requirements and to design, debug, and evolve software products. Over the last ten years, a vast amount of research has been produced to study what useful information might be found in app reviews, and how to mine and organise such information as efficiently as possible. This paper presents a comprehensive survey of this research, covering 182 papers published between 2012 and 2020. This survey classifies app review analysis not only in terms of mined information and applied data mining techniques but also, and most importantly, in terms of supported software engineering activities. The survey also reports on the quality and results of empirical evaluation of existing techniques and identifies important avenues for further research. This survey can be of interest to researchers and commercial organisations developing app review analysis techniques and to software engineers considering to use app review analysis.

Keywords App store analysis · Mining app reviews · User feedback · Mining software repository · Software engineering · Systematic literature review

Communicated by: David Lo

✉ Jacek Dąbrowski
j.dabrowski@cs.ucl.ac.uk

Emmanuel Letier
e.letier@cs.ucl.ac.uk

Anna Perini
perini@fbk.eu

Angelo Susi
susi@fbk.eu

[1] University College London, London, UK

[2] Fondazione Bruno Kessler, Trento, Italy

1 Introduction

App stores have become important platforms for the distribution of software products. In 2020, Google Play Store and Apple Store host over 5 million apps and are widely used for the discovery, purchase and updates of software products (Clement 2020). The emergence of these App Stores have had important effects on software engineering practices, notably by bridging the gap between developers and users, by increasing market transparency and by affecting release management (AlSubaihin et al. 2019). In 2017, Martin et al. (2017) used the term 'app store analysis' to denote the emerging research using app store data for software engineering. Their survey identified the richness and diversity of research using App Store data, notably for API analysis, feature analysis, release engineering, security and review analysis (Martin et al. 2017).

This paper focuses on analysing app reviews for software engineering. App reviews are textual feedback associated with a star rating that app users can provide to other App Store users and app developers about their experience of an app (App Store 2021). Most reviews have length up to 675 characters (Pagano and Maalej 2013); and convey information on variety of topics such as feature requests, bug reports or user opinions (Martin et al. 2017; Al-Hawari 2020). Analysing these reviews can benefit a range of software engineering activities. For example, for requirements engineering, analyzing app reviews can help software engineers to elicit new requirements about app features that users desire (Johann et al. 2017; Dąbrowski et al. 2020); for testing, app reviews can help in finding bugs (Maalej and Nabil 2015; Iacob et al. 2016; Shams et al. 2020) and evaluating users' reactions to released beta versions of their apps (Gao et al. 2019; AlSubaihin et al. 2019); during product evolution, analysing app reviews may help in identifying and prioritizing change requests (Villarroel et al. 2016; Gao et al. 2018b; Gao et al. 2019; Dąbrowski et al. 2020).

In recent years, scholars have been also studying on-line user feedback from other digital sources such as microblogs e.g., Twitter (Guzman et al. 2017), on-line forums e.g., Reddit (Khan et al. 2019), or issue tracking systems e.g., JIRA (Nyamawe et al. 2019). Most research efforts, however, have been focused on analyzing app reviews (Lim et al. 2021). Supposedly, the large number of this data, their availability and their usefulness make app reviews unique and thus the most frequently studied type of on-line user feedback (Lim et al. 2021).

Significant research has been devoted to study what relevant information can be found in app reviews; how the information can be analysed using manual and automatic approaches; and how the information can help software engineers. However, this knowledge is scattered in literature, and consequently there is no clear view on how app review analysis can support software engineering. The previous survey on app store data analysis (Martin et al. 2017) identified app review analysis as one important topic within the broader area of app store analysis but does not present a detailed comprehensive analysis of app review analysis techniques. Other literature reviews focus on specific types of review analysis such as opinion mining (Genc-Nayebi and Abran 2017) and information extraction (Tavakoli et al. 2018; Noei and Lyons 2019) but they do not cover the whole range of research on analysing app reviews. In contrast, this paper provides a systematic literature review of the whole range of research on analysing app reviews from the first paper published in 2012 up to the end of 2020. The paper objectives are to:

- identify and classify the range of app review analysis proposed in the literature;
- identify the range of natural language processing and data mining techniques that support such analysis;
- identify the range of software engineering activities that app review analysis can support;

– report the methods and results of the empirical evaluation of app review analysis approaches.

To accomplish these objectives, we have conducted a systematic literature review following a well-defined methodology that identifies, evaluates, and interprets the relevant studies with respect to specific research questions (Kitchenham 2004). After a systematic selection and screening procedure, we ended up with a set of 182 papers, covering the period 2012 to 2020, that were carefully examined to answer the research questions.

The primary contributions of the study are: (i) synthesis of approaches and techniques for mining app reviews, (ii) new knowledge on how software engineering scenarios can be supported by mining app reviews, (iii) a summary of empirical evaluation of review mining approaches, and finally (iv) a study of literature growth patterns, gaps, and directions for future research.

2 Research Method

To conduct our systematic literature review, we followed the methodology proposed by Kitchenham (2004). We first defined research questions and prepared a review protocol, which guided our conduct of the review and the collection of data. We then performed the literature search and selection based on agreed criteria. The selected studies were read thoroughly, and data items as in Table 3 were collected using a data extraction form. Finally, we synthesized the results for reporting.

2.1 Research Questions

The primary aim of the study is to understand how analysing app reviews can support software engineering. Based on the objective, the following research questions have been derived:

– **RQ1:** What are the different types of app review analyses?
– **RQ2:** What techniques are used to realize app review analyses?
– **RQ3:** What software engineering activities are claimed to be supported by analysing app reviews?
– **RQ4:** How are app review analysis approaches empirically evaluated?
– **RQ5:** How well do existing app review analysis approaches support software engineers?

The aim of RQ1 is to identify and classify the different types of app review analysis presented in primary literature; where an app review analysis refers to a task of examining, transforming, or modeling data with the goal of discovering useful information. The aim of RQ2 is to identify the range of techniques used to realize the different types of app review analysis identified in RQ1; where a technique stands for a way for facilitating an app review analysis. The aim of RQ3 is to identify the range of software engineering activities that have been claimed to be supported by analyzing app reviews; where a software engineering activity refers to a task performed along the software development life cycle (Bourque et al. 1999). The aim of RQ4 is to understand how primary studies obtain empirical evidences about effectiveness and the perceived-quality of their review analysis approaches. The aim of RQ5 is to summarize the results of empirical studies about effectiveness and user-perceived quality of different types of app review analysis.

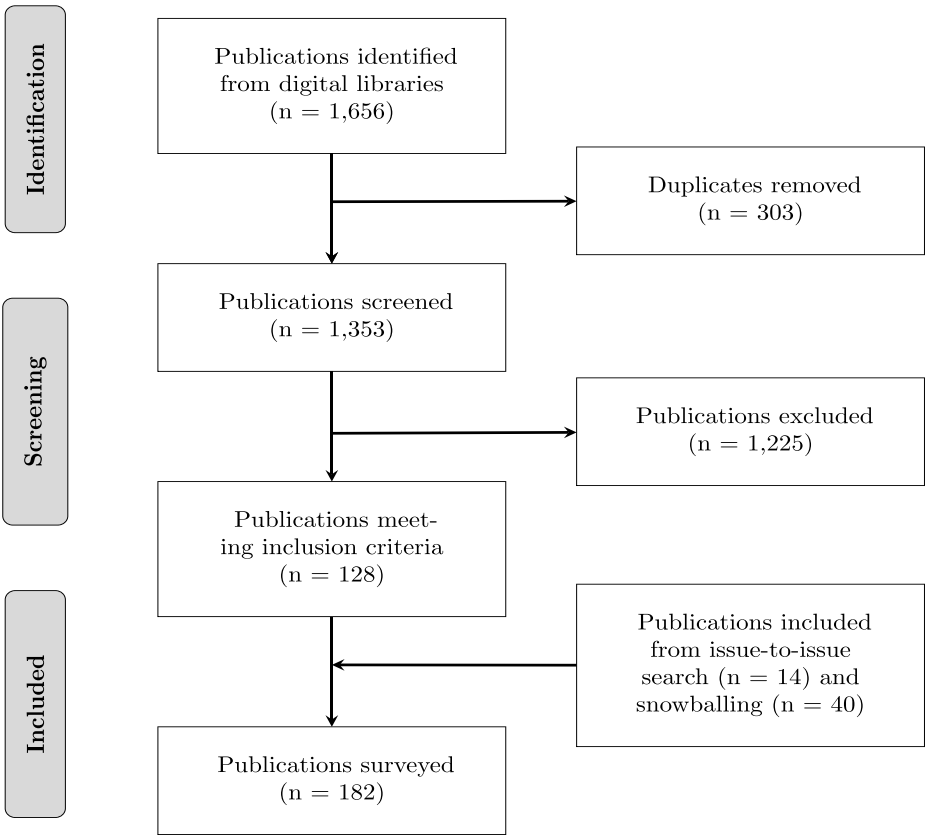

Fig. 1 PRISMA diagram showing study search and selection

2.2 Literature Search and Selection

We followed a systematic search and selection process to collect relevant literature published between January 2010[1] and December 2020. Figure 1 outlines the process as a PRISMA diagram[2]; it illustrates the main steps of the process and their outcomes (the number of publications).[3]

The initial identification of publications was performed using keyword-based search on six major digital libraries: *ACM Digital Library*, *IEEE Xplore Digital Library*, *Springer Link Online Library*, *Wiley Online Library* and *Elsevier Science Direct*. We defined two search queries that we applied in both the meta-data and full-text (when available) of the

[1]We selected 2010 to be the initial period of our search as the earliest study of app store analysis had been reported that year (Martin et al. 2017).

[2]A description of the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) method can be found in (Moher et al. 2009).

[3]The first author conducted the entire literature search and selection process.

publications. To construct the first query, we looked at the content of several dozen publications analysing reviews for software engineering.[4] We identified key terms that these papers share and used the terms to formulate a specific query:

```
('app review mining' OR 'mining user review' OR 'review mining' OR
'review analysis' OR 'analyzing user review' OR 'analyzing app review')
AND ('app store')
```

To not omit other relevant papers not covered by this specific query, we formulated a general query based on phrases reflecting key concepts of our research objective:

```
('app review' OR 'user review' OR 'app store review' OR 'user feedback')
AND ('software engineering' OR 'requirement engineering' OR 'software
requirement' OR 'software design' OR 'software construction' OR
'software testing' OR 'software maintenance' OR 'software configuration'
OR 'software development' OR 'software quality' OR 'software coding')
AND ('app store')
```

The initial search via digital libraries resulted in 1,656 studies, where 303 of them were duplicated. We screened 1,353 studies obtained through the initial search and selected them in accordance with the inclusion and exclusion criteria (see Table 1). To ensure the reliability of our screening process, the four authors of this paper independently classified a sample of 20 papers[5] (each paper was assigned to two authors). We then assessed their inter-rater agreement (Cohen's Kappa = 0.9) (Viera and Garrett 2005).

Due to the conservative searching, the majority of the studies were found to be unrelated to the scope of the survey. We excluded 1,225 publications that did not meet the inclusion criteria. Subsequently, we complemented our search process with two other strategies to find relevant papers that could have been missed in the initial search. We performed a manual *issue-by-issue* search of major conference proceedings and journals in software engineering in the period from January 2010 to December 2020. The searched journal and proceedings are listed in Table 2. That step produced another 14 unique publications. Finally, we completed the searching with a *snowballing* procedure following guidelines proposed by Wohlin (2014). We performed backward snowballing considering all the references from relevant studies found by previous searching strategies. Moreover, we conducted forward snowballing based on the 10 most cited papers. Using snowballing procedure, an additional 40 relevant articles were found to match our inclusion criteria. We used these criteria to screen the papers based on the title, abstract and full-text (if needed). Accordingly, we ended up with 182 articles included in the survey.

2.3 Data Extraction

The first author created a data extraction form to collect detailed contents for each of the selected studies. They used extracted data items to synthesize information from primary studies and answer research questions RQ1-RQ5. Table 3 presents the data items the first author extracted:

[4]We identified papers from previous surveys on app store analysis (Martin et al. 2017).

[5]We selected this number of studies to satisfy the sample size requirements for Cohen's Kappa calculation (Bujang and Baharum 2017).

Table 1 Inclusion and exclusion criteria

No.	Inclusion Criteria
1	Primary studies related to software engineering and may have actionable consequences for engineers or researchers
2	Peer-reviewed studies published as conference, journal, or workshops papers or a book chapter
3	Studies related to the use of app reviews in support to at least one software engineering activity (directly or indirectly) (Bourque et al. 1999)
No.	Exclusion Criteria
1	Papers not written in English
2	Papers analyzing app reviews without the purpose to support software engineering
3	Secondary or tertiary studies (e.g., systematic literature reviews, surveys, etc.) technical reports or manuals

- *Title, Author(s), Year, Venue, Citation* (F1-F5) are used to identify the paper and its bibliographic information. For F5, we record the citation count for each paper according to Google Scholar as of the 4th of August 2021).
- *Review Analysis* (F6) records the type of app review analysis (F6.1) (e.g. review classification), mined information (F6.2) (e.g. bug report) and supplementary description (F6.3).
- *Technique* (F7) records what techniques are used to realize the analysis. We recorded the technique type (F7.1) e.g., machine learning and its name (7.2) e.g., Naïve Bayes.
- *Software Engineering Activity* (F8) records the specific software engineering activities (e.g. requirements elicitation) mentioned in the paper as being supported by the proposed app review analysis method. We used widely known taxonomy of software engineering phases and activities to identify and record these items (Bourque et al. 1999).
- *Justification* (F9) records the paper's explanation for how the app review analysis support the software engineering activities. Some papers do not provide any justification.

Table 2 Selected conference proceedings and journals for manual search

Venue	Abbr.
International Conference on Software Engineering	ICSE
European Software Engineering Conference and Symposium on the Foundations of Software Engineering	ESEC/FSE
International Conference on Automated Software Engineering	ASE
International Conference on Software Maintenance and Evolution	ICSM/ICSME
Conference on Advanced Information Systems Engineering	CAiSE
International Requirements Engineering Conference	RE
IEEE Transactions on Software Engineering	TSE
ACM Transactions on Software Engineering and Methodology	TOSEM
IEEE Software	IEEE SW
Empirical Software Engineering	EMSE
Information and Software Technology	IST
Requirements Engineering Journal	REJ

Table 3 Data extraction form

Item ID	Field	Use
F1	Title	Documentation
F2	Author(s)	Documentation
F3	Year	Documentation
F4	Venue	Documentation
F5	Citation	Documentation
F6	Review Analysis	RQ1
F7	Mining Technique	RQ2
F8	Software Engineering Activity	RQ3
F9	Justification	RQ3
F10	Evaluation Objective	RQ4
F11	Evaluation Procedure	RQ4
F12	Evaluation Metrics and Criteria	RQ4
F13	Evaluation Result	RQ5
F14	Annotated Dataset	RQ4
F15	Annotation Task	RQ4
F16	Number of Annotators	RQ4
F17	Quality Measure	RQ4
F18	Replication Package	RQ4

- *Evaluation Objective* (F10) records the general objective of the paper's evaluation section (F10.1) (e.g. quantitative effectiveness, or user-perceived usefulness) and the type of evaluated app review analysis (F10.2).
- *Evaluation Procedure* (F11) records the paper's evaluation method and detailed evaluation steps.
- *Evaluation Metrics and Criteria* (F12) records the quantitative metrics (e.g. precision and recall) and criteria (e.g. usability) used in the evaluation.
- *Evaluation Result* (F13) records the result of empirical evaluation with respect to the evaluation metrics and criteria.
- *Annotated Dataset* (F14) records information about the datasets used in the study. We stored information about App Store name from which reviews were collected (F14.1) e.g., Google Play, and the number of annotated reviews (F14.2).
- *Annotation Task* (F15) records the task that humans annotators performed when labeling a sample of app reviews e.g., classify reviews by discussed issue types.
- *Number of Annotators* (F16) records number of human annotators labeling app reviews for empirical evaluation.
- Quality Measure (F17) are the measures used for assessing reliability of the annotated dataset e.g., Cohen's Kappa.
- *Replication Package* (F18) records whether a replication package is available. When one is available, we also recorded details about its content such as the availability of an annotated dataset, analysis method implementation, and experiment's scripts. In addition to the reported information; we contacted the authors of primary studies to check the availability of the replication packages.

The reliability of data extraction was evaluated through inter- and intra- rater agreements (Ide and Pustejovsky 2017). The agreements were measured using percentage

agreement on a recommended sample size (Graham et al. 2012; Bujang and Baharum 2017). To evaluate intra-rater agreement, the first author re-extracted data items from a random sample of 20% of selected papers. An external assessor[6] then validated the extraction results between the first and second rounds; and computed percentage agreement. To evaluate inter-rater agreement, the assessor cross-checked data extraction; the assessor independently extracted data items from a new random sample of 10% of selected papers. The first author and the assessor then compared their results and computed agreement. The intra-rater agreement was at the level of 93% whereas the inter-rater agreement was of 90%, indicating nearly the complete agreement (Ide and Pustejovsky 2017).

2.4 Data Synthesis

Most data in our review are grounded in qualitative research. As found by other researchers, tabulating the data is useful for aggregation, comparison, and synthesis of information (Kitchenham 2004). The data was thus stored in the spreadsheets, manually reviewed, and interpreted to answer research questions. Parts of the extracted data we synthesized using descriptive statistics.

We also used three classification schemas to group collected information on app review analysis (F6), mining techniques (F7) and SE activity (F8). We constructed each schema following the same general procedure based on the content analysis method (Bauer 2007); the first author initially examined all the collected information of a specific data item type; then performed an iterative coding process. During the coding, each information was labeled with one of the categories identified in the literature or inferred from the collected data.

To create the schema of app review analyses, we adopted 5 categories proposed in the previous survey (Martin et al. 2017). As the categories were not exhaustive for the coding; we extended them with 14 additional categories: 7 categories from the taxonomy of mining tasks (Cannataro and Comito 2003), and 7 standard types of text analytics (Miner et al. 2012); we referred to data and text mining areas as they have well defined terminology for text analysis. We then merged semantically-related categories; and removed those unrelated to the domain of app review analysis. The resulting list of 8 categories we then extended by adding the Recommendation category abstracted from the remaining unlabelled data. With 9 categories, the first author performed the final coding. Table 7, in the corresponding result section, presents the nine types of app review analyses.

The classification schema of mining techniques is informed by categories in previous survey on intelligent mining techniques (Tavakoli et al. 2018) and text analytics (Miner et al. 2012; Singh 2021; Software 2021). We first identified 5 categories of mining techniques: 4 categories proposed in the previous survey (Tavakoli et al. 2018); and 1 category identified from text analytics i.e., statistical analysis (Miner et al. 2012; Singh 2021; Software 2021). While coding, we however excluded feature extraction category referring to an instance of general information extraction task rather than a type of technique (Miner et al. 2012); and performed the final coding using the remaining 4 categories. The resulting mining techniques categories can be found in Table 9.

We derived the schema of SE activities based on the terminology from the software engineering body of knowledge (Bourque et al. 1999); we first identified 258 terms related to the main software engineering concepts; and then selected 58 terms describing candidate

[6]The assessor has an engineering background and experience with manual annotation; they has no relationship with this research.

Table 4 The intra- and inter-rater agreement for the classification schemas

Classification Schema	Intra-Rater Agreement	Inter-Rater Agreement
App Review Analysis	93%	87%
Software Engineering Task	100%	87%
Mining Technique	90%	80%

activities for the coding process. While coding, we excluded 44 terms as they did not match any data items; and performed the final coding using the remaining 14 terms (from now SE activities). Table 13 list the the resulting software engineering activities in the corresponding result section.

We validated the coding reliability of each schema using inter- and intra- rater agreement. We measured the reliability using percentage agreement on a recommended sample size (Graham et al. 2012; Bujang and Baharum 2017). To evaluate intra-rater agreement, the first authors re-coded a random sample of 20% of selected papers. The external assessor then checked the coding between the first and second coding. To evaluate inter-rater agreement, both the first author and the assessor coded a new random sample of 10% of the papers. They then cross-checked their results. The percentage intra- and inter-rater agreements were equal or above 90% and 80% for coding each schema, indicating their very good quality (Ide and Pustejovsky 2017); Table 4 provides detail statistics for the reliability evaluation.

The spreadsheets resulting from our data extraction and data grouping can be found in the supplementary material of this survey (Dąbrowski 2021).

3 Result Analysis

3.1 Demographics

Figure 2 shows the number of primary studies per year, including breakdown of publication type (Journal, Conference, Workshop, and Book). The publication date of primary studies ranges from 2012 to 2020.[7] We observed that 53% of the primary studies were published in the last 3 years, indicating a growing interest in research on analyzing app reviews to support software engineering.

Figure 3 shows the distribution by venue type: 65% of papers are published in conferences, 23% in journals, 10% in workshops and 2% as book chapters. Table 5 lists the top ten major venues in terms of the number of published papers.[8] The venues include the main conferences and journals in the software engineering community. Table 6 lists twenty most cited papers in the field of app review analysis for software engineering; and summarize their contributions. These studies advanced the field in substantial ways, or introduced influential ideas.

[7] No study was published in 2010 and 2011.

[8] The complete list of venues can be found in supplementary material (Dąbrowski 2021).

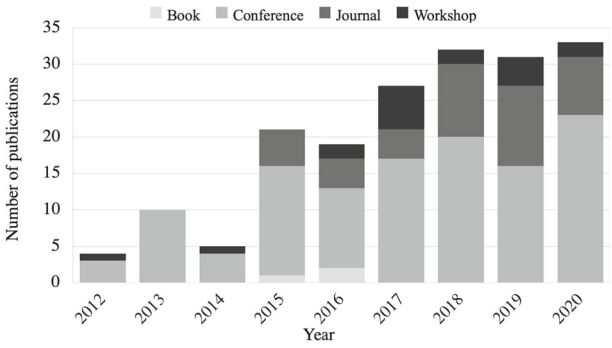

Fig. 2 Number of publications per year. The first papers on app review analysis were published in 2012

Key insights from demographics

- The interest in the research on app review analysis rose substantially in the last 3 years.
- The main venues publishing research in app review analysis include the main general software engineering conferences and journals (ICSE, FSE, ASE, IEEE Software) as well as the main specialized venues in empirical software engineering (ESEM) and requirements engineering (RE, REFSQ).

3.2 RQ1: App Review Analysis

In this section, we answer RQ1 (what are the different types of app review analysis) based on data extracted in F6 (review analyses). To answer the question, we grouped data items into one of nine general categories, each representing a different review analysis type (F6.1). We performed the grouping following the classification schema we had constructed for this study (see Section 2.4); and categories previously proposed in the context of app store analysis (Martin et al. 2017) as well as data and text mining (Cannataro and Comito 2003; Miner et al. 2012). Here, we focused on an abstract representation, because primary studies sometimes use slightly different terms to refer to the same type of analysis. Table 7 lists the different types of app review analyses and their prevalence in the literature.

3.2.1 Information Extraction

App reviews are unstructured text. Manually extracting relevant information from large volume of reviews is not cost-effective (Vu et al. 2015a). To address the problem, 56 of the primary studies (31%) proposed approaches facilitating information extraction. Formally, information extraction is the task of extracting specific (pre-specified) information from the content of a review; this information may concern app features (Guzman and Maalej 2014; Johann et al. 2017; Dąbrowski et al. 2020), qualities (Groen et al. 2017; Wang et al. 2020b), problem reports and/or new feature requests (e.g., Iacob and Harrison 2013; Wang et al. 2017; Gao et al. 2019; Shams et al. 2020), opinions about favored or unfavored features (e.g., Guzman and Maalej 2014; Gu and Kim 2015; Vu et al. 2015a; Li et al. 2017) as well

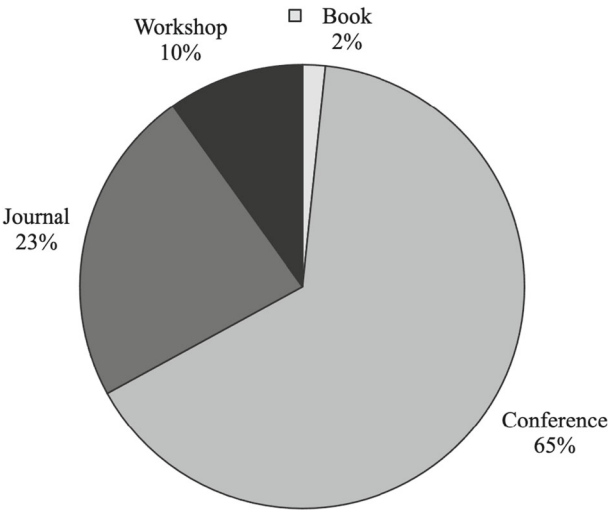

Fig. 3 Pie chart showing the distribution of research papers per venue type in the period from 2010 to December 31, 2020

as user stories (Guo and Singh 2020). Relevant information can be found at any location in the reviews. For instance, a problematic feature can be discussed in a middle of a sentence (Guzman and Maalej 2014; Williams et al. 2020), or a requested improvement can be expressed anywhere in a review (Gao et al. 2015; Guo and Singh 2020).

3.2.2 Classification

Classification consists of assigning predefined categories to reviews or textual snippets (e.g., sentences or phrases). Classification is by far the most common type of app review analysis found in the literature: 58% of publications describe techniques for classifying reviews. Classification can be used to separate informative reviews from those that are uninformative (e.g., Oh et al. 2013; Chen et al. 2014; Di Sorbo et al. 2016; Di Sorbo et al. 2020), spam (Chandy and Gu 2012) or fake (Martens and Maalej 2019b). Informative reviews can

Table 5 Top ten venues publishing papers on app review analysis between 2010 and 2020

Venues	No. Studies
International Requirements Engineering Conference (RE)	11
Empirical Software Engineering Journal (EMSE)	10
International Working Conference on Requirements Engineering (REFSQ)	7
International Conference on Software Engineering (ICSE)	7
IEEE Software (IEEE Softw)	6
International Symposium on Foundations of Software Engineering (FSE)	6
International Conference on Automated Software Engineering (ASE)	6
International Workshop on App Market Analytics (WAMA)	5
Intl. Conference on Mobile Software Engineering and Systems (MOBILESoft)	5
Intl. Conference on Evaluation and Assessment in Software Engineering (EASE)	5

Table 6 Twenty most influential papers in the field of app reviews analysis for software engineering, ordered by year of publication

Reference	Contribution	Citat.
Vasa et al. (2012)	Preformed the first preliminary analysis of mobile app reviews.	123
Carreño and Winbladh (2013)	Proposed an approach extracting requirements from feedback.	329
Fu et al. (2013)	Proposed WisCom system for analyzing millions of reviews.	415
Iacob and Harrison (2013)	Developed a tool extracting and summarizing user requests.	334
Pagano and Maalej (2013)	Studied the content and the usefulness of app reviews for RE.	514
Chen et al. (2014)	Developed AR-Miner tool for filtering and prioritizing reviews.	480
Guzman and Maalej (2014)	Proposed an approach for feature-based sentiment analysis.	531
Guzman et al. (2015)	Proposed ensemble methods for app review classification.	101
Khalid et al. (2015)	Studied user complains in reviews and their impact on ratings.	415
Maalej and Nabil (2015)	Benchmarked techniques for automatically classifying reviews.	381
Martin et al. (2015)	Studied the app sampling problem for app store mining.	121
Panichella et al. (2015)	Taxonomy and an approach for identyfing users' intentions.	352
Palomba et al. (2015)	CRISTALS approach facilitating reviews-to-code traceability.	156
Gu and Kim (2015)	Developed and evaluated SUR-Miner tool for opinion mining.	110
Vu et al. (2015a)	MARK framework searching and analyzing user opinions.	140
Di Sorbo et al. (2016)	SURF tool summarizing users' needs and topics from reviews.	197
Maalej et al. (2016)	Large-scale empirical study on classification techniques.	166
Maalej et al. (2016)	Proposal for utilizing on-line user feedback to support RE.	209
McIlroy et al. (2016)	Automatically analyzed the types of user issues in reviews.	126
Villarroel et al. (2016)	Automatic approach for release planning by review analysis.	205

be subsequently classified to detect user intentions (e.g., Maalej et al. 2016; Zhou et al. 2020) and discussion topics (e.g., Di Sorbo et al. 2017; van Vliet et al. 2020). User intentions include reporting an issue or requesting a new feature (Panichella et al. 2015; Panichella et al. 2016; Srisopha et al. 2020b).

Discussion topics include a variety of concerns such as installation problems, user interface, or price (Mujahid et al. 2017; Ciurumelea et al. 2018; Williams et al. 2020); topics concerning user perception e.g., rating, user experience or praise (Pagano and Maalej 2013;

Table 7 App review analysis types and their prevalence in the literature

App Review Analysis	No. Studies	Percentage
Information Extraction	56	31%
Classification	105	58%
Clustering	44	24%
Search and Information Retrieval	24	13%
Sentiment Analysis	40	22%
Content Analysis	54	30%
Recommendation	30	16%
Summarization	25	14%
Visualization	20	11%

Li et al. 2020); or topics reporting different types of issues (Khalid 2013; McIlroy et al. 2016; Tao et al. 2020). For instance, review classification has been proposed to detect different types of usability and user experience issues (Bakiu and Guzman 2017; Alqahtani and Orji 2019), quality concerns (Mercado et al. 2016; Wen and Chen 2020) or different types of security and privacy issues (Cen et al. 2014; Tao et al. 2020). Similarly, app store feedback can be classified by their reported requirements type (Yang and Liang 2015; Deocadez et al. 2017a; Lu and Liang 2017; Wang et al. 2018; Wang et al. 2018; Wen and Chen 2020). This could help distinguish reviews reporting functional requirements from those reporting non-functional requirements (Yang and Liang 2015; Deocadez et al. 2017a; Wang et al. 2018; Wang et al. 2020b); distilling non-functional requirements into fine-grained quality categories such as reliability, performance, or efficiency (Lu and Liang 2017; Wang et al. 2018). Another key use of the classification task is rationale mining; it involves detecting types of argumentations and justification users describe in reviews when making certain decisions, e.g. about upgrading, installing, or switching apps (Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018; Kunaefi and Aritsugi 2020).

3.2.3 Clustering

Clustering consists of organizing reviews, sentences, and/or snippets into groups (called a cluster) whose members share some similarity. Members in the same group are more similar (in some sense) to each other than to those in other groups. Unlike classification, clustering does not have predefined categories. Clustering is thus widely used as an exploratory analysis technique to infer topics commonly discussed by users (Pagano and Maalej 2013; Guzman et al. 2014; Guzman and Maalej 2014; Liu et al. 2018) and aggregate reviews containing semantically related information (Chen et al. 2014; Guzman et al. 2015; Palomba et al. 2017; Zhou et al. 2020). Clustering can be used for grouping reviews that request the same feature (Peng et al. 2016; Di Sorbo et al. 2016), report similar problems (Martin et al. 2015; Villarroel et al. 2016; Gao et al. 2018b; Williams et al. 2020), or discuss a similar characteristic of the app (Vu et al. 2016; Chen et al. 2019; Xiao et al. 2020). The generated clusters might help software engineers synthesize information from a group of reviews referring to the same topics rather than examining each review individually (Fu et al. 2013; Gao et al. 2015; Wang et al. 2017; Hadi and Fard 2020).

3.2.4 Search and Information Retrieval

Search and information retrieval concerns finding and tracing reviews (or their textual snippets) that match needed information. The task can be used to find reviews discussing a queried app feature (Vu et al. 2015a; Vu et al. 2015b; Dąbrowski et al. 2019), to obtain the most diverse user opinions in reviews (Guzman et al. 2015), or to trace what features described in the app description are discussed by users (Johann et al. 2017; Li et al. 2018). Information retrieval is also used to establish traceability links between app reviews and other software engineering artefacts (Palomba et al. 2015; Palomba et al. 2018), such as source code (Palomba et al. 2017; Zhou et al. 2020; Shams et al. 2020), stack tracers (Pelloni et al. 2018), issues from tracking systems (Palomba et al. 2015; Noei et al. 2019), and warnings from static analysis tools (Wei et al. 2017) in order to locate problems in source code (Palomba et al. 2017; Ciurumelea et al. 2017; Grano et al. 2018), suggest potential changes (Palomba et al. 2015; Palomba et al. 2017), or to flag errors and bugs in an application under test (Wei et al. 2017). Such traceability links can be also detected between reviews and feedback from other source like Twitter to study if the same issues are discussed

in both digital channels (Yadav and Fard 2020; Yadav et al. 2020; Oehri and Guzman 2020); or between reviews and goals in goal-model to understand the extent to which app satisfies the users' goals (Liu et al. 2020; Gao et al. 2020).

Table 8 summarizes types of data that have been combined with app reviews using search and information retrieval; indicates the purpose of the analysis; and provides references to primary studies.

3.2.5 Sentiment Analysis

Sentiment analysis (also known as opinion mining) refers to the task of interpreting user emotions in app reviews. The task consists in detecting the sentiment polarity (i.e., positive, neutral, or negative) in a full review (Martens and Johann 2017; Martens and Maalej 2019a; Srisopha et al. 2020c), in a sentence (Guzman and Maalej 2014; Panichella et al. 2015; Panichella et al. 2016), or on in a phrase (Gu and Kim 2015; Dąbrowski et al. 2020).

App reviews are a rich source of user opinions (Guzman and Maalej 2014; Malik et al. 2018; Masrury and Alamsyah 2019; Martens and Maalej 2019a; Wen and Chen 2020). Mining these opinions involves identifying user sentiment about discussed topics (Gu and Kim 2015; Dąbrowski et al. 2020), features (Guzman and Maalej 2014; Gunaratnam and Wickramarachchi 2020) or software qualities (Bakiu and Guzman 2017; Masrury and Alamsyah 2019; Franzmann et al. 2020). These opinions can help software engineers understand how users perceive their app (Guzman and Maalej 2014; Gu and Kim 2015; Huebner et al. 2018; Franzmann et al. 2020), discover users' requirements (Dąbrowski et al. 2019; Dalpiaz and Parente 2019) and preferences (Guzman and Maalej 2014; Bakiu and Guzman 2017;

Table 8 Types of data that have been combined with app reviews using search and information retrieval

Type of Data	Purpose
App Description	Use features from app descriptions to filter informative reviews (Johann et al. 2017); to discover 'hot' features (Johann et al. 2017); to understand users' preferences (Li et al. 2018); and to identify domain features (Liu et al. 2019).
Git Commit	Detect links between reviews and source code changes to analyze the impact of user feedback on the development process; to keep track on requests that have (not) been implemented (Palomba et al. 2015; Palomba et al. 2018).
Goal Model	Detect links between reviews and goals in goal model; to identify users' satisfaction w.r.t. these goals; or to recommend new goals that need to be satisfied by the app (Liu et al. 2020; Gao et al. 2020).
Issue Report	Detect links between reviews and issue to understand what reports have (not) be addressed (Palomba et al. 2015; Palomba et al. 2018); to identify issue report duplications; and to prioritize the issues (Noei et al. 2019).
Lint Warning	Recover the links between warnings from static analysis tools and app user reviews to support warning prioritization (Wei et al. 2017).
Source Code	Link reviews to source-code to locate components related to requested changes; to recommend software changes (Palomba et al. 2017; Zhou et al. 2020); to estimate the impact of the changes (Ciurumelea et al. 2017).
Stack Trace	Link reviews to stack trace to integrate user feedback into app testing (Grano et al. 2018); to augment testing report with contextual information that can ease the understanding a failure (Pelloni et al. 2018).
Tweet	Link reviews to user feedback from Twitter (Oehri and Guzman 2020); to integrate the feedback from both channel; and to understand what different issues are discussed by app users (Yadav and Fard 2020; Yadav et al. 2020).

Malik et al. 2018; Nicolai et al. 2019), and factors influencing sales and downloads of the app (Liang et al. 2015). Not surprisingly, knowing user opinions is an important information need developers seek to satisfy (Buse and Zimmermann 2012; Begel and Zimmermann 2014; Dąbrowski et al. 2020).

3.2.6 Content Analysis

Content analysis studies the presence of given words, themes, or concepts within app reviews.

For example, studies have analysed the relation between user ratings and the vocabulary and length of their reviews (Hoon et al. 2012; Vasa et al. 2012). Studies have shown that users discuss diverse topics in reviews (Pagano and Maalej 2013; Shams et al. 2020), such as app features, qualities (Williams and Mahmoud 2018; Franzmann et al. 2020), requirements (Wang et al. 2018; Wang et al. 2018) or issues (Khalid 2013; Alqahtani and Orji 2019; Kalaichelavan et al. 2020; Williams et al. 2020). For example, using content analysis, researchers analysed recurring types of issues reported by users (McIlroy et al. 2016; Wang et al. 2020a; Shams et al. 2020), their distribution in reviews as well as as relations between app issue type and other information such as price and rating (Iacob et al. 2013b; Hassan et al. 2018) or between issue type and code quality indicators (Di Sorbo et al. 2020). Interestingly, studies have pointed out that users' perception for the same apps can vary per country (Srisopha et al. 2019), user gender (Guzman and Paredes Rojas 2019), development framework (Malavolta et al. 2015a), and app store (Ali et al. 2017). Content analysis can be also beneficial for software engineers to understand whether cross-platform apps achieve consistency of users' perceptions across different app stores (Hu et al. 2018; Hu et al. 2019), or whether hybrid development tools achieve their main purpose: delivering an app that is perceived similarly by users across platforms (Hu et al. 2019). Finally, studying the dialogue between users and developers has shown evidences that the chances of users to update their rating for an app increase as result of developer's response to reviews (McIlroy et al. 2015; Hassan et al. 2018).

3.2.7 Recommendation

Recommendation task aims to suggest course of action that software engineers should follow. Several mining approaches, for instance (Chen et al. 2014; Villarroel et al. 2016; Scalabrino et al. 2019; Gao et al. 2020), have been proposed to recommend reviews that software engineers should investigate. These approaches typically assign priorities to a group of comments reporting the same bug (Gao et al. 2015; Man et al. 2016; Gao et al. 2018b), requesting the same modification or improvement (Villarroel et al. 2016; Keertipati et al. 2016; Scalabrino et al. 2019; Zhou et al. 2020). Such assigned priorities indicate relative importance of the information that these reviews convey from the users' perspective. Factors affecting the importance vary from e.g., the number of reviews in these groups (Chen et al. 2014; Zhou et al. 2020), to the influence of this feedback on app download (Tong et al. 2018), and the overall sentiment these comments convey (Licorish et al. 2017; Gunaratnam and Wickramarachchi 2020). In line with this direction, mining approaches have been elaborated to recommend feature refinement plans for the next release (Licorish et al. 2017; Zhang et al. 2019), to highlight static analysis warnings that developers should check (Wei et al. 2017), to recommend test cases triggering bugs (Shams et al. 2020), to indicate mobile devices that should be tested (Khalid et al. 2014), and to suggest reviews that developers should reply (Greenheld et al. 2018; Gao et al. 2019; Srisopha et al. 2020c); the

approaches can analogously recommend responses for these reviews (Greenheld et al. 2018; Gao et al. 2019), stimulating users to upgrade their ratings or to revise feedback to be more positive (McIlroy et al. 2015; Vu et al. 2019).

3.2.8 Summarization

Review summarization aims to provide a concise and precise summary of one or more reviews. Review summarisation can be performed based on common topics, user intentions, and user sentiment for each topic (e.g., Guzman and Maalej 2014; Ciurumelea et al. 2018; Liu et al. 2020). For example, Di Sorbo et al. (2016, 2017) proposed summarizing thousands of app reviews by an interactive report that suggest to software engineers what maintenance tasks need to be performed (e.g., bug fixing or feature enhancement) with respect to specific topics discussed in reviews (e.g., UI improvements). Other review summarization techniques give developers a quick overview about users' perception specific to core features of their apps (Iacob and Harrison 2013; Guzman and Maalej 2014; Xiao et al. 2020), software qualities (Ciurumelea et al. 2018), and/or main users' concerns (Iacob et al. 2013a; Iacob et al. 2016; Ciurumelea et al. 2017; Tao et al. 2020). With the addition of statistics e.g., the number of reviews discussing each topic or requesting specific changes, such a summary can help developers to prioritize their work by focusing on the most important modifications (Ciurumelea et al. 2017). In addition, such a summary can be exported to other software management tools e.g., GitHub, JIRA (Iacob et al. 2016) to generate new issue tickets and help in problems resolution (Phetrungnapha and Senivongse 2019).

3.2.9 Visualization

Visualization can aid developers in identifying patterns, trends and outliers, making it easier to interpret information mined from reviews (Guzman et al. 2014; Liu et al. 2020). To communicate information clearly and efficiently, review visualization uses tables, charts, and other graphical representations (Guzman et al. 2014; Maalej et al. 2016), accompanied by numerical data (Maalej et al. 2016; Bakiu and Guzman 2017). For example, Maalej et al. (2016) demonstrated that trend analysis of review type (e.g., bug report, feature request, user experience) over time can be used by software engineers as an overall indicator of how the project's health. Other studies proposed visualizing dynamics of main themes discussed in reviews to identify emerging issues (Gao et al. 2015; Gao et al. 2015; Gao et al. 2018b; Gao et al. 2019), or to show the issue distribution for an app across different app stores (Man et al. 2016). Simple statistics about these issue (e.g., 'How many reviews reported specific issues?') can give an overall idea about the main problems, in particular if compared against other apps (e.g., 'Do users complain more about security of my app compared to similar apps?'). Similarly, analyzing the evolution of user opinions and bug reports about specific features can help software engineers monitor the health of these features and to prioritize maintenance tasks (Vu et al. 2015a; Vu et al. 2016; Bakiu and Guzman 2017; Shah et al. 2019c). For instance, software engineers can analyse how often negative opinions emerge, for how long these opinions have been reported, and whether their frequency is rising or declining (Vu et al. 2015a; Gu and Kim 2015; Tao et al. 2020). This information could provide developers with evidence of the relative importance of these opinions from a users' perspective (Bakiu and Guzman 2017; Dąbrowski et al. 2019).

RQ1: App Review Analysis

- 9 broad types of review analyses have been identified in the literature: (1) information extraction; (2) classification; (3) clustering; (4) search and information retrieval; (5) sentiment analysis; (6) content analysis; (7) recommendation; (8) summarization and (9) visualization.
- Reviews classification, clustering, and information extraction are the mostly frequently applied automatic tasks; they help to group reviews, discover hidden patterns and to focus on relevant parts of reviews.
- Content analysis is used to characterize reviews, to identify discussed topics, and to explore information needs that can be satisfied by the feedback.
- Searching and information retrieval aids software engineers to query reviews with information of their interest, and to trace it over other software artefacts (e.g., stack traces, issue tracking system or goal-models) or other sources of on-line user feedback (e.g. tweets).
- Summarizing and visualizing information scattered across a large amount of reviews can aid developers in interpreting the information that could be costly and time-consuming to undertake if done manually.
- Mined information is commonly used to recommend engineers a course of their maintenances actions e.g., bugs in need of urgent intervention, or localizing the problem in the source code.

3.3 RQ2: Mining Techniques

App review analyses (see Section 3.2) are realized using different text mining techniques. In this section, we address RQ2 (what techniques are used to realize app review analysis) based on extracted data in F7 (mining technique) that we grouped following the classification schema we had constructed for this study (see Section 2.4). The categories of this schema comes from the survey on intelligent mining techniques and tools (Tavakoli et al. 2018) and text analytics area (Miner et al. 2012; Singh 2021; Software 2021).

In answer to this question, we identified 4 broad categories of mining techniques: content analysis (CA), natural language processing (NLP), machine learning (ML) and statistical analysis (SA). Table 9 lists the techniques and their prevalence in the literature. It can be observed more than a half of studies employed NLP or ML; whereas MA and SA were present in 25% and 29% of the studies. Table 10 reports how many studies used a certain technique to realize a given type of app review analysis. We observe that the NLP or ML are dominant for realizing app review analyses, except for Content Analysis that is mostly performed using MA or SA technique.

A single study frequently used the same type of technique for realizing several app review analyses (e.g., Clustering, Classification)[9]; on the other hand, we also recorded studies frequently combined the techniques together to perform a single app review analysis. Table 11

[9]No. studies, in the furthest right column, is thus less or equal than the sum of a row.

Table 9 Mining techniques and their prevalence in the literature

Mining Techniques	No. Studies	Percentage
Manual Analysis	45	25%
Natural Language Processing	113	62%
Machine Learning	108	59%
Statistical Analysis	53	29%

reports what combinations of techniques were used in the literature and how many studies used each combination for realizing a specific app review analysis.[10] The results indicates NLP and ML were mostly combined for Classification; MA and SA were used together for Content Analysis; whereas NLP and SA was adopted for Information Extraction. The following sections discuss each type of technique.

3.3.1 Manual Analysis

Scholars have shown an interest in manual analysis of app reviews (Kurtanovic and Maalej 2018; van Vliet et al. 2020). The technique is used to facilitate Content Analysis e.g., to understand topics users discuss (Pagano and Maalej 2013; Franzmann et al. 2020; Williams et al. 2020) and to develop a ground truth dataset for training and evaluating mining techniques (Kurtanović and Maalej 2017; Dąbrowski et al. 2020). Manual analysis typically takes a form of tagging a group of sample reviews with one or more meaningful tags (representing certain concepts). For example, tags might indicate types of user complaint (Khalid et al. 2015; Wang et al. 2020a), feature discussed in reviews (Maalej and Nabil 2015; Dąbrowski et al. 2020), or sentiment users expresses (Sänger et al. 2016). To make replicable and valid inferences upon manual analysis, studies perform it in a systematic manner. Figure 4 illustrates the overall procedure of manual analysis. Scholars first formulate the analysis objective corresponding to the exploration of review content (e.g., understanding types of user complaints) or the development of ground truth (e.g., labelling types of user feedback). They then select the reviews to be analysed, and specify the unit of analysis (e.g., a review or a sentence). Next, one or more humans (called 'coders') follow a coding process to systematically annotate the reviews. A coder examines a sample of reviews and tags them with specific concepts. Unless these concepts are known in advance or coders agree about the tagging, the step is iterative; When, for example, new concepts are identified, coders examine once again all the previously tagged reviews and check if they should be also tagged with the new concepts. Such iterations minimize the threat of human error when tagging the reviews. Once all the reviews are tagged, authors either analyse findings or use the dataset to evaluate other mining techniques (Stanik et al. 2019; Williams et al. 2020; Dąbrowski et al. 2020).

Manual analysis is time-consuming and require a vast human effort (Pagano and Maalej 2013; Guzman and Maalej 2014; van Vliet et al. 2020); a pilot study typically proceeds an actual analysis (Sänger et al. 2016; Kurtanović and Maalej 2017; Dąbrowski et al. 2020); subsequently the actual tagging, focusing on a statistically representative sample of reviews, takes places (Khalid et al. 2015). For example, Guzman and Maalej (2014) involved seven coders who independently tagged 2800 randomly sampled user reviews. For each review,

[10] A single study could use a certain combination of techniques to facilitate multiple review analyses. The total number, on the right hand side, is thus less than the sum of a row.

Table 10 How often primary studies used certain mining techniques to realise a type of app review analysis

		App Review Analysis								Studies	
		Information Extraction	Classification	Clustering	Search and Info. Retrieval	Sentiment Analysis	Content Analysis	Recommendation	Summarization	Number	Percentage
Technique	Manual Analysis	1	11	1	0	2	37	0	0	45	25%
	Natural Language Processing	36	55	13	24	19	5	9	10	113	62%
	Machine Learning	10	73	36	2	7	3	13	2	108	59%
	Statistical Analysis	9	2	1	1	0	23	11	12	53	29%

Table 11 How often primary studies used certain combination of techniques to realise a type of app review analysis; MA stands for manual analysis; NLP denotes natural language processing; ML marks machine learning; and SA signifies statistical analysis

		App Review Analysis								Studies	
		Information Extraction	Classification	Clustering	Search and Info. Retrieval	Sentiment Analysis	Content Analysis	Recommendation	Summarization	Number	Percentage
Comb. of Techniques	MA	1	11	1	0	2	25	0	0	31	17%
	NLP	32	15	4	22	13	1	5	7	67	37%
	ML	2	32	28	0	1	2	8	2	62	34%
	SA	0	0	0	0	0	11	10	9	30	16%
	MA + ML	0	0	0	0	0	1	0	0	1	1%
	MA + NLP	0	0	0	0	0	1	0	0	1	1%
	MA + SA	0	1	0	0	0	9	0	0	9	5%
	ML + SA	0	0	0	0	0	0	1	0	2	1%
	NLP + ML	8	39	8	2	6	0	4	0	53	29%
	NLP + SA	9	0	1	0	0	2	0	3	15	8%
	MA + NLP + SA	0	0	0	0	0	1	0	0	1	1%
	NLP + ML + SA	0	1	0	1	0	0	0	0	2	1%

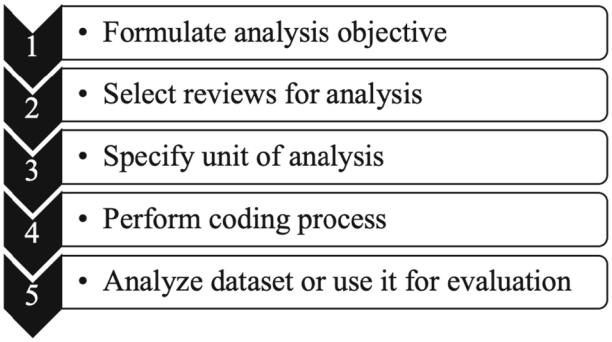

Fig. 4 Figure showing the overall process of manual analysis

two coders independently tagged the type of user feedback, features mentioned in the review and sentiments associated to these features. The study reports that coders spent between 8 and 12.5 hours for coding around 900 reviews.

3.3.2 Natural Language Processing

User-generated content of app reviews takes the form of text (Hoon et al. 2012; Vasa et al. 2012). Such text has plenty of linguistic structure intended for human consumption rather than for computers (Jurafsky and Martin 2009). The content must, therefore, undergo a good amount of natural language processing (NLP) before it can be used (Manning et al. 2008; Jurafsky and Martin 2009). Given this fact, it is not surprising that the majority of primary studies (62% of surveyed papers) adopt NLP techniques to support review analysis (see Section 3.2). At a high level, pre-processing can be simply seen as turning review content into a form that is analysable for a specific mining task (see Section 3.2). There are different ways to pre-process reviews including text normalization, cleaning and augmenting (Manning et al. 2008; Jurafsky and Martin 2009; Panichella et al. 2015; Gao et al. 2020). These pre-processing steps typically involve converting texts into lowercase (Fu et al. 2013; Sänger et al. 2016; Hadi and Fard 2020), breaking up a text into individual sentences (Lu and Liang 2017; Jha and Mahmoud 2017a; Zhou et al. 2020), separating out words i.e., tokenization (Iacob et al. 2016; Palomba et al. 2017; Al-Hawari 2020), spelling correction (Palomba et al. 2017; Grano et al. 2018) as well as turning words into their base forms e.g., stemming or lemmatization (Maalej and Nabil 2015; Lu and Liang 2017; Panichella et al. 2015; Xiao 2019). Of course, not all the review content is meaningful (Guzman and Maalej 2014; Chen et al. 2014; Oehri and Guzman 2020). Some parts are noisy and obstruct text analysis (Palomba et al. 2015; Palomba et al. 2017; Gunaratnam and Wickramarachchi 2020). The content is thus cleaned by removing punctuation (Puspaningrum et al. 2018; Hu et al. 2019), filtering out noisy words like stop words (Johann et al. 2017; Ciurumelea et al. 2017; Gunaratnam and Wickramarachchi 2020), or non-English words (Palomba et al. 2015; Stanik et al. 2019). Such normalized and cleaned text tends to be augmented with additional information based on linguistic analysis e.g., part-of-speech tagging (PoS) (Puspaningrum et al. 2018; Zhang et al. 2019; Gunaratnam and Wickramarachchi 2020) or dependency parsing (Gu and Kim 2015; Liu et al. 2018; Song et al. 2020).

A review can be modelled as a words sequence (Johann et al. 2017), bag-of-words (BoW) (Maalej and Nabil 2015) or in vector space model (VSM) (Vu et al. 2015a) to sereve as input for other mining techniques. In particular, primary studies refers to NLP techniques

comparing text similarity (Vu et al. 2015b; Wang et al. 2018), pattern matching (Groen et al. 2017; Johann et al. 2017; Song et al. 2020) and collocations finding (Guzman and Maalej 2014; Li et al. 2018; Dalpiaz and Parente 2019; Xiao et al. 2020).

Text similarity techniques (employed in 21 studies) determine how "close" two textual snippets (e.g., review sentences) are (Manning et al. 2008). These snippets, represented in VSM or BoW, are compared using similarity measure like Cosine similarity (Vu et al. 2015a; Shams et al. 2020), Dice similarity coefficient (Palomba et al. 2015; Zhou et al. 2020) or Jaccard index (Iacob et al. 2016). These techniques support Searching and Information Retrieval e.g., to link reviews with issue reports from issue tracking systems (Noei et al. 2019), Recommendation e.g., to recommend review responses based on old ones that have been posted to similar reviews (Greenheld et al. 2018), Clustering e.g., to group semantically similar user opinions (Vu et al. 2016; Malgaonkar et al. 2020), and Content Analysis e.g., to compare review content (Malavolta et al. 2015a).

Pattern matching techniques (employed in 22 studies) localize parts of review text (or its linguistic analysis) matching hand-crafted patterns. Such patterns can take many forms, such as, regular expressions (Yang and Liang 2015; Groen et al. 2017; Uddin et al. 2020), PoS sequences (Vu et al. 2016; Johann et al. 2017), dependencies between words (Gu and Kim 2015; Peng et al. 2016; Di Sorbo et al. 2017; Srisopha et al. 2020c) or simple keyword matching (Yang and Liang 2015; Maalej et al. 2016; Di Sorbo et al. 2017; Tao et al. 2020). The technique has been adopted in Information Extraction e.g., to extract requirements from reviews (Yang and Liang 2015; Groen et al. 2017), Classification e.g., to classify requirements into functional and non-functional (Yang and Liang 2015) and Summarization e.g., to provide a bug report summary (Groen et al. 2017).

Collocation finding techniques are employed for Information Extraction e.g., to extract features (Guzman and Maalej 2014; Xiao 2019) or issues (Gao et al. 2018b) from reviews. Such collocations are phrases consisting of two or more words, where these words appear side-by-side in a given context more commonly than the word parts appear separately (Jurafsky and Martin 2009). The two most common types of collocation detected in the primary studies are bigrams i.e., two adjacent words (Guzman and Maalej 2014; Dalpiaz and Parente 2019). Co-occurrences may be insufficient as phrases such as 'all the' may co-occur frequently but are not meaningful. Hence, primary studies explore several methods to filter out the most meaningful collocations, such as Pointwise Mutual Information (PMI) (Gao et al. 2018b; Malgaonkar et al. 2020) and hypothesis testing (Jurafsky and Martin 2009; Guzman and Maalej 2014; Dąbrowski et al. 2020).

3.3.3 Machine Learning

Overall, 108 of 182 primary studies (59%) reported the use of machine learning (ML) techniques to facilitate mining tasks and review analysis. Table 12 reports ten most commonly applied ML techniques. Most of them (i.e., 8 techniques) are supervised, whereas 2 of them are unsupervised (Bishop 2006). The widespread interest in ML techniques may be attributed to the fact that Clustering e.g., to group reviews discussing the same topics (Fu et al. 2013; Srisopha et al. 2020b) and Classification e.g., to categorize user feedback based on user intention (Dhinakaran et al. 2018; Zhou et al. 2020), among the most common review analysis types (see Table 7), are mainly facilitated using ML. When looking at the whole spectrum of review analysis these ML techniques support, we have also recorded their use for Sentiment Analysis e.g., to identify feature-specific sentiment (Gu and Kim 2015), Recommendation e.g., to assign priorities to reviews reporting bugs (Villarroel et al. 2016) and Information Extraction e.g., to identify features (Sänger et al. 2017; Wang et al. 2020b).

Table 12 Distribution of machine learning techniques used in primary studies in the period form 2010 to December 31, 2020

Type	Machine Learning Techniques	No. Studies	Percentage
Supervised	Naïve Bayes	43	24%
	Support Vector Machine	39	21%
	Decision Tree	31	187%
	Logistic Regression	23	13%
	Random Forest	20	1%
	Neural Network	12	7%
	Linear Regression	7	4%
	K-Nearest Neighbor	4	2%
Unsupervised	Latent Dirichlet Allocation	36	20%
	K-Means	8	4%

Scholars experimented with many textual and non-textual review properties[11] to make ML techniques work best (Maalej and Nabil 2015; Guzman et al. 2015). Choosing informative and independent properties is a crucial step to make these techniques effective (Bishop 2006; Maalej et al. 2016). Textual properties, for example, concern: text length, tense of text (Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018), importance of words e.g., td-idf (Lu and Liang 2017; Williams et al. 2020), a word sequence e.g., n-gram (Maalej and Nabil 2015; Al-Hawari 2020) as well as linguistic analysis e.g., dependency relationship (Shah et al. 2018). These properties are commonly combined with non-textual properties like user sentiment (Maalej et al. 2016; Srisopha et al. 2020a), review rating (Kurtanović and Maalej 2017) or app category (Gao et al. 2019). We found that primary studies experiment with different properties (Maalej et al. 2016; Kurtanovic and Maalej 2018; Al-Hawari 2020).

3.3.4 Statistical Analysis

Statistical analysis is used in many papers to report research results (Martin et al. 2015; Sänger et al. 2016; Di Sorbo et al. 2020), demonstrate their significance (Vasa et al. 2012; Khalid et al. 2016), and draw conclusions of a large population of reviews by analysing their tiny portion (Pagano and Maalej 2013; Mercado et al. 2016; Wang et al. 2020a). We observed an interest in use of descriptive and inferential techniques for Content Analysis e.g., Vasa et al. (2012), Pagano and Maalej (2013), Mercado et al. (2016), Guzman et al. (2018), and Wang et al. (2020a). Summary statistics, box plots, and cumulative distribution charts help to gain understanding of review characteristics like their vocabulary size (Hoon et al. 2012; Vasa et al. 2012), issue type distribution (McIlroy et al. 2016; Hu et al. 2018; Williams et al. 2020), or topics these reviews convey (Pagano and Maalej 2013; Srisopha and Alfayez 2018). Scholars employ different statistical tests to test check their hypothesis (Khalid et al. 2016; Guzman and Paredes Rojas 2019; Franzmann et al. 2020), to examine relationship between reviews characteristics (Srisopha and Alfayez 2018; Guzman and Paredes Rojas 2019; Di Sorbo et al. 2020), and to study how sampling bias affects the validity of research results (Martin et al. 2015).

[11] We refer to a property as a concept denoting a feature in the machine learning domain.

Guzman et al. (2018) and Guzman and Paredes Rojas (2019), for example, conducted an exploratory study investigating 919 reviews from eight countries. They studied how reviews written by male and female users differ in terms of content, sentiment, rating, timing, and length. The authors employed Chi-square (e.g., content) and Mann-Whitney (e.g., rating) non-parametric tests for nominal and ordinal variables respectively (Guzman and Paredes Rojas 2019). Srisopha and Alfayez (2018) studied whether a relationship exists between user satisfaction and the application's internal quality characteristics. Having employed Pearson correlation coefficient, the authors studied to what extent do warnings reported by static code analysis tools correlate with different types of user feedback and the average user ratings. Similarly, another study employed the Mann-Whitney test to examine if densities of such warnings differ between apps with high and low ratings (Khalid et al. 2016).

RQ2: Mining Techniques

- Primary studies employ 4 broad types of techniques to realize app review analyses: (1) manual analysis; (2) natural language processing; (3) machine learning and (4) statistical analysis.
- Manual analysis is used to study review content; and to develop datasets for training/evaluating data mining techniques. The technique is time-consuming and requires substantial human effort.
- NLP techniques play an important role for review analysis. The majority of primary studies (62%) use the techniques for a wide spectrum of review analyses: Search and Information Retrieval, Classification, Clustering, Content Analysis, Information Extraction, Summarization or Recommendation.
- ML is employed by ca. 59% of papers for Clustering, Classification, Sentiment Analysis, Recommendation, or Information Extraction. Scholars experiment with textual and non-textual review properties to boost the effectiveness of the techniques.
- Statistical analysis is used to support Content Analysis: to summarize findings; to draw statistically significant conclusions; or to check their validity.

3.4 RQ3: Supporting Software Engineering

To answer RQ3 (what software engineering activities might be supported by analysing app reviews), we used data extracted in F8 (software engineering activity) and F9 (justification) as well as the classification schema of SE activities derived from the software engineering body of knowledge (see Section 2.4). Table 13 provides mapping between primary studies and SE activities that the studies claim to support[12]; it also reports the number and the percentage of the studies per each activity. We can observe that primary studies motivated their approaches to support activities across different software engineering phases, including requirements (36%), maintenance (36%), testing (15%) and design (4%); 14 SE activities are supported in total; mostly research effort is focused on requirements elicitation (26%),

[12]It is worth noting that some papers fall into more than one category i.e., claim to support more than one activity. In such case, we assigned the study to all the claimed activities.

requirements prioritization (10%), validation by users (11%), problem and modification analysis (23%), and requested modification prioritization (11%). We also recorded that 62 studies (34%) did not specify any SE activity that their approaches support.

To support the SE activities, primary studies used 9 broad types of app review analysis we identified with answer to RQ1 (see Section 3.2). Table 14 shows how often a type of review analysis was used for a SE activity.[13] It can be observed that each SE activity was supported using multiple analyses; classification was the most commonly used one; this was also the only analysis motivated for all the activities. A further result analysis revealed studies used the analyses in combination to mine useful information and support SE activities; we recorded 53 unique combinations; each composed of 1 to 5 types of analysis with the median of 2. Table 15 lists combinations used at least in 2 primary studies. The following sections provides a through synthesis on how mining useful information from app reviews might support SE activities.

3.4.1 Requirements

Requirements engineering includes involving system users, obtaining their feedback and agreeing on the purpose of a software to be built (Maalej et al. 2016). It therefore is not surprising that review analysis has received much attention to support requirements engineering activities, including requirements elicitation, requirements classification, requirements prioritization and requirements specification (see Table 13).

Requirements Elicitation In app reviews, users give feedback describing their experience with apps, expressing their satisfaction with software products and raising needs for improvements (Pagano and Maalej 2013; AlSubaihin et al. 2019). Software engineers can make use of the reviews to elicit new requirements (AlSubaihin et al. 2019; Dalpiaz and Parente 2019; Dąbrowski et al. 2019; 2020). For instance, they can employ opinion mining approaches to examine reviews talking negatively about app features (Guzman and Maalej 2014; Shah et al. 2016; Li et al. 2018; Shah et al. 2019c; Liu et al. 2019; Dalpiaz and Parente 2019; Dąbrowski et al. 2019; 2020). This can help developers to understand user concerns about problematic features, and potentially help eliciting new requirements (Johann et al. 2017; Dalpiaz and Parente 2019; Dąbrowski et al. 2019; 2020). Additionally, searching and retrieving users reviews that refer to a specific feature they are responsible for will allow them to quickly identify what users have been saying about their feature (Li et al. 2018; Dąbrowski et al. 2019; Liu et al. 2019). In line with this direction, approaches have been proposed to classify reviews by their user intention (e.g., reviewer requesting a new feature) (Iacob et al. 2013a; Maalej and Nabil 2015; Maalej et al. 2016; Villarroel et al. 2016; Scalabrino et al. 2019; Song et al. 2020) and by the type of requirements these reviews formulate (e.g., functional or non-functional) (Yang and Liang 2015; Lu and Liang 2017; Al Kilani et al. 2019; Jha and Mahmoud 2019; Wen and Chen 2020). Such aggregated information can be further summarized and visualized to developers as a report of all the feature requests reported for an app (Iacob et al. 2013a; Iacob et al. 2016; Di Sorbo et al. 2016; Di Sorbo et al. 2017; Ciurumelea et al. 2018; Liu et al. 2020).

Requirements Classification User feedback can be classified in a number of dimensions (Bourque et al. 1999). Several studies classified user comments based on types of

[13] Table excludes papers that did not specify any SE activity; in case of papers supporting multiple SE activities, we assigned their facilitated analyses to all the claimed activities.

Table 13 Software engineering activities supported by app review analysis

SE Activity	No. Studies	Percentage	Reference
REQUIREMENTS	66	36%	
Requirements Elicitation	55	30%	Goul et al. (2012), Iacob and Harrison (2013), Carreño and Winbladh (2013), Oh et al. (2013), Pagano and Maalej (2013), Iacob et al. (2013a), Guzman and Maalej (2014), Sun and Peng (2015), Gao et al. (2015), Maalej and Nabil (2015), Yang and Liang (2015), Maalej et al. (2016), Shah et al. (2016), Iacob et al. (2016), Di Sorbo et al. (2016), Peng et al. (2016), Maalej et al. (2016), Villarroel et al. (2016), Di Sorbo et al. (2017), Johann et al. (2017), Abad et al. (2017), Groen et al. (2017), Zhang et al. (2017), Deocadez et al. (2017b), Lu and Liang (2017), Williams and Mahmoud (2018), Dhinakaran et al. (2018), Ciurumelea et al. (2018), Li et al. (2018), Wang et al. (2018), AlSubaihin et al. (2019), Liu et al. (2019), Chen et al. (2019), Jha and Mahmoud (2019), Dąbrowski et al. (2019), Dalpiaz and Parente (2019), Al Kilani et al. (2019), Scalabrino et al. (2019), Shah et al. (2019c), Dąbrowski et al. (2020), Srisopha et al. (2020b), Song et al. (2020), Uddin et al. (2020), Wen and Chen (2020), Gao et al. (2020), Tizard et al. (2020), van Vliet et al. (2020), Guo and Singh (2020), Li et al. (2020), Hadi and Fard (2020), Wang et al. (2020b), Kunaefi and Aritsugi (2020), Xiao et al. (2020), Liu et al. (2020), and Sharma and Bashir (2020)
Requirements Classification	10	5%	Yang and Liang (2015), Deocadez et al. (2017a), Groen et al. (2017), Lu and Liang (2017), Wang et al. (2018), Wang et al. (2018), Jha and Mahmoud (2019), Wen and Chen (2020), van Vliet et al. (2020), and Wang et al. (2020b)
Requirements Prioritization	19	10%	Hoon et al. (2012), Vasa et al. (2012), Pagano and Maalej (2013), Guzman and Maalej (2014), Guzman et al. (2015), Maalej et al. (2016), Villarroel et al. (2016), Johann et al. (2017), Kurtanović and Maalej (2017), Groen et al. (2017), Kurtanovic and Maalej (2018), Scalabrino et al. (2019), Shah et al. (2019c), Zhang et al. (2019), AlSubaihin et al. (2019), Dąbrowski et al. (2020), Srisopha et al. (2020b), Oehri and Guzman (2020), and Di Sorbo et al. (2020)
Requirements Specification	6	3%	Pagano and Maalej (2013), Maalej et al. (2016), Maalej et al. (2016), Kurtanović and Maalej (2017), Kurtanovic and Maalej (2018), and Williams et al. (2020)
DESIGN	8	4%	
Design Rationale Capture	5	3%	Groen et al. (2017), Kurtanović and Maalej (2017), Kurtanovic and Maalej (2018), Jha and Mahmoud (2019), and Kunaefi and Aritsugi (2020)
User Interface Design	3	2%	Alqahtani and Orji (2019), Franzmann et al. (2020), and Sharma and Bashir (2020)
TESTING	28	15%	
Validation by Users	20	11%	Fu et al. (2013), Iacob and Harrison (2013), Iacob et al. (2013a), Iacob et al. (2013b), Guzman et al. (2014), Guzman and Maalej (2014), Maalej and Nabil (2015), Gu and Kim (2015), Man et al. (2016), Maalej et al. (2016), Bakiu and Guzman (2017), Ciurumelea et al. (2018), Liu et al. (2018), Durelli et al. (2018), Shah et al. (2019c), Gao et al. (2019), AlSubaihin et al. (2019), Xiao (2019), Dąbrowski et al. (2020), and Xiao et al. (2020)

Table 13 (continued)

SE Activity	No. Studies	Percentage	Reference
Test Documentation	3	2%	Iacob et al. (2016), Grano et al. (2018), and Pelloni et al. (2018)
Test Design	4	2%	Man et al. (2016), Maalej et al. (2016), Groen et al. (2017), and Shams et al. (2020)
Test Prioritization	3	2%	Khalid et al. (2014), Khalid et al. (2015), and Man et al. (2016)
MAINTENANCE	66	36%	
Problem and Modification Analysis	46	25%	Fu et al. (2013), Khalid (2013), Cen et al. (2014), Guzman et al. (2014), Gomez et al. (2015), Gao et al. (2015), Khalid et al. (2015), Khalid et al. (2015b), Guzman et al. (2015), Palomba et al. (2015), Gao et al. (2015), Panichella et al. (2015), Vu et al. (2016), Di Sorbo et al. (2016), Malik and Shakshuki (2016), Iacob et al. (2016), Johann et al. (2017), Palomba et al. (2017), Wei et al. (2017), Bakiu and Guzman (2017), Licorish et al. (2017), Wang et al. (2017), Deocadez et al. (2017b), Pelloni et al. (2018), Palomba et al. (2018), Gao et al. (2018b), Tong et al. (2018), Muñoz et al. (2018), Malik et al. (2018), Dalpiaz and Parente (2019), Phetrungnapha and Senivongse (2019), Gao et al. (2019), Shah et al. (2019c), AlSubaihin et al. (2019), Uddin et al. (2020), Wen and Chen (2020), Malgaonkar et al. (2020), Zhou et al. (2020), Wang et al. (2020a), Tizard et al. (2020), Tao et al. (2020), Al-Hawari (2020), Guo and Singh (2020), Li et al. (2020), Hadi and Fard (2020), and Xiao (2019)
Requested Modification Prioritization	18	10%	Khalid (2013), Gu and Kim (2015), Gao et al. (2015), Khalid et al. (2015), Keertipati et al. (2016), Iacob et al. (2016), Villarroel et al. (2016), Man et al. (2016), Licorish et al. (2017), Wei et al. (2017), Muñoz et al. (2018), Hu et al. (2019), Noei et al. (2019), Noei et al. (2019), Dąbrowski et al. (2019), Scalabrino et al. (2019), Oehri and Guzman (2020), and Di Sorbo et al. (2020)
Help Desk	7	4%	McIlroy et al. (2015), Hassan et al. (2018), Greenheld et al. (2018), Gao et al. (2019), Vu et al. (2019), Srisopha et al. (2020c), and Srisopha et al. (2020a)
Impact Analysis	5	3%	Palomba et al. (2015), Ciurumelea et al. (2017), Palomba et al. (2017), Palomba et al. (2018), and Zhou et al. (2020)
NOT SPECIFIED	62	34%	Chandy and Gu (2012), Ha and Wagner (2013), Hoon et al. (2013), Chen et al. (2014), Martin et al. (2015), Khalid et al. (2015a), Guzman et al. (2015), Vu et al. (2015a), Vu et al. (2015b), Liang et al. (2015), Malavolta et al. (2015a), Malavolta et al. (2015b), Hoon et al. (2016), Sänger et al. (2016), McIlroy et al. (2016), Nagappan and Shihab (2016), Panichella et al. (2016), Mercado et al. (2016), Simmons and Hoon (2016), Khalid et al. (2016), Sänger et al. (2017), Nayebi et al. (2017), Grano et al. (2017), Ali et al. (2017), Martens and Johann (2017), Jha and Mahmoud (2017a), Mujahid et al. (2017), Li et al. (2017), Sun et al. (2017), Jha and Mahmoud (2017b), Wang et al. (2017), Mcilroy et al. (2017), Nayebi et al. (2018), Hu et al. (2018), Srisopha and Alfayez (2018), Deshpande and Rokne (2018), Puspaningrum et al. (2018), Guzman et al. (2018), Scoccia et al. (2018), Noei et al. (2018), Huebner et al. (2018), Gao et al. (2018a), Mujahid et al. (2018), Jha and Mahmoud (2018), Shah et al. (2018), Hassan et al. (2018), Shah et al. (2019a), Martens and Maalej (2019b), Nicolai et al. (2019), Stanik et al. (2019), Srisopha et al. (2019), Guzman and Paredes Rojas (2019), Masrury and Alamsyah (2019), Martens and Maalej (2019a), Weichbroth and Baj-Rogowska (2019), Shah et al. (2019b), Maalej et al. (2019), Bailey et al. (2019), Yadav and Fard (2020), Yadav et al. (2020), Gunaratnam and Wickramarachchi (2020), and Kalaichelavan et al. (2020)

Table 14 How often a type of app review analysis are used to realise a SE activity

		Software Engineering Activity														Studies	
		Requirements				Design		Testing				Maintenance					
		Requirements Elicitation	Requirements Classification	Requirements Prioritization	Requirements Specification	Design Rationale Capture	User Interface Design	Validation by Users	Test Documentation	Test Design	Test Prioritization	Problem and Mods. Analysis	Requested Mods. Prioritization	Help Desk	Impact Analysis	**Number**	**Percentage**
App Review Analysis	Information Extraction	25	4	6	1	1	0	13	1	3	1	20	5	0	0	56	31%
	Classification	32	9	7	5	4	1	9	2	1	1	27	11	4	3	105	58%
	Clustering	13	0	4	2	0	0	8	0	0	0	13	7	0	2	44	24%
	Search and Info. Retrieval	8	0	1	0	0	0	0	3	1	0	9	5	0	5	24	13%
	Sentiment Analysis	14	1	3	0	0	1	10	0	0	0	12	3	1	0	40	22%
	Content Analysis	10	2	6	2	1	3	4	0	1	2	8	5	4	0	54	30%
	Recommendation	7	0	3	0	0	2	2	0	2	1	12	8	3	2	30	16%
	Summarization	14	0	1	0	0	0	5	2	0	0	9	1	0	3	25	14%
	Visualization	5	0	1	1	0	0	8	0	1	1	10	4	1	0	20	11%

requirements the feedback conveys (Yang and Liang 2015; Deocadez et al. 2017a; Lu and Liang 2017; Groen et al. 2017; Wang et al. 2018; Wang et al. 2018; Jha and Mahmoud 2019; Wen and Chen 2020). These works typically classified the feedback into two broad categories: functional requirements (FRs) specifying the behavior of an app, and non-functional requirements (NFRs) describing the constraints and quality characteristics of the app. The

Table 15 How often certain combination of app review analyses are used to realise a SE activity; IE refers to Information Extraction; CL denotes Classification; CU signifies Clustering; CA presents Content Analysis; SA denotes Sentiment Analysis; SIR refers to Search and Information Retrieval; RE presents Recommendation; SU denotes Summarization; and VI signifies Visualization

		Software Engineering Activity														Studies	
		Requirements				Design		Testing				Maintenance					
		Requirements Elicitation	Requirements Classification	Requirements Prioritization	Requirements Specification	Design Rationale Capture	User Interface Design	Validation by Users	Test Documentation	Test Design	Test Prioritization	Problem and Mods. Analysis	Requested Mods. Prioritization	Help Desk	Impact Analysis	**Number**	**Percentage**
Comb. of App Review Analysis	CL	9	4	4	2	3	0	1	0	0	0	6	1	1	0	18	10%
	CU	2	0	0	0	0	0	0	0	0	0	1	1	0	0	3	2%
	CA	1	0	2	0	0	1	1	0	0	2	3	1	1	0	9	5%
	CL + CA	2	2	0	0	1	1	0	0	0	0	2	1	0	0	6	3%
	CL + SU	2	0	0	0	0	0	1	0	0	0	0	0	0	0	2	1%
	SIR + SU	0	0	0	0	0	0	0	0	0	0	2	0	0	2	2	1%
	CA + RE	0	0	0	0	0	0	0	0	0	0	1	2	1	0	2	1%
	IE + CA + SU	2	0	0	0	0	0	2	0	0	0	0	0	0	0	2	1%
	IE + CL + CU	1	0	0	0	0	0	1	0	0	0	1	1	0	0	4	2%
	CL + SIR + SU	0	0	0	0	0	0	0	1	0	0	1	0	0	1	2	1%
	CL + CU + SU	1	0	0	0	0	0	0	0	0	0	2	0	0	0	2	1%
	IE + SA + RE	1	0	0	0	0	0	0	0	0	0	1	0	0	0	2	1%
	IE + SA + CL + VI	1	0	1	0	0	0	2	0	0	0	2	0	0	0	2	1%
	IE + CL + CU + RE + VI	0	0	0	0	0	0	0	0	0	0	2	1	0	0	2	1%

classification at a further level of granularity has been also demonstrated (Lu and Liang 2017; Wang et al. 2018; Jha and Mahmoud 2019; Wen and Chen 2020; van Vliet et al. 2020); User feedback can be classified into the concrete quality characteristics it refers to e.g., defined by ISO 25010 model (ISO/IEC 25010 2011) so that software engineers could analyse candidate requirements more efficiently.

Requirements Prioritization Statistics about user opinions and requests can help prioritizing software maintenance and evolution tasks (Pagano and Maalej 2013; Guzman and Maalej 2014; Maalej et al. 2016; Johann et al. 2017; Dąbrowski et al. 2019; 2020). Bugs and missing features that are more commonly reported can be prioritized over those less commonly reported (Villarroel et al. 2016; Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018; Scalabrino et al. 2019; Di Sorbo et al. 2020). Users' request may not by themselves be sufficient for prioritization (one must also consider costs and the needs of other stakeholders) but can provide valuable evidence-based information to support prioritization (Maalej et al. 2016; Shah et al. 2019c; Oehri and Guzman 2020).

Requirements Specification Requirements specification consists in structuring and documenting detailed descriptions of the software required behaviour and quality properties (van Lamsweerde 2009). App reviews can instead serve for generating lightweight partial documentation of user requirements; they conveys information about functional and non-functional requirements, usage scenarios and user experience (Pagano and Maalej 2013; Maalej et al. 2016; Maalej et al. 2016; Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018; Williams et al. 2020). Software engineers can immediately benefit from review mining approaches to facilitate this information in the form of first drafts of software requirements specifications (SRS) or user stories (Pagano and Maalej 2013; Maalej et al. 2016; Maalej et al. 2016). These approaches can for example classify reviews by the type of requests users make (e.g., asking for new functions); summarise reviews referring to the same requests and generate provisional SRS based on the information. Such SRS may list new functions that users require; recap scenarios in which these functions are used; and report statistics indicating relative importance of the requirements e.g., by the number of users requesting the functions (Maalej et al. 2016). Since users often justify their needs and opinions, SRS may also document user rationales serving later for requirements negotiation or design decisions (Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018).

3.4.2 Design

A few studies motivated app review analysis to assist software design activities: user interface (UI) design (Alqahtani and Orji 2019; Sharma and Bashir 2020; Franzmann et al. 2020) and capturing design rationale (Groen et al. 2017; Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018; Jha and Mahmoud 2019; Kunaefi and Aritsugi 2020).

User Interface Design The success of mobile applications depends substantially on user experience (AlSubaihin et al. 2019; Franzmann et al. 2020). For the app to be successful, software engineers should design the interface to match the experience, skills and needs of users (Bourque et al. 1999). Alqahtani and Orji performed the content analysis of user reviews to identify usability issues in mental health apps (Alqahtani and Orji 2019). They manually tagged 1,236 reviews with different types of usability issues for 106 apps from

Apple's App Store and Google Play. Poor design of user interface was the second most frequently reported issue. It has been found that user-submitted content concerning interface may provide valuable design recommendations on how to improve interface layout, boost readability and easy app navigation. UI/UX designers should therefore take advantage of the feedback. If addressed, it would likely increase user engagement with the apps and reduce the attrition rate (Franzmann et al. 2020).

Design Rationale Capture Design rationale is essential for making the right design decisions and for evaluating architectural alternatives for a software system (Nuseibeh 2001; Burge et al. 2008). A few studies motivated their approaches to capture potential reasons for design decisions (Groen et al. 2017; Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018; Jha and Mahmoud 2019; Kunaefi and Aritsugi 2020). Kurtanović and Maalej devised a grounded theory for gathering user rationale and evaluated different review classification approaches to mine the information from app reviews (Kurtanović and Maalej 2017; Kurtanovic and Maalej 2018). User justifications e.g., on problems they encounter or criteria they chose for app assessment (e.g., reliability or performance) can enrich documentation with new design rationale and guide design decisions. Similarly, user-reported NFR can convey architecturally significant requirements and serve as rationale behind an architecture decision (Nuseibeh 2001; Groen et al. 2017; Kunaefi and Aritsugi 2020). To capture such requirements, app reviews can be classified by quality characteristics users discuss (Nuseibeh 2001; Groen et al. 2017)

3.4.3 Testing

App reviews analysis can be used to support various testing activities: validation by users (Iacob et al. 2013a; Iacob et al. 2013b; Iacob and Harrison 2013; Guzman et al. 2014; Guzman and Maalej 2014; Maalej and Nabil 2015; Gu and Kim 2015; Maalej et al. 2016; Bakiu and Guzman 2017; Ciurumelea et al. 2018; Durelli et al. 2018; Liu et al. 2018; Shah et al. 2019c; Gao et al. 2019; AlSubaihin et al. 2019; Dąbrowski et al. 2020; Xiao et al. 2020), test documentation (Iacob et al. 2016; Grano et al. 2018; Pelloni et al. 2018), test design (Man et al. 2016; Maalej et al. 2016; Groen et al. 2017; Shams et al. 2020) and test prioritization (Khalid et al. 2014).

Validation by Users Evaluating a software system with users usually involves expensive usability testing in a laboratory (Iacob et al. 2013a) or acceptance testing performed in a formal manner (IEEE 1990). In the case of mobile apps, software engineers can exploit user feedback to assess user satisfaction (Fu et al. 2013; Iacob et al. 2013a; Iacob et al. 2013b; Gu and Kim 2015; Bakiu and Guzman 2017; Ciurumelea et al. 2018; Shah et al. 2019c; Xiao 2019; Dąbrowski et al. 2020) and to identify any glitches with their products (Iacob et al. 2013a; Maalej and Nabil 2015; Gu and Kim 2015; Maalej et al. 2016; Ciurumelea et al. 2018; AlSubaihin et al. 2019; Gao et al. 2019). A recent survey with practitioners has shown that developers release the alpha/beta version of their apps to test the general reaction of users and to discover bugs (AlSubaihin et al. 2019).

In line with the direction, several approaches have been proposed to mine user opinions (Guzman and Maalej 2014; Guzman et al. 2014; Gu and Kim 2015; Bakiu and Guzman 2017; Shah et al. 2019c; Dąbrowski et al. 2020; Xiao et al. 2020) and to generate bug reports (Iacob et al. 2013a; Maalej and Nabil 2015; Maalej et al. 2016; Man et al. 2016; Ciurumelea

et al. 2018; Liu et al. 2018; Shah et al. 2019c). Opinion mining approaches help to discover the most problematic features and to quantify the number of negative opinions. Knowing what features users praise or hate can give a developer a hint about user acceptance of these features (Bakiu and Guzman 2017; AlSubaihin et al. 2019; Dąbrowski et al. 2020). Assuming core features have been modified, the team may want to know how users react to these features so that they can fix any issues quickly and refine these features. Analogously, identifying and quantifying reported bugs within a given time frame can help a development team during beta testing before official release (Iacob et al. 2013a; Iacob et al. 2013b; Ciurumelea et al. 2018; Gao et al. 2019; Shah et al. 2019c). If the number of reported issues is unusually high, development teams can reschedule the release of a new version in order to refocus on quality management and testing (Maalej and Nabil 2015; Maalej et al. 2016).

Test Documentation Test documentation can be partly supported by analysing app reviews (Iacob et al. 2016; Pelloni et al. 2018; Grano et al. 2018). Iacob et al. developed a tool that produce a summary of bugs reported in reviews with breakdown by app version and features that these bugs refer to (Iacob et al. 2016). Such summary can form the basis for later debugging the app and fixing the problems. User comments can also be integrated into mobile app testing tools (Pelloni et al. 2018; Grano et al. 2018). Originally, the tools generate a report of stack traces leading to an app crash (Pelloni et al. 2018; Grano et al. 2018). Analyzing the information to understand the root of the problems can be often counterintuitive. In such case, user comments can be used as a human readable companion for such report; linked to a related stack trace, user-written description of the problem can instantly guide testers where to look up for the emerged fault (Pelloni et al. 2018; Grano et al. 2018).

Test Design Analysing app reviews can support test case design (Man et al. 2016; Maalej et al. 2016; Groen et al. 2017; Shams et al. 2020). Analysing reported issues can help testers determine the app behavior, features, and functionality to be tested (Man et al. 2016). Reviews may describe particular use of the software in which users encountered an unusual situation (e.g., crashing without informing users of what happened) or inform about the lack of supporting users in finding a workaround (Maalej et al. 2016). Such information may help testers to design test cases capturing exceptions leading to a problem or to exercise new alternative scenarios other those initially considered (Maalej et al. 2016; Groen et al. 2017; Shams et al. 2020). Additionally, identifying negative comments on quality characteristics can help in specifying acceptance criteria an app should hold (Groen et al. 2017). For example, user complaints about performance efficiency can indicate performance criteria for functions that are expected to finish faster or more smoothly (Groen et al. 2017).

Test Prioritization Reviews and their ratings have been found to correlate with a download rank, a key measure of the app's success (Khalid et al. 2015; Martin et al. 2017). User complaints about specific issues can have a negative impact on rating, and in turn discourage users from downloading apps (Khalid et al. 2015). Therefore, it has been therefore suggested to prioritize issue-related test cases based on frequency and impact of these complaints (Khalid et al. 2015; Man et al. 2016). To address device-specific problems a development team must test their apps on a large number of devices, which is inefficient and costly (Erfani et al. 2013). The problem can be partially ameliorated by selecting devices submitted from reviews having the greatest impact on app ratings (Khalid et al. 2014). The strategy can be particularly useful for the team with limited resources that can only afford

to buy a few devices. Using the strategy, they can determine the optimal set of devices they can buy on which to test their app (Khalid et al. 2014).

3.4.4 Maintenance

In attempt to support software maintenance, review analysis has been proposed for problem and modification analysis, requested modification prioritization, help desk and impact analysis (see Table 13).

Problem and Modification Analysis Software engineers strive continuously to satisfy user needs and keep their app product competitive in the market (AlSubaihin et al. 2019). To this end, they can exploit approaches facilitating problem and modification analysis (Fu et al. 2013; Khalid 2013; Cen et al. 2014; Guzman et al. 2014; Gao et al. 2015; Gomez et al. 2015; Panichella et al. 2015; Gao et al. 2015; Palomba et al. 2015; Guzman et al. 2015; Khalid et al. 2015; Khalid et al. 2015b; Malik and Shakshuki 2016; Vu et al. 2016; Di Sorbo et al. 2016; Iacob et al. 2016; Wei et al. 2017; Licorish et al. 2017; Johann et al. 2017; Bakiu and Guzman 2017; Deocadez et al. 2017b; Wang et al. 2017; Palomba et al. 2017; Malik et al. 2018; Gao et al. 2018b; Muñoz et al. 2018; Palomba et al. 2018; Pelloni et al. 2018; Tong et al. 2018; Dalpiaz and Parente 2019; Phetrungnapha and Senivongse 2019; Gao et al. 2019; Shah et al. 2019c; AlSubaihin et al. 2019; Li et al. 2020; Hadi and Fard 2020; Zhou et al. 2020). The approaches detect user requests in app store feedback and classify them as problem reports and modifications requests (Zhou et al. 2020). Fine-grained classification can be carried out too, for example, to detect specific issues like privacy (Khalid 2013; Cen et al. 2014; Tao et al. 2020) or concrete change requests like features enhancement (Palomba et al. 2017; Al-Hawari 2020). Mining such information allows software engineers to determine and analyze user demands in timely and efficient fashion (Gao et al. 2015; Wang et al. 2017; Gao et al. 2018b; Gao et al. 2019; Guo and Singh 2020). By analysing the dynamics of reported problems over time, software engineers can immediately spot when a "hot issue" emerges and link it to a possibly flawed release (Fu et al. 2013; Guzman et al. 2014; Gao et al. 2015; Shah et al. 2019c). Moreover, they can generate a summary of user demands to obtain interim documentation serving as change request/problem report (Iacob et al. 2016; Di Sorbo et al. 2016; Phetrungnapha and Senivongse 2019).

Requested Modification Prioritization App developers may receive hundreds or even thousands of reviews requesting modifications and reporting problems (Khalid 2013; Villarroel et al. 2016; Noei et al. 2019). It is therefore not a trivial task for developers to select those requests which should be addressed in the next release (Villarroel et al. 2016). As with requirements, developers can investigate statistics concerning these requests (e.g., how many people requested specific modifications), estimate their impact on perceived app quality (e.g., expressed as user rating) or analyze the how these requests change over time (Gu and Kim 2015; Gao et al. 2015; Khalid et al. 2015; Man et al. 2016; Keertipati et al. 2016; Villarroel et al. 2016; Iacob et al. 2016; Licorish et al. 2017; Wei et al. 2017; Muñoz et al. 2018; Scalabrino et al. 2019; Dąbrowski et al. 2019; Hu et al. 2019; Noei et al. 2019; Noei et al. 2019; Oehri and Guzman 2020). Assuming developers have to decide which change to address first, they could select one with the largest share in the numbers of requests, or the one whose feedback most drives down the most app rating (Gu and Kim 2015; Dąbrowski et al. 2019; Di Sorbo et al. 2020). Similarly, observing a sharp growth in feedback reporting

of a specific problem (e.g., security and privacy), it may suggest that the issue is harmful to users and should be resolved quickly.

Help Desk Help desk typically provides end-users with answers to their questions, resolve their problems or assist in troubleshooting (Bourque et al. 1999). Analogously, app developers can respond to specific user reviews to answer users' questions, to inform about fixing problems or to thank users for their kind remarks about apps (McIlroy et al. 2015; Hassan et al. 2018; Srisopha et al. 2020a; Srisopha et al. 2020c). Though the task is not traditionally included in the typical responsibilities of software engineers, user support and managing the product reputation on the app store are essential to the app success; they should be viewed as important activities in in the software lifecycle. In fact, responding to reviews motivate app users to revise their feedback and ratings to be more positive (McIlroy et al. 2015). Some users even update their feedback to inform developers that the response solved users' problems or to thank for help (McIlroy et al. 2015; Hassan et al. 2018). Since responding to a large number of reviews can be time-consuming, developers can make use of approaches highlighting reviews that are more likely to require a response Srisopha et al. (2020a) and Srisopha et al. (2020c); and generating automatic replies to these reviews (Greenheld et al. 2018; Hassan et al. 2018; Vu et al. 2019; Gao et al. 2019).

Impact Analysis Review mining approaches help developers to discover modification requests posted in reviews; to identify app source code affected by these modifications (Zhou et al. 2020); and to estimate how implementing the modifications may impact users' satisfaction; (Palomba et al. 2015; Ciurumelea et al. 2017; Palomba et al. 2017; Palomba et al. 2018). The approaches typically cluster feedback requesting the same modifications (Ciurumelea et al. 2017; Palomba et al. 2017; Zhou et al. 2020), then search and retrieve links between review clusters and corresponding source code artefacts referring to the modifications (Palomba et al. 2015; Ciurumelea et al. 2017; Palomba et al. 2017; Palomba et al. 2018; Zhou et al. 2020). Such information can be useful for engineers before an issue of new release as well as afterwards. Software engineers can track which requests have (not) been implemented; monitor the proportion of reviews linked to software changes; and estimate the number of users affected by these changes. After the release has been issued, software engineers can also use the approaches to observe gain/loss in terms of average rating with respect to implemented changes.

RQ3: Supporting Software Engineering

- Analysing app reviews can support software engineers in requirements, design, testing, and maintenance activities.
- Most primary studies analyse app reviews to support (i) requirements elicitation, (ii) requirements prioritization, (iii) validation by users, (iv) problem and modification analysis, and (v) requested modification prioritization.
- 62 of primary studies (34%) do not describe software engineering use cases of their review mining approaches.

3.5 RQ4: Empirical Evaluation

To answer RQ4 (how are app review analysis approaches empirically evaluated), we used data items: F10 (evaluation objective), F11 (evaluation procedure), F12 (metrics and criteria), F14 (annotated datasets), F15 (annotation task), F16 (number of annotators), F17 (quality measure) and F18 (replication package). We found that 109 primary studies performed empirical evaluation of review mining approaches; 105 studies included evaluation of effectiveness and 23 of user-perceived quality.

3.5.1 Effectiveness Evaluation

A common procedure for effectiveness assessment consists of four steps: (i) formulate an evaluation objective, (ii) create an annotated dataset, (iii) apply the approach on the annotated dataset, and (iv) quantify the effectiveness. The evaluation objective refers to assessing the degree to which an approach can correctly perform a specific mining task or analysis (see Section 3.2). Human judgement is usually required to create the annotated dataset.

Primary studies involved humans performing the task manually on a sample of reviews and annotating the sample with correct solutions. Such annotated dataset (called the "ground truth") served as a baseline for evaluating the approach and quantifying the outcome.

Most studies provided a detail description of how each step of their evaluation methods have performed. Hence, we could record additional information:

Availability of Dataset and Tool Most studies have not released their annotated datasets nor the tools they evaluated.[14] Tables 16 provides an overview of 23 annotated datasets that are publicly available, reporting the reference to the paper, a short description of the dataset and its size in terms of number of reviews, whereas Table 17 presents 16 available tools,[15] providing the reference to the paper and a short description of the characteristics of the tool.

Evaluation Objective Scholars evaluated the effectiveness of their app review mining approaches in performing: Classification, Clustering, Sentiment Analysis, Information Extraction, Searching and Information Retrieval, Recommendation and Summarization.

Annotation Procedure The number of annotators labeling the same review sample (or their fragment) ranged from 1 to 5 with the median of 2 human annotators. Only 26 primary studies (25%) reported how the quality of their annotated datasets has been measured. The three most common metrics for inter-rater agreement evaluation were Cohen's Kappa (Pustejovsky and Stubbs 2012), Percentage Agreement (Hallgren 2012) and Jaccard index (Manning et al. 2008). Percentage Agreement and Cohen's Kappa were used to measure the quality of human annotation for Classification, Sentiment Analysis, or Feature Extraction; Jaccard index was used for assessing the human agreement for the task of Searching and Information Retrieval; whereas Fleiss' Kappa was used to assess the quality of manual Clustering. No study reported how the agreement was measured when annotators performed, Recommendation, or Summarization task.

[14] In addition to the reported information in the surveyed literature; we also contacted the authors of 105 primary studies to request replication packages.

[15] The references to the tools and the datasets are available in the supplementary material (Dąbrowski 2021)

Table 16 Publicly-available datasets of annotated reviews

Ref.	Description	Size
Chen et al. (2014)	Indicated whether the content of each review is informative or uninformative.	12,000
Guzman et al. (2015)	Tagged reviews with topics (e.g., bug report, feature shortcoming, complaint, usage scenario).	4,500
Gu and Kim (2015)	Identified type of user request each review convey (e.g., bug report, feature requests).	2,000
Maalej and Nabil (2015)	Reviews labeled with a type of user requests (bug report, feature request, rating, user experience).	4,400
Di Sorbo et al. (2016)	Reviews labeled with 12 topics (e.g. security) and user intention (e.g., problem discovery).	3,439
Panichella et al. (2016)	Reviews labeled with 5 categories useful from maintenance perspective (e.g., problem discovery).	852
Sänger et al. (2016)	Identified user opinions (feature and sentiment).	1,760
Ciurumelea et al. (2017)	Tagged reviews with mobile specific categories (e.g. performance, resources, battery, memory).	
Groen et al. (2017)	Labeled reviews with software quality requirements (e.g., usability, reliability, portability, compatibility).	360
Lu and Liang (2017)	Reviews labeled with functional and non-functional requirements (e.g., usability, performance).	2,000
Grano et al. (2018)	Annotated reviews with their topics and a type of issue users reports.	6,600
Jha and Mahmoud (2018)	Annotated a type of user feedback (feature request, bug reports, and others).	2,930
Nayebi et al. (2018)	Annotated reviews with a type of a user request (e.g., problem discovery).	2,383
Pelloni et al. (2018)	Reviews labeled with a crash report category.	534
Scoccia et al. (2018)	Annotated reviews with 10 categories of users's concern.	1,000
Al Kilani et al. (2019)	Labeled reviews with 5 categories: bug, new feature, performance, security, usability or sentimental.	7500
Dąbrowski et al. (2019)	Reviews annotated with 20 app features.	200
Jha and Mahmoud (2019)	Labeled reviews with non-functional requirements user discuss (e.g. usability, dependability).	6,000
Scalabrino et al. (2019)	Reviews labeled with feedback category (e.g., bug report, feature request).	3,000
Shah et al. (2019a)	Identified features discussed in reviews.	3,500
Stanik et al. (2019)	Annotated a type of user feedback (problem reports, inquiries, and irrelevant).	6,406
Dąbrowski et al. (2020)	Annotated reviews with 1,521 user opinions i.e., pairs of features and their related users' perceived sentiment.	1,000
Guo and Singh (2020)	Annotated reviews with user stories i.e., action-problem pairs.	200

Characteristics of Dataset Most annotated datasets were created using reviews coming from Google Play and Apple Store (84% in total); the remaining datasets have been created using reviews from Amazon Appstore, Black Berry App World; Huawei Store, Windows Phone Store and 360 Mobile Assistant. On average, an annotated dataset has been prepared using 2,800 reviews collected from a single app store; the reviews were collected for 19

Table 17 Publicly-available app review mining tools

Ref.	Description
Di Sorbo et al. (2016)	SURF tool classifies reviews by users' intention; cluster them then generates their summaries.
Panichella et al. (2016)	ARdoc tool classifies reviews with a type of user requests (e.g., feature request, problem discovery, information seeking, information giving and other.)
Johann et al. (2017)	SAFE tool extracts features from reviews and match them with features present in app descriptions.
Wei et al. (2017)	OASIS tool classifies reviews by reported issue; links them to warnings from static analysis tools; and recommend the priorities of these warnings.
Deshpande and Rokne (2018)	The tool classifies reviews by a type of a user request (e.g., problem discovery).
Dhinakaran et al. (2018)	The tool classifies reviews based on types of user feedback i.e., feature request, bug report, user experience and rating
Scoccia et al. (2018)	The tool classifies app reviews into users' concerns related to android run-time permission.
Shah et al. (2018)	A tool classifying app reviews based on their feedback type (e.g., feature request, problem report).
Jha and Mahmoud (2019)	Tool classifies app reviews by non-functional requirements user discuss (e.g. usability, dependability).
Pelloni et al. (2018)	BECLoMA tool links stack traces from testing tools to user reviews referring to the same crash.
Scalabrino et al. (2019)	CLAP tool classifying reviews by their types; clustering them; then recommend their relative-importance.
Shah et al. (2019a)	SAFE tool reimplementation facilitating feature extraction from reviews and app descriptions.
Shah et al. (2019b)	A reimplementation of a tool facilitating feature extraction using supervised ML technique.
Stanik et al. (2019)	A tool classifying reviews by the the type of user feedback (problem reports, inquiries, and irrelevant).
Guo and Singh (2020)	CASPER tool for extracting and synthesizing user stories of problems from app reviews.
Hadi and Fard (2020)	AOBTM tool discovers coherent and discriminative topics in reviews.

apps from 6 app categories. Table 18 provides five-number summary that details descriptive statistics about the datasets.

Effectiveness Quantification Three most common metrics used for assessing the effectiveness of app review mining approach are precision, recall, and F1-measure (Manning et al. 2008). The metrics were employed for evaluating Classification, Clustering, Information Extraction, Searching and Information retrieval, Sentiment Analysis, Recommendation and Summarization.

A few studies deviate from the common procedure outlined above. The studies evaluated their review mining approaches without annotated datasets:

– Eight studies asked annotators to assess the quality of output produced by their approaches, instead of creating an annotated dataset before applying the mining

Table 18 Five-summary numbers providing descriptive statistics of annotated datasets that primary used to evaluate app review mining approaches

Characteristics	Min.	Q1	Med.	Q3	Max.
No. App Stores	1	1	1	2	3
No. Apps	1	7	19	185	1,430,091
No. App Categories	1	4	6	10	35
No. App Reviews	80	1,000	2,800	4,400	41,793

approach. This was practiced for evaluating Classification (Li et al. 2017), Clustering (Guzman and Maalej 2014; Vu et al. 2015a; Palomba et al. 2017), Information Extraction (Johann et al. 2017; Li et al. 2017), Searching and Information Retrieval (Wei et al. 2017), and Recommendation (Shams et al. 2020).

– Seven studies used other software artefacts as an evaluation baseline rather than creating an annotated dataset (Gao et al. 2015; Man et al. 2016; Gao et al. 2018b; Uddin et al. 2020; Srisopha et al. 2020a; Srisopha et al. 2020c; Xiao et al. 2020). To evaluate Recommendation (e.g., determining priorities for reported issues), the studies compared recommended priorities for issues with priorities for the issues reported in user forums or changelogs; to assess the quality of Clustering, the studies benchmarked the output of their approaches with topics from app changelogs; whereas to evaluate their approaches in Recommending reviews that need to be responded, the studies used information of already responded reviews that developers posted in app stores.

3.5.2 User Study

Twenty three studies evaluated their review mining approaches through user studies (Guzman et al. 2014; Chen et al. 2014; Gu and Kim 2015; Guzman et al. 2015; Villarroel et al. 2016; Maalej et al. 2016; Panichella et al. 2016; Di Sorbo et al. 2016; Di Sorbo et al. 2017; Ciurumelea et al. 2017; Palomba et al. 2017; Ciurumelea et al. 2018; Greenheld et al. 2018; Liu et al. 2018; Gao et al. 2018b; Dalpiaz and Parente 2019; Scalabrino et al. 2019; Liu et al. 2019; Zhou et al. 2020; Gao et al. 2020; Tao et al. 2020; Shams et al. 2020; Liu et al. 2020). The objective of these evaluation was to qualitatively assess how the approach and/or their facilitated analysis are perceived by intended users (e.g., software engineers). Such evaluation procedure typically consists of the following steps: (i) define an evaluation subject and assessment criteria, (ii) recruit participants, (iii) instruct participants to perform a task with an approach or a produced analysis, (iv) elicit participant's opinion of the approach through questionnaire and/or interviews.

We looked in details at how studies perform each of the steps. The extracted data yields the following insights:

Evaluation Subjects User studies evaluated the following types of app review analyses: Clustering, Classification, Sentiment Analysis, Information Extraction, Search and Information Retrieval, Recommendation, Summarization, and Visualization.

Table 19 Reference mapping of user studies with breakdown of evaluation criterion and app review analysis

Criterion	App Review Analysis
Accuracy	Information Extraction (Gao et al. 2018b; Dalpiaz and Parente 2019), Classification (Villarroel et al. 2016; Panichella et al. 2016; Di Sorbo et al. 2016; Scalabrino et al. 2019), Clustering (Palomba et al. 2017; Zhou et al. 2020), Summarization (Di Sorbo et al. 2017).
Efficiency	Classification (Chen et al. 2014; Ciurumelea et al. 2017; Ciurumelea et al. 2018), Recommendation (Greenheld et al. 2018; Shams et al. 2020), Summarization (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Liu et al. 2019; Tao et al. 2020).
Informativeness	Classification (Ciurumelea et al. 2017; Liu et al. 2018; Dalpiaz and Parente 2019), Recommendation (Gao et al. 2020) Summarization (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Tao et al. 2020), Visualization (Guzman et al. 2014; Gao et al. 2018b).
Usability	Recommendation (Greenheld et al. 2018), Summarization (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Dalpiaz and Parente 2019).
Usefulness	Information Extraction (Guzman et al. 2015; Gao et al. 2018b; Dalpiaz and Parente 2019), Classification (Panichella et al. 2016; Di Sorbo et al. 2016; Maalej et al. 2016; Ciurumelea et al. 2017; Ciurumelea et al. 2018; Liu et al. 2018), Clustering (Palomba et al. 2017), Search and Information Retrieval (Palomba et al. 2017), Sentiment Analysis (Guzman et al. 2015), Recommendation (Villarroel et al. 2016; Scalabrino et al. 2019; Gao et al. 2020; Shams et al. 2020), Summarization (Di Sorbo et al. 2017; Liu et al. 2019; Tao et al. 2020), Visualization (Gu and Kim 2015; Liu et al. 2020).

Assessment Criteria Five evaluation criteria were typically taken into account: 1) Usefulness denoting the quality of being applicable or having practical worth; 2) Accuracy indicating the ability of being correct; 3) Usability signifying the quality of being easy

Table 20 Reference mapping of user studies with breakdown of the types of participants taking part in the studies

Sector	Participant	Reference
Academia	Student	(Di Sorbo et al. 2017; Ciurumelea et al. 2017; Greenheld et al. 2018; Liu et al. 2018; Gao et al. 2018b; Liu et al. 2019; Tao et al. 2020; Shams et al. 2020; Liu et al. 2020)
	Researcher	(Chen et al. 2014; Maalej et al. 2016; Di Sorbo et al. 2016; Di Sorbo et al. 2017; Ciurumelea et al. 2018; Liu et al. 2019)
Industry	Architect	(Maalej et al. 2016)
	Business Analyst	(Dalpiaz and Parente 2019)
	Developers	(Guzman et al. 2014; Guzman et al. 2015; Gu and Kim 2015; Panichella et al. 2016; Maalej et al. 2016; Di Sorbo et al. 2016; Palomba et al. 2017; Di Sorbo et al. 2017; Ciurumelea et al. 2017; Liu et al. 2018; Liu et al. 2019; Zhou et al. 2020; Gao et al. 2020; Liu et al. 2020)
	Product Manger	Dalpiaz and Parente (2019)
	Project Manager	(Maalej et al. 2016; Villarroel et al. 2016; Di Sorbo et al. 2016; Scalabrino et al. 2019)
	Requirement Engineer	(Maalej et al. 2016)
	Software Engineer	(Di Sorbo et al. 2016; Di Sorbo et al. 2017; Liu et al. 2018; Dalpiaz and Parente 2019)
	Software Tester	(Di Sorbo et al. 2016; Di Sorbo et al. 2017)

to use; 4) Efficiency indicating the capability of producing desired results with little or no human effort; and 5) Informativeness denoting the condition of being informative and instructive. Table 19 provides reference mapping of user studies with a breakdown of evaluation criteria and evaluated subjects.

Study Participants The number of participants involved in the study ranges from 1 to 85 with the median of 9 participants. The participants included professionals, scientists and students; Table 20 details the types of participants taking part in user studies and provide references to the corresponding studies.

Evaluation Procedure A The participants were instructed to either perform specific task with or without the use of the mining approach being evaluated, to review the outputs produced by the approach, or to simply trial the proposed approach without being given any specific tasks.

RQ4: Empirical Evaluation

- Review mining approaches are evaluated in terms of their effectiveness in performing review analyses and their user-perceived quality.
- To evaluate effectiveness, studies compare outputs of mining approaches with human-generated ones on sample datasets. Most datasets, however, have not been published.
- To assess perceived quality, studies perform user studies with software professionals, scientists and students. Participants are typically tasked to use a mining approach with a certain objective; then assess it based on specific quality criteria e.g., usefulness.

3.6 RQ5: Empirical Results

We answered RQ5 (how well do existing app review analysis approaches support software engineers) based on data item F13 (evaluation result). The data come from 87 studies reporting results of their empirical evaluations: effectiveness evaluations (83 studies) and user studies (18 studies). We synthesize results of these studies in the subsequent subsections.

3.6.1 Effectiveness Evaluation Results

The methodology that primary studies employed for effectiveness evaluation was too diverse to undertake a meta-analysis or other statistical synthesis methods (Higgins et al. 2019); these studies characterized for example diversity in their treatment (e.g., review mining approach), population (e.g., review dataset) or study design (e.g., annotation procedure). We thus employed 'summarizing effect estimates' method (Higgins et al. 2019); Table 21 reports the magnitude and range of effectiveness results that primary studies reported for different review analyses with breakdown of mined information type.[16]

[16]No effectiveness evaluation was performed w.r.t. content analysis and visualization.

Table 21 Summary of the results of controlled experiments measuring the effectiveness of app review analysis techniques

App Review Analysis	Mined Information	Results
Information Extraction	Features	Precision range from 21% to 84% (Shah et al. 2019a; Gao et al. 2020) with the median precision of 58% (Guzman and Maalej 2014); recall range from 42% to 77% (Malik et al. 2018; Shah et al. 2019a) with the median recall of 62% (Sänger et al. 2016).
	User Request	Precision range from 85% to 91% (Iacob and Harrison 2013; Iacob et al. 2013a) with the median precision of 91% (Iacob et al. 2016); recall range from 87% to 89% (Iacob and Harrison 2013; Iacob et al. 2013a) with the median recall of 89% (Iacob et al. 2016).
	NFR	Precision at the level of 92% (Groen et al. 2017).
Classification	User Request Type	Precision range from 35% to 94% (Martin et al. 2015; Jha and Mahmoud 2017b) with the median precision of 80% (Song et al. 2020); recall range from 51% to 99% (Jha and Mahmoud 2017b; Puspaningrum et al. 2018) with the median recall of 82% (Peng et al. 2016; Oh et al. 2013).
	NFR Type	Precision range from 63% to 100% (Mercado et al. 2016; Wang et al. 2018) with the median precision of 74% (Yang and Liang 2015; Lu and Liang 2017); recall range from 63% to 92% (Yang and Liang 2015; Wang et al. 2018) with the median recall of 79% (Lu and Liang 2017; Jha and Mahmoud 2019).
	Issue Type	Precision range from 66% to 100% (McIlroy et al. 2016; Mercado et al. 2016) with the median precision of 76% (Scoccia et al. 2018); recall range from 65% to 92% (McIlroy et al. 2016; Mercado et al. 2016) with the median recall of 79% (Scoccia et al. 2018).
Clustering	Similar Reviews	Accuracy range from 80% to 99% (Phetrungnapha and Senivongse 2019; Noei et al. 2019) with the median accuracy of 83% (Vu et al. 2015a); MojoFM range from 73% to 87% with the median MojoFM of 80% (Villarroel et al. 2016; Scalabrino et al. 2019).
Search and Information Retrieval	Feature-Specific Review	Precision range from 36% to 83% (Dąbrowski et al. 2019; Liu et al. 2019) with the median precision of 70% (Johann et al. 2017); recall range from 26% to 70% (Dąbrowski et al. 2019; Liu et al. 2019) with the median recall of 56% (Johann et al. 2017).
	Review-Goal Links	Precision range from 84% to 85% with the median precision of 85% (Liu et al. 2020; Gao et al. 2020); recall range from 71% to 74% with the median recall of 73% (Liu et al. 2020; Gao et al. 2020).
	Review-Issue Links	Precision range from 77% to 79% (Palomba et al. 2015; Noei et al. 2019) with the median precision of 77% (Palomba et al. 2018); recall at the level of 73% (Palomba et al. 2015; Palomba et al. 2018).
	Review-Code Links	Precision range from 51% to 82% (Ciurumelea et al. 2017; Grano et al. 2018) with the median precision of 82% (Palomba et al. 2017; Pelloni et al. 2018); recall range from 70% to 79% (Palomba et al. 2017; Ciurumelea et al. 2017) with the median recall of 75% (Pelloni et al. 2018; Grano et al. 2018).
Sentiment Analysis	Feature-Specific	Precision ranges from 68% to 74% with the median precision of 71%; recall ranges from 64% to 70% with the median of. 67% (Bakiu and Guzman 2017; Dąbrowski et al. 2020)
	Review	Accuracy at the level of 91% (Masrury and Alamsyah 2019).
Recommendation	User Request Priority	Accuracy range from 72% to 83% with the median accuracy of 78% (Villarroel et al. 2016; Scalabrino et al. 2019); precision range from 60% to 63% with the median precision of 62% (Gao et al. 2015; Gao et al. 2018b).
	Review Response	BLEU-4 at the level of 36% (Gao et al. 2019).
Summarization	Review Summary	Recall at the level of 71% (Jha and Mahmoud 2018).

Information Extraction The effectiveness of extracting information from reviews depends on the type of mined information. Techniques for extracting features from reviews has the lowest performance: median precision of 58% (Guzman and Maalej 2014) and median recall of 62% (Sänger et al. 2016); and the most diverging results: precision varies from 21% to 84% (Shah et al. 2019a; Gao et al. 2020). Techniques for extracting user requests and NFRs from reviews have higher performance with a median precision above 90% (Iacob et al. 2016; Groen et al. 2017) and only small variations between techniques.

Classification App reviews can be classified by information types these reviews contain, such as user requests, NFRs and issues. State-of-the-art review classification techniques have a median precision above 81% (Yang and Liang 2015; Lu and Liang 2017; Deshpande and Rokne 2018; Scoccia et al. 2018) and median recall around 83% (Peng et al. 2016; Lu and Liang 2017; Scoccia et al. 2018; Nayebi et al. 2018; Jha and Mahmoud 2019).

Clustering Studies have shown the accuracy of clustering semantically related reviews to be 83% (Vu et al. 2015a); this result is in line with findings concerning the quality of review clustering, where authors reported MojoFM of 80% (Villarroel et al. 2016; Scalabrino et al. 2019).

Search and Information Retrieval Mining approaches showed effectiveness in retrieving reviews to specific information needs; in particular, the results show that tracing information between reviews and issues in ticketing systems and between reviews and source code can be precise: the median precision above 75% (Palomba et al. 2017; Palomba et al. 2018; Pelloni et al. 2018); and complete: median recall above 70% (Palomba et al. 2015; Palomba et al. 2018; Pelloni et al. 2018; Grano et al. 2018); whereas linking reviews to goals in goal-models have been achieved with the median precision of 85%; and the median recall of 73% (Liu et al. 2020; Gao et al. 2020) Similarly, finding reviews related to specific features has been reported with 70% of precision and recall of 56% (Johann et al. 2017). The variability of the results e.g., precision between 36%-80% (Dąbrowski et al. 2019; Liu et al. 2019), however, may lead to inconclusive findings.

Sentiment Analysis The overall sentiment of a review can be identified with an accuracy of 91% (Masrury and Alamsyah 2019). Identifying the sentiment of a review with respect to a specific app feature is less effective with the median precision of 71% and the median recall of 67% (Bakiu and Guzman 2017; Dąbrowski et al. 2020).

Recommendation Recommending priorities for user requests was reported with medium to high effectiveness: the median accuracy of 78% (Villarroel et al. 2016; Scalabrino et al. 2019) and precision of 62% (Gao et al. 2015; Gao et al. 2018b). Whereas, generating review responses was reported with BLEU-4[17] greater than 30% (Gao et al. 2019), which reflects human-understandable text.

Summarization Mining techniques were recorded to generate a compact description outlining the main themes present in reviews with recall of 71% (Jha and Mahmoud 2018).

[17] The metrics quantifying the quality of generated text on a scale of 0% to 100%.

3.6.2 User Study Results

Twenty three studies evaluated user-perceived quality of review mining approaches. Table 22 provides synthesis of user study results that primary studies reported for different review analyses with breakdown of evaluation criterion.

Information Extraction Extracting information from reviews e.g., issue reports and user opinions is useful for developers (Gao et al. 2018b); it can help to elicit new requirements or prioritize development effort (Guzman et al. 2015; Dalpiaz and Parente 2019). In particular, machine learning techniques are able to identify issues with acceptable accuracy (Gao et al. 2018b); feature extraction methods instead produce too imprecise analyses to be applicable in practice (Dalpiaz and Parente 2019).

Classification Review classification showed their utility for identifying different users' needs e.g., feature requests, or bug reports (Di Sorbo et al. 2016; Panichella et al. 2016; Maalej et al. 2016; Ciurumelea et al. 2017; Liu et al. 2018; Ciurumelea et al. 2018; Zhou et al. 2020). Such categorized feedback is informative and ease further manual review inspection (Ciurumelea et al. 2017; Liu et al. 2018; Dalpiaz and Parente 2019). Practitioners reported to save up to 75% of their time thanks to the analysis (Chen et al. 2014; Ciurumelea et al. 2017; Ciurumelea et al. 2018); and that their accuracy is sufficient for the practical application (Villarroel et al. 2016; Di Sorbo et al. 2016; Panichella et al. 2016; Scalabrino et al. 2019).

Clustering Review clustering is convenient for grouping feedback conveying similar content; for example, those reporting the same feature request or discussing the same topic (Palomba et al. 2017; Zhou et al. 2020). Evaluated approaches can perform the analysis with a high level of precision and completeness (Palomba et al. 2017; Zhou et al. 2020).

Searching and Information Retrieval Developers admitted the usefulness linking reviews to the source code components to be changed (Palomba et al. 2017); the task traditionally requires an enormous manual effort and is highly error-prone.

Sentiment Analysis Analyzing user opinions can help to identify problematic features and to prioritize development effort to improve these features (Guzman et al. 2015).

Recommendation Project managers found recommending priorities of user requests useful for release planning (Villarroel et al. 2016; Scalabrino et al. 2019); it can support their decision-making w.r.t. requirements and modifications that users wish to address. Developers perceived an automatic review response system as more usable than the traditional mechanism (Greenheld et al. 2018); recommending reviews that require responding and suggesting responses to the reviews can reduce developers' workload (Greenheld et al. 2018). Similarly, recommending goals that an app needs to satisfy is informative and may guide this app evolution (Gao et al. 2020); whereas suggesting test cases triggering bugs can be useful for developers to reproduce bug-related user reviews; and save cost on manual bug reproduction (Shams et al. 2020).

Summarization Compact description outlining most important review content is useful for developers in their software engineers activities (Di Sorbo et al. 2017; Liu et al. 2019; Tao

Table 22 Summary of user studies evaluating the perceived quality of app review analysis techniques

App Review Analysis	Criterion	Results
Information Extraction	Accuracy	Machine learning techniques extracts issues with acceptable accuracy (Gao et al. 2018b); collocation algorithms generates too many false positives for feature extraction (Dalpiaz and Parente 2019).
	Usefulness	Extracting issues is useful for app maintenance (Gao et al. 2018b); feature extraction may help to analyze user opinions, and to prioritize development effort (Guzman et al. 2015; Dalpiaz and Parente 2019).
Classification	Accuracy	Classifying reviews by types of user requests, topics is sufficient for practical use (Panichella et al. 2016; Villarroel et al. 2016; Di Sorbo et al. 2016; Scalabrino et al. 2019); ML techniques are superior in accuracy to simple NLP (Dalpiaz and Parente 2019).
	Efficiency	Classifying reviews automatically (e.g., by user requests) could reduce up to 75% of the time that app developers spend for a manual feedback inspection (Chen et al. 2014; Ciurumelea et al. 2017; Ciurumelea et al. 2018).
	Informativeness	Grouping feedback by topics is instructive; a group may express shared users' needs specific to e.g., app features, emerging issues, new requirements (Ciurumelea et al. 2017; Liu et al. 2018; Dalpiaz and Parente 2019).
	Usefulness	Classifying reviews is useful for identifying different users' needs (Di Sorbo et al. 2016; Panichella et al. 2016; Liu et al. 2018); comprehending reported issues and problematic features (Maalej et al. 2016; Ciurumelea et al. 2017; Ciurumelea et al. 2018; Liu et al. 2018).
Clustering	Accuracy	Clustering reviews conveying semantically similar information is performed with high precision and completeness (Palomba et al. 2017; Zhou et al. 2020).
	Usefulness	Useful for grouping reviews bearing a similar message e.g. reporting the same issues (Palomba et al. 2017).
Search and Information Retrieval	Usefulness	Helpful for linking reviews to source components (e.g., methods, class or field name) (Palomba et al. 2017).
Sentiment Analysis	Usefulness	Helpful for detecting user opinions about app features; it can support prioritising development effort to improve these features (Guzman et al. 2015).

Table 22 (continued)

App Review Analysis	Criterion	Results
Recommendation	Efficiency	Suggesting reviews requiring responses lower app developers' workload (Greenheld et al. 2018); suggesting test cases triggering bugs can save developers' effort on manual bug reproduction (Shams et al. 2020).
	Informativeness	Recommending users' goals that an app needs to satisfy is informative from the developers' perspective (Gao et al. 2020).
	Usability	Automatic review response system shows higher usability compared to the original app store mechanism; survey respondents favored the the system (Greenheld et al. 2018).
	Usefulness	Recommending priority of user requests is useful for release planning (Villarroel et al. 2016; Scalabrino et al. 2019); suggesting goals an app needs to satisfy may support this app evolution (Gao et al. 2020); whereas recommending test cases triggering bugs can help developers to reproduce bug-related user reviews (Shams et al. 2020).
Summarization	Accuracy	Summarizing topics and requests that users post can be made with high accuracy (Di Sorbo et al. 2017).
	Efficiency	Prevents more than half of the time required for analyzing feedback (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Tao et al. 2020); it helps to immediately understand user opinions, user requests or security issues (Liu et al. 2019; Tao et al. 2020).
	Informativeness	Listing user requests (e.g., bug reports), topics or security issues through a summary table is informative and expressive for software engineers (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Tao et al. 2020).
	Usability	Table showing frequently posted user requests, and topics is easy to read (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Dalpiaz and Parente 2019).
	Usefulness	Summarizing user requests, topics and/or security issues that reviews convey is useful for better understanding users' needs (Di Sorbo et al. 2017; Liu et al. 2019; Tao et al. 2020).
Visualization	Informativeness	Presenting what topics users discuss and how they change over time can inform about emerging issues (Gao et al. 2018b), or relevant user opinions (Guzman et al. 2014).
	Usefulness	Showing heat-map of feature-specific sentiments or their trends over time can help developers to identify problematic features (Gu and Kim 2015); and to understand to what extent an app satisfied users' goals (Liu et al. 2020).

et al. 2020); in particular, summaries conveying information about frequently discussed topics, user opinions, user requests and security issues. Facilitating this information in a tabular form is easy to read and expressive (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Dalpiaz and Parente 2019). Such summaries are generated with sufficient accuracy to be used in practical scenarios (Di Sorbo et al. 2017; Tao et al. 2020); in fact, developers reported to save up to 50% of their time thanks to the analysis (Di Sorbo et al. 2016; Di Sorbo et al. 2017; Liu et al. 2019; Tao et al. 2020).

Visualization Presenting trends of frequently discussed topics can inform developers about urgent issues, 'hot features', or popular user opinions (Guzman et al. 2014; Gao et al. 2018b). Heat-map illustrating feature-specific sentiment (i.e., user options) help developers to understand users experience with these features (Gu and Kim 2015); it indicates which features users praise and which are problematic. Visualizing how user opinions change over time aids developers in examining users' reactions e.g., to newly implemented modifications for these features; and understanding to what extent an app satisfies users' goals (Liu et al. 2020).

RQ5: Evaluation Results

- Mining approaches perform well for 5 broad app review analyses: (1) Classification; (2) Clustering; (3) Searching and Information Retrieval; (4) Recommendation and (5) Summarization.
- Software engineers generally consider app review analyses useful. The analyses can ease their SDLC activities; reduce their workload; and support their decision-making.
- Software engineers find accuracy of most app review analyses promising for the practical usage; yet the quality of (6) Information Extraction and (7) Sentiment Analysis seem to be insufficient.

4 Discussion

In this section we highlight and discuss some of the findings from our study, summarize literature gaps, pointing to directions for future research.

4.1 Mining App Reviews Is a Growing Research Area

Mining app reviews for software engineering is a relatively new research area. The first use of app reviews for software engineering purposes can be dated back to 2012. Nevertheless, the analysis of demographics has revealed that the research area increasingly attracts the attention of scholars. The number of papers published in line with the directions has grown substantially in the last three years. A recent survey in app store analysis found 45 papers relevant to app review analysis published up to 2015 (Martin et al. 2017). Our findings show that the number of published papers in the area has quadrupled by the end of 2020. The most frequent venues where scholars have published their work concern high-quality software engineering conferences and journals (see Table 5). These imply there is not only increasing effort on exploring the research direction, but also suggest contributions of these efforts are relevant from a software engineering perspective; in fact, empirical evidences

(RQ5) demonstrate that software engineers find mining app reviews useful in support of their SDLC activities; mining approaches can reduce their workload; facilitate knowledge that would be difficult to obtain manually. As other work (Martin et al. 2017), we also hypothesize factors leading to the research interest in the field concerns increased popularity of mobile apps, an easy access to user feedback on a scale not seen before as well as a general interest in adopting data mining techniques for mining software repository.

4.2 Software Engineering Goals and Use Cases

App reviews analysis has broad applications in software engineering (RQ3). It can be used to support a variety of activities in requirements, design, testing and maintenance (see Table 6). Researchers however do not always clearly describe the envisioned software engineering use cases for their techniques.

So far, research in this area has been driven mostly by the opportunity to apply ML techniques on app reviews. Most studies (61%) relate their approaches to potential software engineering activities, but the remain vague about details of how they envision the techniques to be used in practice. A greater focus on software engineering goals and use cases would increase the relevance and impacts of app review analysis techniques. This systematic literature review includes a complete inventory of already envisioned software engineering use cases for the various app review analysis technique (RQ3). This inventory can provide the basis for a more detailed investigation of software engineering goals and use cases for app review analysis tools. This investigation will contribute to designing future app review analysis tools that best serves the needs of software engineers.

4.3 Need Of Reference Model For Review Mining Tools

Reference model of stakeholders goals, use cases and system architectures for review mining tools would help structuring research efforts in this area, and communicate how fitting review mining techniques together help to address real stakeholders' needs. In the future, scholars can elaborate such model by generalizing existing review mining solutions; explaining how different components help to realize intended use cases and satisfy stakeholders' goals. The model would also help researchers to identify and reuse common components in a typical architecture of review mining tools as well as explain the novelty and contribution of their work within that framework.

4.4 Small Size Of Evaluation Datasets

A great deal of effort has been made to evaluate the effectiveness of data mining techniques (RQ4). Primary studies, however, used evaluation datasets of small size (on average 2,800 reviews). This is a tiny portion of user-submitted feedback in app stores. Popular mobile apps (like WhatsApp or Instagram) can receive more than 5,000 reviews per day, and more than one million reviews in a year (App Annie 2020). This is a significant threat to the validity of their results when trying to generalize them e.g., (Ciurumelea et al. 2017; Deocadez et al. 2017a; Dąbrowski et al. 2019). The problem is attributed to the substantial effort of manual review annotation; labeling 900 reviews can take up to 12.5 hours (Guzman and Maalej 2014). As none of the surveyed studies tried to tackle the problem, it opens an avenue for future research. Researchers may experiment with semi-automated data labeling techniques currently exploited to minimize effort for preparing training datasets (Deocadez et al. 2017b; Dhinakaran et al. 2018; Miller et al. 2020). Providing the problem was

handled, scholars should still be mindful of a sampling bias when curating dataset (Annis 2005). Techniques to ameliorate the latter problem, however, has been well-studied in a recent study (Martin et al. 2015).

4.5 Replication Packages

Most papers did not make available their review mining tools and evaluation datasets (see Table 16 and Table 17). This hinders the replicability of these works as well as new comparative studies. Our survey contains a single replication study and that study reported the challenge in validating results of the original work due the absence of annotated dataset and insufficiently documented evaluation procedure (Shah et al. 2019a). Future studies should provide replication packages, including evaluation datasets, procedures, and approaches so that researchers will be able to validate existing works and confirm reported findings. It will also help in benchmarking approaches and provide a baseline for evaluating new approaches aiming at improving performance of review mining techniques.

4.6 Impacts On Software Engineering Practice

It is not yet clear whether app review analysis techniques are already good enough to be useful in practice (RQ5). Identifying what performance the approaches should have to be useful for software engineers is an important open question (Berry 2017; 2018). Essentially, an approach facilitating review analysis should synthesize reviews so that the effort for further manual inspection of the outcomes of that analysis would be negligible or at least manageable. Clearly, the effort would depend on a scenario an approach aims to realize. In addition to evaluating review analysis tools in terms of ML performance metrics (e.g precision and recall), it will become increasingly important to evaluate them in terms of software engineering concerns: Does it save time? Does it improve the quality of, for example, the requirements elicitation and prioritisation process? etc. Evaluating techniques with respect to software engineering concerns is more difficult but necessary to ensure research efforts are aligned with real stakeholders' goals. Such evaluation will involve a combination of quantitative and quantitative studies aimed at reducing our current uncertainty about potential impacts of review mining techniques on software engineering activities.

4.7 Practitioners' Requirements For App Review Mining Tools

Numerous tools have been developed in the context of app review analysis research; they satisfy requirements coming mainly from scholars rather than practitioners. We have recorded no research studying what features the tools should facilitate nor what goals they should satisfy. The current research is *data-driven* rather than *goal-driven*. The studies apply different types of app review analyses and techniques to mine information from app reviews without explicitly examining the practitioners' perspective. It is not clear to what extent the tools satisfy the real practitioners' goals. Though existing user studies provides evidences software practitioners find certain types of analyses valuable e.g., Classification (Palomba et al. 2017), yet more systematic research is necessary in such directions to understand practitioners' needs. Future research should plan to actively involve practitioners, for example via interview sessions or the analysis of their development practices, to understand why the tools are needed; what SE goal they want to satisfy with the tools; what features the tools should facilitate; and how the tool would be used in the organizational settings. Such knowledge will help to understand the actual use cases scenarios of the tools, and to

identify whether there is misalignment between what state-of-the-art tools offer and what practitioners actually need.

4.8 Verifying the Industrial Needs for App Review Analysis

Most studies motivated their mining approaches to reduce the manual effort for app review analysis. Such rationale seems to be reasonable in the context of popular apps (e.g., WhatsApp or Facebook Messenger) that are frequently commented and receive hundreds or thousands reviews per day. However, an average app receives 22 reviews per day (Pagano and Maalej 2013). It seems therefore legitimate to study the potential impact of the app review analysis research on the app store industry; and to what extent the mining tools would be useful in the industrial settings. Such a study could address this problem from multiple perspectives e.g., what small, medium and large app development organization are interested in app review mining tools? who in the organization would use the tools? is the manual app review analysis 'the real pain' of the practitioners? if so, how 'the pain' manifests itself? are any tasks obstructed? is the problem generating additional costs? Answering the questions could help to understand who are the actual beneficiaries of the app review analysis research; and what is the size of that market. Not only it would help to scope and justify the future research directions, but it would also provide insights to commercializing this research.

4.9 Pay Attention to Efficiency and Scalability of Mining Tools

Primary studies are mostly focused on evaluating effectiveness and perceived quality of their mining tools. We however recorded no study focused on assessing the efficiency and scalability of their tools; studying the efficiency informs how much time the tools take to produce their outcomes; whereas scalability informs how the time changes when the input of the tools increase. Efficiency and scalability are fundamental qualities of analytics tools (Talia 2019); app review mining tools are no exception. The number of reviews that an app receives can vary from a few to more than thousands. Existing approaches e.g., for feature extraction (Guzman and Maalej 2014) or app review classification (Maalej and Nabil 2015) rely on NLP and ML techniques that may be challenging to scale-up (Analytics India Mag 2020). Future studies, therefore, should take the efficiency and scalability into consideration when developing and evaluating their mining tools to demonstrate the tools can be used in the practical settings.

4.10 The Problem of Training ML Techniques

Machine learning is the most frequent type of techniques used for app review analysis (RQ2). Most of these techniques, however, are supervised one and requires a training dataset consisting of manually annotated reviews. Preparing manually annotated dataset is time-consuming and often error-prone (Guzman and Maalej 2014). More importantly, such annotated dataset might be domain- and time-specific; an annotated reviews of one app might not be re-usable for training a technique for the feedback of the other app; further, the dataset may be prone to data drift - a phenomenon in which the characteristics of app reviews change over the time. In such a case, ML technique must be periodically trained with up-to-date training dataset to maintain their predictive abilities (Explorium 2020). Recent studies thus experimented with active learning (Dhinakaran et al. 2018) and semi-supervised techniques (Deocadez et al. 2017b) to reduce the cost of annotating a large amount of data.

More research is however needed to understand how many reviews should be annotated for preparing a training dataset when the techniques is used in the industrial settings; how often such dataset needs to prepared; and whether or not the practitioners would accept the cost of preparing this dataset.

5 Threats to Validity

One of the main threats to the validity of this systematic literature review is incompleteness. The risk of this threat highly depends on the selected list of keywords forming search queries. To decrease the risk of an incomplete keyword list, we have used an iterative approach to keyword-list construction. We constructed two queries: generic and one specific. The generic query was formed using keywords appearing in the index of terms in sample studies analysing app reviews for SE. Specific query was formed based on a set of keywords representing concepts of our research objective. As in any other literature survey, we are also prone to a publication bias. To mitigate this threat, we complemented a digital library search with other strategies. We conducted an issue-by-issue search of top-level conferences and journals as well as performed a backward and forward snowballing.

To ensure the quality and reliability of our study, we defined a systematic procedure for conducting our survey, including research questions to answer, searching strategies and selection criteria for determining primary studies of interest. We conducted a pilot study to assess the technical issues such as the completeness of the data form and usability issues such as the clarity of procedure instructions. The protocol was reviewed by the panel of researchers in addition to the authors of the study. It was then revised based on their critical feedback. Consequently, the selection of primary studies followed a strict protocol in accordance to well-founded guidelines (Kitchenham 2004; Kitchenham et al. 2004; Ralph et al. 2020).

Another threat to validity we would like to highlight is our subjectivity in screening, data extraction and classification of the studied papers. To mitigate the threat, each step was performed by one coder, who was the first author of this paper. Then, the step was cross-checked by a second coder. Each step was validated on a randomly selected sample of 10% of the selected papers. The percentage inter-coder agreement reached for all the phases was equal or higher than 80%, indicating high agreement between the authors (Ide and Pustejovsky 2017). In addition, the intra-rater agreement was performed. The first author re-coded once again a randomly selected sample of 20% of studied papers. Then an external evaluator, who has no relationship with the research, verified the agreement between the first and the second rounds. The percentage intra-coder agreement was higher than 90%, indicating near complete agreement (Ide and Pustejovsky 2017).

A similar threat concerns whether our taxonomies are reliable enough for analysing and classifying extracted data. To mitigate this threat, we used an iterative content analysis method to continuously develop each taxonomy. New concepts which emerged when studying the papers were introduced into a taxonomy and changes were made respectively. These taxonomies were discussed between all the authors and agreed upon their final form.

6 Related Work

This review is not the first effort synthesizing knowledge from the literature analysing app reviews for SE (Martin et al. 2017; Genc-Nayebi and Abran 2017; Tavakoli et al. 2018; Noei

and Lyons 2019). Our SLR, however, differs substantially from previous studies in scope of the literature surveyed and depth of our analysis. Table 23 shows the differences between our study and previous works in accordance with dimensions we considered for the comparison. We grouped the dimensions into information related to study characteristics and topics surveyed in our study. The characteristics concern study type (i.e., systematic literature review or survey), time period covered and number of papers surveyed. The topics concern: Paper Demographics, App Reviews Analyses (RQ1), Mining Techniques (RQ2), Supporting Software Engineering (RQ3), Empirical Evaluation (RQ4) and Empirical Results (RQ5).

Martin et al. (2017) surveyed literature with the aim to demonstrate a newly emerging research area i.e., app store analysis for software engineering. The scope of their survey is much broader than of our study, as it covers literature analyzing various types of app store data (e.g., API, rank of downloads, or price). Our work has much narrower scope, focussing only on app review analysis, but studies the paper in greater depths in order to answer our five research questions.

Though the related survey also addresses (RQ1), our study is more up-to-date and larger in scale, covering 182 papers. More importantly, most dimensions of our SLR i.e., RQ2-RQ5, are missing in this other study.

Two other studies addressed our RQ2, but partially, as they are narrower in scope (Genc-Nayebi and Abran 2017; Tavakoli et al. 2018). Tavakoli et al. (2018) surveyed the literature in the context of techniques and tools for mining app reviews. Similarly, Genc-Nayebi and Abran (2017) consolidated literature to synthesize information on techniques for opinion mining. Our SLR addresses the dimension more broadly, rather than in context of techniques for a specific review analysis or tool-supported approaches. We have made an effort to consolidate general knowledge on techniques the literature employs for 9 broad types of review analyses. We also provided mapping between different review analyses and techniques facilitating their realization.

Noei and Lyons (2019) summarized 21 papers analysing app reviews from Google Play. The authors provided an overview of each paper, briefly explaining the applications, and mention their limitations. The surveyed papers were selected subjectively, rather than following a systematic searching procedure. In contrast, our study is a SLR rather than a summary. Following a systematic procedure, we selected 182 studies that we carefully read and then synthesized to answer five research questions. The related work marginally covers information for RQ1 and RQ2.

Table 23 Main differences between our study and previous surveys

Dimensions	Our Study	Martin (2017)	Genc-Nayebi (2017)	Tavakoli (2018)	Noei (2019)
Study Type	SLR	Survey	SLR	SLR	Survey
Time Period	’10-’20	’00-’15	’11-’15	’11-’17	’12-’19
No. Papers	182	45	24	34	21
Paper Demographics	✓	✓	✓	✓	
App Review Analyses (RQ1)	✓	✓			✓
Mining Techniques (RQ2)	✓		✓	✓	✓
Supporting SE (RQ3)	✓				
Empirical Evaluation (RQ4)	✓				
Empirical Results (RQ5)	✓				

In summary, previous studies do not cover our research questions related to software engineering activities (RQ3) and empirical evaluations (RQ4 and RQ5). They partly cover our research questions RQ1 and RQ2 but on a smaller set of papers and in less details.

7 Conclusion

In this paper, we presented a systematic literature review of the research on analysing app reviews for software engineering. Through systematic search, we identified 182 relevant studies that we thoroughly examined to answer our research questions. The findings have revealed a growing interest in the research area. Research on analysing app reviews are published in the main software engineering conferences and journals e.g., ICSE, TSE or EMSE and the number of publications has tripled in the last four years. The research in this area will likely continue to gain importance as a consequence of increased interest in mobile app development.

This systematic literature review structures and organizes the knowledge on the different types of app review analyses as well as data mining techniques used for their realization. With that knowledge, researchers and practitioners can understand what useful information can be found in app reviews, and how app review analysis can be facilitated at abstract and technical levels. More importantly, the literature review provides a new light on why mining app reviews can be useful; the findings identifies 14 software engineering activities that have been the target of previous research on app review analysis. Important future research for app review analysis will involve developing a deeper understanding of the stakeholders' goals and context for app review analysis tools in order to increase the applicability, relevance and value of these tools.

The findings have revealed that software engineers find mining approaches useful and with promising performance to generate different app review analyses. It however remains unclear to what extent these approaches are already good enough to be used in practice.

It will become increasingly important to evaluate them in terms of software engineering specific concerns: Does it improve the quality of, for example, the requirements elicitation and prioritization process? We also recommend empirical evaluation will continue to improve in scale and reproducibility. Research in this area is currently inconsistent quality in terms of evaluation method and ability for the research to be reproduced. Future studies should share evaluation datasets and mining tools, allowing their experiments to be replicated. They should also pay more attention to the scalability and the efficiency of their mining approaches.

In conclusion, this study helps to communicate knowledge on analyzing app reviews for software engineering purposes. We hope our effort will inspire scholars to advance the research area and assist them in positioning their new works.

References

Abad ZSH, Sims SDV, Cheema A, Nasir MB, Harisinghani P (2017) Learn more, pay less! lessons learned from applying the wizard-of-oz technique for exploring mobile app requirements. In: 2017 IEEE 25th international requirements engineering conference workshops (REW). pp 132–138

Al-Hawari A (2020) Najadat H, Shatnawi R, Classification of application reviews into software maintenance tasks using data mining techniques. Softw Qual J. https://doi.org/10.1007/s11219-020-09529-8

Al Kilani N, Tailakh R, Hanani A (2019) Automatic classification of apps reviews for requirement engineering: Exploring the customers need from healthcare applications. In: 2019 sixth international conference on social networks analysis, management and security (SNAMS). pp 541–548

Ali M, Joorabchi ME, Mesbah A (2017) Same app, different app stores: A comparative study. In: Proceedings of the 4th international conference on mobile software engineering and systems, MOBILESoft '17. IEEE Press, pp 79–90

Alqahtani F, Orji R (2019) Usability issues in mental health applications. In: Adjunct publication of the 27th conference on user modeling, adaptation and personalization, USA, UMAP'19 Adjunct. ACM, New York, pp 343–348

AlSubaihin A, Sarro F, Black S, Capra L, Harman M (2019) App store effects on software engineering practices. IEEE Trans Softw Eng :1–1

Analytics India Mag (2020) https://analyticsindiamag.com/challenges-of-implementing-natural-language-processing/, Accessed: 2021-06-01

Annis DH (2005) Probability and statistics: The science of uncertainty, Michael J. Evans and Jeffrey S. Rosenthal. Am Stat 59:276–276

App Annie (2020) https://www.appannie.com/, Accessed: 2020-07-01

App Store (2021) Ratings, Reviews, and Responses. https://developer.apple.com/app-store/ratings-and-reviews/, Accessed: 2021-06-01

Bailey K, Nagappan M, Dig D (2019) Examining user-developer feedback loops in the ios app store. In: 52nd Hawaii international conference on system sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019, pp 1–10

Bakiu E, Guzman E (2017) Which feature is unusable? detecting usability and user experience issues from user reviews. In: 2017 IEEE 25th international requirements engineering conference workshops (REW). pp 182–187

Bauer M (2007) Content analysis. an introduction to its methodology – by klaus krippendorff from words to numbers. narrative, data and social science – by roberto franzosi, vol 58, pp 329–331. https://doi.org/10.1111/j.1468-4446.2007.00153_10.x

Begel A, Zimmermann T (2014) Analyze this! 145 questions for data scientists in software engineering. In: 36th international conference on software engineering. pp 12–13

Berry D (2018) Keynote: Evaluation of NLP tools for hairy RE tasks. In: Joint proceedings of REFSQ-2018 workshops, doctoral symposium, live studies track, and poster track co-located with the 23rd international conference on requirements engineering: foundation for software quality (REFSQ 2018), Utrecht, The Netherlands, March 19, 2018

Berry DM (2017) Evaluation of tools for hairy requirements and software engineering tasks. In: IEEE 25th international requirements engineering conference workshops, RE 2017 Workshops, Lisbon, Portugal, September, 4-8, 2017, pp 284–291

Bishop CM (2006) Pattern recognition and machine learning (information science and statistics). Springer, Berlin

Bourque P, Dupuis R, Abran A, Moore J, Tripp L (1999) The guide to the software engineering body of knowledge. IEEE Softw 16:35–44

Bujang M, Baharum N (2017) Guidelines of the minimum sample size requirements for kappa agreement test. Epidemiol Biostat Public Health 14

Burge JE, Carroll JM, McCall R, Mistrk I (2008) Rationale-based software engineering, 1st edn. Springer Publishing Company, Incorporated, Berlin

Buse RPL, Zimmermann T (2012) Information needs for software development analytics. In: 34th international conference on software engineering. pp 987–996

Cannataro M, Comito C (2003) A data mining ontology for grid programming. In: Proc. 1st int. workshop on semantics in peer-to-peer and grid computing, in conjunction with WWW2003. pp 113–134

Carreño LVG, Winbladh K (2013) Analysis of user comments: An approach for software requirements evolution. In: Proceedings of the 2013 international conference on software engineering, ICSE '13. IEEE Press, pp 582–591

Cen L, Si L, Li N, Jin H (2014) User comment analysis for android apps and cspi detection with comment expansion. In: Proceeding of the 1st international workshop on privacy-preserving IR (PIR). pp 25–30
Chandy R, Gu H (2012) Identifying spam in the ios app store. In: Proceedings of the 2nd Joint WICOW/airweb Workshop on Web Quality. ACM, pp 56–59
Chen N, Lin J, Hoi SCH, 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, ICSE 2014. ACM, New York, pp 767–778
Chen R, Wang Q, Xu W (2019) Mining user requirements to facilitate mobile app quality upgrades with big data. Electron Commer Res Appl 38:100889
Ciurumelea A, Schaufelbühl A, Panichella S, Gall HC (2017) Analyzing reviews and code of mobile apps for better release planning. In: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). pp 91–102
Ciurumelea A, Panichella S, Gall HC (2018) Poster: Automated user reviews analyser. In: 2018 IEEE/ACM 40th international conference on software engineering: companion (ICSE-Companion). pp 317–318
Clement J (2020) Number of apps available in leading app stores as of 1st quarter 2020. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, Accessed: 2020-07-01
Dalpiaz F, Parente M (2019) RE-SWOT: from user feedback to requirements via competitor analysis. In: Requirements engineering: foundation for software quality - 25th international working conference, REFSQ 2019, Essen, Germany, March 18-21, 2019, Proceedings. pp 55–70
Deocadez R, Harrison R, Rodriguez D (2017) Automatically classifying requirements from app stores: A preliminary study. In: 2017 IEEE 25th international requirements engineering conference workshops (REW). pp 367–371
Deocadez R, Harrison R, Rodriguez D (2017) Preliminary study on applying semi-supervised learning to app store analysis. In: Proceedings of the 21st international conference on evaluation and assessment in software engineering, EASE'17. ACM, New York, pp 320–323
Deshpande G, Rokne J (2018) User feedback from tweets vs app store reviews: An exploratory study of frequency, timing and content. In: 2018 5th international workshop on artificial intelligence for requirements engineering (AIRE). pp 15–21
Dhinakaran VT, Pulle R, Ajmeri N, Murukannaiah PK (2018) App review analysis via active learning: Reducing supervision effort without compromising classification accuracy. In: 2018 IEEE 26th international requirements engineering conference (RE). pp 170–181
Di Sorbo A, Panichella S, Alexandru CV, Shimagaki J, Visaggio CA, Canfora G, Gall HC (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, FSE 2016. ACM, New York, pp 499–510
Di Sorbo A, Panichella S, Alexandru CV, Visaggio CA, Canfora G (2017) Surf: Summarizer of user reviews feedback. In: Proceedings of the 39th international conference on software engineering companion, ICSE-C '17. IEEE Press, pp 55–58
Di Sorbo A, Grano G, Aaron Visaggio C, Panichella S (2020) Investigating the criticality of user-reported issues through their relations with app rating. J Softw Evol Process 33(3):e2316. https://doi.org/10.1002/smr.2316. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2316, e2316 smr.2316
Dąbrowski J (2021) Supplementary material for system literature review: analysing app reviews for software engineering. https://github.com/jsdabrowski/SLR-SE/
Dąbrowski J, Letier E, Perini A, Susi A (2019) Finding and analyzing app reviews related to specific features: A research preview. In: Requirements engineering: foundation for software quality - 25th international working conference, REFSQ 2019, Essen, Germany, March 18-21, 2019, Proceedings. pp 183–189
Dąbrowski J, Letier E, Perini A, Susi A (2020) Mining user opinions to support requirement engineering: An empirical study. In: Dustdar S, Yu E, Salinesi C, Rieu D, Pant V (eds) Advanced information systems engineering - 32nd international conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings, Springer, Lecture Notes in Computer Science, vol 12127. pp 401–416. https://doi.org/10.1007/978-3-030-49435-3_25
Durelli VHS, Durelli RS, Endo AT, Cirilo E, Luiz W, Rocha L (2018) Please please me: Does the presence of test cases influence mobile app users' satisfaction. In: Proceedings of the XXXII Brazilian symposium on software engineering, SBES '18. ACM, New York, pp 132–141
Erfani M, Mesbah A, Kruchten P (2013) Real challenges in mobile app development. In: 2013 ACM/IEEE international symposium on empirical software engineering and measurement (ESEM). pp 15–24
Explorium (2020) Understanding and handling data and concept drift. https://www.explorium.ai/blog/understanding-and-handling-data-and-concept-drift/, Accessed: 2021-06-01
Franzmann D, Eichner A, Holten R (2020) How mobile app design overhauls can be disastrous in terms of user perception: The case of snapchat. Trans Soc Comput 3(4). https://doi.org/10.1145/3409585

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, KDD '13. ACM, New York, pp 1276–1284

Gao C, Wang B, He P, Zhu J, Zhou Y, Lyu MR (2015) Paid: prioritizing app issues for developers by tracking user reviews over versions. In: 2015 IEEE 26th international symposium on software reliability engineering (ISSRE). pp 35–45

Gao C, Xu H, Hu J, Zhou Y (2015) Ar-tracker: Track the dynamics of mobile apps via user review mining. In: 2015 IEEE symposium on service-oriented system engineering, SOSE '15. pp 284–290

Gao C, Zeng J, Lo D, Lin CY, Lyu MR, King I (2018a) Infar: Insight extraction from app reviews. In: Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering, ESEC/FSE 2018. ACM, New York, pp 904–907

Gao C, Zeng J, Lyu MR, King I (2018b) Online app review analysis for identifying emerging issues. In: Proceedings of the 40th international conference on software engineering, ICSE '18. ACM, New York, pp 48–58

Gao C, Zeng J, Xia X, Lo D, Lyu MR, King I (2019) Automating app review response generation. In: 2019 34th IEEE/ACM international conference on automated software engineering (ASE). pp 163–175

Gao C, Zheng W, Deng Y, Lo D, Zeng J, Lyu MR, King I (2019) Emerging app issue identification from user feedback: Experience on wechat. In: Proceedings of the 41st international conference on software engineering: software engineering in practice, ICSE-SEIP '19. IEEE Press, pp 279–288

Gao S, Liu L, Liu Y, Liu H, Wang Y (2020) Updating the goal model with user reviews for the evolution of an app. J Softw Evol Process 32(8):e2257. https://doi.org/10.1002/smr.2257. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2257, e2257 JSME-19-0105.R2

Genc-Nayebi N, Abran A (2017) A systematic literature review: Opinion mining studies from mobile app store user reviews. J Syst Softw 125:207–219

Gomez M, Rouvoy R, Monperrus M, Seinturier L (2015) A recommender system of buggy app checkers for app store moderators. In: 2nd ACM international conference on mobile software engineering and systems. IEEE

Goul M, Marjanovic O, Baxley S, Vizecky K (2012) Managing the enterprise business intelligence app store: Sentiment analysis supported requirements engineering. In: 2012 45th Hawaii international conference on system sciences. pp 4168–4177

Graham M, Milanowski AT, Miller J (2012) Measuring and promoting inter-rater agreement of teacher and principal performance ratings

Grano G, Di Sorbo A, Mercaldo F, Visaggio CA, Canfora G, Panichella S (2017) Android apps and user feedback: A dataset for software evolution and quality improvement. In: Proceedings of the 2nd ACM SIGSOFT international workshop on app market analytics, WAMA 2017. ACM, New York, pp 8–11

Grano G, Ciurumelea A, Panichella S, Palomba F, Gall HC (2018) Exploring the integration of user feedback in automated testing of android applications. In: 2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER). pp 72–83

Greenheld G, Savarimuthu BTR, Licorish SA (2018) Automating developers' responses to app reviews. In: 2018 25th Australasian software engineering conference (ASWEC). pp 66–70

Groen EC, Kopczyńska S, Hauer MP, Krafft TD, Doerr J (2017) Users — the hidden software product quality experts?: A study on how app users report quality aspects in online reviews. In: 2017 IEEE 25th international requirements engineering conference (RE). pp 80–89

Gu X, Kim S (2015) "What parts of your apps are loved by users?" (T). In: 30th IEEE/ACM international conference on automated software engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015. pp 760–770

Gunaratnam I, Wickramarachchi D (2020) Computational model for rating mobile applications based on feature extraction. In: 2020 2nd international conference on advancements in computing (ICAC), vol 1, pp 180–185. https://doi.org/10.1109/ICAC51239.2020.9357270

Guo H, Singh MP (2020) Caspar: extracting and synthesizing user stories of problems from app reviews. In: 2020 IEEE/ACM 42nd international conference on software engineering (ICSE). pp 628–640

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

Guzman E, Paredes Rojas A (2019) Gender and user feedback: An exploratory study. In: 2019 IEEE 27th international requirements engineering conference (RE). pp 381–385

Guzman E, Bhuvanagiri P, Bruegge B (2014) Fave: Visualizing user feedback for software evolution. In: 2014 Second IEEE working conference on software visualization. pp 167–171

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

Guzman E, El-Halaby M, Bruegge B (2015) Ensemble methods for app review classification: An approach for software evolution. In: Proceedings of the 30th IEEE/ACM international conference on automated software engineering, ASE '15. IEEE Press, pp 771–776

Guzman E, Ibrahim M, Glinz M (2017) A little bird told me: Mining tweets for requirements and software evolution. In: Moreira A, Araùjo J, Hayes J, Paech B (eds) 25th IEEE international requirements engineering conference, RE 2017, Lisbon, Portugal, September 4-8, 2017, IEEE Computer Society, pp 11–20. https://doi.org/10.1109/RE.2017.88

Guzman E, Oliveira L, Steiner Y, Wagner LC, Glinz M (2018) User feedback in the app store: A cross-cultural study. In: 2018 IEEE/ACM 40th international conference on software engineering: software engineering in society (ICSE-SEIS). pp 13–22

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

Hadi MA, Fard FH (2020) Aobtm: Adaptive online biterm topic modeling for version sensitive short-texts analysis. In: 2020 IEEE international conference on software maintenance and evolution (ICSME). pp 593–604. https://doi.org/10.1109/ICSME46990.2020.00062

Hallgren K (2012) Computing inter-rater reliability for observational data: An overview and tutorial. Tutor Quant Methods Psychol 8:23–34

Hassan S, Bezemer C, Hassan AE (2018) Studying bad updates of top free-to-download apps in the google play store. IEEE Trans Softw Eng :1–1

Hassan S, Tantithamthavorn C, Bezemer C, Hassan AE (2018) Studying the dialogue between users and developers of free apps in the google play store. Empir Softw Eng 23(3):1275–1312

Higgins JP, Thomas J, Chandler J, Cumpston M, Li T, Page MJ, Welch VA (2019) Cochrane handbook for systematic reviews of interventions, 2nd edn. Wiley, Chichester

Hoon L, Vasa R, Schneider JG, Mouzakis K (2012) A preliminary analysis of vocabulary in mobile app user reviews. In: Proceedings of the 24th Australian computer-human interaction conference. ACM, pp 245–248

Hoon L, Vasa R, Martino GY, Schneider JG, Mouzakis K (2013) Awesome! conveying satisfaction on the app store. In: Proceedings of the 25th Australian computer-human interaction conference: augmentation, application, innovation, collaboration, OzCHI '13. ACM, New York, pp 229–232

Hoon L, Rodriguez-García M, Vasa R, Valencia-García R, Schneider JG (2016) App reviews: Breaking the user and developer language barrier. In: Trends and applications in software engineering, vol 405. Springer International Publishing, pp 223–233

Hu H, Bezemer C, Hassan AE (2018) Studying the consistency of star ratings and the complaints in 1 & 2-star user reviews for top free cross-platform android and ios apps. Empir Softw Eng 23(6):3442–3475

Hu H, Wang S, Bezemer C, Hassan AE (2019) Studying the consistency of star ratings and reviews of popular free hybrid android and ios apps. Empir Softw Eng 24(1):7–32

Huebner J, Frey RM, Ammendola C, Fleisch E, Ilic A (2018) What people like in mobile finance apps: An analysis of user reviews. In: Proceedings of the 17th international conference on mobile and ubiquitous multimedia, MUM 2018, Cairo, Egypt, November 25-28, 2018, pp 293–304

Iacob C, Harrison R (2013) Retrieving and analyzing mobile apps feature requests from online reviews. In: Proceedings of the 10th working conference on mining software repositories, IEEE Press. pp 41–44

Iacob C, Harrison R, Faily S (2013a) Online reviews as first class artifacts in mobile app development. In: Proceedings of the 5th international conference on mobile computing, applications, and services. MobiCASE '13

Iacob C, Veerappa V, Harrison R (2013b) What are you complaining about?: A study of online reviews of mobile applications. In: Proceedings of the 27th international BCS human computer interaction conference. British Computer Society, pp 29:1–29:6

Iacob C, Faily S, Harrison R (2016) Maram: Tool support for mobile app review management. In: Proceedings of the 8th EAI international conference on mobile computing, applications and services, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), MobiCASE'16. pp 42–50

Ide N, Pustejovsky J (eds) (2017) Handbook of linguistic annotation. Springer Netherlands, Dordrecht

IEEE (1990) IEEE standard glossary of software engineering terminology

ISO/IEC 25010 (2011) ISO/IEC 25010:2011, systems and software engineering — systems and software quality requirements and evaluation (SQuaRE) — system and software quality models

Jha N, Mahmoud A (2017a) MARC: A mobile application review classifier. In: Joint proceedings of REFSQ-2017 workshops, doctoral symposium, research method track, and poster track co-located with the 22nd international conference on requirements engineering: foundation for software quality (REFSQ 2017), Essen, Germany, February 27, 2017

Jha N, Mahmoud A (2017b) Mining user requirements from application store reviews using frame semantics. In: Requirements engineering: foundation for software quality - 23rd international working conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings. pp 273–287

Jha N, Mahmoud A (2018) Using frame semantics for classifying and summarizing application store reviews. Empir Softw Eng 23(6):3734–3767

Jha N, Mahmoud A (2019) Mining non-functional requirements from app store reviews. Empir Softw Eng 24(6):3659–3695

Johann T, Stanik C, B AMA, Maalej W (2017) Safe: A simple approach for feature extraction from app descriptions and app reviews. In: 2017 IEEE 25th international requirements engineering conference (RE). pp 21–30

Jurafsky D, Martin JH (2009) Speech and language processing, 2nd edn. Prentice-Hall, Inc., Hoboken

Kalaichelavan K, Malik H, Husnu N, Sreenath S (2020) What do people complain about drone apps? a large-scale empirical study of google play store reviews. Procedia Comput Sci 170:547–554. https://doi.org/10.1016/j.procs.2020.03.124. https://www.sciencedirect.com/science/article/pii/S1877050920305627, the 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops

Keertipati S, Savarimuthu BTR, Licorish SA (2016) Approaches for prioritizing feature improvements extracted from app reviews. In: Proceedings of the 20th international conference on evaluation and assessment in software engineering, EASE '16. ACM, New York

Khalid H (2013) On identifying user complaints of ios apps. In: Proceedings of the 2013 international conference on software engineering. IEEE Press, pp 1474-1476

Khalid H, Nagappan M, Shihab E, Hassan AE (2014) Prioritizing the devices to test your app on: a case study of android game apps. In: Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering, (FSE-22), Hong Kong, China, November 16-22, 2014, pp 610–620

Khalid H, Shihab E, Nagappan M, Hassan AE (2015) What do mobile app users complain about? IEEE Softw 32(3):70–77

Khalid H, Nagappan M, Hassan AE (2016) Examining the relationship between findbugs warnings and app ratings. IEEE Softw 33(4):34–39

Khalid M, Asif M, Shehzaib U (2015a) Towards improving the quality of mobile app reviews. Int J Inf Technol Comput Sci (IJITCS) 7(10):35

Khalid M, Shehzaib U, Asif M (2015b) A case of mobile app reviews as a crowdsource. Int J Inf Eng Electron Bus (IJIEEB) 7(5):39

Khan J, Xie Y, Liu L, Wen L (2019) Analysis of requirements-related arguments in user forums. https://doi.org/10.1109/RE.2019.00018

Kitchenham BA (2004) Procedures for performing systematic reviews

Kitchenham BA, Dyba T, Jorgensen M (2004) Evidence-based software engineering. In: Proceedings of the 26th international conference on software engineering, ICSE '04. IEEE Computer Society, pp 273–281

Kunaefi A, Aritsugi M (2020) Characterizing user decision based on argumentative reviews. In: 7th IEEE/ACM international conference on big data computing, applications and technologies, BDCAT 2020, Leicester, United Kingdom, December 7-10, 2020, IEEE. pp 161–170. https://doi.org/10.1109/BDCAT50828.2020.00002

Kurtanović Z, Maalej W (2017) Mining user rationale from software reviews. In: 2017 IEEE 25th international requirements engineering conference (RE). pp 61–70

Kurtanovic Z, Maalej W (2018) On user rationale in software engineering. Requir Eng 23(3):357–379

van Lamsweerde A (2009) Requirements engineering: from system goals to UML models to software specifications. Wiley, Hoboken

Li S, Guo J, Fan M, Lou JG, Zheng Q, Liu T (2020) Automated bug reproduction from user reviews for android applications. In: 2020 IEEE/ACM 42nd international conference on software engineering: software engineering in practice (ICSE-SEIP). pp 51–60

Li T, Zhang F, Wang D (2018) Automatic user preferences elicitation: A data-driven approach. In: Requirements engineering: foundation for software quality - 24th international working conference, REFSQ 2018, Utrecht, The Netherlands, March 19-22, 2018, Proceedings. pp 324–331

Li Y, Jia B, Guo Y, Chen X (2017) Mining user reviews for mobile app comparisons. Proc ACM Interact Mob Wearable Ubiquitous Technol 1(3)

Liang TP, Li X, Yang CT, Wang M (2015) What in consumer reviews affects the sales of mobile apps: A multifacet sentiment analysis approach. Int J Electron Commer 20(2):236–260

Licorish SA, Savarimuthu BTR, Keertipati S (2017) Attributes that predict which features to fix: Lessons for app store mining. In: Proceedings of the 21st international conference on evaluation and assessment in software engineering, EASE'17. ACM, New York, pp 108–117

Lim S, Henriksson A, Zdravkovic J (2021) Data-driven requirements elicitation: A systematic literature review. SN Comput Sci 2. https://doi.org/10.1007/s42979-020-00416-4
Liu Y, Liu L, Liu H, Wang X (2018) Analyzing reviews guided by app descriptions for the software development and evolution. J Softw Evol Process 30(12):e2112. e2112 JSME-17-0184.R2
Liu Y, Liu L, Liu H, Yin X (2019) App store mining for iterative domain analysis: Combine app descriptions with user reviews. Softw Pract Exper 49(6):1013–1040. sPE-19-0009.R1
Liu Y, Liu L, Liu H, Gao S (2020) Combining goal model with reviews for supporting the evolution of apps. IET Softw 14(1):39–49. https://doi.org/10.1049/iet-sen.2018.5192
Lu M, Liang P (2017) Automatic classification of non-functional requirements from augmented app user reviews. In: Proceedings of the 21st international conference on evaluation and assessment in software engineering, EASE'17. ACM, New York, pp 344–353
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
Maalej W, Kurtanovic Z, Nabil H, Stanik C (2016) On the automatic classification of app reviews. Requir Eng 21(3):311–331
Maalej W, Nayebi M, Johann T, Ruhe G (2016) Toward data-driven requirements engineering. IEEE Softw 33(1):48–54
Maalej W, Nayebi M, Ruhe G (2019) Data-driven requirements engineering: An update. In: Proceedings of the 41st international conference on software engineering: software engineering in practice, ICSE-SEIP '19. IEEE Press, pp 289–290
Malavolta I, Ruberto S, Soru T, Terragni V (2015a) End users' perception of hybrid mobile apps in the google play store. In: Proceedings of the 4th international conference on mobile services (MS). IEEE
Malavolta I, Ruberto S, Terragni V, Soru T (2015b) Hybrid mobile apps in the google play store: an exploratory investigation. In: Proceedings of the 2nd ACM international conference on mobile software engineering and systems, ACM
Malgaonkar S, Licorish SA, Savarimuthu BTR (2020) Towards automated taxonomy generation for grouping app reviews: A preliminary empirical study. In: Shepperd MJ, e Abreu FB, da Silva AR, Pérez-Castillo R (eds) Quality of information and communications technology - 13th international conference, QUATIC 2020, Faro, Portugal, September 9-11, 2020, Proceedings, Communications in Computer and Information Science, vol 1266. Springer, pp 120–134. https://doi.org/10.1007/978-3-030-58793-2_10
Malik H, Shakshuki EM (2016) Mining collective opinions for comparison of mobile apps. Procedia Comput Sci 94:168–175. the 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops
Malik H, Shakshuki EM, Yoo WS (2018) Comparing mobile apps by identifying 'hot' features. Future Gener Computer Syst
Man Y, Gao C, Lyu MR, Jiang J (2016) Experience report: Understanding cross-platform app issues from user reviews. In: 2016 IEEE 27th international symposium on software reliability engineering (ISSRE). pp 138–149
Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, Cambridge
Martens D, Johann T (2017) On the emotion of users in app reviews. In: Proceedings of the 2nd international workshop on emotion awareness in software engineering, SEmotion '17. IEEE Press, pp 8–14
Martens D, Maalej W (2019) Release early, release often, and watch your users' emotions: Lessons from emotional patterns. IEEE Softw 36(5):32–37
Martens D, Maalej W (2019) Towards understanding and detecting fake reviews in app stores. Empir Softw Eng 24(6):3316–3355
Martin W, Harman M, Jia Y, Sarro F, Zhang Y (2015) The app sampling problem for app store mining. In: Proceedings of the 12th working conference on mining software repositories, MSR '15. IEEE Press, pp 123–133
Martin WJ, Sarro F, Jia Y, Zhang Y, Harman M (2017) A survey of app store analysis for software engineering. IEEE Trans Software Eng 43(9):817–847
Masrury RA, Alamsyah A (2019) Analyzing tourism mobile applications perceived quality using sentiment analysis and topic modeling. In: 2019 7th international conference on information and communication technology (ICoICT). pp 1–6
McIlroy S, Shang W, Ali N, Hassan A (2015) Is it worth responding to reviews? a case study of the top free apps in the google play store. IEEE Software PP
McIlroy S, Ali N, Khalid H, Hassan AE (2016) Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empir Softw Eng 21(3):1067–1106

Mcilroy S, Shang W, Ali N, Hassan AE (2017) User reviews of top mobile apps in apple and google app stores. Commun ACM 60(11):62–67
Mercado IT, Munaiah N, Meneely A (2016) The impact of cross-platform development approaches for mobile applications from the user's perspective. In: Proceedings of the international workshop on app market analytics, WAMA 2016. ACM, New York, pp 43-49
Miller B, Linder F, Mebane WR (2020) Active learning approaches for labeling text: Review and assessment of the performance of active learning approaches. Polit Anal :1–20
Miner G, Elder J, Hill T, Nisbet R, Delen D, Fast A (2012) Practical text mining and statistical analysis for non-structured text data applications, 1st edn. Academic Press, Cambridge
Moher D, Liberati A, Tetzlaff J, Altman D (2009) Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. Br Med J 8:336–341
Mujahid S, Sierra G, Abdalkareem R, Shihab E, Shang W (2017) Examining user complaints of wearable apps: A case study on android wear. In: 2017 IEEE/ACM 4th international conference on mobile software engineering and systems (MOBILESoft). pp 96–99
Mujahid S, Sierra G, Abdalkareem R, Shihab E, Shang W (2018) An empirical study of android wear user complaints. Empir Softw Eng 23(6):3476–3502
Muñoz S, Araque O, Llamas AF, Iglesias CA (2018) A cognitive agent for mining bugs reports, feature suggestions and sentiment in a mobile application store. In: 2018 4th international conference on big data innovations and applications (innovate-data). pp 17–24
Nagappan M, Shihab EMenzies T, Williams L, Zimmermann T (eds) (2016) Mobile app store analytics. Morgan Kaufmann, Boston
Nayebi M, Cho H, Farrahi H, Ruhe G (2017) App store mining is not enough. In: 2017 IEEE/ACM 39th international conference on software engineering companion (ICSE-C). pp 152–154
Nayebi M, Cho H, Ruhe G (2018) App store mining is not enough for app improvement. Empir Softw Eng 23(5):2764–2794
Nicolai M, Pascarella L, Palomba F, Bacchelli A (2019) Healthcare android apps: a tale of the customers' perspective. In: Proceedings of the 3rd ACM SIGSOFT international workshop on app market analytics, WAMA@ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 27, 2019, pp 33–39
Noei E, Lyons K (2019) A survey of utilizing user-reviews posted on google play store. In: Proceedings of the 29th annual international conference on computer science and software engineering, IBM Corp., USA, CASCON '19. pp 54–63
Noei E, Da Costa DA, Zou Y (2018) Winning the app production rally. In: Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering, New York, NY, USA, ESEC/FSE 2018, pp 283-294
Noei E, Zhang F, Wang S, Zou Y (2019) Towards prioritizing user-related issue reports of mobile applications. Empir Softw Eng 24(4):1964–1996
Noei E, Zhang F, Zou Y (2019) Too many user-reviews, what should app developers look at first?IEEE Trans Softw Eng 1–1
Nuseibeh B (2001) Weaving together requirements and architectures. Computer 34(3):115–119
Nyamawe A, Liu H, Niu N, Umer Q, Niu Z (2019) Automated recommendation of software refactorings based on feature requests. pp 187–198. https://doi.org/10.1109/RE.2019.00029
Oehri E, Guzman E (2020) Same same but different: Finding similar user feedback across multiple platforms and languages. In: Breaux TD, Zisman A, Fricker S, Glinz M (eds) 28th IEEE international requirements engineering conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020, IEEE, pp 44–54. https://doi.org/10.1109/RE48521.2020.00017
Oh J, Kim D, Lee U, Lee JG, Song J (2013) Facilitating developer-user interactions with mobile app review digests. In: CHI '13 extended abstracts on human factors in computing systems, CHI EA '13. ACM, New York, pp 1809–1814
Pagano D, Maalej W (2013) User feedback in the appstore: An empirical study. In: 2013 21st IEEE international requirements engineering conference (RE). pp 125–134
Palomba F, Linares-Vásquez M, Bavota G, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2015) User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In: 2015 IEEE international conference on software maintenance and evolution (ICSME). pp 291–300
Palomba F, Salza P, Ciurumelea A, Panichella S, Gall H, Ferrucci F, De Lucia A (2017) Recommending and localizing change requests for mobile apps based on user reviews. In: Proceedings of the 39th international conference on software engineering, ICSE '17. IEEE Press, pp 106–117
Palomba F, Linares-Vásquez M, Bavota G, Oliveto R, Penta MD, Poshyvanyk D, Lucia AD (2018) Crowdsourcing user reviews to support the evolution of mobile apps. J Syst Softw 137:143–162
Panichella S, Di SorboA, Guzman E, Visaggio CA, Canfora G, Gall HC (2015) How can i improve my app? classifying user reviews for software maintenance and evolution. In: 2015 IEEE international conference on software maintenance and evolution (ICSME). pp 281–290

Panichella S, Di Sorbo A, Guzman E, Visaggio CA, Canfora G, Gall HC (2016) Ardoc: App reviews development oriented classifier. In: Proceedings of the 2016 24th ACM SIGSOFT international symposium on foundations of software engineering, FSE 2016. ACM, New York, pp 1023–1027

Pelloni L, Grano G, Ciurumelea A, Panichella S, Palomba F, Gall HC (2018) Becloma: Augmenting stack traces with user review information. In: 2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER). pp 522–526

Peng Z, Wang J, He K, Tang M (2016) An approach of extracting feature requests from app reviews. In: Collaborate computing: networking, applications and worksharing - 12th international conference, CollaborateCom 2016, Beijing, China, November 10-11, 2016, Proceedings. pp 312–323

Phetrungnapha K, Senivongse T (2019) Classification of mobile application user reviews for generating tickets on issue tracking system. In: 2019 12th international conference on information communication technology and system (ICTS). pp 229–234

Puspaningrum A, Siahaan D, Fatichah C (2018) Mobile app review labeling using lda similarity and term frequency-inverse cluster frequency (tf-icf). In: 2018 10th international conference on information technology and electrical engineering (ICITEE). pp 365–370

Pustejovsky J, Stubbs A (2012) Natural language annotation for machine learning - a guide to corpus-building for applications. O'Reilly, Newton

Ralph P, Baltes S, Bianculli D, Dittrich Y, Felderer M, Feldt R, Filieri A, Furia CA, Graziotin D, He P, Hoda R, Juristo N, Kitchenham BA, Robbes R, Mėndez D, Molleri J, Spinellis D, Staron M, Stol K, Tamburri D, Torchiano M, Treude C, Turhan B, Vegas S (2020) ACM SIGSOFT empirical standards. arXiv:2010.03525

Sänger M, Leser U, Kemmerer S, Adolphs P, Klinger R (2016) SCARE - the sentiment corpus of app reviews with fine-grained annotations in German. In: Proceedings of the tenth international conference on language resources and evaluation (LREC'16)

Sänger M, Leser U, Klinger R (2017) Fine-grained opinion mining from mobile app reviews with word embedding features. In: Natural language processing and information systems - 22nd international conference on applications of natural language to information systems, NLDB 2017, Liège, Belgium, June 21-23, 2017, Proceedings. pp 3–14

Scalabrino S, Bavota G, Russo B, Penta MD, Oliveto R (2019) Listening to the crowd for the release planning of mobile apps. IEEE Trans Softw Eng 45(1):68–86

Scoccia GL, Ruberto S, Malavolta I, Autili M, Inverardi P (2018) An investigation into android run-time permissions from the end users' perspective. In: Proceedings of the 5th international conference on mobile software engineering and systems, MOBILESoft '18. ACM, New York, pp 45–55

Shah FA, Sabanin Y, Pfahl D (2016) Feature-based evaluation of competing apps. In: Proceedings of the international workshop on app market analytics, WAMA 2016. ACM, New York, pp 15–21

Shah FA, Sirts K, Pfahl D (2018) Simplifying the classification of app reviews using only lexical features. In: Software Technologies - 13th International Conference, ICSOFT 2018, Porto, Portugal, July 26-28, 2018, Revised Selected Papers. pp 173–193

Shah FA, Sirts K, Pfahl D (2019a) Is the SAFE approach too simple for app feature extraction? A replication study. In: Requirements Engineering: Foundation for Software Quality - 25th International Working Conference, REFSQ 2019, Essen, Germany, March 18-21, 2019, Proceedings. pp 21–36

Shah FA, Sirts K, Pfahl D (2019b) Simulating the impact of annotation guidelines and annotated data on extracting app features from app reviews. International Conference on Software Technologies (ICSOFT, In

Shah FA, Sirts K, Pfahl D (2019c) Using app reviews for competitive analysis: Tool support. In: Proceedings of the 3rd ACM SIGSOFT international workshop on app market analytics, WAMA 2019. ACM, New York, pp 40-46

Shams RA, Hussain W, Oliver G, Nurwidyantoro A, Perera H, Whittle J (2020) Society-oriented applications development: Investigating users' values from bangladeshi agriculture mobile applications. In: Proceedings of the ACM/IEEE 42nd international conference on software engineering: software engineering in society, ICSE-SEIS '20. Association for Computing Machinery, New York, pp 53–62. https://doi.org/10.1145/3377815.3381382

Sharma T, Bashir MN (2020) Privacy apps for smartphones: An assessment of users' preferences and limitations. In: Moallem A (ed) HCI for cybersecurity, privacy and trust - second international conference, HCI-CPT 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19-24, 2020, Proceedings, Springer, Lecture Notes in Computer Science, vol 12210. pp 533–546. https://doi.org/10.1007/978-3-030-50309-3_35

Simmons A, Hoon L (2016) Agree to disagree: on labelling helpful app reviews. In: Proceedings of the 28th Australian conference on computer-human interaction, OzCHI '16. ACM, New York. pp 416–420

Singh V (2021) South Asian University - Department of Computer Science. http://www.sau.int/research-themes/text-analytics.html, Accessed: 2021-06-01
Software T (2021) What is text analytics? http://www.tibco.com/reference-center/what-is-text-analytics, Accessed: 2021-06-01
Song R, Li T, Ding Z (2020) Automatically identifying requirements-oriented reviews using a top-down feature extraction approach. In: 2020 27th Asia-Pacific software engineering conference (APSEC). pp 450–454. https://doi.org/10.1109/APSEC51365.2020.00054
Srisopha K, Alfayez R (2018) Software quality through the eyes of the end-user and static analysis tools: A study on android oss applications. In: Proceedings of the 1st international workshop on software qualities and their dependencies, SQUADE '18. ACM, New York, pp 1–4
Srisopha K, Phonsom C, Lin K, Boehm B (2019) Same app, different countries: A preliminary user reviews study on most downloaded ios apps. In: 2019 IEEE international conference on software maintenance and evolution (ICSME). pp 76–80
Srisopha K, Link D, Swami D, Boehm B (2020a) Learning features that predict developer responses for ios app store reviews. In: Proceedings of the 14th ACM / IEEE international symposium on empirical software engineering and measurement (ESEM), ESEM '20. Association for Computing Machinery, New York. https://doi.org/10.1145/3382494.3410686
Srisopha K, Phonsom C, Li M, Link D, Boehm B (2020b) On building an automatic identification of country-specific feature requests in mobile app reviews: Possibilities and challenges. In: Proceedings of the IEEE/ACM 42nd international conference on software engineering workshops, ICSEW'20. Association for Computing Machinery, New York, pp 494–498. https://doi.org/10.1145/3387940.3391492
Srisopha K, Swami D, Link D, Boehm B (2020c) How features in ios app store reviews can predict developer responses. In: Proceedings of the evaluation and assessment in software engineering, EASE '20. Association for Computing Machinery, New York, pp 336–341. https://doi.org/10.1145/3383219.3383258
Stanik C, Haering M, Maalej W (2019) Classifying multilingual user feedback using traditional machine learning and deep learning. In: 2019 IEEE 27th international requirements engineering conference workshops (REW). pp 220–226
Sun D, Peng R (2015) A scenario model aggregation approach for mobile app requirements evolution based on user comments. In: Requirements engineering in the big data era, vol 558. Springer, Berlin, pp 75–91
Sun Z, Ji Z, Zhang P, Chen C, Qian X, Du X, Wan Q (2017) Automatic labeling of mobile apps by the type of psychological needs they satisfy. Telematics Inform 34(5):767–778
Talia D (2019) A view of programming scalable data analysis: from clouds to exascale. J Cloud Comput 8(1):4
Tao C, Guo H, Huang Z (2020) Identifying security issues for mobile applications based on user review summarization. Inform Softw Technol 122:106290. https://doi.org/10.1016/j.infsof.2020.106290. https://www.sciencedirect.com/science/article/pii/S0950584920300409
Tavakoli M, Zhao L, Heydari A, Nenadić G (2018) Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools. Expert Syst Appl 113:186–199
Tizard J, Rietz T, Blincoe K (2020) Voice of the users: A demographic study of software feedback behaviour. In: Breaux TD, Zisman A, Fricker S, Glinz M (eds) 28th IEEE international requirements engineering conference, RE 2020, Zurich, Switzerland, August 31 - September 4, 2020. IEEE, pp 55–65. https://doi.org/10.1109/RE48521.2020.00018
Tong G, Guo B, Yi O, Zhiwen Y (2018) Mining and analyzing user feedback from app reviews: An econometric approach. In: 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp 841–848
Uddin MDK, He Q, Han J, Chua C (2020) App competition matters: How to identify your competitor apps? In: 2020 IEEE International Conference on Services Computing, SCC 2020, Beijing, China, November 7-11, 2020. IEEE, pp 370–377. https://doi.org/10.1109/SCC49832.2020.00055
Vasa R, Hoon L, Mouzakis K, Noguchi A (2012) A preliminary analysis of mobile app user reviews. In: Proceedings of the 24th Australian Computer-Human Interaction Conference, ACM. pp 241–244
Viera AJ, Garrett JM (2005) Understanding interobserver agreement: the kappa statistic. Fam Med 37(5):360–3
Villarroel L, Bavota G, Russo B, Oliveto R, Di PentaM (2016) Release planning of mobile apps based on user reviews. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). pp 14–24
van Vliet M, Groen EC, Dalpiaz F, Brinkkemper S (2020) Identifying and classifying user requirements in online feedback via crowdsourcing. In: Madhavji NH, Pasquale L, Ferrari A, Gnesi S (eds) Requirements engineering: foundation for software quality - 26th International Working Conference, REFSQ 2020,

Pisa, Italy, March 24-27, 2020, Proceedings [REFSQ 2020 was postponed], Springer, Lecture Notes in Computer Science, vol 12045. pp 143–159. https://doi.org/10.1007/978-3-030-44429-7_11

Vu PM, Nguyen TT, Pham HV, Nguyen TT (2015a) Mining user opinions in mobile app reviews: A keyword-based approach. In: Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering, ASE '15. IEEE Press, pp 749–459

Vu PM, Pham HV, Nguyen TT, Nguyen TT (2015b) Tool support for analyzing mobile app reviews. In: 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015, pp 789–794

Vu PM, Pham HV, Nguyen TT, Nguyen TT (2016) Phrase-based extraction of user opinions in mobile app reviews. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016, pp 726–731

Vu PM, Nguyen TT, Nguyen TT (2019) Why do app reviews get responded: A preliminary study of the relationship between reviews and responses in mobile apps. In: Proceedings of the 2019 ACM Southeast Conference, ACM SE '19. ACM, New York, pp 237–240

Wang C, Zhang F, Liang P, Daneva M, van Sinderen M (2018) Can app changelogs improve requirements classification from app reviews? an exploratory study. In: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '18. ACM, New York

Wang H, Wang L, Wang H (2020a) Market-level analysis of government-backed covid-19 contact tracing apps. In: Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASE '20. Association for Computing Machinery, New York, pp 79–84. https://doi.org/10.1145/3417113.3422186

Wang S, Wang Z, Xu X, Sheng QZ (2017) App update patterns: How developers act on user reviews in mobile app stores. In: Service-oriented computing - 15th International Conference, ICSOC 2017, Malaga, Spain, November 13-16, 2017, Proceedings. pp 125–141

Wang T, Liang P, Lu M (2018) What aspects do non-functional requirements in app user reviews describe? an exploratory and comparative study. In: 2018 25th Asia-Pacific Software Engineering Conference (APSEC). pp 494–503

Wang Y, Wang H, Fang H (2017) Extracting user-reported mobile application defects from online reviews. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW). pp 422–429

Wang Y, Zheng L, Li N (2020b) Rom: A requirement opinions mining method preliminary try based on software review data. In: Proceedings of the 2020 4th International Conference on Management Engineering, Software Engineering and Service Sciences, ICMSS 2020. Association for Computing Machinery, New York, pp 26-33. https://doi.org/10.1145/3380625.3380665

Wei L, Liu Y, Cheung SC (2017) Oasis: Prioritizing static analysis warnings for android apps based on app user reviews. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017. ACM, New York, pp 672–682

Weichbroth P, Baj-Rogowska A (2019) Do online reviews reveal mobile application usability and user experience? the case of whatsapp. In: 2019 Federated Conference on Computer Science and Information Systems (FedCSIS). pp 747–754

Wen P, Chen M (2020) A new analysis method for user reviews of mobile fitness apps. In: Kurosu M (ed) Human-computer interaction. human values and quality of life - thematic Area, HCI 2020, Held as Part of the 22nd International Conference, HCII 2020, Copenhagen, Denmark, July 19-24, 2020, Proceedings, Part III, Springer, Lecture Notes in Computer Science, vol 12183. pp 188–199. https://doi.org/10.1007/978-3-030-49065-2_14

Williams G, Mahmoud A (2018) Modeling user concerns in the app store: A case study on the rise and fall of yik yak. In: 2018 IEEE 26th international requirements engineering conference (rE). pp 64–75

Williams G, Tushev M, Ebrahimi F, Mahmoud A (2020) Modeling user concerns in sharing economy: the case of food delivery apps. Autom Softw Eng 27(3):229–263. https://doi.org/10.1007/s10515-020-00274-7

Wohlin C (2014) Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, EASE '14, ACM, New York

Xiao J (2019) Ospaci: Online sentiment-preference analysis of user reviews for continues app improvement. In: Yangui S, Bouguettaya A, Xue X, Faci N, Gaaloul W, Yu Q, Zhou Z, Hernandez N, Nakagawa EY (eds) Service-oriented computing - ICSOC 2019 workshops - WESOACS, ASOCA, ISYCC, TBCE, and STRAPS, Toulouse, France, October 28-31, 2019, Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 12019. pp 273–279. https://doi.org/10.1007/978-3-030-45989-5_23

Xiao J, Chen S, He Q, Wu H, Feng Z, Xue X (2020) Detecting user significant intention via sentiment-preference correlation analysis for continuous app improvement. In: Kafeza E, Benatallah B, Martinelli

F, Hacid H, Bouguettaya A, Motahari H (eds) Service-oriented computing - 18th International Conference, ICSOC 2020, Dubai, United Arab Emirates, December 14-17, 2020, Proceedings, Springer, Lecture Notes in Computer Science, vol 12571. pp 386–400. https://doi.org/10.1007/978-3-030-65310-1_27

Yadav A, Fard FH (2020) Semantic analysis of issues on google play and twitter. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). pp 308–309

Yadav A, Sharma R, Fard FH (2020) A semantic-based framework for analyzing app users' feedback. In: Kontogiannis K, Khomh F, Chatzigeorgiou A, Fokaefs M, Zhou M (eds) 27th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2020, London, ON, Canada, February 18-21, 2020. IEEE, pp 572–576. https://doi.org/10.1109/SANER48275.2020.9054843

Yang H, Liang P (2015) Identification and classification of requirements from app user reviews. In: The 27th International Conference on Software Engineering and Knowledge Engineering, SEKE 2015, Wyndham Pittsburgh University Center, Pittsburgh, PA, USA, July 6-8, 2015, pp 7–12

Zhang J, Wang Y, Xie T (2019) Software feature refinement prioritization based on online user review mining. Inf Softw Technol 108:30–34

Zhang L, Huang X, Jiang J, Hu Y (2017) Cslabel: An approach for labelling mobile app reviews. J Comput Sci Technol 32(6):1076–1089

Zhou Y, Su Y, Chen T, Huang Z, Gall HC, Panichella S (2020) User review-based change file localization for mobile applications. IEEE Trans Softw Eng :1–1. https://doi.org/10.1109/TSE.2020.2967383

Jacek Dąbrowski is a Doctoral Researcher in Software Engineering in a joint program between University College London, UK and Fondazione Bruno Kessler, Italy. His thesis is about mining online user feedback to support software engineering. He received his MSc and BSc in Electrical Engineering from Warsaw University of Technology, Poland. During his MSc he worked as a Junior Researcher at Aalto University, Finland where he completed his master thesis in Robotics. Before his doctoral studies, he worked in Accenture as a Technology Consulting Analyst where he was responsible for requirement engineering, software system design and digital transformation. His current research interests concern requirement engineering, mining software repositories and software analytics.

Emmanuel Letier is an Associate Professor in the Department of Computer Science at University College London where he teaches and research systems requirements engineering and software architecture.

Anna Perini is FBK Distinguished Fellow, former senior researcher at the Software Engineering research unit of Fondazione Bruno Kessler, Trento (Italy). Anna Perini teaches Requirements Engineering at the University of Trento, MSc degree in Computer Science. Her research interests include requirements engineering, agent-oriented software development methodologies, conceptual modelling, decision making in requirements engineering, and empirical studies. H-Index 43 (Google Scholar October 2021). Anna Perini is serving as program co-chair of RCIS 2021 served as program co-chair of IEEE RE'19, REFSQ 2017, and of STAIRS 2006. She is member of the Steering Committee of the IEEE RE Int. conference and chair of the Steering Committee of the REFSQ conference. Moreover, she has served as program committee member of several Int. Conferences, (e.g., ICSE, RE, CAiSE), and international workshops, and regularly reviews papers for top journals in the Software Engineering area.

Angelo Susi is a research scientist in the Software Engineering unit at Fondazione Bruno Kessler in Trento, Italy. His research interests are in the areas of requirements engineering, goal-oriented software engineering, formal methods for requirements validation, and search-based software engineering. He published more than 100 refereed papers in journals and international conferences such as TSE, TOSEM, IST, SoSyM, FSE, ICSE, RE. He participated in the organization committee of several conferences, such as SSBSE'12 (General Chair), REFSQ (workshop and industry chair), RE (Financial chair) and in program committees of international conferences and workshops (such as ICSE, RE, REFSQ, CAiSE and SSBSE). He also served as reviewer for several Journals such as TSE, REJ, IST, JSS. He has been the scientific manager of the EU FP7 project RISCOSS.