Empir Software Eng (2018) 23:1275–1312
DOI 10.1007/s10664-017-9538-9

Studying the dialogue between users and developers of free apps in the Google Play Store

Safwat Hassan[1] · Chakkrit Tantithamthavorn[2] · Cor-Paul Bezemer[1] · Ahmed E. Hassan[1]

Published online: 8 September 2017

Abstract The popularity of mobile apps continues to grow over the past few years. Mobile app stores, such as the Google Play Store and Apple's App Store provide a unique user feedback mechanism to app developers through the possibility of posting app reviews. In the Google Play Store (and soon in the Apple App Store), developers are able to respond to such user feedback. Over the past years, mobile app reviews have been studied excessively by researchers. However, much of prior work (including our own prior work) incorrectly assumes that reviews are static in nature and that users never update their reviews. In a recent study, we started analyzing the dynamic nature of the review-response mechanism. Our previous study showed that responding to a review often has a positive effect on the rating that is given by the user to an app. In this paper, we revisit our prior finding in more depth by studying 4.5 million reviews with 126,686 responses for 2,328 top free-to-download apps in the Google Play Store. One of the major findings of our paper is that the assumption that reviews are static is incorrect. In particular, we find that developers and users in some cases use this response mechanism as a rudimentary user support tool, where dialogues emerge between users and developers through updated reviews and responses. Even though

Communicated by: Andreas Zeller

✉ Safwat Hassan
shassan@cs.queensu.ca

Chakkrit Tantithamthavorn
chakkrit.tantithamthavorn@adelaide.edu.au

Cor-Paul Bezemer
bezemer@cs.queensu.ca

Ahmed E. Hassan
ahmed@cs.queensu.ca

[1] Software Analysis and Intelligence Lab (SAIL), Queen's University, Kingston, Ontario, Canada

[2] School of Computer Science, The University of Adelaide, Adelaide, Australia

the messages are often simple, we find instances of as many as ten user-developer back-and-forth messages that occur via the response mechanism. Using a mixed-effect model, we identify that the likelihood of a developer responding to a review increases as the review rating gets lower or as the review content gets longer. In addition, we identify four patterns of developers: 1) developers who primarily respond to only negative reviews, 2) developers who primarily respond to negative reviews or to reviews based on their contents, 3) developers who primarily respond to reviews which are posted shortly after the latest release of their app, and 4) developers who primarily respond to reviews which are posted long after the latest release of their app. We perform a qualitative analysis of developer responses to understand what drives developers to respond to a review. We manually analyzed a statistically representative random sample of 347 reviews with responses for the top ten apps with the highest number of developer responses. We identify seven drivers that make a developer respond to a review, of which the most important ones are to thank the users for using the app and to ask the user for more details about the reported issue. Our findings show that it can be worthwhile for app owners to respond to reviews, as responding may lead to an increase in the given rating. In addition, our findings show that studying the dialogue between user and developer can provide valuable insights that can lead to improvements in the app store and user support process.

Keywords Google play store · User-developer dialogue · Developer reply · Developer response · Mixed-effect model · Android mobile apps · Empirical study · Software engineering

1 Introduction

Mobile apps continue to rapidly gain popularity over the last few years. Mobile apps can be downloaded from app stores, such as the Google Play Store, which has more than 3.1 million apps available as of July 2017 (AppBrain 2017). These app stores allow users to express their opinion about an app through posting reviews, including a rating, that other potential users of the app see in the store.

A 2015 survey shows that 69% of the users consider the app rating as an important or very important deciding factor when downloading an app. In addition, 77% of the users will not download an app that has a rating that is lower than 3 stars (Martin 2017). Hence, the success of an app is closely tied to the reviews and ratings that it receives.

The reviewing and rating processes have always been one-way mechanisms, as developers[1] were not allowed to respond to the reviews or ratings. As a result, issues that are raised in reviews can discourage other users from downloading the app, even though the raised issue may be inaccurate or might be easily solved.

Recently, app stores have given developers the opportunity to engage in a dialogue with the users of their apps by responding to user reviews. The Google Play Store provides guidelines for responding to a posted review (Google 2017). In addition, Apple's App Store is expected to add support for responding to a posted review soon (Perez 2017). Such dialogue allows developers to provide possible solutions for the issues that are raised in the review, or to ask users to clarify their unsatisfaction with their app. In addition, developers can also

[1]Throughout this paper we use 'developer' to indicate the person(s) or company who are responsible for making an app.

encourage users to change their review or rating, which in turn can result in more downloads as reviews become more positive.

Prior work on mobile app reviews focuses on extracting useful information for developers from those reviews, such as bug reports or feature requests. However, prior work (including our own) falsely assumes the following: (1) app reviews and ratings are considered to be immutable, even though a user may change them over time, and (2) reviewing and rating apps is considered a one-way mechanism, even though the dynamic nature of app reviews and the response mechanism can lead to a rich dialogue between developers and users.

In this paper, we empirically study the dynamic nature of app reviews. In particular, we study the dialogue that takes place between users and developers in 2,328 free-to-download apps in the Google Play Store. We study 126,686 dialogues that contain messages between the user and the developer. First, we conduct a preliminary study of the benefit of responding to a user review. We show that responding to a review increases the chances of users updating their given rating for an app by up to six times compared to not responding. Second, we conduct the following studies:

Study I: A Study of the Characteristics of User-Developer Dialogues

Motivation To understand how users and developers interact via the review system in the Google Play Store, we study the characteristics of these dialogues that emerge through back-and-forth (updated) app reviews and developer responses.

Results A user-developer dialogue was triggered by 2.8% of the user reviews. If users change the review rating after a developer response, users tend to increase the review rating (in 4.4% of the reviews that received a response). In comparison, only 0.7% of the reviews that do not receive a response change their rating. Hence, app owners should assign more effort to responding to user reviews as a response is likely to lead to an increased rating.

Study II: A Quantitative Study of the Likelihood of a Developer Responding

Motivation In our preliminary study and Study I, we observed that responding to reviews can be beneficial for developers. Therefore, in our second study, we investigate which metrics are related to the likelihood of a developer responding to a review. The goal of Study II is to provide recommendations for store owners and developers to easier identify reviews that may require a response. For example, store owners could automatically highlight reviews that a developer is likely to respond to.

Results When developers respond, they tend to respond to reviews that are longer and have a low rating. In addition, we find four patterns of developers: (1) developers who primarily respond to only negative reviews, (2) developers who primarily respond to negative reviews or to reviews based on their contents, (3) developers who primarily respond to reviews which are posted shortly after the latest release of their app, and (4) developers who primarily respond to reviews which are posted long after the latest release of their app.

Study III: A Qualitative Study of What Drives a Developer to Respond

Motivation In Study II we found how we can identify reviews that may require a response. However, the results obtained from Study II do not explain the actual contents of a response.

Understanding the contents of developer responses can lead to improvements in the review-response mechanism, or to improvements in the way that developers use reviews and responses. For example, if developers turn out to ask users for more details in many of their responses, such requests can be better accommodated by next-generation automated review mechanisms. Therefore, we conduct a manual study of the contents of responses to understand better what drives developers to respond to reviews.

Results We manually examined a statistically representative random sample of 347 reviews for the top ten apps with the highest number of developer responses. We identify seven different drivers for responding, of which the most important ones are to thank the user for using the app and to ask for more details about the reported issue. In addition, we uncover interesting opportunities for developers, such as the opportunity to automatically generate frequently asked questions (FAQs) that can be published to improve the user support process.

The main contributions of this paper are as follows:

1. Our paper is the first work to demonstrate the dynamic nature of reviews. Much of prior work (including our own prior work) incorrectly assumes that reviews are static in nature and users never update their reviews. Our paper shows that this is an incorrect assumption – an assumption that impacts studies in this nascent research area. For example, prior work on studying user complaints in mobile app reviews is impacted, as the static view that such work currently has of reviews cannot reflect changing user complaints.
2. Furthermore, we are the first to demonstrate a peculiar use of the app-review platforms as a user support medium – we do acknowledge that not as many developers are using the review system in this way. Nevertheless, this is a use which further work needs to explore, especially given that the Apple App Store is expected to add support for responding to a posted review soon (Perez 2017). Given the nascent nature of app reviews, it might be worthwhile that such type of use is encouraged and better supported in next generation app-review platforms.

   By encouraging such use, future mining studies of reviews will have access to a richer and more complete view of user concerns. Currently, in many instances developers encourage users to continue discussions via other mediums (e.g., email). Hence, the reviews do not provide an in-depth view of the user concerns (as previously thought – for instance, this observation impacts recent publications which mine reviews to gather requirements).
3. Furthermore, our work is the first work to deeply explore developer responses in a systematic manner. Our analysis is much deeper and uses considerably more data. The more in-depth analysis led us to reformulate some of our prior observations/recommendations. For instance, in comparison to our prior work (McIlroy et al. 2015), we still see that responding to user reviews may increase the rating but we find that the chances of a user updating a review based on a developer response to be quite low (even though they are up to six times higher than without a response). A deeper investigation shows that all too often, the developer responses are rather simplistic and are just asking users to raise their ratings.
4. Finally, our classification of developer-responses highlights the value of providing canned or even automated responses in next generation app-review platforms. For instance, 45% of the developer-responses ask for more details. Furthermore, the ability to auto-construct FAQs based on the commonly repeated questions and answers in the user-developer dialogue is a valuable one for next generation app-review platforms.

The rest of this paper is organized as follows. Section 2 gives background information about the user-developer dialogue in the Google Play Store. Section 3 describes our methodology for collecting the user-developer dialogues from the Google Play Store. Section 4 describes a preliminary study for the user-developer interactions. Section 5 discusses the characteristics of user-developer dialogues in the Google Play Store. Sections 6 and 7 present our quantitative and qualitative studies of what drives developers to respond. Section 8 discusses the implications of our studies. Section 9 discusses the limitations and threats to the validity of our findings. Section 10 discusses the related work and describes the difference between our prior work and the current work that is presented in this paper. Finally, Section 11 presents our conclusion.

2 Background

In this section, we give background information for our study by explaining how the user-developer dialogue in the Google Play Store works. Figure 1 shows an abstraction of the user-developer dialogue in the Google Play Store. A user initiates the dialogue by posting a review, including a rating, for an app (S_1). User reviews can contain valuable information such as bug reports, feature requests and complaints or praise about the user experience (Maalej and Nabil 2015). In turn, the app developer engages in the dialogue by responding to the review (S_2). The Google Play Store notifies a user when a developer responds to the user review. Both the user review and the developer response can be updated at any time. Hence, by considering the changes made to the review and the response in a chronological order, we can see the dialogue between the user and developer. Note that the app store shows only the latest versions of the review and the response. Hence, to study such a dialogue we must continuously crawl the app store reviews to spot any changes to user reviews or developer responses. Today, the Google Play Store is the only store with support for developer responses. Apple's App Store is expected to add support for responding to reviews soon (Perez 2017).

The Google Play Store sends an email notification to app users when a developer responds to their comments (i.e., when the user-developer dialogue changes from state S_1 to state S_2) (Google 2017). In addition, the Google Play Store sends email notifications to app developers when users change their posted reviews after a developer response (i.e., when the user-developer dialogue changes from state S_2 to state S_1). However, the store does not notify developers when users keep updating their review (i.e., remain in state S_1). In

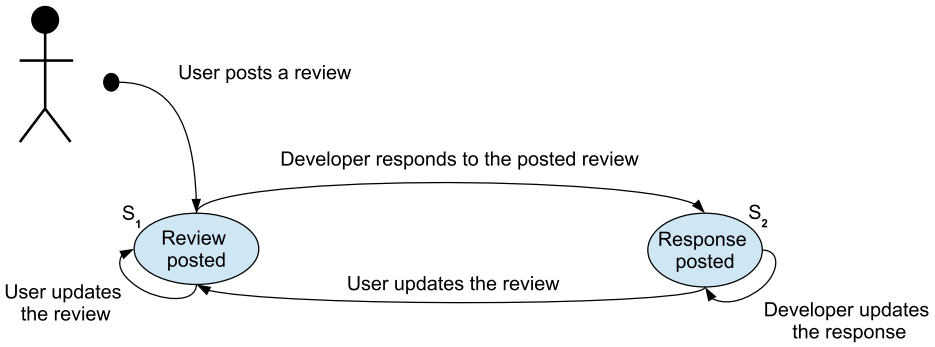

Fig. 1 User-developer dialogue in the Google Play Store

addition, the store does not notify users when developers keep changing their response (i.e., remain in state S_2). In summary, developers and users are not always notified when a review or response changes.

Table 1 shows an example of how such a dialogue emerges when a user and a developer update their review and response for the *AppLock* app. The dialogue shows how a user raises an issue in his review and how the developer helps the user to solve that issue. After solving the issue, the user increases the given rating from 2 stars to 5 stars out of 5 stars.

3 Data Collection

In this section, we describe our approach for collecting the user-developer dialogues from the Google Play Store. Figure 2 gives an overview of our approach.

3.1 Selecting Apps

We select the apps to study based on the following criteria:

1. **App popularity**: We focus on popular apps since popular apps contain more reviews than unpopular apps (Harman et al. 2012), which should facilitate enough data for studying user-developer dialogues.
2. **App maturity**: We focus on mature apps to ensure that the apps have had enough time to gather reviews and responses.

We select the top popular 12,000 free-to-download apps in 2013 according to App Annie (App Annie 2013) as these apps were top apps one year prior to our study. This decision was made to ensure that the studied apps had enough reviews and that their development teams were concerned about their apps (i.e., not abandoned apps). In our study, we focus on free-to-download apps to avoid the influence of the pricing of the app. The price of a paid app is very likely to be a confounding factor, as the app price may affect the expectation of a user.

After verification, we find that 8,218 apps out of these 12,000 free-to-download apps are still available in the Google Play Store. As an additional measure for selecting mature apps

Table 1 User-developer dialogue for the *AppLock* app

Dialogue	State	Rating
User: *"Applock stopped locking apps suddenly.... help"*	S_1	★★☆☆☆
Developer: *"Hi Vikhyath, thanks for using AppLock. To solve this problem, please open phone settings, then select security, then select apps with usage access, enable AppLock."*	S_2	–
User: *"Applock stopped locking apps suddenly.... help . edit: why is it asking for usage access all of a sudden ? Reply please."*	S_1	★★☆☆☆
Developer: *"Hi Vikhyath, since Lollipop system, there are some changes in Android system. In order to make AppLock work, please open phone settings, then select security, then select apps with usage access, enable AppLock."*	S_2	–
User: *"Thank you .thank you for response"*	S_1	★★★★★
Developer: *"Hi Vikhyath, thanks for your support all along. :) We will keep working to provide the best user experience. Have a nice day!"*	S_2	–

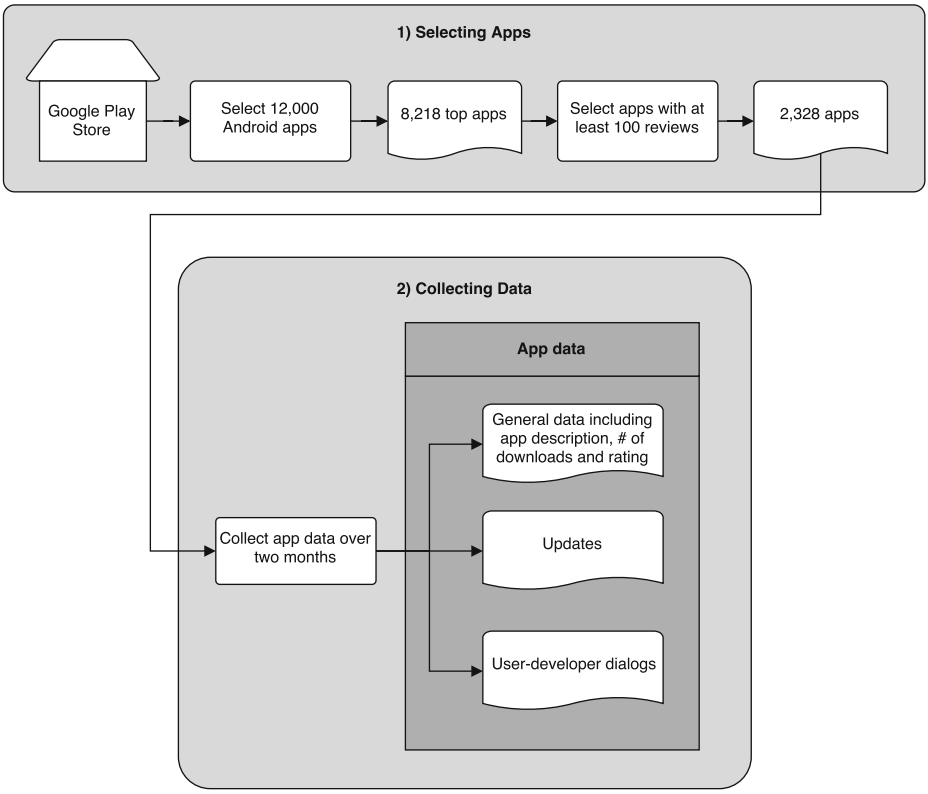

Fig. 2 An overview of our approach for collecting user-developer dialogues

only, we select only apps with at least 100 reviews. After removing all apps that have less than 100 reviews, we end up with 2,328 apps that match our selection criteria.

3.2 Collecting Data

We use a Google Play crawler (Akdeniz 2013) to collect data from the Google Play Store. For each studied app, we collect the following data:

1. **General data:** app title, app description, number of downloads, app rating and app developer.
2. **Updates:** date of each update.
3. **User-developer dialogues:** review title, review text, review time, reviewer name, rating, user device model, the time it took a developer to respond and the text in the developer response.

The crawler uses the Samsung S3 device model to connect to the Google Play Store, as the Samsung S3 is a popular mobile device (Top Android phones 2017). When the crawler connects to the store, the crawler collects the latest 500 reviews for each configured app.

During our study, we noticed that apps differ in the amount of new data that can be collected for an app per day. For example, the Facebook app receives thousands of new reviews per day while other apps receive only a small number of new reviews per day.

Table 2 Dataset description

Number of studied apps	2,328
Number of collected reviews (A)	4,474,023
Number of collected changes in reviews	355,600
Number of collected responses (B)	126,686
Percentage of reviews with responses $\left(100 * \frac{B}{A}\right)$	2.8%

To avoid flooding the Google Play Store with requests, we adapted the crawler to automatically adjust the number of times that it checks for new data to match the rate with which new review content is being posted for each studied app.

Every time the crawler connects to the Google Play Store, it checks whether there is new or updated data available for each studied app. The crawler collects the new or the updated data and appends that data to a database in which data for all apps is stored. As a result, we get a chronological overview of newly-posted reviews, ratings and responses and changes to those reviews, ratings and responses.

We run our crawler from September 22^{nd} 2015 to December 3^{rd} 2015. During that period, we collect almost 4.5 million reviews for 2,328 apps and over 355,000 changes that are made to those reviews. More than 126 thousand collected reviews have received a response from the app developer. Table 2 describes our dataset.

Table 3 shows the mean and five-number summary of the number of reviews, the number of reviews with responses and the percentage of responses for each studied app.

4 Preliminary Study

Ideally, responding to a review results in either a rating increase or an updated review that contains more information about the issue that was raised by the user. In this section, we describe the results of our preliminary study to demonstrate the chances of a user updating the review or rating after a developer responded to the review.

A user-developer dialogue was triggered by 2.8% of the user reviews. As shown in Table 2, developers responded to 2.8% of the user reviews during the studied period. Figure 3 shows the distribution of the percentage of reviews that have triggered a dialogue between the user and developer, i.e., reviews that have received a developer response, for each studied app. For clarity we display only the data for the 794 apps for which the developer responded at least once to a user review. As shown by Fig. 3, there is a broad variation in the percentage of reviews that receive a response for each studied app. The broad variation suggests that developers of different apps assign a different value to the opportunity of engaging in a dialogue with the user through the Google Play Store.

The chances of a user updating their rating after receiving a response are six times as high compared to users who did not receive a response. Table 4 shows the percentage

Table 3 Mean and five-number summary of collected data for every studied app

	Mean	Min.	1st Qu.	Median	3rd Qu.	Max.
Number of reviews per app (A_{app})	1,922	100	177	357	1,026	184,693
Number of reviews with responses per app (B_{app})	53	0	0	0	8	6,055
Percentage of reviews with responses per app $\left(100 * \frac{B_{app}}{A_{app}}\right)$	4.9%	0.0%	0.0%	0.0%	1.4%	98.6%

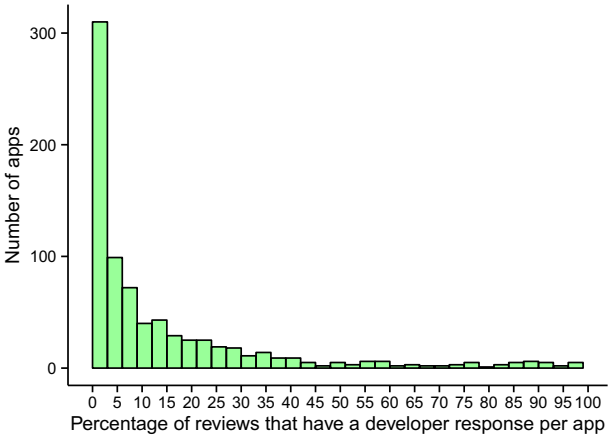

Fig. 3 The percentage of reviews to which a developer responded for each studied app. The figure displays only the 794 apps that have at least one review to which a developer responded

of reviews and ratings that change with and without receiving a response. Of the reviews that did not receive a response, only 0.7% increased their rating, while 4.4% of the reviews that received a response increased their rating. The percentage of decreased ratings does not change (0.7%), indicating that a response is more likely to have a positive effect on the rating. In addition, we studied the number of times a given rating both increased and decreased over time (0.0% of reviews without a response and 0.1% of reviews with a response). These rare cases are caused by a user changing the rating several times, usually over a longer period of using the app. For example, Table 5 shows an example dialogue in which the user increased and decreased the rating. As illustrated in Section 2, for the represented example in Table 5, a developer should receive four email notifications (i.e., an email each time the user changes in the review). In some cases, users may update their reviews several times after receiving developer responses. For example, a user may follow up to a developer response by writing few details about an issue and then the user modifies the posted review to add more details. As illustrated in Section 2, the Google Play Store will not notify the developer with the new modification. Thus, the developer may lose potential information that may help in solving the reported issue.

> *There lies great potential in responding to reviews. The chances of seeing an increase in the given rating are six times as high after responding to a review.*

Now that the potential of responding to reviews is clear, we study the phenomenon of the user-developer dialogue in the Google Play store in more depth in the remainder of the paper.

5 Study I: A Study of the Characteristics of User-Developer Dialogues

5.1 Motivation

Most prior studies of the app review mechanism in mobile app stores focus on extracting developer-relevant information from *reviews*. However, reviews form only part of the app

Table 4 The percentage of reviews and ratings that change with and without receiving a response

Reviews without a response:	Total number	% of total reviews without a response
Collected reviews	4,350,541	100%
Changed review contents	100,142	2.3%
Increased ratings	28,903	0.7%
Decreased ratings	31,344	0.7%
Both increased and decreased ratings	4,469	0.0%
Reviews with a response:	Total number	% of total reviews with a response
Collected reviews	123,556	100%
Changed review contents	9,246	7.5%
Increased ratings	5,457	4.4%
Decreased ratings	925	0.7%
Both increased and decreased ratings	81	0.1%

review mechanism. As explained in the previous sections, the possibility of *responding* to such reviews leads to new opportunities for developers to increase the rating of their app and elicit more information about issues that are raised by their users. Prior work (McIlroy et al. 2015) has only touched the surface of the emerging phenomenon of developers responding to reviews. In this section, we examine the characteristics of this phenomenon in more detail.

5.2 Approach

We studied the length and speed of the dialogue between a user and developer, and the effect of a developer responding to a review.

To calculate the length of the dialogue, we counted the number of iterations in the user-developer dialogue during the studied period for each review. We counted a transition from state S_1 to S_2 in the state diagram in Fig. 1 as a single iteration. Hence, the dialogue in Table 1 consists of 3 iterations.

To calculate the speed of the dialogue, we measured the developer response time as the time difference (in hours) between the time when a user posts a review and the time when a developer responds to the user review.

To quantify the effect of a developer responding to a review, we studied the difference in the review rating before and after receiving a developer response. We used the Wilcoxon signed-rank test (Gehan 1965; Wilcoxon 2017), a paired statistical test that validates if two samples have the same distribution. A p-value of less than 0.05 means that the difference between the distributions of the two samples is statistically significant and that there is a change in the review rating after a developer response.

In addition, we calculated Cliff's delta (d) (Long et al. 2003) effect size to quantify the difference in the distributions of the metrics. We used the following thresholds for interpreting d, as provided by Romano et al. (2006):

$$\text{Effect size} = \begin{cases} negligible(N), & \text{if } |d| \leq 0.147. \\ small(S), & \text{if } 0.147 < |d| \leq 0.33. \\ medium(M), & \text{if } 0.33 < |d| \leq 0.474. \\ large(L), & \text{if } 0.474 < |d| \leq 1. \end{cases}$$

Table 5 An excerpt from a single user-developer dialogue for the *Piano+* app

Dialogue	State	Rating
User: *"I can't play a song from the song book :("*	S_1	★★★★★
Developer: *"Hi Brandon, we feel so sorry for any inconvenience you experienced. Please leave us a message at support@rubycell.com to let us know more clearly about your problem. If possible, show us a screenshot of the issue so that we can offer you some help. We truly look forward to your response. Many thanks. "*	S_2	–
User: *"I can't play a song from the song book :(On Samsung galaxy S4"*	S_1	★★★★★
Developer: *"Hi Brandon, we feel so sorry for any inconvenience you experienced. Please contact us at support@rubycell.com and have a detailed discussion. We faithfully promise to help you address this problem. Hope for your response soon! "*	S_2	–
User: *"I can't play a song from the song book :(On Samsung galaxy S4... Sorry about this but I love this app and not being able to play it sucks."*	S_1	★★★★★
Developer: *"Hi again, as we said before, we really want to receive your message at support@rubycell.com and have a detailed discussion about this problem with you. Please give us a hand to get it fixed. Many thanks!"*	S_2	–
User: *"Sorry the problem was my phone I fixed it The best piano app on here by a long shot Download this app if you haven't already you'll enjoy it!!"*	S_1	★★★★★
Developer: *"Hi Brendon, we are very glad to know that your problem was handled and hope you will find more good time with Piano+. Best wishes. "*	S_2	–
User: *"I think this version was Ok, but there are many things I don't like, like the absence of the play all button (instead the play button is play all) sometimes I just want to listen/ play to the 1st part of the song but with this update you need to listen to both, (also the play/play all button is glitchy, letting the game play the song for you is glitchy with that), also some parts of songs (for ex. MIDIs) won't display the not for that part but instead the other part will play it, AlsoNotAsPolished"*	S_1	★★★★★
Developer: *"Hi there, in new version, though "Play all" is not available we sure that the current "Play" button has same function. It is the matter of change in name. If you still have a problem with this button, do not hesitate to send us your email. We promise to provide you with the most suitable answer. Many thanks."*	S_2	–

The dialogue emerged as the user and developer updated their review and response. We omitted the rest of the dialogue as the review and response updates became repetitive

We conducted several manual analyses to better understand our findings. For example, we manually studied why users decrease their rating after a developer responds to their review. In all manual analyses, we went through the following process:

Step 1 We used an iterative process that is similar to the coding method (Seaman 1999) to study the following cases:

– To understand why user-developer dialogues last for just one iteration, we read the user-developer dialogues which lasted for only one iteration and we identified why such dialogues did not continue.

– To understand why the percentage of reviews with responses is high for some apps, we read the user-developer dialogues for the five apps whose developers responded to at least 95% of the reviews and we identified why such developers responded to almost all the reviews for their apps.
– To understand why users decrease the rating after a developer response, we read the user-developer dialogues in which the rating is decreased and we identified the most likely explanation for the decrease.
– To understand why users increase the rating after a developer response, we read the user-developer dialogues in which the rating is increased and we identified the most likely explanation for the increase.

To eliminate human bias, the first and the second author (as two *coders*) conducted the manual analysis. Both coders independently read the user-developer dialogues and identified the explanations for the studied cases. When there was no exact explanation specified in the dialogue, the explanation was identified based on the experience and intuition of the coders.

Step 2 Both coders compared their results. To resolve conflicts, the coders discussed the differences to come to a consensus on the most likely explanation.

5.3 Results

The vast majority of user-developer dialogues last for just one iteration. Table 6 shows the distribution of the number of iterations within a user-developer dialogue. During our studied period, we found that user-developer dialogues can be as elaborate as up to ten iterations. However, 97.5% of the dialogues end after one iteration. After manual inspection of a statistically representative random sample (with a confidence level of 95% and a confidence interval of 10%) of 96 user-developer dialogues that ended after one iteration, we find the following explanations for the large number of short dialogues:

1. Users do not always require a response from the developer (67% of the sample). For example, the developer responds simply to thank the user for downloading the app or posting a review.

Table 6 The length and speed of the user-developer dialogues

Number of reviews	Number of review-response iterations	Median response time (hours)	Median review update time (days)	Number of apps
4,350,541	0	NA	NA	2,328
123,556	1	14.7	4.1	794
2,774	2	15.9	5.4	308
255	3	12.9	4.8	78
61	4	10	4.1	35
18	5	14	2.0	15
11	6	6.1	4.6	9
5	7	3.1	3.6	5
4	8	0.4	2.0	4
1	9	0.5	24.2	1
1	10	0.2	NA	1

2. Users are requested by the developer to continue the dialogue through another channel, such as email (29% of the sample).
3. Users do not provide the additional information that is requested by the developer response (10% of the sample).

Table 5 shows an excerpt of a longer dialogue between a user and app developer that we found in our study. In this dialogue, the review-response mechanism is used in two ways. First, the user is asked to provide more details through another channel (email). Second, the user is explained that the 'play all' functionality is now provided by the 'play' button. In addition, the dialogue demonstrates how a developer response can impact the user rating (i.e., a rating increase and decrease occur – the given ratings vary from three to five stars!).

Developers tend to respond faster as the number of iterations in the user-developer dialogue increases. As shown in Table 6, the median value for the developer response time decreases as the number of iterations in the user-developer dialogue increases. We manually analyzed the user-developer dialogues that have more than five iterations. We observed that users and developers leverage longer dialogues as a user support mechanism. During the user-developer dialogue, users either continue the discussion (e.g., follow up on the reported issue) or start a new discussion (e.g., raise a new complaint). As explained in Section 2, the Google Play Store notifies app developers when users follow up to a response. Table 6 suggests that developers give priority to responding to such notifications.

An interesting observation in Table 6 is that the median review update time is much longer (i.e., several days) than the median response time (i.e., several hours). Upon inspection, the longer median review update time appears to be caused by users posting new issues that they encountered after using the app for some time in the updated reviews. Next generation reviewing systems should somehow consider the changes in users' long term impressions while calculating the app rating. Moreover, store owners should notify app developers about such changes so developers can track the changes in users' impressions about their app.

When more than 95% of the reviews of an app have a response, the responses are mostly repetitive. As shown in Table 3, the maximum percentage of reviews with responses for an app is 98.6%. This high percentage indicates that the developer of such apps responded to almost all reviews. To understand better why these the percentage of reviews with responses is high for some apps, we studied the five apps whose developers responded to at least 95% of the reviews. Together, these five apps responded 4,301 times to 4,431 reviews.

The first and second author of this paper manually analyzed a statistically representative random sample of 353 out of the 4,301 responses with a confidence level of 95% and a confidence interval of 5%. We found that in 275 out of 353 (78%) cases in the sample, the response is generic in nature and based on a template. 55% of the responses in the sample ask the user to contact the company email in case they have issues with the app. For example, the "PrankDial - #1 Prank Call" app uses the following template response: *"Please send us a mail to support@prankdial.com if you are having any issues with the app. Thank you."* In 61% of the sample, the developer simply expresses appreciation for using the app using a template response. For example, the "Podcast Player - Free" app uses the template response: *"Thanks for the 5 stars, *user*!"*, in which **user** is replaced with the name of the user posting the review.

The value of the type of responses above for a user is questionable. On the one hand, a response shows that the developer is appreciative of the user. On the other hand, such unremarkable responses lead to more clutter in the Google Play Store. Google has set a

Table 7 Rating change after a developer response

Before response	After response				
	★☆☆☆☆	★★☆☆☆	★★★☆☆	★★★★☆	★★★★★
★☆☆☆☆	**48.8%**	4.6%	7.7%	11.5%	27.5%
★★☆☆☆	15.3%	**31.9%**	10.4%	14.4%	28.0%
★★★☆☆	7.1%	7.8%	**33.1%**	17.7%	34.3%
★★★★☆	3.4%	2.1%	7.2%	**43.2%**	44.1%
★★★★★	4.5%	3.3%	4.5%	3.9%	**83.7%**

The diagonal values (i.e., bold values) mean that there is no change in the user rating value. The values above the diagonal mean that the rating value is increased and values below the diagonal mean that the rating value is decreased

quota of 500 responses per day per app in the Google Play Developer API (Google 2017), which suggests that Google is trying to prevent developers from posting a large number of automated responses to their reviews.

To study the value of template-based responses, we compared the percentage of rating increases and decreases after template-based and non-template-based responses. The percentage of rating decreases was similar (approximately 8.0%) across template-based and non-template-based responses. Surprisingly, the percentage of ratings that are increased after a non-template-based response is lower (42.6%) than after a template-based response (48.2%). However, after manual inspection, we observed that the template-based responses that resulted in an increased rating were responses that contained useful information for the user. For example, the response contained an email address that could be used to contact the developers, or the developers responded using a template-based solution for an issue that was raised in the review. Hence, these observations suggest that providing useful information in the response improves the chances of a rating increase, even if that information is template-based.

A developer response can have a positive impact on the given rating. We tracked the changes in the review rating during the user-developer dialogue. We observed that in 11,813 out of the 126,686 iterations (9.3%),[2] users follow-up on the developer response. The Wilcoxon signed-rank test indicates that there is a significant change in the review rating before and after a developer response, with a small effect size.

Table 7 shows the percentage of the change in the user review rating after a developer responds to a review. The bold values indicate that the rating does not change. As illustrated by Table 7, users tend to increase the review rating (in particular, to 5-stars) if they change the review rating after a developer response. We calculated that in 5,504 reviews, the rating increases after a developer response. We found 942 reviews in which the user decreases the rating after a developer response. The first and the second author of this paper manually examined a statistically representative random sample of 87 of such reviews (a confidence level of 95% and a confidence interval of 10%). We identified the following reasons for decreasing a rating:

[2]Note that this number is different from the numbers in Table 4, since in Table 4 we count the number of reviews/ratings that changed at least once during the dialogue.

1. The user raises a new issue after a developer response, and the decreased rating is based on the severity of the new issue (31% of the sample).
2. The user has an issue and the developer responds that the issue cannot be fixed or the issue will be solved in one of the next releases (13% of the sample).
3. The user has an issue and the developer recommends a solution for the raised issue. The user complains that the provided solution does not work (15% of the sample).
4. The user has an issue and the developer responds to that issue by asking for more details. The user decreases the rating as follows:
   (a) The user decreases the rating without providing the requested details to the developer (15% of the sample). We believe that the user was expecting the issue to be solved directly without asking for more details.
   (b) The user provides the requested details and expresses their disappointment that the issue is not solved yet (23% of the sample).

We found two dialogues in which a user decreased the rating without explanation after having a positive experience with the app. A possible explanation is that those users misinterpreted the scale of the rating system (i.e., they mistakenly assumed that a 1-star rating is better than a 5-star rating).

App developers can solve 34% of the reported issues without deploying an app update. We found that in 5,504 reviews, the rating increased after a developer response. To understand why users increased the rating after a developer response, we manually investigated a statistically representative random sample of 94 of such reviews (a confidence level of 95% and a confidence interval of 10%). Table 8 shows the identified reasons for a rating increase. As illustrated in Table 8, developers could guide users to solve 34% of the reported issues without having to deploy an app update. This percentage shows that in one-third of the studied dialogues, it was relatively easy for the developer to achieve a rating increase.

> *The majority of user-developer dialogues last for one iteration. However, if the user decides to change the given rating during the dialogue, the rating is increased in most cases. In one-third of the studied dialogues, a rating increase was achieved by simply explaining to the user how to resolve the reported issue.*

Table 8 The identified reasons for rating increase (ranked by the percentage of responses)

Percentage of responses	Reason for rating increase
34%	The developer guides the user to solve the reported issue without having to deploy an app update
24%	The developer deploys an app update to address the reported issue
13%	The details of the solution are communicated outside the store
7%	The developer asks a satisfied user for a rating increase
6%	The user appreciates that the developer cared by responding, even though the issue cannot be fixed
5%	The developer indicates that the reported issue is temporary and the rating is increased when the issue is addressed
5%	The user raises a new issue
4%	No reason could be identified

6 Study II: A Quantitative Study of the Likelihood of a Developer Responding

6.1 Motivation

In the previous section, we illustrated that responding to user reviews can have a positive effect on the user rating. Reading and responding to user reviews is a time-consuming process, especially if an app has a large number of user reviews. Providing a better understanding of the metrics that drive a developer to respond, can be used by app developers and store owners to facilitate a better and more efficient user-developer dialogue. For example, the identified metrics can be used by store owners to highlight reviews that are more likely to require a response, thereby helping developers to prioritize the reviews to read and respond to.

6.2 Approach

In this section, we describe our approach for analyzing the relationship between the studied metrics and the likelihood of a developer responding. Figures 4 and 5 depict the steps of our approach. As shown in Figs. 4 and 5, our approach contains three steps. First, we collected metrics that may have a relationship with the likelihood of a developer responding. Second, we constructed statistical models (i.e., mixed effect models) to capture the relationship between the studied metrics and the likelihood of a developer responding. Finally, we analyzed the constructed models to understand which metrics are related to the likelihood of a developer responding. In the next sections, we detail every step.

6.2.1 Data Selection and Metrics Collection

In order to create a model to understand which metrics are related to a developer responding to a user review, we need to study apps that have sufficient user reviews and developer

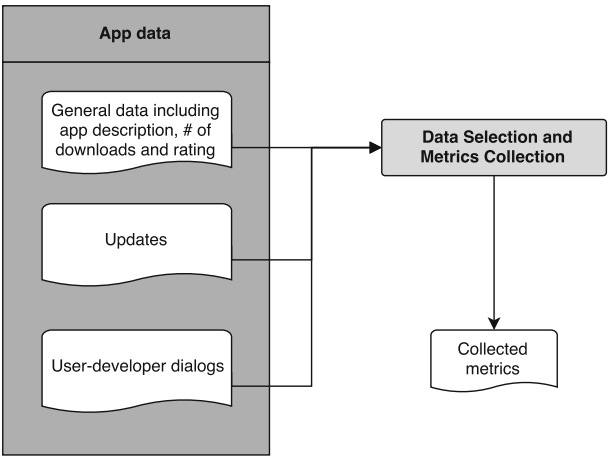

Fig. 4 An overview of our data selection and metrics collection step

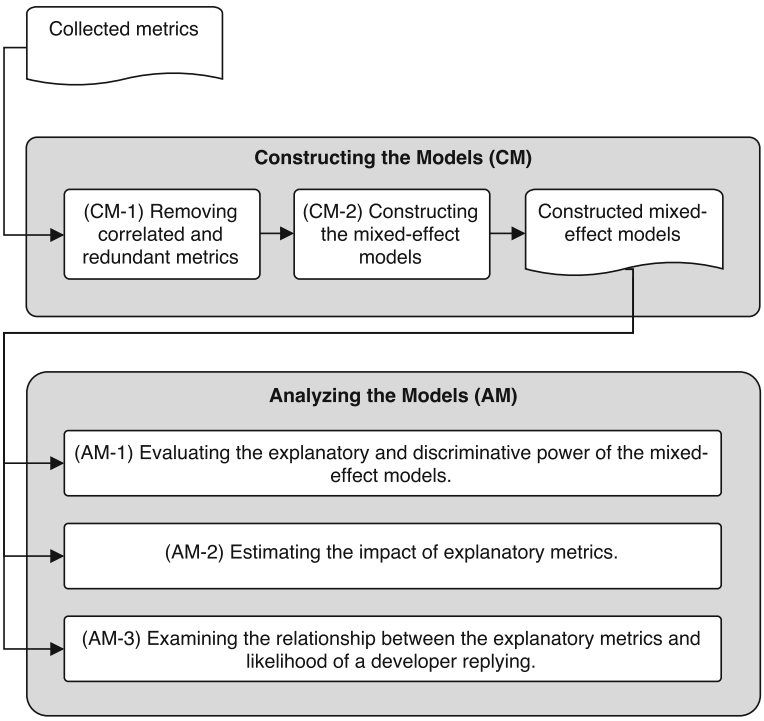

Fig. 5 An overview of our approach for studying the relationship between the studied metrics and the likelihood of a developer responding to a user review

responses. Hence, we removed apps in which less than 5%[3] of the reviews have a response. After removing such apps, there remained 415 apps with 556,314 reviews and 103,612 responses in total. Consequently, the results that are presented in this section are valid for developers who respond to at least 5% of the posted reviews.

Table 9 presents the two levels of metrics that we collected in our study. app-level metrics are at the app level (e.g., the app category) while review-level metrics are at the review level (e.g., the review rating).

In the app-level metrics, we collected the app ID, app rating and app category. In the review-level metrics, we collected the review title length, review text length, days since last release, review rating and positive/negative sentiment.

We used the SentiStrength tool (SentiStrength 2017) to measure the review sentiment metrics. SentiStrength is designed for analyzing sentiments in a short text such as online reviews (Thelwall et al. 2010). Several studies have shown that the SentiStrength tool is applicable to the software engineering domain (e.g., Tourani et al. 2014 and Guzman et al. 2014). In particular, the SentiStrength tool was previously used to analyze user reviews of mobile apps (e.g., Guzman and Maalej 2014, Maalej and Nabil 2015 and McIlroy et al. 2016).

[3]We experimented with several thresholds, as explained in Section 9, which resulted in very similar models.

Table 9 Collected metrics for each studied review

Metric category	Metric	Values	Description (*D*) - Rationale (*R*)
App-level metrics	App ID	Categorical	***D:*** Identifier for the studied app ***R:*** Including the app identifier can be used to understand how each app behaves when it comes to user-developer interactions
	App rating	Categorical	***D:*** The app rating (e.g., 3.4) ***R:*** Apps with certain popularity (i.e., app rating value) may provide better user support than apps with different rating
	App category	Categorical	***D:*** The category of the app (e.g., games or tools) ***R:*** Apps in certain categories may provide better user support than apps in other categories
Review-level metrics	Review title length	Numerical	***D:*** The number of characters in the review title ***R:*** Developers may tend to respond to issues with well-described titles
	Review text length	Numerical	***D:*** The number of characters in the review text ***R:*** Developers may tend to respond to issues with well-described text
	Days since last release	Numerical	***D:*** The number of days between the review posting date and the last released update of the app ***R:*** Users tend to post reviews in the short period after a new update (Pagano and Maalej 2013). Hence, developers may tend to respond to reviews that are posted shortly after an update
	Review rating	Numerical	***D:*** The star rating of a user review ***R:*** The review rating reflects the review content (e.g., negative reviews have a low rating) (Khalid et al. 2015). Hence, developers may tend to respond to reviews with a low rating to increase the overall rating of their app
	Positive/Negative sentiment	Numerical	***D:*** Two metrics that indicate the positive and negative sentiment values for the review content. The positive sentiment metric has values that range from 1 to 5. The negative sentiment metric has values that range from -1 to -5 ***R:*** The review sentiment can be used to indicate the purpose of the user review (e.g., user complaints have a negative sentiment) (Maalej and Nabil 2015). Developers may tend to respond to reviews based on the review sentiment

SentiStrength provides for each analyzed sentence two values: (1) a positive sentiment value that ranges from 1 to 5 and (2) a negative sentiment value that ranges from − 1 to − 5. Tourani et al. (2014) studied the sentiment analysis in developer emails and found that the maximum sentiment value per line can be used to represent the sentiment value for the entire email text. Hence, we followed the same approach by calculating the sentiment value per sentence and consider the maximum sentiment values across all sentences as the sentiment value for the entire review text.

For example, a user for the *"Daily Yoga - Get Fit & Relaxed"* app posts a review ***"Brilliant App Very easy to use and extremely good fun!*** *Would recommend it to all ages. I would down load it twice if I could!"*. The highlighted sentence has the highest positive sentiment value (i.e., a sentiment value of 5) while the other sentences have lower sentiment values (i.e., sentiment values of 1 and 2), so we considered this review to have a positive sentiment value of 5.

6.2.2 Constructing the Models (CM)

In order to examine the metrics that have a relationship with the likelihood of a developer responding, we performed the following steps to construct our models. Note that we are not trying to predict whether a developer will respond to a review.

(CM-1) Removing correlated and redundant metrics. We removed highly correlated review-level metrics before constructing our models to remove the risk that those correlated metrics interfere with our interpretation of the models (Tantithamthavorn et al. 2016). We measured the correlation between the review-level metrics using the Spearman rank correlation test (cut-off value for ρ is 0.7). We then used a metric clustering approach to construct a hierarchical overview of the inter-metric correlation using the implementation of the `varclus` function that is provided by the `Hmisc R` package (Harrell 2017). To detect multicollinearity, we performed a redundant analysis using the implementation of the `redun` function that is provided by the `Hmisc R` package (Harrell 2017). After doing the correlation and redundancy analysis, we found that no metrics were correlated or redundant. Hence, no metrics were removed from our dataset.

(CM-2) Constructing the mixed-effect models. As explained in Section 6.2.1, the collected metrics represent two hierarchical levels (i.e., the app level and the review level). Constructing a simple regression model that ignores the hierarchical nature of the collected metrics may produce inaccurate results (Anderson et al. 2001). As the user reviews were collected from different mobile apps, the likelihood of a developer responding may vary depending on the mobile app. For example, developers of one particular app may have the policy to never respond to reviews or the metrics that are related to the response likelihood might differ between groups of apps (something that we observe later on in our analysis).

In a traditional linear regression model, all subjects have the same relation with the outcome y:

$$y = \alpha x + \beta + \epsilon \tag{1}$$

with the slope α and the intercept β being fixed and ϵ being the standard error. Because all subjects have the same relation with y, such a model cannot express differences for subjects at different hierarchical levels.

To study the user review characteristics while considering mobile application context, we used a mixed-effect model (Snijders and Bosker 2012) instead. A mixed-effect model

includes two types of metrics, i.e., explanatory metrics and context metrics. Explanatory metrics (review-level metrics) are used to explain the data at the user review level, while context metrics refer to the app level (i.e., app category and app ID). A mixed-effect model expresses the relationship between the outcome (i.e., a developer responding) and the review-level metrics (e.g., review rating), while taking into consideration the different app-level metrics (e.g., the app ID).

There are two ways to construct a mixed-effect model: (1) a random intercept model and (2) a random slope and intercept model (Snijders 2005). A random intercept model contains different intercepts (for the app-level metrics) and fixed slopes (for the review-level metrics). Such a model assumes that each app has its own baseline likelihood to respond, while the final likelihood to respond is related to a general model that is the same across all apps (i.e., a fixed slope for the review-level metrics).

A random slope and intercept model contains different intercepts (for the app-level metrics) and different slopes (for the review-level metrics). Such a model assumes that the likelihood to respond varies per app and per review-level metric for each app. In our study, we constructed both types of mixed-effect models using the `glmer` function of the `lme4` `R` package (Bates et al. 2017).

6.2.3 Analyzing the Models (AM)

After building the mixed-effect models, we evaluated the explanatory and discriminative power of the models and we estimated the impact of the review-level metrics. In addition, we examined the relationship of each metric to the likelihood of a developer responding.

(AM-1) Evaluating the explanatory and discriminative power of the mixed-effect models. Explanatory power measures the goodness-of-fit of a model (Eisenhauer 2009). To calculate the explanatory power $expl_{\text{mixed}}$ of a mixed-effect model, we calculated:

$$expl_{\text{mixed}} = \frac{D_{\text{null}} - D_{\text{mixed}}}{D_{\text{null}}} \tag{2}$$

where D_{null} is the deviance (which is a goodness-of-fit measure) of the null model and D_{mixed} is the deviance of the mixed-effect model. The null model is the simplest possible mixed-effect model that contains only app-level metrics. The explanatory power explains the proportion of the data that can be explained by the review-level metrics, with 0 being the lowest proportion, and 1 being the highest proportion.

The discriminative power of a model measures the model's ability to discriminate between whether a developer will or will not respond to a review. We calculated the discriminative power using the Area Under the receiver operator characteristic Curve (AUC). The Receiver Operator Characteristic (ROC) curve plots the true positive rate against the false positive rate for different threshold settings. AUC values range between 0 (worst performance), 0.5 (performance of random guessing), and 1 (best performance) (Hanley and McNeil 1982; AUC 2017).

(AM-2) Estimating the impact of review-level metrics. To understand the most impactful metrics in our mixed-effect models, we used Wald statistics to estimate the relative contribution (χ^2) and the statistical significance (p-value) of each review-level metric in the model. The larger the χ^2 value, the larger the impact that a particular review-level metric has on the performance of the model. We used the ANOVA implementation that is

provided by the `Anova` function of the `car` R package (Fox and Weisberg 2017) to calculate the Wald χ^2 and the p-value of each review-level metric in the model.

(AM-3) Examining the relationship between the review-level metrics and likelihood of a developer responding. To study the relationship between the review-level metrics and the likelihood of a developer responding, we plotted the likelihood that developers will respond corresponding to the changes in each review-level metric while holding the other review-level metrics at their median values.

6.3 Results

Table 10 shows the statistics for the two mixed-effect models.

Table 10 Summary of the model analysis of the mixed-effect models that we used to understand the relationship between review metrics and the likelihood of a developer responding to a review

Model statistics	Random intercept model	Random intercept & random slope model	
Explanatory power	0.28	0.34	
Discriminative power (AUC)	0.92	0.93	
Random intercept statistics			
App-level metrics	Variance	Variance	
App ID	4.4	9.4	
App rating	1.8	2.2	
App category	0.1	0.4	
Random slope statistics			
Review-level metrics	Variance	Variance	
Review rating	–[a]	1.6	
Review-level metrics statistics for the random intercept model			
Review-level metrics	Coeff ± std. error	χ^2	
Review rating	-1.018 ± 0.00	62,533	***
Review text length	0.003 ± 0.00	2,317	***
Days difference	-0.007 ± 0.00	504	***
Review title length	0.005 ± 0.00	192	***
Negative sentiment	0.040 ± 0.01	39	***
Positive sentiment	0.010 ± 0.01	4	–
Review-level metrics statistics for the random intercept & random slope model			
Review-level metrics	Coeff ± std. error	χ^2	
Review text length	0.003 ± 0.00	2,278	***
Days difference	-0.006 ± 0.00	457	***
Review title length	0.006 ± 0.00	235	***
Negative sentiment	0.024 ± 0.01	13	***
Positive sentiment	0.011 ± 0.01	3	–

Statistical significance of the χ^2 test: '***' < 0.001, '–' ≤ 1

[a]There is no variance value for the review rating metric for the 'Random Intercept' model because the random intercept model has a fixed slope for the review-level metrics

Different mobile apps have a different likelihood of a developer responding. Table 10 shows that the random intercept of the app ID metric varies across different apps, which indicates that apps have a different likelihood of a developer responding. As shown in Table 10, the random intercept of the app rating metric varies across different apps, which indicates that apps with different ratings have a different likelihood of a developer responding. Table 10 shows that the app category metric has the lowest influence in the mixed-effect model.

Reviews with a low rating have the highest likelihood of receiving a response. Table 10 displays the ANOVA results of the random intercept model. The review rating metric accounts for the largest χ^2 value in the model, indicating that review rating contributes the most to our model.

Figure 6 shows the relationship between the likelihood of a developer responding and each review-level metric in the random intercept model. We omit the plots for the random intercept and random slope model because the plots are very similar to Fig. 6. As illustrated by Fig. 6, the likelihood that a developer will respond increases as the review rating gets lower. Hence, we conclude that developers are likely to respond to reviews with a low rating.

The relation that the review rating has with the likelihood of a developer responding is different across mobile apps. Table 10 shows that there is a variance in the review rating coefficient (for the random slope and random intercept model) which indicates that app developers differ in the way their likelihood of responding is related to the review rating.

Longer reviews have a higher likelihood of receiving a response. Figure 6 shows the likelihood of a developer responding for each review-level metric. The grey area in Fig. 6 represents the confidence interval of 95%. As shown in Fig. 6, the likelihood of a developer responding increases as the length of the review text increases. However, as most reviews are short, the confidence interval for the longer reviews is larger.

6.4 Identifying Response Patterns

Since we found that each mobile app exhibits different behavior (as the app-level metrics and the review rating slope change per app), we performed a deeper investigation to identify the common patterns for developer responses. Figure 7 shows an overview of our approach.

To group apps that share a behavioral pattern when it comes to responding to a user review, we constructed a logistic regression model for each app using the `glm` function that is provided by the `stats` `R` package (Documentation for package 'stats', 2017). The logistic regression model indicates the likelihood of a developer responding based on the values of the review-level metrics.

We found that the constructed models differ in the importance of each review-level metric and the relation of each review-level metric (i.e., the review-level metric is either positively or negatively related) to the likelihood that a developer responds. Hence, we digested the generated app model by extracting the following key features from each model:

1. **The percentage of importance for each review-level metric.** To calculate the percentage of importance for each review-level metric, first we measured the power of each review-level metric. We used Wald statistics to estimate the relative contribution (χ^2) for each review-level metric in the model. We then calculated the percentage of χ^2 for each review-level metric to the total χ^2 for all review-level metrics. For example, the percentage of importance for the review rating in the model described by Table 10 is 95.6%.

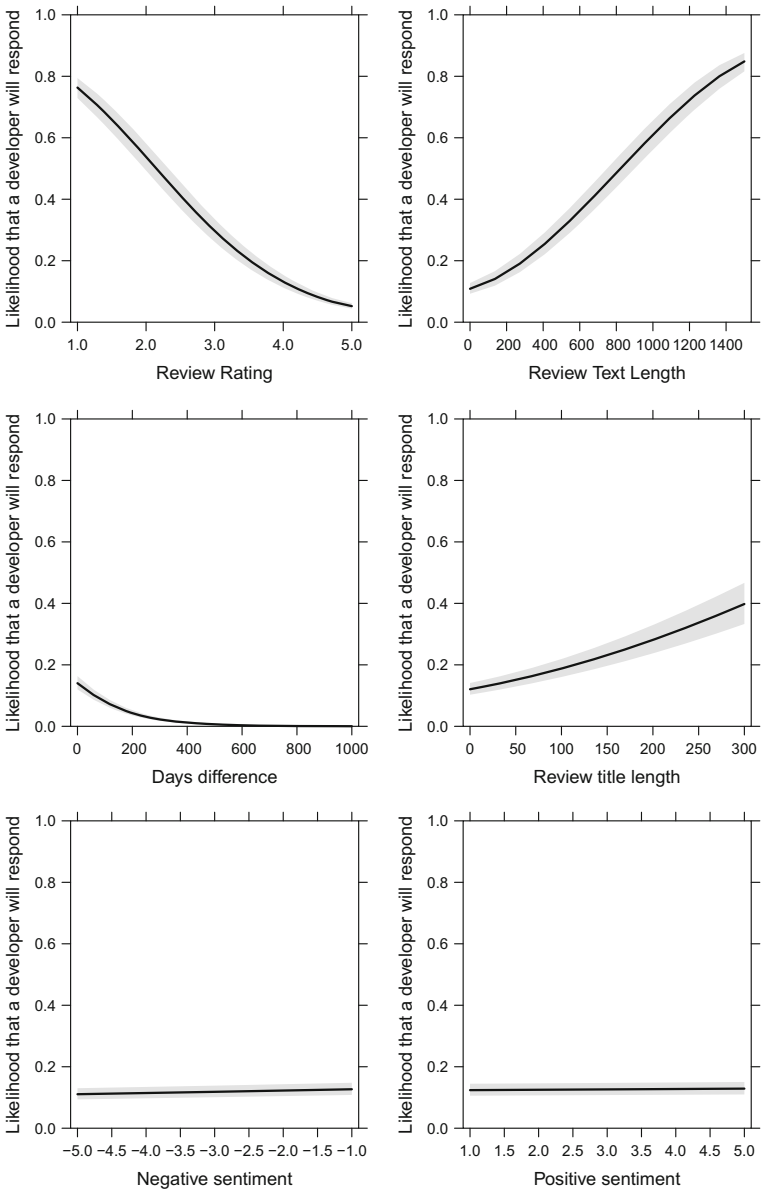

Fig. 6 The relationship between the review-level metrics and the likelihood that a developer will respond. The grey area represents the confidence interval. Note that these plots show the likelihood that a developer will respond for apps with responses to at least 5% of the reviews

2. **The sign of the slope for each review-level metric.** In order to estimate the direction of the relationship of review-level metrics for each model, we extracted the sign of the slope for each review-level metric.

Table 11 shows the extracted key features for the generated logistic regression model for the *"MapFactor GPS Navigation Maps"* app. We used the extracted key features from

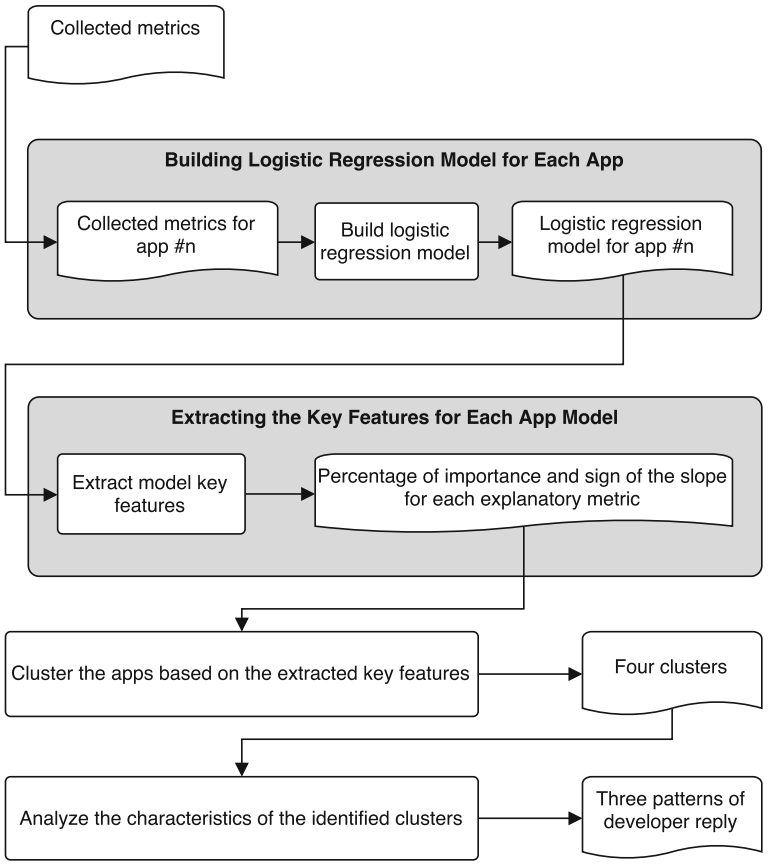

Fig. 7 An overview of our approach for identifying the developer response patterns

each model as input to a clustering technique. We used the clustering implementation of the `kmeans` function that is provided by the `stats` `R` package (Documentation for package 'stats', 2017). In order to identify the optimal number of clusters, we started with the minimal number of clusters (two clusters) and increased that number until the newly-found clusters are only sub-clusters of the previous run. Eventually, we identify four clusters of developer response patterns using our heuristic. The four clusters can be explained as follows:

1. **Developers who primarily respond to only negative reviews.** In the first cluster (231 apps), developers responded mainly to negative reviews (the mean value for the importance of the review rating metric is 78%).
2. **Developers who primarily respond to negative or longer reviews.** In the second cluster (95 apps), developers responded mostly to negative reviews (the mean value for the importance of the review rating metric is 40%). However, in addition to the negative reviews, developers in the second cluster appear to pay attention to the length of a review as well (the mean value for the importance of the review text length is 26%).
3. **Developers who primarily respond to reviews which are posted shortly after the latest release.** In the third cluster (50 apps), developers responded mostly to reviews

Table 11 The extracted key features for the generated logistic regression model for the *"MapFactor GPS Navigation Maps"* app

The review-level metric	χ^2	Slope	Percentage of importance	Sign of the slope
Review rating	55.32	− 5.80	79.2%	Negative
Negative sentiment	6.61	− 1.49	9.5%	Negative
Review title length	2.68	0.03	3.8%	Positive
Review text length	2.24	0.00	3.2%	Positive
Positive sentiment	1.99	− 0.46	2.8%	Negative
Days difference	0.98	0.02	1.4%	Positive

which are posted soon after the latest update (the mean value for the importance of the days since last release metric is 47%).

4. **Developers who primarily respond to reviews which are posted longer after the latest release.** In the fourth cluster (39 apps), developers have a higher likelihood of responding to reviews that are posted long after the deployment of the latest update (the mean value for the importance of the days since last release metric is 42%).

The explanation for the first and second cluster is quite obvious, as negative and longer reviews are the most likely to be candidates for a rating increase. The third and fourth cluster are less intuitive. We manually read a sample review to which a developer responded for each app in the third and fourth cluster. We noticed that for both clusters developers responded shortly after the review is posted. For the third cluster, we noticed that in most cases users complain about an issue in the latest update and developers try to solve the issue or announce that they are working on fixing the issue. For example, we noticed that in 18 apps (out of 50 apps belonging to this pattern) the median number of days between the update and a posted review is less than a week. Hence, the third cluster can be explained as developers focusing on issues that arise directly after an update.

The fourth cluster can be explained as developers who focus mostly on issues that are raised longer after an update. A possible explanation for the focus of these developers is that they focus on reviews that raise issues that need to be fixed. Hence, shortly after an update, the same obvious issues are reported by many users, yet these developers respond to only a few of them. Instead, they shift their focus to responding to issues in reviews that are raised at a later stage, which are less likely to be noticed by users, therefore taking a longer time to be reported in reviews.

7 Study III: A Qualitative Study of What Drives a Developer to Respond

7.1 Motivation

In the previous section, we gave a quantitative view of the metrics that are related to the likelihood of a developer responding to a review. These metrics can be used to identify reviews that are likely to require a response. However, these metrics do not explain the actual contents of a response, nor do they explain the rationale for a developer responding to a review. In this section, we study more qualitatively what drives developers to respond. Getting a better understanding of what drives developers to respond to a user review can be beneficial for developers, as it allows them to identify flaws in their user support process.

For example, it can help them to improve or extend the Frequently Asked Questions (FAQs) on their website. In addition, identifying the drivers for responding can assist the app store owners by revealing useful functionality for developers that can be added to their store.

7.2 Approach

In order to identify what drives developers to respond to a user review, we conducted a manual analysis of their responses. First, we ranked the apps by the number of responses by a developer. To avoid bias in our manual study, we examined a statistically representative random sample (with a 95% confidence level and a 5% confidence interval) of the responses for each of the 10 apps with the highest number of responses. Table 12 shows a summary of the selected ten apps together with the size of the selected sample. In total, we selected 3,431 responses.

For our manual analysis, we examined a statistically representative random sample of 347 responses from the selected 3,431 responses (with a 95% confidence level and a 5% confidence interval). The first author and the third author of the paper (the *coders*) went through the following process to identify the drivers that make developers respond to a user review:

Step 1 The coders independently followed a method similar to the open coding method (Khandkar 2017; Borgatti 2017) to identify the drivers that make developers response. The open coding method iteratively builds a list of identified drivers. For every studied developer response, the coder identifies the driver. If the driver is not in the list of identified drivers, the coder extends the list and revisits all developer responses using the new list of identified drivers. The open coding method terminates when there are no newer drivers to be identified and all developer responses are studied.

Table 12 Data summary of the top ten apps with the highest number of reviews with responses

App name	Number of collected reviews with responses	Number of downloads	Average rating	Category	Number of selected responses
Clean Master (Boost & AppLock)	6,128	500M–1,000M	4.66	Tools	362
Daily News, eBooks & Magazines	4,554	10M–50M	4.24	News & Magazines	354
PicsArt Photo Studio	4,481	100M–500M	4.44	Photography	354
AppLock	4,471	100M–500M	4.35	Tools	354
Solo Launcher Clean Smooth Diy	4,187	100M–500M	4.48	Personalization	352
Hungama Music - Songs & Videos	3,949	10M–50M	4.11	Music & Audio	350
The PCH App	2,452	1M–5M	4.46	Lifestyle	332
UC Browser - Fast Download	2,193	100M–500M	4.55	Communication	327
FrostWire - Torrent Downloader	2,026	10M–50M	3.93	Media & Video	323
WeChat	2,023	100M–500M	4.25	Communication	323

Using the open coding method, we can identify multiple drivers for a single developer response. For example, a user posted a review *"Good"* and the developer response is *"Dear Rama Thanks for your valuable feedback. Please revise your ratings to support us better since 3 star that you have rated is lower and 5 star is the highest."*. We tagged this response with the "Thank the user" and "Ask for endorsement" labels.

The output from this step is the list of drivers that the coders identified for each studied response.

Step 2 The first coder compared the drivers that were identified by both coders for all studied responses and marked the differences.

Step 3 The coders discussed the differences and came to a consensus about the final codes for each studied response. We found 19 developer responses that had differences in the identified drivers between the two coders. After discussion, all differences were easy to resolve.

7.3 Results

We identified seven drivers that make a developer response to a review. Table 13 shows the identified drivers for responding along with their description and examples of a developer response. Table 14 shows statistics about the identified drivers for responding. In particular, Table 14 shows for each driver the number of responses, the percentage of the total number of responses in the sample, the number of apps whose developers had that driver at least once for responding, the number of reviews that were changed after a developer response and the number of reviews that changed their rating after a developer response.

In 24% of the responses, app developers use the review mechanism to ask users to increase their given rating. In general, users tend to ignore these requests and do not update their rating. Table 14 shows that the "Ask for endorsement" label occurs in responses for nine out of ten studied apps, which indicates that this kind of request is common. Only in 3 out of 83 developer responses, users change their review after a developer asks for a rating increase and in only two cases the rating actually increases. The observation about developers asking for a rating increase is in line with our finding in Section 6.3, where we show that developers are more likely to respond to reviews with low ratings. These observations combined confirm (in accordance with literature (Martin 2017)) the importance of ratings to developers.

In 45% of the developer responses, developers ask for more details about the reported issues. Users tend to post reviews that contain a complaint but do not contain enough information for the developer to address the complaint. For example, developers ask for more details about the hardware or the OS version on which an issue occurred. Developers ask the user to provide the additional details through one of the following channels: (1) directly in the dialogue (i.e., by updating the review), (2) using the crash reporting mechanism in the app or (3) by email.

Moran et al. (2015) propose an approach for guiding users to report the steps that produce reported issues. Store owners and developers can include more detailed information about the reported issues by following Moran et al.'s approach.

In 25% of the responses, developers provide guidance to users to solve their issues. We notice that often, users encounter the same issue as others. During our manual analysis, we find eight cases of the same issue that is reported by multiple users to which the

Table 13 The identified drivers for responding

Driver for responding	Description (*D*) - Example (*E*)
Advertise other products	***D:*** The developer advertises one of his other products. ***E:*** *"Hi Michael we hope you enjoy the PCH App. Please feel free to try one of our other fun and exciting opportunities to win like PCH Lotto Blast and PCH Cash Slots. If you ever need any assistance please check out our app FAQ page within the app itself."*
Ask for endorsement	***D:*** The developer asks a user to market for the apps by increasing the review rating or by sharing the good experience with others. ***E:*** *"If you like AppLock please rate 5 stars to support us and share it thanks."*
Ask for more details	***D:*** The developer asks for additional details in order to solve the user complaint. ***E:*** *"Hi Pratik Could you please let us know if this is happening all the time or is it a recent occurrence? Does it happen with a particular song or all of them? Kindly share more details about this issue and we will provide you with optimum support."*
Justify the advertisements	***D:*** The developer explains the rationale for the existing advertisements in the app. ***E:*** *"Sorry for making you trouble with ads and we also completely understand your concern. We are always try to improve our app and we want to provide our user a good and free clean master. So we hope you could understand that we need this part for our income. Thank you!"*
Provide guidance	***D:*** The developers provides guidance and additional details to the users. ***E:*** *"Please open phone settings, then select security, then select apps with usage access, enable AppLock."*
Wait for updates	***D:*** The developer notifies the user that the reported issue will be solved or the requested feature will be added and asks the user to wait for upcoming updates. ***E:*** *"Your feedback is received and will be handled as soon as possible. Stay tuned with us for updates. Thanks for your support!"*
Thank the user	***D:*** The developer thanks the user for using the app or posting a review. ***E:*** *"Hi Shani Thank you for your fantastic review!!"*
Unspecified	***D:*** The developer response is generic or the developer response is not in the English language. ***E:*** *"Dear User We hope you enjoy using the app and becomes a satisfied users of the app."*

developer responses mostly the same. For each case, we manually extract keywords from the responses (e.g., 'clear cache') and search for responses containing those keywords for that app in the total set of reviews. We then manually check how many times the response occurs in the total set of reviews (not just our manually coded reviews). Table 15 shows the top five responses that occur the most often, together with a description of the reported problem and the given response.

Table 14 Statistics for the drivers for responding (ranked by the number of responses)

Driver	Number of responses	Percentage of responses[a]	Number of apps	Number of changed reviews	Number of changed rating
Thank the user	219	63%	10	8	5
Ask for more details	155	45%	9	12	6
Provide guidance	86	25%	10	7	3
Ask for endorsement	83	24%	9	3	2
Advertise other products	47	14%	3	1	0
Wait for updates	31	9%	8	2	1
Justify the advertisements	5	1%	3	0	0
Unspecified	3	1%	2	0	0

[a]Since a response can have several drivers, this column does not add up to 100%

As Table 15 shows, the repeated responses often indicate an issue that occurs for many users. Knowing about such issues can be beneficial for developers, as they can improve their app to address the issue. Another possibility is to address the issue in the Frequently Asked Questions (FAQs) section of the app to help users solve the issue without having to post a review (which often includes a low rating). For example, the "AppLock" issue that is addressed by 520 reviews can be summarized by the question *"How can I use the AppLock app to lock other apps?"* By studying the reviews and responses, developers can semi-automatically (e.g., using keywords) extract issues that should be addressed in the FAQ section.

> *Developers can benefit from the analysis of user reviews and developer responses by extracting repeated questions. These questions can help to identify issues that should either be fixed, or discussed in the Frequently Asked Questions (FAQs) section of the app.*

8 Implications

In this section, we discuss the implications of our work for researchers and app store owners.

8.1 Implications for Researchers

App reviews and app ratings are not immutable. Prior work on app reviews has always assumed that reviews and ratings are posted and thereafter not changed. However, our work shows that users may change their reviews and ratings over time. Even though the portion of changed reviews (3.7% in our studied period) and ratings (1.4%) is fairly low, researchers need to be aware of the potential impact of changes in reviews and ratings on their work. Future studies are necessary to better understand the implications of changing reviews and ratings on prior research findings.

Reviews only do not provide an in-depth view of user concerns. Recently, several approaches were proposed for extracting requirements from app reviews. For example, Martin et al.'s (2016) survey shows that existing research uses reviews as a tool for requirements

Table 15 The identified similar responses

App	Number of similar reviews	Description (*D*) - Response (*R*)
AppLock	520	***D:*** The user asks how to use the *"AppLock"* app to lock other apps. ***R:*** *"Please open phone settings security apps with usage access enable AppLock."*
PicsArt	271	***D:*** The user complains that the app is very slow. ***R:*** *"Hi Vincent this issue can sometimes be solved by clearing the cache. To do so go to your device's Settings - Apps - PicsArt and tap 'Clear data' and 'Clear cache'. If the issue persists please feel free to contact us at support@picsart.com and we will be happy to investigate the issue so you can use the app without problems. "*
AppLock	109	***D:*** The user complains that the app stops working. ***R:*** *"Please open AppLock to check whether the app is locked. Then try to enable advanced protection."*
WeChat	70	***D:*** The user complains that the app does not display or upload pictures. ***R:*** *"You can try following tips to troubleshoot your problems: 1.Download and install the latest version of WeChat. We suggest you migrate your chat history before uninstalling WeChat to prevent data loss. 2.Select any one of the chat pages and send '//switchsdcard'. If the above tips can't fix your issue contact us again."*
FrostWire	26	***D:*** The user complains that the app does not work after updating to Android Marshmallow (Android version 6). ***R:*** *"Have you just updated your operating system to Android 6? We are still working on a compatible release... lots to change. In the mean time you can go to Phone Settings and go to Apps find FrostWire then select Permissions and enable all of the permissions listed. Everything else should work just fine :)"*

engineering (e.g., by extracting user complaints and feature requests). In our study, we observed that in many cases, developers ask a user for more information based on the review (in particular the final revision of a review). Hence, studying only reviews does not give a complete overview of user concerns. Future studies on extracting requirements from app reviews should investigate how user concerns in app reviews can be combined with concerns expressed through other channels, such as email or a support forum.

Developers use the review mechanism as a user support medium. Even though not many developers use the review mechanism in this way, future studies should monitor this type of usage to identify possible improvements to the review mechanism that may better facilitate developers and users.

8.2 Implications for Store Owners

Developers need to resort to alternative channels, such as email, to be of full assistance to a user. We encountered in many cases developers asking users to contact them through an alternative channel, such as email. One possible explanation is that the user is forwarded to a

more technical support team. Another possible explanation is that the review mechanism of the app store is not sufficient to provide adequate support. Mobile app store owners should study in depth how developers and users are using the review mechanism to find out how the mechanism can be improved.

Developers need to be notified whenever users update their reviews. As illustrated in Section 2, developers are not always notified when users update their reviews, which prevents developers from tracking users' changes in their impressions about the app. Moreover, missing notifications about user updates may hinder developers from noticing any additional information that is added by users to the posted reviews.

Developers can be assisted by highlighting reviews that are more likely to require a response. We showed that, depending on the app, there can be several types of reviews identified that developers respond to. Store owners can improve the response mechanism by highlighting reviews that are longer or have a low rating. The highlighting could be done using thresholds for the length of the review text, and the number of given stars. These thresholds can be automatically defined for developers based on their response behavior, or they could be configured by the developers themselves.

Not all developers follow the published guidelines for responding to reviews. Although Google has set guidelines for responding to reviews in the Google Play store (Google 2017), we showed that not all developers follow these guidelines. For example, several developers thank every user for their review. Developers are discouraged from posting such responses in the guidelines as they do not contain useful information for other users. A possible solution for avoiding such responses is to automatically filter repetitive or very short responses. Another possible solution to lower the number of useless responses is to put a limit on the total number of responses that can be posted per day. With such a limit, developers need to be more selective in deciding on which reviews to respond to. Google has recently introduced a limit of 500 responses per day per app in the Google Play store API (Google 2017). Future studies must investigate whether a limit of 500 responses is sufficient to ban useless responses from the store.

Many users may share the same concerns about an app. Store owners can improve the way in which reviews are displayed by clustering reviews and responses based on the concerns that are expressed and addressed in them. The most often-expressed concerns (and if available, their responses/solutions!) can then be displayed on the product page of an app. Clustering reviews and responses is beneficial for both developers and users, as (1) it gives developers the chance to explain to users how to handle a concern after downloading the app, and (2) it gives users the chance to adjust their expectations of the app.

9 Threats to Validity

9.1 Construct Validity

In our quantitative study of the metrics that are related to the response likelihood of a developer, we filtered out apps for which less than 5% of the reviews have responses. To evaluate whether our filtering has an impact on our results, we constructed our models using a threshold of 3%, 5% and 8%. We found that the most important metrics (i.e., the review rating and review text length) do not change across thresholds. Hence, we conclude that the threshold of 5% does not impact our results.

There are several approaches to understand what drives a developer to respond. For example, interviewing and surveying developers might be one way. In this paper, we chose

to instead look at the actual responses. Both approaches have their benefits and limitations. For example, with surveys developers might miss reporting on some reasons since we are depending on their recollection. Nevertheless, a mining approach has its limitation as well. For example, the collected data cannot represent all reasons for responding to user reviews. Thus, we suggest that future studies are needed to triangulate our observations through developer surveys.

9.2 Internal Validity

In our study, we analyzed the top free popular apps from 2013 for a two months period. Including more apps and extending the study period may provide more insights about developer responses. While we studied considerably more data than prior work on developer responses (McIlroy et al. 2015), future studies should revisit our findings for data that is collected during a longer period (i.e., years).

We assume throughout this paper that the posted responses are written by app developers. We were unable to find information about the team size and structure of the teams that built the top 25 apps with the highest percentage of developer responses. Clutch,[4] a company that lists 2,532 mobile app development firms, shows that 1,732 out of 2,532 (68%) of these firms have less than 50 employees. Hence, while we do not have strong evidence that responses were written by actual software developers, we do have evidence that most mobile app development firms are small. Therefore, due to the small team size, it is likely that responses to reviews are posted either by app developers themselves or by other team members who are in close contact with the app developers. Future studies should investigate further the team size and structure of mobile app development teams.

The results for our manual studies are impacted by the experience of the coders and the amount of the collected data. To reduce the errors in the manual analysis, three authors of this paper (at least two coders in each manual study) with past experience in analyzing mobile app data were involved in the manual analysis process. In addition, while doing the manual analysis, we extracted reasons for posting reviews and responses based on the implications of their contents. Hence, while we put considerable effort to mitigate the bias, the extracted reasons may be biased by our experience and intuition.

9.3 External Validity

App store data changes very rapidly and the Google Play Store provides only the latest 500 reviews per app. Therefore, the crawler may miss to crawl data when the crawls are not executed as often as required. Martin et al. (2015) illustrate the sampling problem in analyzing app store data. In order to overcome this issue, we scheduled our crawler to connect to the Google Play Store several times per day to collect as many reviews as possible. In our study, we collect the data for 8,218 apps. During the study period (from September 22^{nd} 2015 to December 3^{rd} 2015), in only 307 out of 547,966 crawls (0.06%) the maximum of 500 reviews (either new or updated reviews) could be downloaded. This shows that in at least 99.94% of the crawls, we were able to collect all the new reviews and the changes that were made to the existing reviews.

We studied developer responses for reviews of free apps only. One of the main reasons for removing non-free apps is that the pricing of an app is very likely to act as a major

[4] https://clutch.co/

confounding factor. Developers and users of paid apps may have different expectations and attitudes than developers and users of free apps. Hence, our findings are valid for free apps only.

In this paper, we classify developer response patterns based on general characteristics (e.g., review rating). The characteristics of developer responses may vary based on other factors such as the company size, team structure and cultural differences. Further studies are needed to investigate the nature of user-developer dialogues based on different factors (e.g., the cultural differences). However, our documentation of the type of patterns is an essential first step for future studies.

10 Related Work

While mobile app stores have been the subject of many studies recently, few studies have focused on the user-developer dialogue inside these stores. Instead, most studies focus on the analysis of user reviews only. In the remainder of this section, we give an overview of the most important related work. We refer to Martin et al.'s survey for an extensive overview of mobile app store research (Martin et al. 2016).

10.1 User Reviews

User reviews provide useful information for software developers. In this section, we represent relevant research on analyzing user reviews.

Iacob and Harrison (2013) manually analyzed 3,279 reviews for 161 apps from the Google Play Store. They found that 23.3% of the analyzed reviews contain feature requests. Iacob et al. proposed the MARA (Mobile App Review Analyzer) approach for extracting and analyzing feature requests from reviews. Later, they improved their MARA tool to use linguistic rules to identify reviews that contain bug reports or feature requests (Iacob et al. 2013).

Khalid (2013) and Khalid et al. (2015) manually analyzed 6,390 user reviews for 20 apps in the Apple Store. They identified 12 labels for different user complaints. Khalid et al. found that functional errors, feature requests and app crashes are the most frequent labels in the studied reviews.

McIlroy et al. (2016) built on the Khalid et al. (2015) work by developing an automated approach which analyzes 601,221 reviews for 12,000 apps in the Google Play Store. They found that 30% of the studied reviews contain different types of user complaints. McIlroy et al.'s automated approach multi-labels user reviews with the corresponding complaint types. The approach was evaluated to have a 66% precision and 65% recall.

Khalid et al. (2015) surveyed the existing mobile reviews research work. They proposed an approach for presenting and grouping user reviews based on different factors (such as complaint type, review rating and app update version). The proposed approach can be used by app developers to easily access reviews that are related to a certain factor.

Maalej and Nabil (2015) analyzed 1.3 million reviews for 1,186 apps in the Google Play and the Apple Stores. They classified the reviews into four categories: feature requests, bug reports, app rating and user experience. Maalej et al. used different review analysis techniques (such as sentiment analysis, Natural Language Processing (NLP) and string matching) in order to classify the reviews. Their approach achieves precision and recall values that range from 71% to 97% based on the technique that is used for classifying the reviews.

Table 16 summarizes the previous work that has been performed in analyzing user reviews. Our work differs from the existing review analysis research as our work focuses on studying the bidirectional dialogue between app users and app developers instead of only the unidirectional user reviews.

10.2 User-Developer Dialogue in App Stores

Oh et al. (2013) surveyed 100 users to study the different approaches for user-developer interactions. Oh et al. found that users prefer posting reviews on the mobile stores over giving feedback through other communication channels (such as email or telephone).

McIlroy et al. (2015) studied the impact of a developer response on the rating given by a user. They found that users often increase the given rating after a developer responds to the posted review. McIlroy et al. showed that the majority of developer responses provide assistance to the user or ask the user to contact the developer through another communication channel. In our paper, we revisited McIlroy et al.'s findings by conducting a more in-depth study on a larger dataset (as well tracking such user-developer dialogues over an extended period of time).

McIlroy et al. found that 13.8% of the apps respond to at least one review. In our study, we found that 794 out of 2,328 apps (34.1%) respond to at least one review. A possible explanation for this difference is that we studied apps that have more than 100 reviews only (to ensure maturity of the app). By including all still-active apps (8,218 apps) in our analysis, we found that there are 1,311 (out of 8,218) apps (16%) in which developers respond at least once to a user review. McIlroy et al. indicated that apps with a large number of reviews tend not to respond to reviews as they may be overwhelmed by the large number of reviews. In this paper, we showed that this perception appears to be changing, since we focused on apps that have at least 100 reviews.

In addition, McIlroy et al. found that a developer response has an impact on the review rating in 38.7% of the studied cases. In this paper, we found the number of reviews in which the rating is changed after a developer responds to be much lower (3.8%). Our findings did confirm that the rating change is often positive. A possible explanation for this difference is that we found that in many of the responses, the user is not actually being assisted directly by the developers. The large number of "Thank you" responses may influence the number of rating changes. The lack of assistance in the responses may be due to the large number of reviews that developers are dealing with, leading to (semi-)automated responses.

Table 16 Summary of the previous studies on analyzing user reviews

Study	Store	Number of studied apps	Number of analyzed reviews	Number of manually analyzed reviews	Number of analyzed responses
Iacob and Harrison (2013)	G	161	136,998	3,279	–
Khalid (2013) and Khalid et al. (2015)	A	20	226,797	6,390	–
McIlroy et al. (2016)	A, G	24	230,277	7,456	–
Maalej and Nabil (2015)	A, G	1,186	1,303,182	4,400	–
McIlroy et al. (2015)	G	10,713	111,099	384	111,099
This paper	G	2,328	4,474,023	347	126,686

(G = Google Play Store, A = Apple App Store)

11 Conclusions

The Google Play Store provides a mechanism that allows users and developers to engage in a dialogue. On the one hand, users leverage this mechanism by reading, writing, and updating reviews for an app. On the other hand, developers use this mechanism to respond to the posted user reviews.

In our study, we analyzed 4.5 million reviews with 126 thousand responses for 2,328 top free popular apps in the Google Play Store. We found that users and developers leverage the user-developer dialogues as a user support mechanism. However, in most cases the dialogue between the user and the developer is short.

Below are some of the most notable findings of our study:

1. Reviews and ratings change over time by a user. Hence, researchers need to consider the dynamic nature of reviews while analyzing user reviews.
2. When the rating is changed after a developer response, it is increased in most cases. Hence, app owners should focus more on responding to reviews, especially given that in many cases reported issues can be resolved without having to deploy an app update.
3. Users and developers use the user-developer dialogue as a user support mechanism. However, in most cases, the dialogue is short.
4. In general, developers are more likely to respond to reviews with a low rating and that are longer. The more in-depth analysis led us to identify four different patterns of developers. App store owners can facilitate a better responding mechanism by automatically highlighting reviews that are likely to require a response.
5. In 45% of the responses, the user is asked to provide more details about a reported issue. App store owners can improve the review-response mechanism by automatically providing developers with more details about the raised issues.

The presented findings in this paper show that responding to a review improves the chances of a user rating increase. However, we observed that developers do not leverage the potential of the response mechanism, as a large percentage of the responses consists of template-based responses. In particular, app owners should focus more on sending tailored responses that actually address the concerns that are raised by users. In addition, app store owners should improve their review-response mechanism as currently, developers need to ask the user to contact them via another communication channel to retrieve more details about a raised issue.

References

Akdeniz (2013) Google play crawler. https://github.com/akdeniz/google-play-crawler (last accessed: July 2017)

Anderson DR, Burnham KP, Gould WR, Cherry S (2001) Concerns about finding effects that are actually spurious. Wildl Soc Bull 311–316

App Annie (2013) The app analytics and app data industry standard: Google play store, united states, top overall, free, week 35. https://www.appannie.com/ (last accessed: July 2017)

AppBrain Free versus paid android apps. http://www.appbrain.com/stats/free-and-paid-android-applications. (last accessed: July 2017)

Bates D, Maechler M, Bolker B, Walker S (2017) Package 'lme4'. https://cran.r-project.org/web/packages/lme4/lme4.pdf (last accessed: July 2017)

Borgatti S Introduction to grounded theory. http://www.analytictech.com/mb870/introtogt.htm. (last accessed: July 2017)

Documentation for package 'stats'. https://stat.ethz.ch/r-manual/r-patched/library/stats/html/00index.html (last accessed: July 2017)
Eisenhauer JG (2009) Explanatory power and statistical significance. Teach Stat 31(2):42–46
Fox J, Weisberg S (2017) Package 'car'. https://cran.r-project.org/web/packages/car/car.pdf (last accessed: July 2017)
Gehan EA (1965) A generalized Wilcoxon test for comparing arbitrarily singly-censored samples. Biometrika 52(1–2):203–223
Google: Google my business help - read and reply to reviews. https://support.google.com/business/answer/3474050?hl=en (last accessed: July 2017)
Google: Google play developer API - reply to reviews. https://developers.google.com/android-publisher/reply-to-reviews (last accessed: July 2017)
Guzman E, Azócar D, Li Y (2014) Sentiment analysis of commit comments in GitHub: an empirical study. In: 11tH working conference on mining software repositories (MSR). IEEE, pp 352–355
Guzman E, Maalej W (2014) How do users like this feature? A fine grained sentiment analysis of app reviews. In: 22nd international requirements engineering conference (RE). IEEE, pp 153–162
Hanley JA, McNeil BJ (1982) The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143(1):29–36
Harman M, Jia Y, Zhang Y (2012) App store mining and analysis: MSR for app stores. In: 9th working conference of mining software repositories (MSR). IEEE, pp 108–111
Iacob C, Harrison R (2013) Retrieving and analyzing mobile apps feature requests from online reviews. In: 10th working conference on mining software repositories (MSR). IEEE, pp 41–44
Iacob C, Harrison R, Faily S (2013) Online reviews as first class artifacts in mobile app development. In: 5th international conference on mobile computing, applications, and services (MobiCASE), pp 47–53
Harrell FE Jr (2017) Package 'hmisc'. https://cran.r-project.org/web/packages/hmisc/hmisc.pdf (last accessed: July 2017)
Khalid H (2013) On identifying user complaints of iOS apps. In: 35th international conference on software engineering (ICSE), pp 1474–1476
Khalid H, Shihab E, Nagappan M, Hassan AE (2015) What do mobile app users complain about? IEEE Softw 32(3):70–77
Khalid M, Asif M, Shehzaib U (2015) Towards improving the quality of mobile app reviews. International Journal of Information Technology and Computer Science 35–41
Khandkar SH Open coding. http://pages.cpsc.ucalgary.ca/saul/wiki/uploads/CPSC681/open-coding.pdf. (last accessed: July 2017)
Long JD, Feng D, Cliff N (2003) Ordinal analysis of behavioral data. Wiley
Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? on automatically classifying app reviews. In: 23rd international requirements engineering conference (RE). IEEE, pp 116–125
Martin, P 77% Will not download a retail app rated lower than 3 stars. https://blog.testmunk.com/77-will-not-download-a-retail-app-rated-lower-than-3-stars/. (Last accessed: July 2017)
Martin W, Harman M, Jia Y, Sarro F, Zhang Y (2015) The app sampling problem for app store mining. In: 12th working conference on mining software repositories (MSR). IEEE/ACM, pp 123–133
Martin W, Sarro F, Jia Y, Zhang Y, Harman M (2016) A survey of app store analysis for software engineering. IEEE Trans Softw Eng (TSE) PP(99):1–32
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 (EMSE) 21(3):1067–1106
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(99):1–1
Moran K, Vásquez ML, Bernal-Cárdenas C, Poshyvanyk D (2015) Auto-completing bug reports for android applications. In: Proceedings of the 2015 10th joint meeting on foundations of software engineering (ESEC/FSE). ACM, pp 673–686
Oh J, Kim D, Lee U, Lee J, Song J (2013) Facilitating developer-user interactions with mobile app review digests. In: 2013 conference on human factors in computing systems (CHI). ACM SIGCHI, pp 1809–1814
Pagano D, Maalej W (2013) User feedback in the appstore: an empirical study. In: 21st international requirements engineering conference (RE). IEEE, pp 125–134
Perez S (2017) Apple will finally let developers respond to app store reviews. https://techcrunch.com/2017/01/24/apple-will-finally-let-developers-respond-to-app-store-reviews/. (Last accessed: July 2017)
Romano J, Kromrey JD, Coraggio J, Skowronek J, Devine L (2006) Exploring methods for evaluating group differences on the NSSE and other surveys: are the t-test and Cohen's d indices the most appropriate choices. In: Annual meeting of the southern association for institutional research
Seaman CB (1999) Qualitative methods in empirical studies of software engineering. IEEE Trans Softw Eng (TSE) 25(4):557–572
SentiStrength. http://sentistrength.wlv.ac.uk. (last accessed: July 2017)

Snijders TA, Bosker RJ (2012) Multilevel analysis: an introduction to basic and advanced multilevel modeling. Sage Publications
Snijders TAB (2005) Fixed and random effects. In: Encyclopedia of statistics in behavioral science. Wiley
Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2016) Comments on researcher bias: the use of machine learning in software defect prediction. IEEE Trans Softw Eng (TSE) 11(42):1092–1094
Thelwall M, Buckley K, Paltoglou G, Cai D, Kappas A (2010) Sentiment in short strength detection informal text. JASIST 61(12):2544–2558
Tourani P, Jiang Y, Adams B (2014) Monitoring sentiment in open source mailing lists: exploratory study on the apache ecosystem. In: 24th annual international conference on computer science and software engineering (CASCON), pp 34–44
Top Android phones. http://www.appbrain.com/stats/top-android-phones. (last accessed: July 2017)
What does AUC stand for and what is it? http://stats.stackexchange.com/questions/132777/what-does-auc-stand-for-and-what-is-it. (last accessed: July 2017)
Wilcoxon rank sum and signed rank tests. https://stat.ethz.ch/r-manual/r-devel/library/stats/html/wilcox.test.html. (last accessed: July 2017)

Safwat Hassan is a Ph.D. student at School of Computing, Queens University. Safwat worked as a soft- ware engineer for ten years in different corporations like Egyptian Space Agency (ESA), HP, EDS, VF Germany (outsourced by HP), and Etisalat. During his ten years work experience, he worked on different large-scale systems (varying from Web-Based systems to embedded systems) and in diverse project types (design service, customer support, and R&D) across various domains (Telecommunication, Supply-chain, and Aerospace). His research interest includes data mining, big data analytics, software engineering, mobile app store analytics. Contact him at shassan@cs.queensu.ca.

Chakkrit Tantithamthavorn received the BE degree from Kasetsart University, Thailand, in 2012, and the ME and PhD degrees from the Nara Institute of Science and Technology, Japan, in 2014 and 2016. He is a faculty member in the School of Computer Science, the University of Adelaide. He held one of the most prestigious and selective sources of national funding in Japan, i.e., a JSPS Research Fellowship for Young Researchers and a Grants-in-Aid for JSPS Fellow. He won the Best Ph.D. Student Award for his Ph.D. study at Nara Institute of Science and Technology, Japan. His research has been published at top-tier software engineering venues, such as, IEEE Transactions on Software Engineering (TSE), Empirical Software Engineering (EMSE), and the International Conference on Software Engineering (ICSE). His research aims to improve the fundamentals of statistical modelling for software engineering (e.g., defect prediction models) in order to produce more accurate predictions and reliable insights. More about him and his work is available online at http://chakkrit.com.

Cor-Paul Bezemer currently works as a postdoctoral research fellow in the Software Analysis and Intelligence Lab (SAIL) at Queen's University in Kingston, Canada. His research interests cover a wide variety of software engineering and performance engineering-related topics, including repository mining and performance regression analysis. His work has been published at premier software engineering venues such as the ESEC-FSE, ICSME and ICPE conferences and the EMSE journal. He was born in The Hague (Den Haag) in the Netherlands. Before moving to Canada, he studied at Delft University of Technology, where he received his BSc (2007), MSc (2009) and PhD (2014) degree in Computer Science. The title of his PhD thesis was "Performance Optimization of Multi-Tenant Software Systems". More about Cor-Paul can be read on his website: http://sailhome.cs.queensu.ca/∼corpaul.

Ahmed E. Hassan is the Canada Research Chair (CRC) in Software Analytics, and the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University, Canada. His research interests include mining software repositories, empirical software engineering, load testing, and log mining. Hassan received a PhD in Computer Science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. Hassan also serves on the editorial boards of IEEE Transactions on Software Engineering, Springer Journal of Empirical Software Engineering, and Springer Journal of Computing. Contact him at ahmed@cs.queensu.ca.