A Survey of Utilizing User-Reviews Posted on Google Play Store

Ehsan Noei
e.noei@utoronto.ca
University of Toronto

Kelly Lyons
kelly.lyons@utoronto.ca
University of Toronto

ABSTRACT
Mobile application (app) markets, such as Google Play Store, provide a rating mechanism for users to rate the hosted apps and leave comments and feedback (i.e., user-reviews). User-reviews contain valuable information, such as bug reports, feature requests, and user experiences. Recent studies have shown the unavoidable impact of studying users’ feedback on the success of an app, whereas ignoring users’ feedback can endanger the survival of an app in an app market. In this paper, we survey the research papers and solutions that can help developers and researchers to utilize user-reviews and integrate them into the app development process. We provide an overview of each work, briefly explain their applications, and finally mention the limitations. Moreover, derived from the existing body of research, we provide a guideline for researchers and developers, showing them how to collect, preprocess, and analyze user-reviews. Finally, we conclude the survey and provide directions for future research.

CCS CONCEPTS
• Software and its engineering → Software development process management; • Machine learning → Machine learning algorithms; • Information systems → Data extraction and integration.

KEYWORDS
User-review, Crowdsourcing, Data mining, Mobile application, Software maintenance

ACM Reference Format:

1 INTRODUCTION
Google Play Store has become an immensely competitive market for app developers due to the rapid increase in the number of mobile apps and smartphone users [67]. Google Play Store hosted more than two million Android apps at the time of this research [19].

As shown in Figure 1, Android app developers can publish their apps on Google Play Store, so users would be able to view the published apps and install them on their devices. Moreover, for each published app, Google Play lets users leave their comments and feedback (i.e., user-review). A user-review is an informal piece of text without a predefined structure [57] and it can contain several useful information, such as bug reports, feature requests, and reports of user experience [16, 57, 60].

Recent studies (e.g., [17, 56, 59, 60, 62]) have shown the importance of studying user-reviews, identifying bug reports and feature requests from them, and having them addressed in the next releases. For example, Noei et al. [52] observed that developers who address the issues that are reported in the user-reviews tend to be more successful than the ones who rarely address such issues. In addition, Noei et al. [52] reported that developers should not wait too long to release a newer version of an app. In another study Noei et al. [57] studied open-source Android apps that are available on both Google Play Store and GitHub [18]. They observed that addressing the issue reports that are more similar to the ones in the user-reviews share a statistically significant relationship with positive changes in star-ratings. Villarroel et al. [70] proposed a solution to classify user-reviews into clusters of bug reports and feature requests, and, thereby, helping developers in their release planning. Villarroel et al. [70] also ranked clusters of user-reviews based on some metrics such as the number of user-reviews and star-ratings.

This paper provides a survey of existing research work on utilizing user-reviews. Both researchers and developers should learn about state-of-the-art solutions to process, analyze, and utilize user-reviews. For each paper, we provide an overview of the work, its applications, and limitations. Therefore, researchers would be able to tackle open problems and challenges more effectively. In addition, by learning from the existing work, we provide a set of recommendations and techniques for developers and researchers, along with the steps that should be taken, in order to utilize user-reviews efficiently.

Paper Organization. We proceed by providing background information in Section 2. Then, in Section 3, the surveyed papers are introduced and discussed. Section 4 explains the implications of the existing work for both developers and researchers. Finally, Section 5 concludes the paper and provides directions for future research.
2 BACKGROUND

Google Play Store provides a platform for app developers to publish their apps and introduce them to users. Figure 1 shows an overview of interactions between users and developers on Google Play Store. As shown in Figure 1, once developers created a new app or modified their existing apps, they would be able to publish (i.e., release) it on the Google Play Store platform. Users can search, view, and download the published apps, or update their already installed apps using the Google Play Store app on their mobile devices. Users can also post their feedback for each app. Developers can view the received feedback and respond to any desired feedback.

2.1 Users’ Feedback

Users can rate each app and associate their ratings with user-reviews.

**Star-Ratings.** Google Play Store uses the star-rating mechanism to capture and demonstrate ratings. By the star-rating mechanism, users can rate each app from one star (the lowest) to five stars (the highest). However, unfortunately, there is no predefined standard or agreement on the meaning of star-ratings. For example, a user may interpret a three-star rating as an excellent rating, while another user may perceive it as a horrendous rating.

Star-ratings impact the income of app developers and app development companies [5, 38] as users rely on star-ratings for choosing an app to download [5, 51]. Users usually do not download and install an app with an average star-rating of less than three [54, 55]. Moreover, Harman et al. [26] reported a statistically significant relationship between the number of downloads and star-ratings. As a result, low-rated apps will lose their chance of surviving and succeeding in the competitive market of mobile apps. Developers should refer to the existing body of knowledge (e.g., [16, 17, 51, 54, 56, 57, 59, 60, 62]) to understand the factors that share a significant relationship with star-ratings. Thus, they would proactively quantify the expected star-ratings prior to releasing a newer version.

**Advantages of the Star-Rating Mechanism.** The star-rating mechanism provides users with an easy solution to rate their apps. Simply, users can give a five-star rating if they are completely satisfied with an app. Conversely, they can leave a one-star rating if they are utterly unhappy with an app.

Moreover, as star-ratings are just integer numbers between one and five, the star-rating mechanism makes it easy for developers and the Google itself to interpret, summarize, and visualize the given star-ratings.

**Disadvantages of the Star-Rating Mechanism.** Despite the convenience of the star-rating mechanism for users, developers, and the Google, the definition of star-ratings is not clear for both developers and users. For example, two users with the same user-experience may give two- or three-star ratings to the same app based on their personal interpretation of star-ratings. Different perceptions of star-ratings cause having star-ratings that are inconsistent with their associated user-reviews. For instance, consider the two user-reviews in Figure 2. The first user-review delivers positive feedback from the user, but it is associated with only a one-star rating. Interestingly, the second user-review is associated with a five-star rating! Inconsistent user-reviews introduce noises to statistical analyses. Noei et al. [57] and Fu et al. [16] have proposed solutions to filter-out inconsistent user-reviews (see Section 4.2).

Another major issue with the star-rating mechanism is that it can endanger the survival of an app as star-ratings are resilient to change once a substantial number of users rated an app [65]. It is recommended by Ruiz et al. [65] to remove an app that has received a very low star-rating and have it released as a new app instead of releasing a newer version. This issue will remain until Google modifies the star-rating mechanism and have the overall star-ratings calculated based on the most recent star-ratings or the star-ratings received for the most recent releases.

**User-Reviews.** A user-review is an informal piece of text without a predefined structure [57]. User-reviews are associated with star-ratings. Figure 2 provides two example user-reviews associated with star-ratings. User-Reviews are an important source of knowledge for app developers as they contain critical information, such as bug reports, feature requests, and user experiences [16, 60]. Recent studies have shown that addressing the issues reported in user-reviews helps developers to improve their apps, and, consequently, improve their star-ratings and ranks [52, 56]. The most important pieces of information hidden in the user-reviews are [51] (i) expectations of users from an app, (ii) users’ concerns, (iii) feature requests, (iv) bug reports, and (v) guidelines for a better release planning [51]. Unfortunately, due to the unstructured nature of user-reviews, developers need to take various steps to clean up the user-reviews first (see Section 4).

**Developers’ Response.** App developers can choose to respond to each user-review by posting a reply text. McIlroy et al. [44] studied 10,713 apps and reported that only 13.8% of apps respond to user-reviews. They also reported that 38.7% of negative user-reviews turned into positive ones after getting a proper response from developers explaining that they have addressed the issue or resolved the problem.

2.2 Continuous App Development

To succeed in the competitive market of mobile apps, developers adopt a continuous development paradigm [1, 47, 49]. The continuous app development is the process of continuously releasing...
high-quality versions of an app. To this end, developers need to analyze user-reviews and find out users’ demands, concerns, issues, and feature requests. Figure 3 shows an overview of the knowledge extraction and utilization process. As shown in Figure 3, developers need to constantly investigate the user-reviews and have them addressed in the next releases. Noei et al. [52] showed that the apps that fail to continuously improve their apps will eventually lose their ranks in Google Play Store.

3 SURVEYED PUBLICATIONS

In this section, the research papers that help developers and researchers to summarize user-reviews are introduced and discussed. Please note that, in this work, we do not cover all the existing work that utilizes user-reviews. For a more complete list of such studies please check the survey paper “a survey of app store analysis for software engineering” by Martin et al. [42]. Our criteria for choosing the papers are:

- Made a significant contribution at the time the research was published.
- Introduced an approach that can potentially help developers and researchers to better comprehend and utilize user-reviews.
- The proposed solution is useful and applicable as the time of this research.

Table 1 illustrates a timeline of the related studies and the venue in which each of them has been published. We find all the listed publications valuable where each of them can help developers and researchers from a different perspective. Therefore, we do not sort them based on specific criteria. For each study, first, we provide a brief overview; then, we discuss its applications and limitations.

Goul et al. [21]

**Overview.** Goul et al. [21] applied sentiment analysis on 5,000 user-reviews in order to facilitate the app requirements engineering. Goul et al. [21] reported that sentence-level and feature-based sentiment analysis is an informative solution for identifying user requirements.

**Application.** According to Goul et al. [21], users’ sentiments are important assets for app developers when investigating app requirements. App developers should carefully consider users’ sentiments in order to better identify user requirements.

**Limitation.** The study by Goul et al. [21] was based on the data from Apple App Store [3]. However, we included their work in this survey as it was one of the earliest attempts for putting users’ feedback to use [42].

Galvis and Winbladh [17]

**Overview.** Galvis and Winbladh [17] studied user-reviews in order to identify the changes that need to be made for the next releases of an app. Galvis and Winbladh [17] suggested a solution for extracting user requirements from user-reviews by applying topic modeling on user-reviews.

**Application.** The Galvis and Winbladh [17] solution is a straightforward approach for developers. Their approach is a fast and cheap solution that works better than manually analyzing user-reviews.

**Limitation.** Galvis and Winbladh [17] identified user requirements using a sentiment-aware topic model (i.e., ASUM model [35]). However, even though it improves the requirements summation process, it cannot find all the possible requirements.

Iacob and Harrison [31]

**Overview.** Iacob and Harrison [31] proposed a solution, called MARA, by applying linguistic rules on user-reviews in order to extract feature requests from them. Iacob and Harrison [31] manually trained their linguistic rules using 161 apps and 3,279 user-reviews. They evaluated their trained models using 136,998 user-reviews. Iacob and Harrison [31] observed that 23.3% of user-reviews contain feature requests.

**Application.** By employing MARA, developers would be able to identify the features requested by users. Iacob and Harrison [31]
also identified the topics of user-reviews that are associated with user-reviews.

**Limitation.** Due to the unstructured format of user-reviews and existence of typos and grammatical issues in user-reviews [56], applying linguistic rules may not successfully find all the feature requests reported in user-reviews. However, such an approach can be improved by preprocessing user-reviews (see Section 4).

**Fu et al. [16]**

**Overview.** Fu et al. [16] proposed a solution, called WisCom, to summarize user-reviews in three levels of (i) comments, (ii) apps, and (iii) app market. Fu et al. [16] applied topic modeling and built a linear regression model with the star-ratings as the dependent variable and user-reviews as the independent variable.

**Application.** WisCom is easily scalable and it can analyze millions of user-reviews. Fu et al. [16] also reported the top three complained aspects for each app category (e.g., *business* and *medical*), which can provide developers with more insights into user complaints.

**Limitation.** Fu et al. [16] applied their solution on over 13 million user-reviews. However, due to Google Play Store limitations, one cannot access a large number of user-reviews at once. Therefore, replication of such a study might not be feasible for future research.

**Chen et al. [10]**

**Overview.** Chen et al. [10] trained a classifier to distinguish between informative and uninformative user-reviews (see Section 4.2). Almost half the user-reviews contain no valuable information for developers but expressing users’ praises or hatreds. By having the uninformative user-reviews identified and removed, developers would be able to focus on the feedback that can actually help them to improve their apps. Chen et al. [10] also grouped the informative user-reviews using topic modeling and ranked the grouped user-reviews by considering various metrics such as star-ratings.

**Application.** Chen et al. [10] trained a classifier which is a practical yet simple solution for identifying informative user-reviews. Many recent studies, such as Nayebi et al. [48], adopted their approach to identify and remove inconsistent user-reviews.

**Limitation.** Removing uninformative user-reviews is not always the best solution. Recent studies [56, 70] showed that grouping uninformative set of user-reviews together may produce an informative set of user-reviews. Moreover, user-reviews are associated with star-ratings and users’ sentiments where both are valuable assets for developers and researchers.

**Khalid et al. [37]**

**Overview.** Khalid et al. [37] identified 12 major topics of user complaints from user-reviews by manually investigating a set of user-reviews. Their identified topics are (i) app crashing, (ii) compatibility app, (iii) feature removal, (iv) feature request, (v) functional error, (vi) hidden cost, (vii) interface design, (viii) network problem, (ix) privacy and ethical, (x) resource heavy, (xi) uninteresting content, and (xii) unresponsive app.

**Application.** The observations made by Khalid et al. [37] can help developers and researchers to better understand user-reviews and users’ complaints. Khalid et al. [37] reported that functional errors, feature requests, and app crashes are the most frequent complaints. They also observed that, in 11% of the user-reviews, users are complaining about the changes after an update.

**Limitation.** Khalid et al. [37] trained their models using user-reviews from Apple App Store which may not be generalizable to Android ecosystem [29].

**Guzman et al. [23, 24]**

**Overview.** Guzman et al. [23, 24] extended their earlier work [25] in which they studied users’ sentiment scores for apps from both Google Play Store and Apple App Store. Guzman et al. [23] proposed a tool, called DIVERSE, that identifies the user-reviews that mention similar features and have similar sentiment scores.

**Application.** Developers can use DIVERSE to query different features from their user-reviews. Guzman et al. [24] also classified user-reviews into seven topics of (i) bug reports, (ii) feature strength, (iii) feature shortcoming, (iv) user request, (v) praise, (vi) complaint, and (vii) usage scenario. Guzman et al. [24] achieved a precision of 0.74 and a recall of 0.59 on average.

**Limitation.** DIVERSE relies on user-reviews which are unstructured pieces of text and can be uninformative [10]. However, this limitation can be mitigated by clustering related user-reviews together [56] (see Section 4.2).

**Moran et al. [46]**

**Overview.** Moran et al. [46] introduced a tool, called Fusion, to help developers manage the bug reports mentioned in user-reviews. They also applied statistic and dynamic analysis on the source code or decompiled code (i.e., byte-code) of their subject apps.

**Application.** Developers can use Fusion as a systematic solution for resolving bug reports as it links users’ feedback to the source code. Therefore, developers can better reproduce bugs reports.

**Limitation.** Moran et al. [46] evaluated their approach using 15 bug reports of only 14 apps hosted on F-Droid app market [15]. The generalizability of their approach and the types of bug reports that it can address is not clear.

**Panichella et al. [62]**

**Overview.** Panichella et al. [62] manually analyzed a set of user-reviews and emails at a sentence-level granularity. Panichella et al. [62] identified five topics of user-reviews, including (i) feature requests, (ii) opinion asking, (iii) problem discovery, (iv) solution proposal, (v) information seeking, and (vi) information giving. Then, they applied natural language processing and sentiment analysis techniques on user-reviews to identify and extract feature requests. Panichella et al. [62] used the extracted features to classify user-reviews, so developers can better improve their apps.

**Application.** By applying Panichella et al. [62] approach, developers should be able to identify relevant information from user-reviews, and, therefore, be more responsive to users’ feedback.
Limitation. Panichella et al. [62] have only considered five topics of user-reviews which should not be enough due to the wide range of existing apps on Google Play Store with various functionalities and purposes [2].

Gu and Kim [22]

Overview. Gu and Kim [22] proposed a tool, called SUR-Miner, that summarizes and visualizes user-reviews. Gu and Kim [22] classified user-reviews, applied text analysis techniques, such as parsing user-reviews, and conducted sentiment analysis on user-reviews. By surveying actual developers, Gu and Kim [22] reported that the majority of developers agreed that their proposed tool could be useful in practice.

Application. SUR-Miner can be used by developers to understand user-reviews. Also, visualizing users’ feedback can help developers to better plan for their next releases.

Limitation. Gu and Kim [22] evaluated their approach using 17 popular apps and achieved an F1-measure of 0.81. However, their findings may vary on less popular apps with a limited number of user-reviews. Moreover, Gu and Kim [22] only considered five broad topics of user-reviews which should not be enough to study all the various demands and concerns that are mentioned in user-reviews.

McIlroy et al. [43]

Overview. McIlroy et al. [43] studied user-reviews of 20 apps. They found that users report different issues, including feature requests and bug reports, in a single user-review. McIlroy et al. [43] proposed a solution to assign multiple labels to user-reviews. They reported precision of up to 66% and recall of up to 65% for their labeling solution. Also, by manually analyzing a sample of user-reviews, McIlroy et al. [43] identified 14 types of issues, including (i) additional cost, (ii) functional complaint, (iii) compatibility issue, (iv) crashing, (v) feature removal, (vi) feature request, (vii) network problem, (viii) other, (ix) privacy and ethical issue, (x) resource heavy, (xi) response time, (xii) uninteresting content, (xiii) update issue, and (xiv) user interface.

Application. Developers and researchers can use McIlroy et al. [43] solution to label user-reviews. This will reduce the time and effort required to manage and analyze user-reviews.

Limitation. McIlroy et al. [43] used a limited number of apps (i.e., 20) to identify the labels which shall not a be representative sample set. Future research should replicate their study using a larger set of apps.

Panichella et al. [63]

Overview. Panichella et al. [63] applied natural language processing, text analysis, and sentiment analysis techniques to classify user-reviews into five main topics, including (i) information giving, (ii) information seeking, (iii) feature requests, (iv) problem discovery, and (v) others. They achieved a precision between 84% and 89%, a recall between 84% and 89%, and an F1-measure between 84% and 89% when classifying user-reviews from a maintenance point of view.

Application. Panichella et al. [63] have provided a tool, called ARdoc. Developers and researchers can use ARdoc to have their user-reviews classified into one of the five aforementioned topics. Therefore, developers should manage their time and resources when studying user-reviews.

Limitation. Although Panichella et al. [63] provided a practical tool for classifying user-reviews, their number of topics of user-reviews is limited (i.e., five broad topics). Therefore, developers may still need to look into each category of user-reviews in order to figure out users’ exact demands and concerns.

Di Sorbo et al. [14]

Overview. Di Sorbo et al. [14] proposed an approach, called SURF, to summarize user-reviews. Di Sorbo et al. [14] employed two levels of classification: (i) intention classification [63], and (ii) topic classification. The intentions are the same topics as in their earlier work discussed above [62, 63]. Di Sorbo et al. [14] introduced 12 more topics on top of their five intentions, including (i) app, (ii) graphical user-interface, (iii) contents, (iv) pricing, (v) feature or functionality, (vi) improvement, (vii) updates/versions, (viii) resources, (xi) security, (x) download, (xi) model, and (xiii) company.

Application. Di Sorbo et al. [14] attempted to resolve the limitation of only five intentions by adding 12 more topics on top of those intentions. Therefore, by applying their approach, developers can have a more precise understanding of the received feedback. Di Sorbo et al. [14] surveyed seven developers and the results show that developers find the summaries generated by their tool useful.

Limitation. Although Di Sorbo et al. [14] was an improvement over their previous work, there still exist a notable number of topics, such as speed and advertisement, which are not covered by their approach as they manually identified the topics.

Villarroel et al. [70]

Overview. Villarroel et al. [70] proposed a tool, called CLAP, to classify user-reviews into two major groups of bug reports and feature requests. Villarroel et al. [70] clustered similar user-reviews and ranked the clusters of user-reviews based on several metrics, including (i) the number of reviews in a cluster, (ii) the average rating of a cluster, (iii) the difference between the average rating of a cluster and average rating of an app, (iv) the average difference of the ratings assigned by users in the cluster who reviewed older releases of an app, and (v) the number of different hardware devices in a cluster.

Application. CLAP can help app developer in planning for the next releases of their apps by having similar user-reviews grouped together and ranked. Developers can investigate the clusters of user-reviews based on their priority and address users’ feedback in the next releases.

Limitation. Based on their evaluation [70], 66% of user-reviews get categorized as “other” instead of feature request or bug report. Considering the reported recall, 76% for bug reports and 67% for feature requests, 24% and 33% of bug reports and feature requests are missed, respectively. Moreover, Villarroel et al. [70] considered the number of star-ratings as one of the main factors when ranking
the clusters of user-reviews. However, Noei et al. [57] reported that the number of user-reviews is not always a correct indicator for the importance of an issue. In some cases, users may report an issue or request a feature frequently, but it may impact neither users’ satisfaction nor star-ratings.

Ciurumelea et al. [11]

Overview. Ciurumelea et al. [11] created a two-level taxonomy of concepts from user-reviews. In the highest level, they defined (i) compatibility, (ii) usage, (iii) resources, (iv) pricing, and (v) protection. On top of their taxonomy, they proposed an approach, called UUR, to organize user-reviews concerning users’ requests. Therefore, by having the user-reviews organized, Ciurumelea et al. [11] could suggest some source code modifications using code localization techniques [66].

Application. By employing the approach proposed by Ciurumelea et al. [11], developers would be able to identify the files that are related to the categorized user-reviews. This could save developers time and resources when addressing the user-reviews.

Limitation. Ciurumelea et al. [11] evaluated their approach using open-source apps that are hosted on F-Droid app market [15]. However, their findings may not be generalizable to all the proprietary apps that are hosted on Google Play Store. Moreover, the number of topics they considered (e.g., app usability and performance) shall not precisely cover all the topics of user-reviews.

Palomba et al. [61]

Overview. Palomba et al. [61] proposed an approach, called CHANGEADVISOR, following the approach proposed by Panichella et al. [62], to classify user-reviews and map them to source code. Palomba et al. [61] recommended the required source code changes to address users’ feedback by measuring the asymmetric Dice similarity coefficient [4] between the words in user-reviews and the words in each class of source code. They evaluated their approach using 44,683 user-reviews of 10 open-source apps. They achieved a precision of 81% and a recall of 70% in identifying source code components that are impacted by the suggested changes.

Application. By using the CHANGEADVISOR, developers would be able to localize the required changes based on the received feedback from users.

Limitation. CHANGEADVISOR does not prioritize the suggested changes. However, combining their approach with other prioritization solutions, such as [56, 57, 70], would create a practical solution for app developers and researchers.

Noei et al. [52]

Overview. Noei et al. [52] tracked the changes in the ranks of 900 apps in 30 most searched areas (e.g., dating, mailbox, and messaging) for two years. They reported that 61% of their understudied apps lost their initial ranks over the period of their study, and only 6% could improve their ranks. By studying the changes in ranks, Noei et al. [52] analyzed various factors that are statistically significantly related to the changes in ranks. They observed that constantly addressing the issues that are reported in the user-reviews will prevent an app from losing its rank.

Application. Noei et al. [52] provided a guideline for new app developers in order to succeed in the competitive market of mobile apps. Their guideline is based on their analyses and surveying 51 app developers. Moreover, they suggested developers of new apps, that have received no or a limited number of user-reviews, to study user-reviews, features, and descriptions of other similar apps in the competition.

Limitation. Noei et al. [52] investigated the factors that are statistically significantly related to the ranks. However, for a company that is not concerned about the ranks, their findings may not be interesting.

Hassan et al. [27]

Overview. As developers can directly respond to user-reviews, Hassan et al. [27] studied such an interaction between developers and users. They investigated ~ 4.47 million user-reviews and 126,686 responses of 2,328 top free apps. Hassan et al. [27] observed that in almost one-third of the cases that developers respond to user-reviews, the associated star-ratings have been increased afterward.

Application. According to the findings of Hassan et al. [27], developers should provide users with a proper response explaining the changes that they have made to address the reported issue. Similarly, Noei et al. [52] suggested that the changes should also be included in the release notes.

Limitation. Responding to all the received feedback is not a trivial task for developers and may be expensive for developers. Google Play Store should provide developers with a more productive response mechanism.

Noei et al. [56]

Overview. As addressing the issues that are reported in user-reviews can potentially lead to better star-ratings and ranks [52], Noei et al. [56] proposed a solution to prioritize the user-related issue reports. They integrated user-reviews into the process of issue report prioritization by, first, clustering related user-reviews together. Then, Noei et al. [56] identified the issue reports (from GitHub) that are related to the clusters of user-reviews (from Google Play Store) with a precision of 79%. By having the user-reviews matched with issue reports, they integrated users’ feedback (e.g., star ratings, sentiment scores, and the size of user-reviews) into issue reports prioritization. Noei et al. [56] reported that prioritizing the issue reports that are related to user-reviews shares a statistically significant relationship with star-ratings.

Application. Different developers might follow different prioritization approaches when it comes to closing an issue report on GitHub. Some developers address the issues that are reported by senior developers first [12] while some developers address the issues that are reported in more user-reviews on Google Play Store. Noei et al. [56] proposed a systematic approach to have the user-reviews matched with issue reports, so developers can prioritize the issue reports using metrics from both Google Play Store and
GitHub. Furthermore, developers should be able to avoid issue reports duplication [9] prior to adding issues that are reported in user-reviews to an issue tracking system.

**Limitation.** The proposed approach by Noei et al. [56] has been evaluated using open-source Android apps that are hosted on GitHub. However, their findings may not be generalizable to the apps that are hosted on other code repositories or issue tracking systems. Future research should evaluate their approach in other ecosystems and source code repositories.

**Noei et al. [57]**

**Overview.** Noei et al. [57] studied ~ 4 million user-reviews of 623 apps in ten different categories (e.g., business and social). They identified the topics of user-reviews that are statistically significantly related to star-ratings (key topics). Noei et al. [57] observed that the key topics of user-reviews are not necessarily the most frequent topics of user-reviews. They evaluated their approach using release notes of their subject apps, and reported, for 77% of the apps on average, having a similar release note to the key topics shares a statistically significant relationship with positive changes in star-ratings.

**Application.** As Noei et al. [57] observed that the most frequent issue reports or feature requests are not always associated with a better or worse star-rating, developers should not be distracted by the frequent topics. In fact, user-reviews that contain reports of an issue related to the key topics should be addressed first. Moreover, testing teams should pay special attention to the key topics that are shared amongst the majority of app categories.

**Limitation.** Noei et al. [57] studied ten categories of mobile apps. However, for the remaining categories, a similar study is required to cover all the categories. Moreover, the key topics may change over time and the same approach is required to recalculate the key topics in the future.

## 4 IMPLICATIONS

In this section, the essential steps required to collect, prepare, and analyze user-reviews are explained.

### 4.1 Data Collection

The data collection process is easier for app developers than external researchers (e.g., researchers and students). App developers have access to users’ feedback via Google Play Store developers console [20]. In addition, for each user-review, developers can view the user’s device, language, and hardware specifications. By having access to all this information, developers can study their users’ feedback more precisely and conveniently.

For researchers and students, user-reviews of non-owned apps should be crawled from Google Play Store. However, due to the Google limitations in accessing all the user-reviews of an app, researchers should gradually crawl their required data from Google Play store [16, 56]. Moreover, as recommended by Noei et al. [52], even developers should also study user-reviews of other similar apps in the competition in order to succeed. Therefore, developers should crawl and consider user-reviews of other apps in the competition as well.

Having an incomplete set of users’ feedback can introduce bias to the findings of a study as Martin et al. [41] reported that using an incomplete set of data in Blackberry World App Store [7] biases the final findings. A similar bias may also be introduced to the findings of a study that is conducted using an incomplete set of data from Google Play Store. Future research should shed more light into this.

### 4.2 Data Preprocessing

A user-review is an informal piece of text [19, 57, 60] that usually suffers from grammatical issues and typos. A user-review such as “it ws workin fine till it crashddddd” contains several issues and typos: “wsa”, “workin”, “crashddddd” should be replaced with “was”, “working”, and “crashed”, respectively. Moreover, there are no standards or consistent choices of words and terms to describe an issue. For example, a user may use the term glitch to report an issue while another user with the same issue may use the term problem to report it [56]. In addition, user-reviews contain negations that can disrupt automatic text analysis approaches [57, 70]. For instance, a user-review such as “There is no problem using this app!” may be interpreted as a user-review that reports a problem because of having the term “problem” in it. As a result, preprocessing user-reviews is an essential part of studying user-reviews. In this section, the most important steps that need to be taken are explained.

**Identifying Inconsistent User-Reviews.** User-reviews that are posted on Google Play Store suffer from inconsistencies with the associated star-ratings [16, 30, 57]. Imagine two users with exactly the same perceived quality from the same app. These two users can rate the app differently based on their personal interpretation of star-ratings. Noei et al. [51] observed that some user-reviews with negative content can be associated with high star-ratings, and vice versa (see Figure 2). Consequently, the accuracy of a study can be tainted by inconsistent user-reviews. Various solutions have been proposed to identify inconsistent user-reviews [16, 52].

Fu et al. [16] built a regression model and tested it using 50,000 user-reviews in order to identify the inconsistent user-reviews. They checked the differences between the star-ratings and user-reviews.

As another solution, Noei et al. [52, 57] compares the sentiment scores [69] of user-reviews with the associated star-ratings to identify the inconsistent user-reviews. Different sentiment analysis tools [36], such as SentiSTRENGTH [68], SentiSTRENGTH-SE [33], and Natural Language Toolkit (NLTK) [6], can be used to capture the sentiment scores of user-reviews. However, there is not a solid sentiment analysis tool trained using user-reviews on Google Play Store. Noei et al. [52] employed SentiSTRENGTH-SE [33] which is trained using software engineering artifacts. The generated sentiment scores are between −5 and +5: the most negative user-reviews are scored as −5 and the most positive ones are scored as +5 [68]. The user-reviews are also associated with star-ratings between 1 and 5 [19]. As the majority of users do not download the apps with star-ratings of less than three [54], Noei et al. [51] considers the star-ratings of 3 as neutral ratings, star-ratings below 3 as negative ones, and star-ratings above 3 as positive ones. Noei et al. [51] defines a consistent user-review as the one that holds a positive star-rating with a positive sentiment score, or a neutral star-rating...
with a neutral sentiment score, or a negative star-rating with a negative sentiment score [51].

**Identifying Uninformative User-Reviews.** An uninformative user-review is a user-review that provide no applicable information for app developers [10, 70]. For example, a user-review such as “Very good” expresses praise by a user. On the other hand, a user-review such as “disappointed with the full version. I love this app and I have been using it for a long time, so I decided to get the full version but some features disappeared, like the possibility to add more photos to edit at the same time...” gives more information, regarding an app functionality, to developers. Removing uninformative user-reviews narrows down the number of user-reviews and reduces noises while studying user-reviews. Many solutions have been proposed to remove or utilize uninformative user-reviews.

Chen et al. [10] applies Expectation Maximization for Naïve Bayes method [50] and builds a classifier to distinguish between informative and uninformative user-reviews.

Noei et al. [56] groups similar user-reviews together and argues that even user-reviews that are considered as uninformative user-reviews by Chen et al. [10] can become informative when they are studied as a group of user-reviews. Noei et al. [56] employs linguistic rules to filter out the user-reviews that only express praises or dissatisfactions towards an app.

Villarroel et al. [70] applies preprocessing steps on user-reviews, such as n-grams extraction and negations management. Villarroel et al. [70] clusters user-reviews in groups of feature requests and bug reports. They reported that their solution outperforms the solution proposed by Chen et al. [10].

Removing or keeping uninformative user-reviews depends on the research goals. For example, when users’ satisfaction is desired, uninformative user-reviews should be kept. Also, by grouping user-reviews [56, 70], the elimination of uninformative user-reviews should be restricted. On the other hand, when user-reviews are explicitly used to identify feature requests or bug reports, uninformative user-reviews should be identified and removed.

**Correcting Typos.** Typos and misspelling in the user-reviews can disrupt text analysis techniques, such as topic modeling [58]. To mitigate the risk of missing valuable information and reducing the potential noises, typos should be corrected and replaced with the right words and terms. For example, Noei et al. [51] uses Jazzy Spell Checker [34] with a dictionary of 645,289 English words to correct the typos in user-reviews.

**Coreference Resolution.** Coreference occurs when two or more expressions refer to the same referent [13, 57]. A user-review such as “Great app, a little slow. wish they had transitions between shots! or maybe I’m just not seeing it. good app nonetheless” contains coreferences. The second part of the user-reviews uses a pronoun (i.e., it) that refers to “transitions between shots”. Unfortunately, systematic techniques may not understand such references. A tool such as Stanford deterministic coreference resolution [39] should be useful to resolve the coreferences in the user-reviews. After coreference resolution, the above example user-reviews should be converted to “Great app, a little slow. wish they had transitions between shots! or maybe I’m just not seeing transitions between shots. good app nonetheless”.

**Labeling and Annotation.** As reported by McIlroy et al. [43], users may report several concerns and demands, such as bug reports and feature requests, in one single user-review. For example, a user-review such as “You can only edit photos but whenever I wanted to edit videos I clicked on it and the app stopped working. I’ve tried it multiple times and it still doesn’t work” should be broken into two smaller pieces: (i) “You can only edit photos” and (ii) “whenever I wanted to edit videos I clicked on it and the app stopped working. I’ve tried it multiple times and it still doesn’t work”. Different solutions have been proposed to resolve such an issue [43, 57].

McIlroy et al. [43] suggests an approach to automatically assign multiple labels to user-reviews with a precision of 66% and a recall of 65%. Noei et a. [57] employs Stanford CoreNLP [40] to break the user-reviews with several concerns into smaller pieces. Therefore, each smaller piece shares an independent concern. Stanford CoreNLP (i) annotates the words in the user-reviews, (ii) produces the base forms and the parts of speech, and (iii) identifies the structure of sentences.

**Resolving Synonyms.** Synonyms should be resolved as resolving synonyms increases the precision of statistical analyses and modeling techniques [53, 56, 70]. Unfortunately, there is no thesaurus or dictionary of words related to users’ feedback and the slangs and vocabularies that users use to express their feedback. General-purpose thesaurus, such as WordNet [45], are not sufficient enough to resolve the synonyms of user-reviews [56, 70].

Earlier studies [52, 70] have built their own dictionary of words by manually investigating a sample of user-reviews. For example, Bavota et al. [5] manually analyzed 1,000 user-reviews to build their own dictionary of words. Noei et al. [52] manually studied 5,000 user-reviews to build such a dictionary. Yet having a unified dictionary is an open problem in this area as it impacts the accuracy of research findings.

In addition to synonyms, the abbreviation and informal messaging vocabularies should also be handled or replaced with proper formal terms. For example, “w8” should be replaced with “wait”.

**Resolving Negations.** The negations in user-reviews disrupt text analysis and topic modeling techniques [52, 56, 70]. One way to handle negations, as also employed by Noei et al. [56], is using the Stanford natural language processing toolkit [40] that helps developers and researchers in finding and resolving the negated verbs and terms [52].

**Clustering.** As discussed earlier in this section, clustering user-reviews into groups of similar user-reviews can turn uninformative user-reviews into informative groups of user-reviews. Noei et al. [56] reported a significant 45% increase in the precision of mapping user-reviews (from Google Play Store) to issue report (from GitHub) after grouping similar user-reviews together.

### 4.3 Analysis

Studying and analyzing user-reviews heavily depends on the goals of a company or researcher. Some earlier studies investigate the factors that are statistically significantly related to star-ratings, the number of downloads, or ranks [42], while some other studies attempt to summarize user-reviews and put them to use [42]. Once the user-reviews got cleaned up (see Section 4.2), if knowledge
A Survey of Utilizing User-Reviews Posted on Google Play Store

eering and easiness the development process using state-of-the-art data
ence from user-reviews more efficiently. The existing work provides
developers would be able to extract bug reports and user experi-
itits feedback mechanism and making it more structured. Therefore,
challenge. Google Play Store should ease this process by enhancing
user-reviews, a solid approach is required to generate clear, concise,
all the attempts to extract bug reports and feature request from
Google Play Store rating solution.

not an ideal solution. This issue will remain until Google improves
an app from the store and re-uploading it as a new app, which is
over a period of two years. Ruiz et al. [25] studied 900 apps and
reported that 61% of the understudy apps lost their initial ranks
over a period of two years. Ruiz et al. [64] suggests removing such
an app from the store and re-uploading it as a new app, which is
not an ideal solution. This issue will remain until Google improves
the star-rating mechanism or incorporates recent studies into the
Google Play Store rating solution.

In the future, we expect more research on user-reviews. Besides
all the attempts to extract bug reports and feature request from
user-reviews, a solid approach is required to generate clear, concise,
and complete reports from user-reviews. However, due to the unin-
structed and informal format of user-reviews, this is not an easy
challenge. Google Play Store should ease this process by enhancing
its feedback mechanism and making it more structured. Therefore,
developers would be able to extract bug reports and user experi-
ence from user-reviews more efficiently. The existing work provides
developers with knowledge and insight into app development
process and eases the development process using state-of-the-art data
mining and machine learning techniques. We expect to hear more
about an automated mobile app engineering solution integrating
user-reviews into an adaptable mobile app development paradigm.

5 CONCLUSION

Utilizing user-reviews is an important part of the app development
process as it impacts the star-ratings and rank of an app. In this
paper, we survey the earlier work on utilizing user-reviews. We discuss
and explain the related papers and solutions in chronological
order, and, for each paper, we provide an overview in addition
to applications of findings and limitations. Moreover, we provide
a guideline for future research, including the essential steps of
data collection and data preprocessing. Researchers and developers
should: (i) identify inconsistent user-reviews, (ii) identify uninfor-
mative user-reviews, (iii) correct typos, (iv) resolve coreferences,
(v) annotate user-reviews, (vi) resolve synonyms, (vii) resolve nega-
tions, and (viii) group similar user-reviews together, in order to
achieve better and more accurate results.

Unfortunately, the current rating mechanism in Google Play Store is not fair to app developers. Once an app receives a notable
number of negative star-ratings, it would be impossible for its
developers to fix the overall star-rating. This is a major problem as
users usually do not download the app with an overall star-rating
of less than three. Therefore, developers should always pay extra
attention to users’ feedback. Noei et al. [52] studied 900 apps and
reported that 61% of the understudy apps lost their initial ranks
over a period of two years. Ruiz et al. [64] suggests removing such
an app from the store and re-uploading it as a new app, which is
not an ideal solution. This issue will remain until Google improves
the star-rating mechanism or incorporates recent studies into the
Google Play Store rating solution.

In the future, we expect more research on user-reviews. Besides
all the attempts to extract bug reports and feature request from
user-reviews, a solid approach is required to generate clear, concise,
and complete reports from user-reviews. However, due to the unin-
structed and informal format of user-reviews, this is not an easy
challenge. Google Play Store should ease this process by enhancing
its feedback mechanism and making it more structured. Therefore,
developers would be able to extract bug reports and user experi-
ence from user-reviews more efficiently. The existing work provides
developers with knowledge and insight into app development
process and eases the development process using state-of-the-art data
mining and machine learning techniques. We expect to hear more
about an automated mobile app engineering solution integrating
user-reviews into an adaptable mobile app development paradigm.

REFERENCES

why researchers should care. In Proceedings of 23rd International Conference on
Software Analysis, Evolution, and Reengineering (SANER), Vol. 5. 78–90.
[2] Aiman A Al-Sabahii, Federica Sarro, Sue Black, Lucia Capea, Mark Harman,
Yue Jia, and Yauyan Yu Zhang. 2016. Clustering mobile apps based on mined
textual features. In Proceedings of the 10th ACM/IEEE international symposium
on empirical software engineering and measurement. ACM, 38.
store/.
[5] Gabriele Bavota, Mario Linares-Vasquez, Carlos Eduardo Bernal-Cardenas, Mas-
similiano Di Penta, Rocco Oliveto, and Denys Poshyvanyk. 2015. The impact of
API Change-and Fault-Proneness on the User Ratings of Android Apps. IEEE
with Python: analyzing text with the natural language toolkit. ’O’Reilly Media,
Inc.’.
[9] Yegarasat Chuenprasert Charuvanakit, Paulo Anselmo da Mota Silveira Neto, Daniel
Lucrédio, Tassio Vale, Eduardo Santana de Almeida, and Silvio Romero de Lemos
[10] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Bohden Zhang. 2014. AR-
miner: mining informative reviews for developers from mobile app marketplace.
ACM, 767–778.
Planning. In Proceedings of the 24th International Conference on Software Analysis
Evolution and Reengineering. IEEE.
mapping study of software development with Github. IEEE Access 5 (2017), 7173–
7192.
& Sons.
[14] Andrea Di Sorbo, Sebastiano Panichella, Carol V Alexandru, Junji Shimagaki,
Corrado A Visaggio, Gerardo Canfora, and Harald C Gall. 2016. What would
users change in my app? summarizing app reviews for recommending software
[16] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason Hong, and Norman Sadeh.
2013. Why people hate your app: Making sense of user feedback in a mobile app
store. In Proceedings of the 19th International Conference on Knowledge Discovery
and Data Mining. ACM, 1276–1284.
[17] Laura V Galvis Carreiro and Kristina Wushdh. 2013. Analysis of user comments:
an approach for software requirements evolution. In Proceedings of the 35th
aging the enterprise business intelligence app store: Sentiment analysis supported
requirements engineering. In 45th Hawaii International Conference on System
Sciences. IEEE, 4168–4177.
[22] Xiaodong Gu and Sunghun Kim. 2015. “What Parts of Your Apps are Loved by
Users?” (T). In 30th IEEE/ACM International Conference on Automated Software
Engineering. IEEE, 760–770.
[23] Emilita Guzman, Omar Aly, and Bernd Bruegge. 2015. Retrieving diverse opin-
ions from app reviews. In 2015 ACM/IEEE International Symposium on Empirical
Methods for App Review Classification: An Approach for Software Evolution
(N). In 2015 30th IEEE/ACM International Conference on Automated Software