Empir Software Eng
https://doi.org/10.1007/s10664-018-9601-1

App store mining is not enough for app improvement

Maleknaz Nayebi[1] · Henry Cho[2] · Guenther Ruhe[1]

Abstract The rise in popularity of mobile devices has led to a parallel growth in the size of the app store market, intriguing several research studies and commercial platforms on mining app stores. App store reviews are used to analyze different aspects of app development and evolution. However, app users' feedback does not only exist on the app store. In fact, despite the large quantity of posts that are made daily on social media, the importance and value that these discussions provide remain mostly unused in the context of mobile app development. In this paper, we study how Twitter can provide complementary information to support mobile app development. By analyzing a total of 30,793 apps over a period of six weeks, we found strong correlations between the number of reviews and tweets for most apps. Moreover, through applying machine learning classifiers, topic modeling and subsequent crowd-sourcing, we successfully mined 22.4% additional feature requests and 12.89% additional bug reports from Twitter. We also found that 52.1% of all feature requests and bug reports were discussed on both tweets and reviews. In addition to finding common and unique information from Twitter and the app store, sentiment and content analysis were also performed for 70 randomly selected apps. From this, we found that tweets provided more critical and objective views on apps than reviews from the app store. These results show that app store review mining is indeed not enough; other information sources ultimately provide added value and information for app developers.

Communicated by: Yasutaka Kamei

✉ Maleknaz Nayebi
mnayebi@ucalgary.ca

Henry Cho
henry.cho@mail.utoronto.ca

Guenther Ruhe
ruhe@ucalgary.ca

[1] SEDS Lab, University of Calgary, Calgary, Alberta, Canada

[2] Department of Engineering Science, University of Toronto, Toronto, Ontario, Canada

Published online: 22 February 2018

Keywords App store mining · Twitter · Mobile apps · Topic modeling · Machine learning · Crowdsourcing

1 Introduction

The high demand for mobile devices, combined with the relative ease of app development has caused an exponential growth of the app market in the past half-decade. The resulting quantity of available apps caused competition between app developers to earn a spot on mobile users' devices. Thus, now more than ever, app developers need a source where they can collect user demands, feature requests, and general opinions to cater to the needs of their consumers effectively. The paramount importance of this developer-user feedback loop is confirmed by numerous studies in the area (Martin et al. 2016).

Where do app developers currently search to find all this information? Looking into third party data analytics platforms[1,2,3] and the existing software engineering body of knowledge (Martin et al. 2016; Nayebi et al. 2016; Palomba et al. 2015), it can be seen that the most popular source utilized by app developers and researchers for mining user feedback is the app store. The app store is used to collect reviews and user meta-data to analyze how to enhance the apps (Martin et al. 2016). However, the conversations about apps do not only happen in the app store. App users often employ social media platforms to discuss their experience with an app, whether it would be praise (Maalej and Nabil 2015), feature requests or users sharing the app with their social media connections (Palomba et al. 2015). As a follow up of our former work (Nayebi et al. 2017), the *main objective* of this paper is to show that the evaluation and analysis of these app discussions on social media serve as valuable resources for app developers. We focused on Twitter to evaluate the added value and insight that developers can attain from tweets when compared to reviews.

Over a period of six weeks (June 24^{th} to Aug 7^{th}, 2016), we investigated the relationship between the number of reviews and number of tweets for a sample set of 30,793 Google Play apps. This sample included 7,933 apps from Google Play top charts (all the top chart apps across different categories) as well as random selections of apps with diverse rating and number of reviews (Martin et al. 2016). For the purpose of an in-depth content analysis, we randomly selected 70 apps and compared the content of app store reviews with the content of app-related tweets. These comparisons provided insight into Twitter's potential to give developers extra information that cannot be obtained with app store reviews. To achieve that, we applied automated classification and subsequent topic modeling, following the process employed in previous studies in this context (Chen et al. 2014; Guzman et al. 2017; Maalej and Nabil 2015; Martin et al. 2016). We also went one step further and evaluated the quality of the extracted topics with automatic classification through crowdsourcing. Specifically, we investigated the following research questions:

RQ1: When considered over a period of time, is there a correlation between the number of tweets and the number of reviews about mobile apps? If so, which apps exhibit the strongest correlation?

[1] https://www.appannie.com.

[2] http://www.appbrain.com.

[3] http://www.searchman.com.

Why and How: Prior to an in-depth analysis of content, we focused on the quantitative analysis of review and tweet trends. This means that we wanted to understand if an increase or decrease in user attention on an app brings about similar responses from both the app store and Twitter. We were mostly interested in exploring the types of apps that exhibit stronger/weaker correlations (between number of reviews and tweets) and consequently the types of apps that are potentially suitable for in-depth content analysis. To do so, we investigated the correlation between the number of tweets and the number of reviews over time and compared the distribution of the number of reviews and tweets for each app.

RQ2: What type of information can developers gain from tweet analysis that cannot be attained from the app store reviews?
Why and How: We were interested to see if it is worthwhile to look at tweets in addition to studying the more technically-oriented reviews. To ensure that we compare feature requests and bug reports within tweets with feature requests and bug reports within reviews (respectively), we first classified tweets and reviews into feature requests, bug reports, and others (including user experience, rating, and advertisement). For the two categories of bug reports and feature requests, we applied topic modeling and compared the topics of tweets with the topics of reviews to see if tweets include any extra information for app development in comparison to app reviews.

RQ3: How do app store reviews and app related tweets compare with regards to sentiments, degree of specification, and understandability?
Why and How: Sentiment analysis was used to understand users and prioritize their needs for app evolution and improvement (Martin et al. 2016; Gu and Kim 2015). Based on this, we were further interested in the degree of sentiment alignment between tweets and reviews of mobile apps. We also were interested in understanding how the reviews and tweets are characterized and related to each other with regards to technical specification and understandability. To investigate these questions, we compared the sentiment values of reviews and tweets. The sentiment analysis was done based on polarity (positivity and negativity) and subjectivity (factual or opinionated) using machine learning. The degree of specification and understandability was crowd evaluated for *all* tweets and reviews.

The results of the paper show the importance of analyzing alternative information sources other than just app store reviews. Specifically, *app store mining is not enough* in the sense that additional and complementary information can be extracted from other media to support app development, where Twitter is just one prominent example. With our results, we go beyond the existing work of just analyzing one feedback mechanism at a time. While supporting app development currently is limited to app store mining (Martin et al. 2016; Palomba et al. 2015; Villarroel et al. 2016), our work shows the importance of looking beyond the fence (Maalej et al. 2016) of the app stores to support app development.

We published initial results related to RQ2 in Nayebi et al. (2017). In comparison to that, in this paper we:

- Added two research questions (RQ1 and RQ3).
- Provided a motivating example.
- Extended the related work section.
- Analyzed a bigger sample of apps for unbiased app selection and comprehensively discussed data gathering and processing.

- Provided insight to the quantitative relation between the number of reviews and tweets as part of RQ1.
- Provided further discussion and examples for RQ2.
- Compared reviews and tweets with regards to sentiment, the degree of specification and understandability.
- Discussed the applicability and threats to validity.
- Compared users of twitter with Google Play users based on the user IDs.

In the next section, we provide a motivating example for our study. We give an overview of the related work in Section 3. In Section 4, we describe the data gathering and pre-processing steps. Then in Section 5, we discuss the variety of statistical, natural language processing, and machine learning techniques that we used. In Section 6, we present our results of comparing tweets with app reviews. Threats to validity are studied in Section 7. A discussion of results is provided in Section 8. The paper is wrapped up by conclusions and an outline of future work in Section 9.

2 Motivating example

In this section, we briefly illustrate the main idea of the paper through a recent example. Pokemon Go, the free-to-play and location-based game with augmented reality features was released on July 5^{th}, 2016. It got a lot of attention from users, turning it into a global phenomenon. Pokemon Go users wrote 254 reviews on the first day after the app release. For the same day, we mined 2,103 tweets related to its Android mobile app. Figure 1 compares the number of reviews with the number of tweets for Pokemon Go over six weeks period of time. We observe a similarity in trend, showing that the app drew increased attention in both media. During the first six days of the app release, we can see that users tweeted about the app more than reviewing it in the app store. Beginning of August, there is a jump in both numbers of tweets and number of reviews, but the jump in reviews comes a few days later. Twitter users did not only react faster, but also reported servers and feature crashes to the app owners.

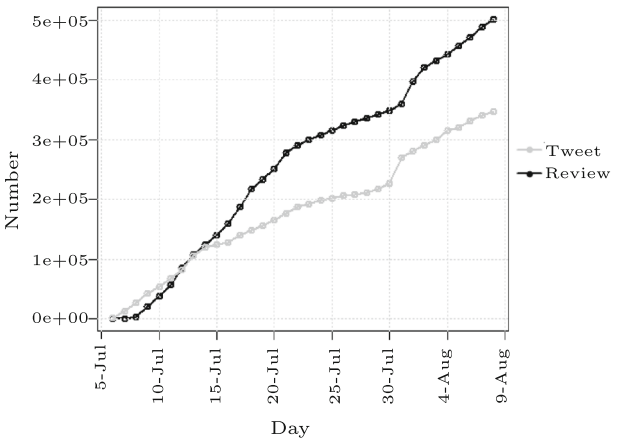


Fig. 1 Cumulative number of app store reviews and tweets about Pokemon Go app for a period of six weeks from June 24^{th} to Aug 7^{th}, 2016

For example, on July 28^{th}, Twitter users warn each other about the new update of Google Maps which adversely affected the accuracy of the Pokemon Go map. The official Pokemon Go News account used this form of Twitter crowd intelligence and officially warned the users about this update and provided a workaround. In comparison, the reviews mined from the app store show very few mentions about Google Map changes, and the reviews are written three days later (on Aug 3^{rd}). Having a comprehensive understanding of users' needs and desires in a timely manner and being aware of the need to change existing features is a key advantage that could be attained by analyzing tweets, not just app store reviews.

Similarly, when a pay wall and yearly subscription fees for accessing several features was introduced for the Evernote app in a release (on Jun 13^{th}), users reacted both in app store reviews and tweets. Looking into the tweets, we found that Twitter users repeatedly submitted the request of "accessing passcode pin for free"; app owners made this feature available in the most recent release of the app.

Figure 2 shows an example result of this paper for Pokemon Go. We mined both app store reviews and app related tweets and found the commonalities and differences between tweets and reviews that are informative for developers.

3 Related work

App store review analysis and Twitter analysis are established fields of research. In this section, we discuss the most related studies of these two fields.

3.1 App review analysis

Reviews have been analyzed with different objectives in previous studies. An overview was provided by Martin et al. (2016) where they analyzed 45 studies on app store reviews performed between 2012 and 2015. Several studies focused on different techniques for extracting app features from reviews (Iacob and Harrison 2013), while other studies focused on developing models and tools for automated classification of reviews into

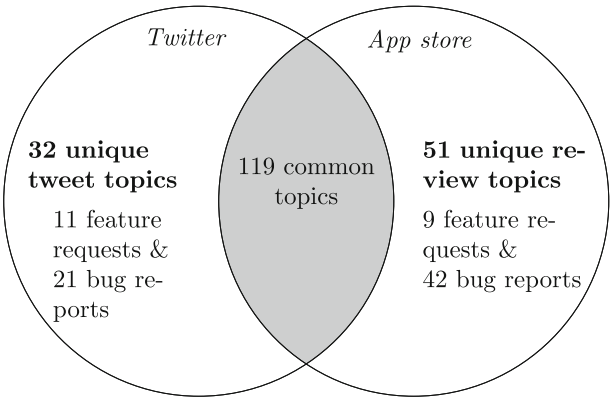


Fig. 2 Comparison of the number of feature request and bug reports between app reviews and tweets for the Pokemon Go app

predefined categories (Chen et al. 2014; Gomez et al. 2015; Gu and Kim 2015; Maalej and Nabil 2015).

Maalej and Nabil (2015) discussed machine learning methods for classifying app reviews. They classified app reviews as bug reports, feature requests, user experiences, and ratings. They described the pre-processing and stemming steps and its threats for app reviews. They also compared the performance of *Naive Bayes*, *Decision Tree* and *Max Ent* classifiers on mobile app reviews and found that *Naive Bayes* achieved high accuracy with a smaller training set and much less time in comparison to other classifiers. Based on these findings, we used *Naive Bayes* as one of the learning methods in our methodology.

For the content analysis of app reviews, Chen et al. (2014) suggested the use of topic modeling in AR-Miner. AR-Miner (Chen et al. 2014) first classifies reviews into informative and non-informative and applies topic modeling within each class. We combined this method with the classification suggested by Maalej and Nabil (2015) to analyze the content in depth. Furthermore, mining app reviews have been used as a basis for decision support. Palomba et al. (2015) analyzed 100 Android apps and showed that addressing user reviews increases the chance of apps' success. Villarroel et al. (2016) introduced CLAP (Crowd Listener for releAse Planning) for categorizing reviews into bug reports, clustering related reviews, and prioritizing reviews. More recently, Ciurumelea et al. (2017) defined a two-level taxonomy based on 1,556 reviews with the aim to assist mobile app developers with planning maintenance and evolution activities. Di Sorbo et al. (2017) defined a taxonomy of reviews and provided a tool for categorizing reviews.

Sentiment analysis of mobile app reviews for analyzing positivity or negativity of users' feedback have been discussed by different authors (Martin et al. 2016). Chen et al. (2014) used sentiment of app reviews to identify informative reviews. In a nutshell, sentiment analysis was used to identify informative reviews, user complains, and priorities tasks for app evolution.

In short, app reviews formed the basis for many studies and decisions ranging from feature extraction to release planning of mobile apps. The underlying assumption for these studies is that developers would benefit from looking into reviews for prioritizing bugs (Chen et al. 2014; Gomez et al. 2015) and planning for releases (Jivani et al. 2011). While this is true, our study shows that the app store review data is not enough and media such as Twitter can provide complementary information for app development. In fact, many app development companies already have an active and official channel on Twitter to communicate with their users (Guzman et al. 2017).

3.2 Tweet analysis

Analyzing Twitter to gain information about peoples' communication is the subject of study in different fields including but not limited to journalism, emergency management, biomedicine, stock market analysis, and most importantly software engineering. On the other hand, several research studies on natural language mining, social media analysis, communications, and sentiment and opinion analysis were introduced as enabling technologies for using Twitter information. While these studies are outside software engineering, we relied on their promising results we introduce and use them for this study.

3.2.1 Tweet analysis in software engineering

Twitter was analyzed from different software engineering perspectives. Bougie et al. (2011) studied the potential of Twitter and micro-blogging for software development while

Tian and Lo (2014) analyzed the software developer microbloggers and their communities on Twitter. Wang et al. (2014) studied tweets for Drupal software and reported that Twitter is used to solicit contributions. In a very recent study, Nayebi et al. (2017) introduced MAPFEAT to map tweets (as a source of user requirements) into mobile app features.

In a very recent study, Guzman et al. (2017) studied the contents of tweets and investigated their potential for software requirement engineering. They analyzed the content of 6,437,282 tweets across 22 software applications and defined their relevance for different stakeholder groups. Furthermore, they manually categorized 1,000 tweets to define the tweet sentiment (Likert scale 1 to 5) and the content categories. They ended up with 26 content categories that were extracted manually. They also discussed the automation potential for classifying the content of tweets. They compared Decision Tree and Support Vector Machine (SVM) classifiers. Their results showed that SVM has better performance when compared to Decision Tree analysis (Guzman et al. 2017). Based on their results, we selected SVM as another classifier for our study. Prasetyo et al. also suggested SVM (Prasetyo et al. 2012) for automatic classification of tweets into software engineering related and unrelated classes.

The value added by Twitter analysis was determined by studying the differences between contents of tweets and app store reviews. We employed both Naive Bayes as suggested by Maalej and Nabil (2015) and SVM as suggested by Guzman et al. (2017) to classify the content and hence contrast tweets and reviews. We also applied topic modeling, which was previously used by Chen et al. (2014) for app review analysis and for Twitter analysis (Hong and Davison 2010). Using topic modeling, we compared feature requests and bug reports in Google Play and Twitter. Williams and Mahmoud (2017) analyzed tweets related to software products for automatic classification of technical feedback. Guzman et al. (2017) performed a study on the prioritization for software related tweets.

3.2.2 Sentiment and content analysis of tweets

Sentiment analysis of tweets is broadly investigated by researchers to understand feelings of the general public toward specific events. Agarwal et al. (2011) investigated on tweet features for training better learners. Their study showed that use of Twitter-specific features such as hashtags does not tremendously change the accuracy of the classifiers. They conclude that sentiment analysis of tweets does not extensively differ from sentiment analysis in other genres. Thelwall et al. (2011) analyzed tweet sentiments of specific events. They found that peaks of interest have stronger positive sentiment during the event in comparison to the time before that event.

Kouloumpis et al. (2011) and Smedt and Daelemans (2012) studied lexical features of tweets for sentiment analysis. Mohammad et al. (2013) and Rosenthal et al. (2015) proposed different methods with high accuracy for sentiment analysis of tweets. These works had triggered many other studies for designing high-performance systems for sentiment and content analysis of tweets.

Content analysis of tweets was done manually in some studies (for example Naaman et al. 2010) while many studies use topic modeling for this purpose. Ramage et al. (2010) used LDA for topic modeling in consideration of emotions, social signals, and hashtags on Twitter. However, O'Connor et al. (2010) used syntactic filtering, language modeling for extracting topics from tweets. Hong and Davison (2010) compared multiple topic modeling techniques on the tweets. Their results showed that topic models learned from tweets of each user have superior performance in classification problems. These topic modeling

techniques were widely adopted to different contexts and were used for content analysis of microblogs.

4 Data gathering and preparation

We gathered app related data from both Google Play and Twitter. The apps we selected for analysis are documented in Section 4.1. The data collection processes for reviews and tweets are detailed in Sections 4.2 and 4.3, respectively.

4.1 Selection of apps

Following the arguments and techniques proposed by Martin et al. (2015), we selected a diverse set of apps for the analysis. We selected top-chart apps as well as non-top chart apps having different prices (free vs. priced), different number of reviewers and different ratings. For our investigation, 30,793 apps were analyzed, with each of them belonging to one of the following categories:

i 7,993 trending apps from Google Play top charts (all the top chart apps).
ii 7,600 apps: Rating $\geq$ 4.0 AND # of reviewers $\geq$ 1,000.
iii 7,600 apps: Rating $\geq$ 4.0 AND # of reviewers $\leq$ 100.
iv 3,800 apps: Rating $<$ 4.0 AND # of reviewers $\geq$ 1,000.
v 3,800 apps: Rating $<$ 4.0 AND # of reviewers $\leq$ 100.

The 7,993 trending apps were collected from all six Google Play defined top charts - *Top Free, Top Paid, Top Grossing, Top Free Games, Top Paid Games and Top Grossing Games*. Top charts are published by the Google Play team based on popularity and market trends. We crawled the top charts *daily* to track changes in trending apps as well as rise and fall in apps' popularity. The remainder of the apps were collected randomly from a sample of the Google Play app store, according to the above classification. For this, we crawled 100,000 apps with a random crawler.[4]

Looking into the rating distribution of the sample apps, we found that about 33% of them had a rating of 4.0 or lower. The rest (67% of apps in our random sample) had ratings above 4.0. We defined categories of apps having below 4.0 and equal or above 4.0 rating. Looking into the 10% percentile of our set of 100,000 random apps, a number of 100 or fewer reviewers is classified as a *low* number of reviewers. Similarly, a number of 1,000 or more reviewers representing about the top 80% of the reviewers, is called *high*. Having these categories, we randomly picked apps among each category in a way to make sure they did not belong to top charts. As the result of this selection, we compared top chart apps with (i) high rating AND high # of reviewers, (ii) high rating AND low # of reviewers, (iii) low rating AND high # of reviewers, and (iv) low rating AND low # of reviewers. For each app, the respective package names were collected. This enabled the investigation of the apps' market-side data, as to be detailed in the next section.

In **RQ1**, we quantitatively analyzed the relationship between reviews and tweets for all the above apps. In **RQ2** and **RQ3**, we focused on 70 apps for a more in-depth sentiment and content analysis involving 1,267,895 reviews and 358,860 tweets. We selected these apps from all Google Play *top charts* only, as we did not get a sufficient number of reviews and

[4] http://nutch.apache.org/.

tweets for the ones outside top charts for the selected six week period as we will discuss in **RQ1**. For app selection, we considered the ratio between # of tweets and # of reviews. The *# of tweets* to *# of review* ratio was in the range [0.083, 251] for the 7,993 apps considering all the data within the six weeks period.

We selected apps with the five lowest and five highest ratios between # of tweets and # of reviews from each top chart. We also randomly selected 5-10 other apps from each category (based on the number of apps in each top chart). Following this method, we picked 20 apps from each of the *top free* and *top free games* categories. We also randomly selected 15 apps from each of the *top paid* and *top paid games* categories. As we will report later, non-top chart apps did not receive many reviews and tweets in the six weeks period, hence were not selected for our study.

4.2 Data from Google Play

For each of the above 30,793 apps, we extracted the market-side data from the Google Play platform. Attaching the individual app's package name at the end of the following URL "https://play.google.com/store/apps/details?id=" allows access to the app's unique web page. Once this page was accessed, the HTML data was crawled and parsed to collect the following information about each app - *app name*, *rank in top chart* (if the app is on the top chart), *ratings*, *app description*, *# of downloads*, *# of reviews*, *last updated date*, and *user reviews*. The average length of reviews in our dataset is 97 words with the median being 92 words.

4.3 Data from Twitter

Tweets about the apps were collected using the Twitter API, as provided through the `CliPS Pattern` module (Smedt and Daelemans 2012). Using the `CliPS` open source module, we connected to the Python implementation of the Twitter API, which allowed for the mining of content, date-stamp and unique ID of tweets. We implemented various strategies to ensure that the tweets collected were complete and accurate.

First, the search was performed with the word *"app"* and *"mobile"* attached to the end of the app name (ex. *"Facebook"* - *"Facebook mobile app"*). This may have filtered out some relevant tweets but ensured that all tweets collected were relevant to the mobile application, rather than its web or desktop counter-parts. We also excluded tweets having any of the words *"apple"*, *"ios"*, *"iphone"*, *"blackberry"*, *"desktop"*, *"Mac"*, *"window"* and *"win"* from our search query, to ensure that all tweets were relevant to the Android application. Using the `CLiPS Pattern` module to access Twitter, we ensured that all tweets collected were in English.

Second, in order to make the search more comprehensive, we integrated common acronyms and short-hand names of an app (e.g. *"Facebook"* and *"FB"*) into the search algorithms. We manually extracted the acronyms by searching the Internet (e.g. we sent the query *"Facebook acronyms"* to the Google search engine and got *"FB"* as a common short-hand).

Finally, we cross-referenced tweet ID between searches to ensure that no duplicate tweets existed in the data collected. We performed the search using this search algorithm every two days to ensure the stable number of *likes* (♥) and *re-tweets* (🔁) starting *Jun* 24^{th}, 2016 for all the 30,793 apps. The resulting data contained 8,349,308 tweets. These tweets were filtered to ensure the accuracy and quality of the data. The average length of tweets in our dataset is 28 words with the median being 34 words.

4.3.1 Eliminating duplicate tweets

Duplicate tweets often emerged from the data collected. These duplicate tweets were either (i) one user repeatedly posting the identical tweet, (ii) a user re-tweeting another user's tweet, or (iii) identical tweets posted by two different users. We removed all three types of duplicate tweets from the dataset due to two main reasons.

First, the analysis of the tweets in later sections included the sentiment analysis. Large numbers of duplicate tweets would have skewed the results of this analysis. For example, if one user repeatedly posted the identical negative feedback of an app, and all of the tweets were considered, the app would be deemed to have an overly negative review, despite the fact that only one user had a negative experience and re-tweeted it several times. Second, applying machine learning techniques for tweet classification and later topic modeling is computationally intensive. Duplicate tweets would have hindered the ability to apply these techniques in a timely fashion. Eliminating duplicate content increased the performance of automated techniques without jeopardizing the content analysis, as only one instance of each tweet would remain in our set.

4.3.2 Eliminating irrelevant and spam tweets

As discussed earlier, we sent queries in the form of *"app name" AND "mobile app"* for retrieving tweets about each of the 30,793 apps. However, irrelevant tweets and spam tweets still inevitably emerged in the tweets collected. Irrelevant tweets were tweets that do not pertain to the app itself. These tweets are prevalent in apps with a common name - for example, apps with the name *"Photo Editor"* have a lot of irrelevant tweets as opposed to an app with a unique name such as *"Clash of Clans"*. We filtered out 41 apps with generic names because of the high likelihood of large amounts of irrelevant tweets such as apps named *"Music Player"* or *"One"* from our sample set to mitigate the risk of irrelevancy. For example, the tweet *"Is anyone aware of a music player app for Android?"* may not be talking about the "Music player" app with the package name `media.music.musicplayer`. To detect apps which are related to a significant amount of irrelevant data (more than 10% of all the tweets), two of the authors manually inspected all the tweets per app and selected the apps to be filtered out from our set.

Furthermore, spam tweets are irrelevant. In defining spam, we used Twitter's official definition of spam, "Spam can be generally described as unsolicited, repeated actions that negatively impact other users".[5] Spam tweets were detected using three methods. First, following observations noted in Guzman et al. (2017), we removed tweets where the posting user's name contains the word *"bot"*, as they have a high likelihood of being spam. Second, we manually scanned the tweets to pinpoint users that share spam tweets. These users were added to our spam list as spammers (Benevenuto et al. 2010). All tweets posted by spam users were deemed to be spam and were removed. Three software engineers (including two of the authors) went through the tweets sorted per user. At least two of the three software engineers checked each user and detected if the user is a spammer or not. Third, it was observed that spam tweets were often detected and deleted after a few days period by Twitter. Following this, we searched for each tweet using its tweet ID after *three* days. If the search request for the tweet failed, it indicated that the tweet had been deleted. Hence, these tweets were excluded from our sample set.

[5] https://support.Twitter.com/articles/64986.

Among the 8,349,308 tweets which we gathered over a six weeks time period, we filtered out 3,481,438 tweets due to irrelevancy, duplication or spam content. Of the 3,481,438 tweets that were removed out of the 8,349,308 collected, 2,868,704 were deleted for being a duplicate, 490,187 were deleted for having 'bot' in the username, and 122,547 were deleted due to being posted by a manually identified spam user. Thus, we continued our analysis with the 4,867,870 remaining tweets.

5 Methodology

To approach the stated research questions, we applied several statistical tests and analytic methods to analyze (i) the correlation between the number of reviews and number of tweets, (ii) the comparison between the content of app reviews and app related tweets in terms of the insight they provide for app developers, and (iii) the alignment of sentiments between tweets and app store reviews. Statistical methods were done for exploratory purposes. To investigate the availability of extra information obtained from mining tweets, we performed classification and topic modeling of reviews and tweets independently and subsequently analyzed the similarities and differences between the extracted topics. Once we observed that extra development information is available in the tweets, we compared the sentiment, degree of specification and understandability of the tweets and reviews. Overview of the techniques we used to answer each research question is provided in Fig. 3.

5.1 RQ1: statistical analysis

The investigation involved extensive comparisons between data sets to study correlations and distribution of data. Various statistical methods were used to achieve this. These methods were used to answer **RQ1**. First, the Pearson correlation test (Gibbons and Chakraborti 2011) was utilized to study the numerical alignment between the time series data of tweets and reviews. We tested the relationship between the number of reviews and the number of tweets using this test.

Second, the Mann-Whitney U-test (Gibbons and Chakraborti 2011) was used to study the commonalities and differences between two independent sets of non-parametric data. We used the test to compare the distribution of the number of tweets and the number of reviews in **RQ1**.

5.2 RQ2: extracting and comparing App review and tweet topics

To compare the content of app reviews with the content of app related tweets, we first classified tweets and reviews into different categories, following Chen et al. (2014). We used two different and independent classifiers, *Naive Bayes* and *SVM*. In past works, Naive Bayes performed best to classify app reviews (Maalej and Nabil 2015) and SVM performed best for software engineering tweet classification (Guzman et al. 2017). We filtered out reviews

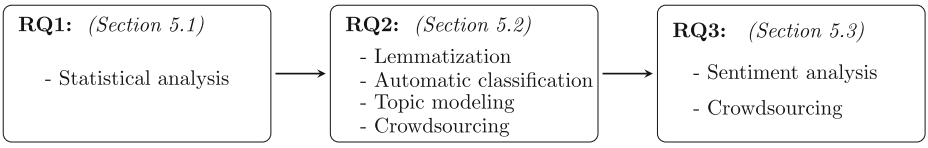


Fig. 3 Overview of the techniques used to answer each RQ

and tweets that were classified inconsistently by the SVM and Naive Bayes classifiers. We used the taxonomy suggested by Maalej and Nabil (2015) as:

– We were interested in the improvement requests and the evolution of the app (and not the praise, user experience, etc.).
– There is no taxonomy based on tweets and reviews of mobile apps and a more fine-grained taxonomy is biased either toward tweets (Guzman et al. 2017) or toward reviews (Di Sorbo et al. 2017; Di Sorbo et al. 2016).

Subsequently, we applied topic modeling for tweets and reviews in each category (being a feature request or bug report) (Chen et al. 2014). Finally, we calculated the similarity between the topics extracted from tweets and the ones extracted from app store reviews. The process is demonstrated in Fig. 4. This analysis was applied to 70 randomly selected apps - entailing 358,860 tweets and 1,267,895 app store reviews. The results were used to answer **RQ2**.

Going from analyzing numbers to content, we applied multiple natural language processing and machine learning techniques to analyze app reviews and tweets, as formulated in **RQ2** and **RQ3**. The process overview is shown in Fig. 4.

We used an open source module, named `Pattern` (Smedt and Daelemans 2012), which is built on top of the `NLTK` comprehensive Python package (Loper and Bird 2002). Python's `NLTK` has been used in a variety of recent app store mining studies in software engineering (Harman et al. 2012). `Pattern` is an open source module that we calibrated, with additional Twitter IDs and specified list of top words, and used for the purpose of this

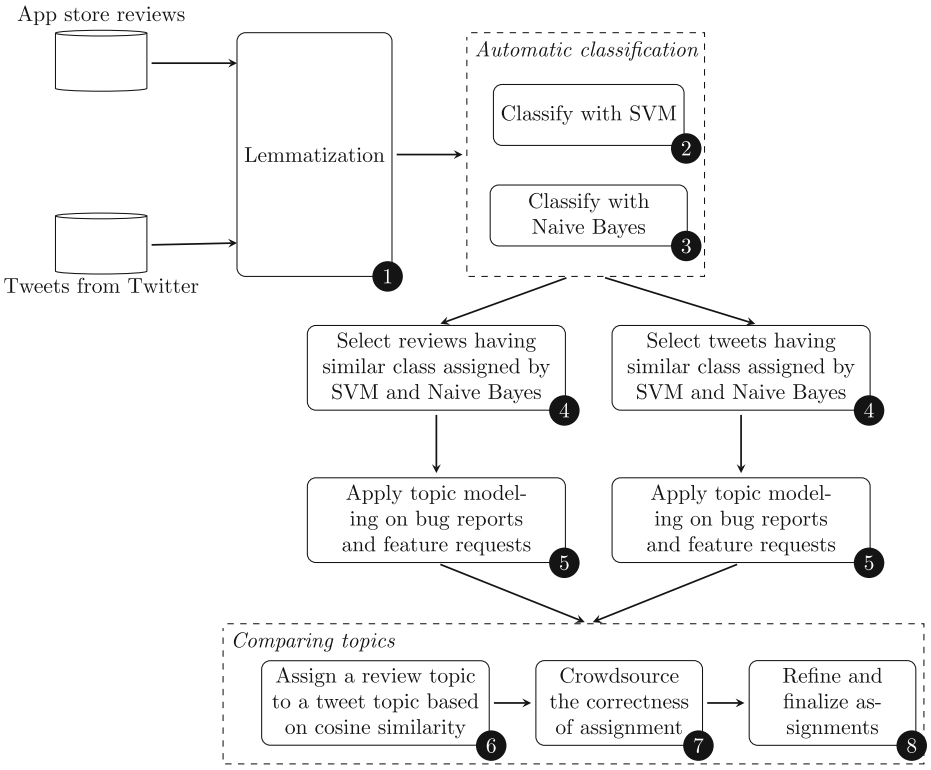


Fig. 4 Process per app for comparing textual content of reviews and tweets

study. The program is used for tweet collection via the Twitter API, as well as for sentiment analysis on the tweets collected. `Pattern` is trusted by many academic scholars, as demonstrated by its wide usage (Smedt and Daelemans 2012).

5.2.1 Lemmatization

Having app reviews and app related tweets, we first removed stop words such as "with", "my", "them" from textual data (Step ❶) in Fig. 4). To do so, we inspected and customized the default set of stop words to keep words such as "should" or "must" as suggested by Maalej and Nabil (2015). We applied lemmatization to map English words into their dictionary form (also named Lemma). For instance, through lemmatization, we mapped *"helped"*, *"helps"*, and *"helping"* to *"help"*. We chose lemmatization over Porter's stemming (Porter 1980) as the contexts of the review and tweet are highly important. Lemmatization maps *"worse"* to *"bad"* while the Porter's stemming misses this link (Jivani et al. 2011; Maalej and Nabil 2015).

5.2.2 Automatic classification of tweets and reviews

Building on top of past research, we automatically classified reviews and tweets into fine-grained categories based on their content as shown in Step ❷ and Step ❸ of Fig. 4. We then applied topic modeling on each class separately (Chen et al. 2014). Maalej and Nabil (2015) defined four categories for app reviews: bug reports, feature requests, user experience, and rating. In another work, Guzman et al. (2017) defined 21 categories for software engineering related tweets after manually analyzing 450 tweets. However, several of these tweet categories are not applicable to app reviews. With the aim of comparing app reviews with app related tweets, we classified reviews and tweets. In a way that each tweet or review belongs to *at most two* of the following categories:

Feature requests, are users' suggestion on functionality or content to enhance the app in future releases.

Bug reports, describe problems with an app such as crashes or performance issues.

Others, this category covers individual and social interactions among users who are looking to share their experience on using the app. This category also encompasses ratings (simple text reflecting numeric rating such as *"great app"*) and advertisements made by users or companies.

Maalej and Nabil (2015) compared different classifiers to categorize *app reviews*. They reported that *Naive Bayes* achieves high accuracy with a small training set and needs much less time in comparison to other studied methods. On the other hand and with the aim to analyze *tweets* for software development, Guzman et al. (2017) suggested the use of SVM method instead of Decision Trees (C4.5 algorithm) as it has higher precision and ability to filter out irrelevant tweets. In Step ❹ of Fig. 4 we applied SVM and Naive Bayes separately for reviews and tweets of each app.

Naive Bayes is relying on the strong assumption that existence of a word in the review or tweet is independent of the existence of another word. For example, this would mean that if the tweet contains the word *"tracker"*, this gives us no further information about *"calorie"*. This algorithm is simple and proven to perform well for small training sets on mobile app reviews (Maalej and Nabil 2015).

Support Vector Machine (SVM) is a universal classifier which finds the decision boundary between each two classes. SVM learns independently from dimensions of the feature space (words in the context of reviews or tweets) (Sebastiani 2002).

We evaluate SVM and Naive Bayes classifiers in terms of precision and recall for category i (i being *feature request* or *bug report*) as below:

$$Precision(i) = \frac{TP(i)}{TP(i) + FP(i)} \tag{1}$$

$$Recall(i) = \frac{TP(i)}{TP(i) + FN(i)} \tag{2}$$

$$F1(i) = \frac{2 \times Precision(i) \times Recall(i)}{Precision(i) + Recall(i)} \tag{3}$$

TP(i): # of reviews or tweets truly classified as category i.
FP(i): # of reviews or tweets falsely classified as category i.
FN(i): # of reviews or tweets falsely classified as not being category i.

For training and evaluating classifiers, we randomly selected 4,500 tweets and 8,300 reviews across different apps. Three of the authors classified these reviews and tweets manually (with the Kappa Blackman and Koval 2000 agreement's degree between 82% and 90%). We used this sample to evaluate our automated classification models applying 10-fold cross validation. The accuracy of Naive Bayes and SVM for tweets and reviews is shown in Table 1. In our subsequent content analysis, for the purpose of the high reliability of results, we only considered reviews and tweets that exhibited the same classification. That way, 72.3% of reviews and 51.8% of all tweets were finally considered.

To understand the content of reviews and tweets, we applied topic modeling only to the ones classified as *bug report* or *feature request* as demonstrated in Step ❺ of Fig. 4. We considered the other classes as non-informative (Chen et al. 2014) as they likely would not contribute to software development.

5.2.3 Topic modeling and crowdsourcing

We used Latent Dirichlet Allocation (LDA) to extract topics from app reviews and app related tweets (Step ❺ of Fig. 4). LDA (Blei et al. 2003) is an established method used to find topics in a set of natural language text documents. Topics are created when words that tend to appear together frequently are found in the documents of the corpus. LDA assumes that there is a fixed number of K topics. It assigns each document a probability distribution

Table 1 Average results of 10 time 10-fold cross validation for precision, recall, and F-score (F1) from applying Naive Bayes and SVM classifiers separately for tweets and reviews

Classifier	Feature request			Bug report		
	Precision	Recall	F1	Precision	Recall	F1
Reviews						
Naive Bayes	0.89	0.81	0.84	0.91	0.82	0.81
SVM	0.84	0.79	0.81	0.78	0.73	0.75
Tweets						
Naive Bayes	0.76	0.61	0.67	0.77	0.70	0.73
SVM	0.56	0.50	0.52	0.61	0.54	0.57

over topics. By looking into the words with heaviest weights, we can assign descriptive names to the probabilities assigned to words.

Because the number of topics existing for tweets and reviews is unknown and the content might be unrelated, we extracted topics from reviews and tweets separately. To decide which number of topics K makes the most sense, we varied K and decided based on (i) the perplexity measure, where low perplexity indicates better generalizability of a topic (Blei et al. 2003), and (ii) intuitive meaningfulness of the results (Chen et al. 2014). In this way, we extracted topics from both tweets and from reviews for each app. Across all 70 apps, the number of topics ranged from [0, 85] for reviews and [0, 170] for tweets.

To determine the consistency and inconsistency between any topics discussed in Twitter and Google Play, we measured cosine similarity which was applied successfully in this context before (Ramage et al. 2010). This is shown as Step ❻ of Fig. 4. Based on human judgment we determined a threshold value of 0.6. For any pair of tweet and review topic with cosine similarity above the threshold, we considered that topic as a *common topic between tweets and reviews*.

As the final step (Step ❼ of Fig. 4), to confirm the results of automated topic modelling and its subsequent similarity assignments, we performed an additional two-step evaluation process of human judgment (Chang et al. 2009). First, we asked human subjects if they see a similarity between any pair of topics or not. Second, we presented a tweet topic with no assigned review topic and asked if a human subject can find similarity with any other topics (we asked for one-by-one topic comparison for each tweet with no similar review topic). This evaluation was done by crowdsourcing. We asked the following two types of questions via Amazon Mechanical Turk (Paolacci et al. 2010):

Question Type 1: Is a tweet topic similar to its assigned review topic?
Question Type 2: Is a tweet topic, with no similar review topic, related to any other unassigned review topics?

The idea of relying on crowd-based judgment for validating the results of LDA was introduced by Chang et al. (2009). First, we randomly submitted 25% of data for calibrating similarity threshold. We calibrated the threshold of our cosine similarity measure to 0.6 as we found that any lower threshold, resulted in a substantially more inaccurate assignment of topics. Then, we randomly submitted 75% of all tweet and review topics across all apps to 658 crowd workers in a way that at least three workers judge the similarity and dissimilarity of "a tweet topic and a review". We considered the majority of votes among the workers as the final decision (democratic voting principle). A sample of questions asked in Amazon Mechanical Turk is presented in Fig. 5.

We found that our topic modeling has an accuracy of 57.1% *precision* and 51.5% *recall*. Crowd workers also determined similar reviews for unassigned topics for 31.3% of the cases. The precision and recall of our approach are slightly better in comparison to past studies on Twitter (Hong and Davison 2010). We finally in Step ❽ of Fig. 4, adjusted our topic assignment using crowd categorization to increase the accuracy of the results for **RQ3**.

5.3 RQ3: comparing sentiments, degree of specification and understandability of tweets and reviews

Having lemmatized texts of the tweets and reviews, we further compare their content with regards to the sentiment, degree of specification and degree of understandability of language and intention.

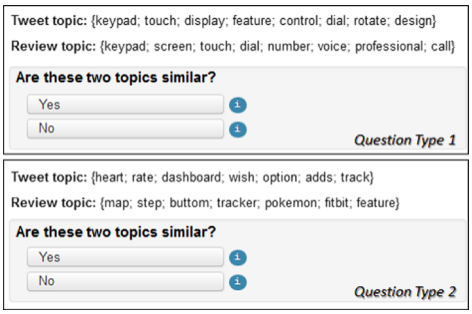


Fig. 5 Sample questions submitted to crowd workers for evaluation of topic modeling and similarity analysis. Each question was answered by at least three workers

5.3.1 Sentiment analysis

Sentiment analysis has been studied to characterize the attitude of people writing reviews and tweets (Guzman et al. 2017; Maalej and Nabil 2015). App reviews and tweets carry people's opinion, and opinions reflect people's sentiment. A user's opinion and sentiment could be positive or negative, defined as *polarity*. Tweets and reviews were studied for sentiments from this polarity aspect (Maalej and Nabil 2015; Martin et al. 2016). For instance, words like *"perfect"* and *"good"* are very positive while the word *"recommended"* is slightly positive. In addition, the sentiment might be factual or opinionated, outlined by subjectivity (Liu 2010). For instance, the word *"hope"* is subjective while the word *"scientific"* is objective. Sentiment analysis of tweets from both aspects of polarity and subjectivity is well established (Nielsen 2011). In this paper, we compared both polarity and subjectivity of app reviews and app related tweets. We performed this sentiment analysis to answer **RQ3**.

5.3.2 Crowdsourcing to evaluate the degree of specification and understandability

We used crowdsourcing to evaluate and adjust the results of automatic topic modeling. Further, in **RQ3** We compared the tweets and reviews based on:

- *Degree of specification*: to what extent the content of a tweet or review specifies a problem or request?
- *Understandability*: to what extend the content of the tweet or review is clear and understandable considering the language and intent?

To compare, we performed another round of crowdsourcing. We submitted all of the informative tweets and reviews to the crowd in Amazon Mechanical Turk and asked them to evaluate each tweet and each review separately on a five-point Likert scale. In the design of this evaluation, we provided descriptions and examples to familiarize the crowd with the intention. Each tweet and each review were evaluated at least by three workers based on these two criteria. Figure 6 shows sample questions submitted to the crowd.

Each review and each tweet were judged initially by three crowd workers. In case the majority agreed with a particular Likert category, the degree of specification or degree of understandability were decided accordingly. Otherwise, the task was resubmitted up to the time that at least two of the workers vote for the same category. 8.1% of the tweets and 11.3% of the reviews were evaluated by more than three crowd workers. In Appendix we discuss the process of crowdsourcing in our study.

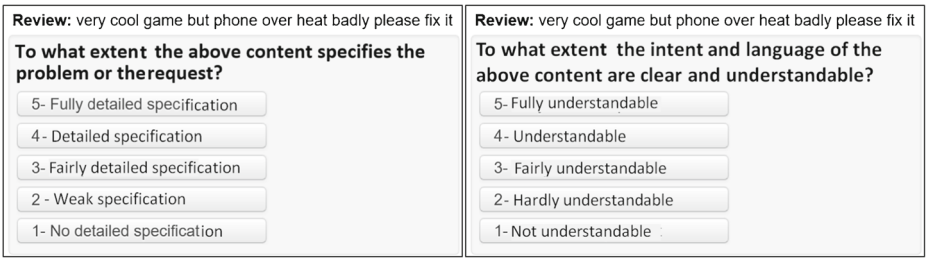


Fig. 6 Sample questions submitted to crowd workers for evaluating the *degree of specification* and *understandability* of tweets and reviews. Each question was answered by at least three workers

6 Results

Results are organized according to the three stated research questions and are presented in Sections 6.1, 6.2 and 6.3. Answering **RQ1** is based on the analysis of reviews and tweets of 30,793 apps. For comparing their content (**RQ2**) as well as for comparing sentiments and other characteristics of app reviews and tweets (**RQ3**), we randomly selected a sample of 70 apps. This sample size is comparable to similar investigations done in this context (Martin et al. 2016).

6.1 RQ1: when considered over a period of time, is there a correlation between the number of tweets and the number of reviews about mobile apps?

For a total of 30,793 apps, we calculated the correlation between the number of reviews and number of tweets when looking at them over a period of six weeks. This initial investigation was aimed to explore if the app related tweets are related in terms of amount and trend with reviews of the same app. We want to know for which apps the relation between a number of reviews and number of tweets is stronger. We performed this comparison for different top charts apps and for apps having combinations of high or low ratings with the high or low number of reviews (Martin et al. 2015). We tested the following null hypothesis using Pearson's correlation:

> H_0: Considered over time and per app, there is no correlation between number of user reviews and number of user tweets.

The null hypothesis was rejected for 98.6% of the apps. The relationship between the number of tweets and number of reviews for all top chart apps (separated by categories) is visualized in Fig. 7. The average Pearson r^2 effect size (with 0.1 being small, 0.3 being medium, and 0.5 considered of being large) of comparing tweets and reviews for *top free* apps was 0.46. Similarly, for *top paid* it was equal to 0.67, for top free games equal to 0.54, for top paid games equal to 0.81, for top grossing equal to 0.18, and for top grossing games equal to 0.23. We can see that apps in the top free category have a higher correlation in comparison to other top chart apps. *Top free games* have the second highest correlation compared to other top chart categories.

The results showed a very high correlation between apps outside top charts ($median = 1$ for all of non top chart apps). For most of the apps outside top charts, the high correlation appears as the app did not receive any review or any tweets over the six weeks of our analysis. In other words, the daily variance of tweets and reviews with six weeks of

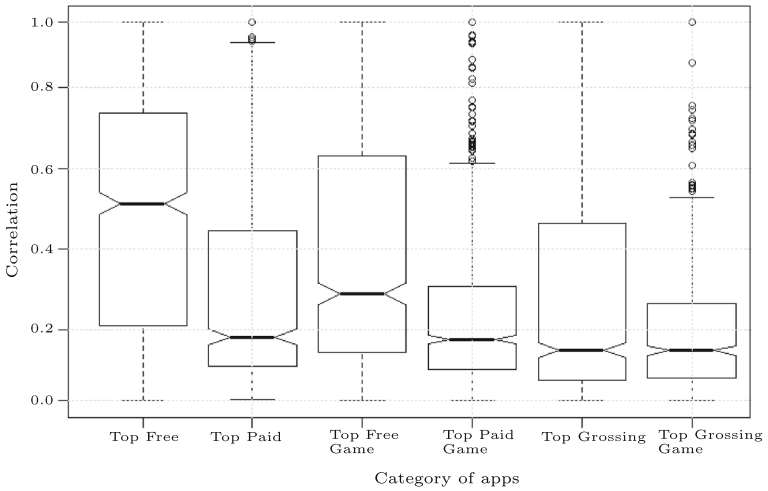


Fig. 7 Correlation between number of reviews and number of tweets for different top chart categories

our study is very low. We compared the distribution of the number of tweets and number of reviews for apps outside top charts. The results of Mann-Whitney test showed that the distribution of tweets and reviews for apps outside top chart are not significantly different ($p - value = 0.748$). As a result, we selected a random sample of apps for the next two questions considering the top chart mobile apps.

6.2 RQ2: what type of information can developers gain from tweet analysis?

In Fig. 8 the distribution of the number of feature requests, the number of bug reports and number of other types of tweets and reviews are compared between tweets and reviews. Looking into the category of informative tweets and reviews for the 70 apps, we found that 45.6% of the tweets are not reporting any bugs or requesting features in comparison to the 39.6% of all the informative reviews not reporting a bug or requesting a feature. However, the reviews are mainly bug reports (41.1%) while 36.9% of tweets are dedicated to this

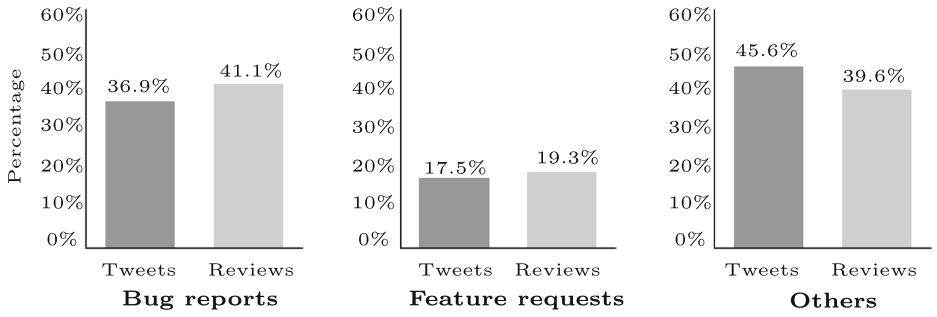


Fig. 8 Comparison of informative reviews and tweets for the 70 apps. The comparison shows the percentage of tweets and reviews in each category

category. We observed that 17.5% of informative tweets and 19.3% of informative reviews are *feature requests*.

We found that in 97.1% of the cases (68 out of 70 apps), mining tweets would provide complementary information to developers about users' feature requests and bug reports. In the rest of cases, we found more than 65% commonality between the feature requests and bug reports in Twitter and app store. Figure 9 compares the number of tweets and reviews from Twitter and app store for all the apps. Among all the 70 apps we analyzed, we found a total of 198 feature requests and 246 bug reports in the app related tweets that we did not find in the app store reviews. We observed that the feature requests in users' tweets have a wider scope than the ones articulated in app store reviews. Bug reports appear to be more detailed in reviews, often specifying the device and Android version that they catch a crash on it. Figure 9 shows the final results of our study on the content of the 70 apps. We found that on average across all the apps 40.78% of feature requests and bug reports are common between app related tweets and app store reviews.

Again, we provide more detailed information for one of the apps introduced in the motivation of the paper. For the Evernote app, we extracted 45 bug reports and 39 feature request from reviews as well as 35 bug reports and 47 feature request tweets. 14 bug reports and 11 features were in common between reviews and tweets. In other words, within six weeks of our analysis, we found 28 feature requests and 31 bug reports in Evernote tweets that were not stated in the app store reviews. Below we provide *a sample* of bug reports and feature requests unique in reviews and tweets and in common between both reviews and tweets for Evernote app:

App store reviews:

- *Need support for mathematical equations.*
- *Ability to hand-write on an inserted photo.*
- *In Jellybean, camera option is not working.*

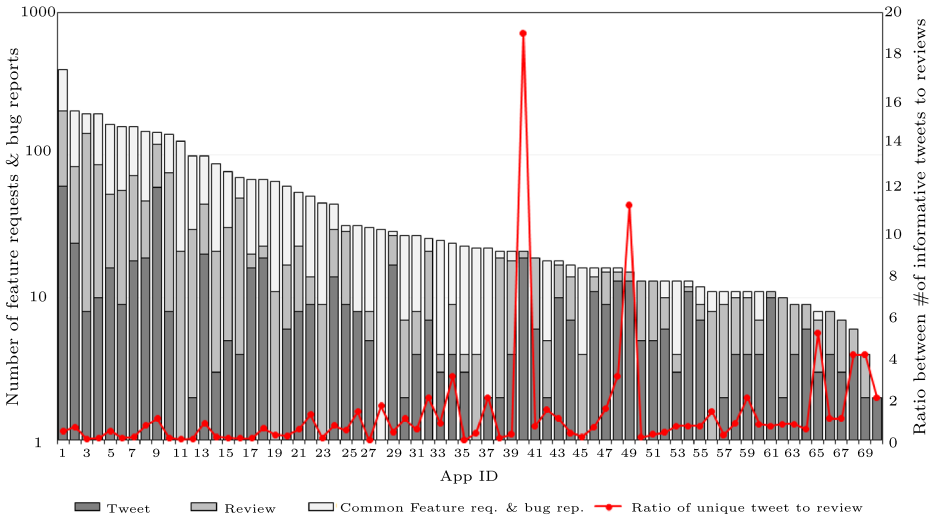


Fig. 9 Number of topics found after classification, topic modeling and crowdsourcing evaluation. Topics are grouped into (i) Twitter unique, (ii) Google Play unique, and (iii) common topics. The line chart on the second y-axis represents the percentage of tweet unique topics detected

– *It won't always allow me to check items off in a checklist.*

Tweets:

– *I need PIN Lock Feature on mobile.*
– *Rhyming feature designed for Songwriters.*
– *My smart watch app doesn't work when this is running*
– *It's slow as death. I type a letter a minute later it appears*

& **Mutual between tweets and reviews:**

– *Integration with SwiftKey keyboard.*
– *Add reminders to the shortcut section.*
– *Can't the camera widget zoom.*
– *Can't access the PDF normally when using the app.*

Considering all the tweets and reviews we analyzed, we mined 1,064 feature requests and 2,277 bug reports in total. Among them, 22.4% of feature requests came from Twitter *only* and 34.09% from app store reviews *only*. Also, 12.8% of bug reports were mined from Twitter *only* while 30.59% *only* came from the app store. 43.51% of feature requests and 56.61% of bug reports were in common between Twitter and Android app store. While in two cases (Apps #27 and #35 in Fig. 9) we could not gain extra information from mining tweets, in four apps with no informative reviews (Apps #26, #28, #37, and #70 in Fig. 9) we mined several feature requests and bug reports from tweets (see Fig. 9). We selected the diversified apps across top charts, and the results showed that we find valuable development information for the majority (97.1%) of these apps (including game applications) by mining Twitter.

We did not find any significant difference between paid and free apps in terms of ratio between #of informative tweets to reviews using Mann-Whitney test. We also did not find any significant difference between game and not game apps in terms of ratio between #of informative tweets to reviews.

6.3 RQ3: How do app store reviews and app related tweets compare with regards to sentiments, degree of specification and understandability?

We focus on 70 randomly selected apps for this analysis. Sentiment analysis was performed from both polarity and subjectivity perspectives. The distributions of review and tweet sentiments for the two sample apps (Pokemon Go and Evernote) mentioned at the beginning are given in Fig. 10. While we have fewer tweets than reviews for both apps, the scatter plots of review and tweet sentiment are largely the same. Both reviews and tweets are rather subjective ($\geq$ 0.5) and have slightly more positive ($\geq$ 0) sentiment. However, if we look closer into the density of data points in the middle charts of Fig. 10, Evernote app has more objective and positive tweets in comparison to its reviews.

To test the difference between review and tweet sentiment of all apps, we compared polarity and subjectivity of reviews and tweets by using the Mann-Whitney test. We tested the following null hypotheses:

$H_{0\_Polarity}$: There is no difference in the polarity distribution (over time) of tweets and reviews.

$H_{0\_Subjectivity}$: There is no difference in the subjectivity distribution (over time) of tweets and reviews.

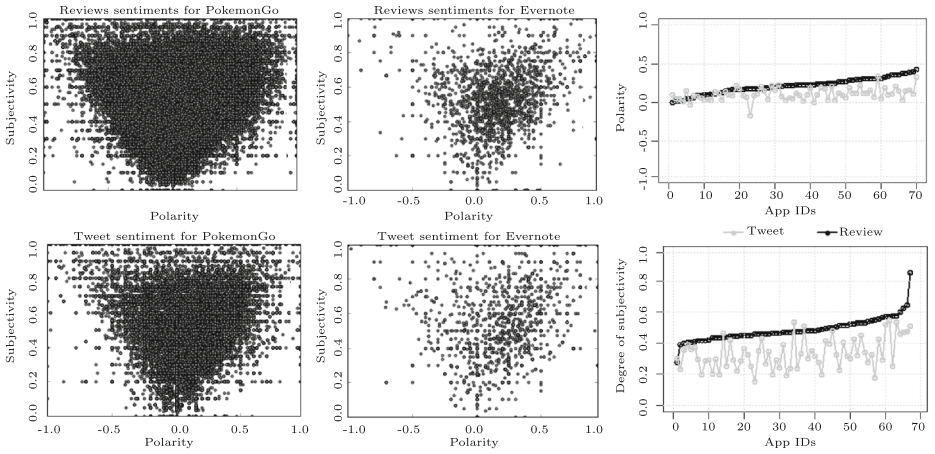


Fig. 10 First two columns are the comparison of polarity and subjectivity between tweets and reviews for Pokemon Go and Evernote reviews and tweets. The last column demonstrates the comparison of sentiments between tweets and reviews of an app - The most right side charts compare the polarity (top) and subjectivity (bottom) of tweets and reviews for 70 apps

Running the Mann-Whitney test, we rejected both null hypotheses $H_{0\_Polarity}$ ($p\text{-}value = 0.021$) and $H_{0\_Subjectivity}$ ($p\text{-}value = 0.004$). The degree of polarity and subjectivity for all apps is shown in the left column of Fig. 10. Looking into these charts for a particular app, tweets are usually less positive and less subjective in comparison to app store reviews. The most right charts of Fig. 11 show the difference between overall polarity and subjectivity of tweets and reviews.

We also compared the degree of understandability and degree of specification of all the informative reviews, and that of all informative tweets. The results of this comparison are presented in Fig. 11. The bigger portion of reviews was evaluated as a fully detailed specification in comparison to the tweets (32.3% versus 25.8%). However, comparing the least specific category, the bigger portion of reviews were judged as they have "no detailed specification" (24.1% vs. 19.6%). Looking into the reviews and tweets classified between three (detailed specification) to five (fully detailed specification), reviews are *slightly* more specific in defining the problem or the request as 66.3% of reviews was judged as having detailed specification or better while this includes 63.1% of the tweets. Also, our analysis did not demonstrate any significant difference between paid and free apps nor between game and non-game apps in terms of polarity and subjectivity of tweets in comparison to reviews.

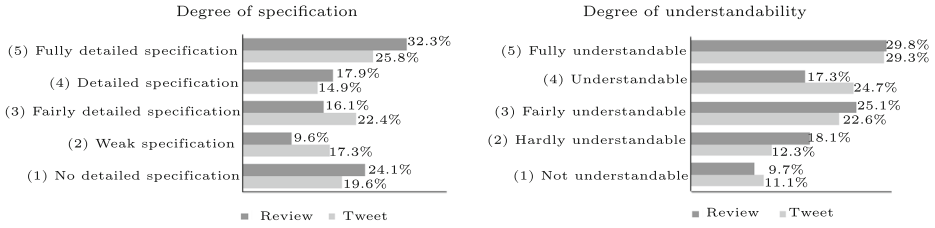


Fig. 11 Comparison between tweets and reviews based on the "degree of specification" and the "degree of understandability"

Looking into the degree of understandability and clarity of the intent and language, reviews and tweets are almost judged equally clear (29.8% vs 29.3%). On the other hand, more tweets considered not understandable at all (11.1%) in comparison to the reviews (9.7%). 72.2% of reviews and 76.6% of tweets were considered understandable, fairly understandable or fully understandable. We speculate that the length limitation of the tweets made the users communicate their intention right on to the point. Our analysis did not demonstrate any significant difference between paid and free apps nor between game and non-game apps in terms of degree of understandability and degree of specification for tweets in comparison to reviews.

7 Threats to validity

As with any study, there exist threats to the validity of our results. We discuss these threats following the categorization which is given by Wohlin et al. (2012):

Conclusion validity- Are we drawing a right conclusion about treatment and outcome relation?: The conclusions made were solely based on apps and related reviews and tweets taken from the Google Play (Android) app platform within six weeks period. App store reviews were collected from the Google Play app store only. Furthermore, tweets from Twitter were searched to specifically target tweets about the Google Play version of the apps. We excluded tweets that pertain to alternate stores through taking out tweets with words such as iOS, iPhone, etc. However, a tweet from a non-Android user might have occurred in our sample if the user did not use any of the mentioned keywords. We also analyzed apps within a limited time frame and compared tweets and reviews only for this period. However, a feature may have been requested or a bug reported in app store prior to this time frame and before the tweet occurred. We argue that 26.2% unique feature requests and bug reports on average over six weeks period is not trivial and not by accident. Also, 42.8% of the apps in our study have been released for the first time or at least had one release within the six weeks period. For more comprehensive analysis the evaluation of data further than six weeks period could be considered.

Taking intersection between two classifiers may potentially take out informative reviews and tweets and affect the conclusion. However, F1 for reviews is high by both classifiers (See Table 1). In addition, we performed a manual inspection on this subject for the 70 apps but did not find any such case which affects the results of the study.

Another threat refers to the trustworthiness of crowd evaluation. We used our experience in designing multiple crowdsourced surveys (Nayebi et al. 2017; Nayebi and Ruhe 2015) and adopted the guidelines of Kittur et al. (2008) for designing crowd surveys. To test the accuracy of the crowdsourcing in **RQ2**, we randomly selected 500 tweet and 500 review topics. Among them, 250 topics were marked as similar and 250 topics were marked as different by the crowd. We asked three app developers (with whom we have formerly worked with) to manually label these 500 topics. They answered the same questions as the crowd did (see Fig. 5). We compared the results and found that developers classified 99.4% of the topics in the same way as we decided based on the crowd evaluation. For 26% of the tasks we had at least one worker in disagreement with another worker and with the results of topic modeling.

In **RQ3**, we used crowdsourcing for defining the degree of specification and understandability. In addition, we also ran a small experiment to understand the degree of difference between surveying an unknown set of developers (crowd) and a set of known developers. We randomly selected 250 tweets and 250 reviews and asked three developers to identify

the degrees of specification and understandability. Comparing the results of crowd classification with the classification made by the developers showed a maximum difference of 3.4% for the degree of specification and a maximum difference of 3.6% for the degree of understandability.

These threats are considered non-critical for the type of investigation made in this paper. The investigations were targeted at a proof of the existence of additional information and insight. Restricting samples is not contradictory to that. It only means that we might have even missed information available.

Construct validity - Are we measuring the right things? Data for this study was collected over a six week time period. Thus, for apps that were released to the market in the middle of our data-collection cycle or occurred in a top chart some time in between, the collected data does not span the full six-weeks. However, we selected apps for content analysis considering different amounts of the ratio between # of tweets and # of reviews. In this way, if an app (such as Pokemon Go) occurred at the end of our third week we still included it in our study due to huge number of reviews and tweets.

We introduced two additional measures to increase construct validity. First, from performing two independent classifications of both reviews and tweets, we have a higher reliability that the content of the messaging is indeed related to the category it is assigned to (this is highly important for tweets). Second, from topic modeling and comparison of all topics extracted, we have increased the validity of comparing the actual content. When the majority of our crowd workers detect a mismatch or found a newly matched topic, we tuned the results of machine learning. However, we expect that by altering the topic modeling technique or defining a more fine-grained classification, the accuracy of the machine learning model would become higher.

The use of sentiment analysis tool might risk the results (Jongeling et al. 2015). This potentially may impact the results of **RQ3** in our study. We carefully analyzed this impact by comparing `Pattern` with `Stanford NLP` (Manning et al. 2014) and `Vader` (Hutto and Gilbert). Testing the $H_{0\_Polarity}$, using each of these libraries showed significant difference between reviews and tweets.

Internal Validity - Can we be sure that the treatment indeed caused the outcome? As we ensured that the reviews and tweets were explicitly related to the apps selected, the risk of violating internal validity appears to be low. Of course, number and sentiment of messages can be influenced by other factors. We would argue that the number of apps and messages studied exclude the random influence of this phenomenon. Despite the rigorous analysis there is still the risk that some of the existence of spam and irrelevant tweets remained in the sample.

To scope the extent of remaining spams in our data we randomly sampled 250 spam tweets and 250 of non-spam tweets, and we asked three developers to read the tweets and label them as spam/not spam. Then, we compared the results between developers' tagged spams and filters we described in Section 4. The results showed that developers detected 257 non-spam tweets and we filtered out tweets that were not spam, however; the difference is about 2.8%. We believe that this is sufficiently small to confirm our stated results.

External Validity - Can the results be generalized beyond the scope of this study? All the investigated apps and related reviews and tweets were randomly taken from Google Play (Android). A generalization of results (getting additional insight) to other social media is outside consideration (even though considered to be likely). A generalization to Twitter seems possible as we showed already for the given sample set of apps that substantial additional insights can be gained. We also observed that the number of tweets and reviews for the majority of apps outside the top chart is low over the considered period of six weeks.

Apps must possess a certain degree of visibility and popularity for users to review and tweet about the app. Thus, developers of apps with little or even no client-base will likely find it difficult to utilize tweets to support their development process.

8 Discussion

We selected the taxonomy suggested by Maalej and Nabil (2015) for the purpose of this study. Maalej and Nabil categories each review into *feature request*, *bug report*, *user experience*, or *other*. We used this high-level taxonomy as a more fine-grained taxonomy is either based toward tweets (Guzman et al. 2017) or reviews (Di Sorbo et al. 2017; Ciurumelea et al. 2017). In this section we discuss why the use of the existing finer grained taxonomies does not suit the purpose of this study.

We compared the review taxonomies suggested by Di Sorbo et al. (2016, 2017), and Ciurumelea et al. (2017) and the tweet taxonomy suggested by Guzman et al. (2017). We took two arguments (called Argument 1 resp. 2):

Argument 1: We compared the tweet taxonomy (Guzman et al. 2017) of software products with app reviews (Ciurumelea et al. 2017).
Argument 2: We analyzed 7,233 Tweets using SURF as suggested by the reviewer. This sample covers all the tweets of Evernote, Dropbox, and FaceTune.

Argument 1- Two of the authors separately compared the taxonomy of tweets provided by Guzman et al. (2017) with the taxonomy of reviews reported by Di Sorbo et al. (2016, 2017) and by Ciurumelea et al. (2017):

- **Comparing tweet taxonomy** (Guzman et al. 2017) **with review taxonomy by Di Sorbo et al. (2017):** We could find only one commonality between these two taxonomies which is "pricing". While tweet taxonomy has separate "Feature shortcoming", "Feature strength", "Feature requests", review taxonomy by Di Sorbo et al. has "feature and functionality". Also, review taxonomy by Di Sorbo et al. suggest "improvement" which could be either "bug report" or "feature request" considering the tweet taxonomy. The rest; 21 categories of tweet taxonomy (80.7%) and nine categories (75%) of Di Sorbo et al.'s review taxonomy; are not explicitly associated.
- **Comparing tweet taxonomy** (Guzman et al. 2017) **with review taxonomy by Ciurumelea et al. (2017):** We could map 2 of the categories "hardware" and "pricing". We did not find any explicit association between the rest of the taxonomies (92.3% of tweet taxonomy and 83.3% of Ciurumelea et al. review taxonomy).
- **Comparing review taxonomy by Di Sorbo et al. (2017) with review taxonomy by Ciurumelea et al. (2017):** We found that there exists a wide array of differences in the taxonomies. Comparing review taxonomies suggested by Di Sorbo et al. and Ciurumelea et al. we only found three categories "app", "GUI" and "Pricing" common between two taxonomies (25% compatibility).

As the result of this inconsistency we used a higher level taxonomy to observe if extra information exists in app relate tweets or not. Future studies should further focus on the taxonomy that can fit both reviews and tweets of mobile applications.

Argument 2- We used SURF tool which was designed based on Di Sorbo et al. (2017) taxonomy to categorize in total 7,233 Tweets. We analyzed the accuracy of the results manually. The tweets were compiled into a compatible XML file, and the SURF script

Table 2 Results of classifying tweets with a tool and taxonomy provided by Di Sorbo et al. (2017) for reviews

SURF Category	Total # of Tweets	# of Correctly Classified Tweets	% of misclassified Tweets
Evernote app			
App	121	55	54.5%
Company	4	0	100.0%
Contents	41	22	46.3%
Download	13	2	84.6%
Feature/Functionality	140	61	56.4%
GUI	29	14	51.7%
Improvement	15	9	33.3%
Model	54	28	48.1%
Pricing	26	15	42.3%
Resources	1	1	0.0%
Security	14	2	85.7%
UpdateVersion	19	11	42.1%
DropBox app			
App	121	44	63.6%
Company	17	4	76.5%
Contents	26	5	80.8%
Download	10	0	100.0%
Feature/Functionality	157	50	68.2%
GUI	40	15	62.5%
Improvement	13	1	92.3%
Model	52	22	57.7%
Pricing	12	4	66.7%
Resources	8	0	100.0%
Security	13	2	84.6%
UpdateVersion	15	9	40.0%
FaceTune app			
App	20	4	80.0%
Company	1	0	100.0%
Contents	5	1	80.0%
Download	4	1	75.0%
Feature/Functionality	19	5	73.7%
GUI	16	4	75.0%
Improvements	1	0	100.0%
Model	7	1	85.7%
Pricing	11	2	81.8%
Resources	0	0	0
Security	1	0	100.0%
UpdateVersion	0	0	0

was run to produce the output XML file. Afterward, we assigned a True/False classification for the assignment of the category done by SURF and made a holistic judgment regarding the validity of its use. We found that (i) SURF categorized 6,187 of the tweets we considered informative as irrelevant and ignored them, and (ii) For the rest of the 1,046 tweets, SURF misclassified 62.3% of tweets as it was tailored and tuned for reviews (and not tweets). Table 2 shows the status of what we found through utilizing SURF with tweets from the Evernote, Dropbox, and Facetune apps.

We often found that the tweets do not belong to any of these categories. A prominent example of this is the pay-wall that Evernote implemented that disables users from having more than two accounts without a paid account. SURF categorized most of the complaints regarding the pay-wall as a bug while it is originally a request to get back a feature. The results showed that development information in tweets is different from the one obtained from reviews. Potentially, an app developer can benefit from looking into sources of users feedback outside app stores. While at this stage we could not generalize the difference between the information inside and outside the app stores, it appears that there exist useful and complementary user reviews outside the app stores that helps in the evolution process of an app.

Also, analyzing users who prefer tweeting to writing app reviews and vice versa could help in understanding users' need better. We analyzed the usernames related to tweets and reviews of the 70 apps by adopting the heuristics suggested by Wiese et al. (2016) and found 181 similar usernames (screen names) between Twitter and Google Play. We then manually looked into the Twitter profile and Google Play of users in case they were publicly available. We also checked their personal website or `LinkedIn` profiles manually in case available. Among these 181 users:

- We could not verify the similarity for 103 users: In these cases, two of the fields "name", "last name", "gender", "age", "occupation', or "location of the person" was totally different between profiles. So we can not say the person on Twitter was the same one on Google Play.
- We could not access the profiles of 74 users: In this case, the user has either deleted or protected her account in Google Play or Twitter. So, we could not check the identity of the user. This is mainly happening as we do this analysis retrospectively and after some delay from the initial study.
- We could verify that four users have been commented on both Google Play and Twitter: among these, one user has commented on the same app and three users have been commented on different apps in Google Play and Twitter.

While our initial analysis of users did not show a significant overlap between users of Twitter and Google Play, we do not know if this happens because of the performance of existing algorithms or if indeed there is no user in common between the two platforms.

9 Conclusions and future work

Understanding and fulfilling users' need is the ultimate goal of app software development which is mainly achieved by providing new features, revising features, fixing bugs, and improving the quality of users' experience with the software. In support of this development and for prioritizing change requests, an increasing number of research studies and businesses have been formed around mining app reviews. The key idea of this paper is to

improve this support by looking "beyond the fence". We analyzed the possibility to combine information sources for supporting development decisions. The analysis was done in a three-staged process: (i) studying the alignment between number of tweets and reviews over time, (ii) comparing the content of tweets and reviews with regards to development information using a variety of machine learning techniques and crowdsourcing, and (iii) similarity and differences between user sentiments on app store reviews and tweets as well as understandability and degree of problem or request specification.

The results of this study imply that the research community should look into the additional sources of information while performing empirical studies on users feedback for app evolution. Our results also showed that App developers could find more information about users' requests by mining social media. Comparing developers' perception along with the actual evidence, we found as well that evaluating the impact of reviews on release decisions for mobile app evolution is a possible future direction of research. The same is true for comparing the posts of users within Twitter and within app store to understand the characteristics of media and content of requested changes.

Acknowledgments We would like to thank Homayoon Farrahi and Ada Lee for their help on this study. We thank all the anonymous reviewers and the Associate editor for their valuable comments and suggestions. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant 250343-12.

Appendix: Crowdsourced evaluation of RQ2 and RQ3

In Section 5 we discussed why and how we used crowdsourcing:

First, we used crowdsourcing to confirm the results of similarity analysis (cosine similarity) between tweet topics and review topics to confirm (i) if assigning a tweet topic to a review topic is correct and (ii) if we missed to assign a tweet topic to a review topic (**RQ2**).

Second, we used crowdsourcing to compare the degree of specification and understandability (**RQ3**).

"The crowd" is composed of workers that are unknown in person to the authors. To elaborate on the validity of crowdsourcing results, we hired three developers being known to the authors from similar former work. We asked them to perform the same tasks done by the crowd. Overall across **RQ2** and **RQ3**, the average Fleiss Kappa among three developers and across different tasks was 0.84, which indicates an almost perfect agreement. We then compared the results that were achieved by the crowd with the results achieved by the three developers.

Evaluating crowdsorced results in RQ2

We randomly selected 500 tweets and 500 review topics. Among them, 250 topics were marked as similar, and 250 topics were marked as different by the crowd. We asked three

Table 3 Comparison of crowdsourcing with known developers results for **RQ2**

Taks	Crowd	Developers
Similar review and tweet topics	250	250
Different review and tweet topics	250	247

app developers we have formerly worked with to manually label these 500 topics. They answered the same questions as the crowd (Fig. 5). We compared the results and found that our developers classified 99.4% of the topics in the same way as we did base on the crowd's evaluation.

We then compared the results that achieved by the crowd with the results achieved by the three developers as presented in Table 3.

Evaluating crowdsorced results in RQ3
We randomly selected 250 tweets and 250 reviews and asked three developers to judge the degree of specification both for degree of specification and degree of undrestandability. We compared the results of this task as it was done by crowd versus the three developers in Tables 4 and 5.

17.2% of reviews and 16.4% of tweets were classified in a different specification category by developers in comparison with the crowd. With the same set up, we asked developers to evaluate the degree of understandability and compared the results with the ones received from the crowd:

11.2% of the reviews and 17.6% of the tweets were classified in a different undrestandability category by developers in comparison with the crowd.

Table 4 Comparison of crowdsourced results with the results from known developers in **RQ3** for degree of specification

	Crowd		Developers	
Taks	Review	Tweet	Review	Tweet
Fully detailed specification	65	51	59	53
Detailed specification	82	74	88	66
Fairly detailed specification	58	76	60	80
Weak specification	37	28	33	30
No detailed specification	8	21	21	10

Table 5 Comparison of crowdsourced results with the results from known developers in **RQ3** for undrestandability

	Crowd		Developers	
Taks	Review	Tweet	Review	Tweet
Fully understandable	38	31	44	35
Understandable	71	65	64	67
Fairly understandable	93	96	88	101
Hardly understandable	26	25	30	34
Not understandable	22	33	24	13

References

Agarwal A, Xie B, Vovsha I, Rambow O, Passonneau R (2011) Sentiment analysis of twitter data. In: Proceedings of the workshop on languages in social media. Association for Computational Linguistics, pp 30–38

Benevenuto F, Magno G, Rodrigues T, Almeida V (2010) Detecting spammers on twitter. In: Collaboration, electronic messaging, anti-abuse and spam conference (CEAS), vol 6, pp 12

Blackman NJ-M, Koval JJ (2000) Interval estimation for cohen's kappa as a measure of agreement. Statist Med 19(5):723–741

Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022

Bougie G, Starke J, Storey M-A, German DM (2011) Towards understanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions. In: Proceedings of the 2nd international workshop on Web 2.0 for software engineering. ACM, pp 31–36

Chang J, Gerrish S, Wang C, Boyd-Graber JL, Blei DM (2009) Reading tea leaves: How humans interpret topic models. In: Advances in neural information processing systems, pp 288–296

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

Ciurumelea A, Schaufelbühl A, Panichella S, Gall HC (2017) Analyzing reviews and code of mobile apps for better release planning. In: Software analysis, evolution and reengineering (SANER). IEEE, pp 91–102

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

Di Sorbo A, Panichella S, Alexandru CV, Visaggio CA, Canfora G (2017) Surf: summarizer of user reviews feedback. In: Proceedings of the 39th international conference on software engineering companion. IEEE Press, pp 55–58

Gibbons JD, Chakraborti S (2011) Nonparametric statistical inference. Springer

Gomez M, Martinez M, Monperrus M, Rouvoy R (2015) When app stores listen to the crowd to fight bugs in the wild. In: Proceedings of the 37th international conference on software engineering (ICSE), vol 2. IEEE Press, pp 567–570

Gu X, Kim S (2015) What parts of your apps are loved by users? (t). In: Automated software engineering (ASE). IEEE, pp 760–770

Guzman E, Alkadhi R, Seyff N (2017) An exploratory study of twitter messages about software applications. Requir Eng 22(3):387–412

Guzman E, Ibrahim M, Glinz M (2017) Mining twitter messages for software evolution. In: Proceedings of the 39th international conference on software engineering companion. IEEE Press, pp 283–284

Harman M, Jia Y, Zhang Y (2012) App store mining and analysis: Msr for app stores. In: Proceedings of the 9th IEEE working conference on mining software repositories. IEEE Press, pp 108–111

Hong L, Davison BD (2010) Empirical study of topic modeling in twitter. In: Proceedings of the first workshop on social media analytics. ACM, pp 80–88

Hutto CJ, Gilbert E Vader: a parsimonious rule-based model for sentiment analysis of social media text. In: Eighth international AAAI conference on weblogs and social media, pp 50–60

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

Jivani AG et al (2011) A comparative study of stemming algorithms. Int J Comp Tech Appl 2(6):1930–1938

Jongeling R, Datta S, Serebrenik A (2015) Choosing your weapons: on sentiment analysis tools for software engineering research. In: Software maintenance and evolution (ICSME). IEEE, pp 531–535

Kittur A, Chi EH, Suh B (2008) Crowdsourcing user studies with mechanical turk. In: Proceedings of the SIGCHI conference on human factors in computing systems. ACM, pp 453–456

Kouloumpis E, Wilson T, Moore JD (2011) Twitter sentiment The good the bad and the omg! Icwsm 11:538–541

Liu B (2010) Sentiment analysis and subjectivity. Handbook Natural Lang Process 2:627–666

Loper E, Bird S (2002) Nltk: the natural language toolkit. In: Proceedings of the ACL-02 workshop on effective tools and methodologies for teaching natural language processing and computational linguistics, vol 1. Association for Computational Linguistics, pp 63–70

Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? On automatically classifying app reviews. In: IEEE 23rd international requirements engineering conference (RE). IEEE, pp 116–125

Maalej W, Nayebi M, Johann T, Ruhe G (2016) Toward data-driven requirements engineering. IEEE Softw 33(1):48–54

Manning CD, Surdeanu M, Bauer J, Finkel JR, Bethard S, McClosky D (2014) The stanford corenlp natural language processing toolkit. In: ACL (System Demonstrations), pp 55–60

Martin W, Harman M, Jia Y, Sarro F, Zhang Y (2015) The app sampling problem for app store mining. In: IEEE/ACM 12th Working conference on mining software repositories. IEEE, pp 123–133

Martin W, Sarro F, Jia Y, Zhang Y, Harman M (2016) A survey of app store analysis for software engineering. RN 16:02

Mohammad SM, Kiritchenko S, Zhu X (2013) Nrc-canada: building the state-of-the-art in sentiment analysis of tweets. arXiv:1308.6242

Naaman M, Boase J, Lai C-H (2010) Is it really about me?: message content in social awareness streams. In: Proceedings of the 2010 ACM conference on computer supported cooperative work. ACM, pp 189–192

Nayebi M, Ruhe G (2015) Analytical product release planning. In: The Art and science of analyzing software data. Morgan Kaufmann, pp 550–580

Nayebi M, Adams B, Ruhe G (2016) Release practices for mobile apps–what do users and developers think? In: Software analysis, evolution, and reengineering (SANER), vol 1. IEEE, pp 552–562

Nayebi M, Farrahi H, Ruhe G, Cho H (2017) App store mining is not enough. In: Proceedings of the 39th international conference on software engineering companion. IEEE Press, pp 152–154

Nayebi M, Quapp R, Ruhe G, Marbouti M, Maurer F (2017) Crowdsourced exploration of mobile app features: a case study of the fort mcmurray wildfire. In: Proceedings of the 39th international conference on software engineering: software engineering in society track. IEEE Press, pp 57–66

Nielsen FÅ (2011) A new anew: Evaluation of a word list for sentiment analysis in microblogs. arXiv:1103.2903

O'Connor B, Krieger M, Ahn D (2010) Tweetmotif: exploratory search and topic summarization for twitter. In: ICWSM

Palomba F, Linares-Vásquez M, Bavota G, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2015) User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In: Software maintenance and evolution (ICSME). IEEE, pp 291–300

Paolacci G, Chandler J, Ipeirotis PG (2010) Running experiments on amazon mechanical turk. Judgment Decis Making 5(5):411–419

Porter MF (1980) An algorithm for suffix stripping. Program 14(3):130–137

Prasetyo PK, Lo D, Achananuparp P, Tian Y, Lim E-P (2012) Automatic classification of software related microblogs. In: Software maintenance (ICSM). IEEE, pp 596–599

Ramage D, Dumais ST, Liebling DJ (2010) Characterizing microblogs with topic models. ICWSM 10:1–1

Rosenthal S, Nakov P, Kiritchenko S, Mohammad SM, Ritter A, Stoyanov V (2015) Semeval-2015 task 10: sentiment analysis in twitter. In: Proceedings of SemEval-2015

Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv (CSUR) 34(1):1–47

Smedt TD, Daelemans W (2012) Pattern for python. J Mach Learn Res 13:2063–2067

Thelwall M, Buckley K, Paltoglou G (2011) Sentiment in twitter events. J Am Soc Inf Sci Technol 62(2):406–418

Tian Y, Lo D (2014) An exploratory study on software microblogger behaviors. In: 2014 IEEE 4th Workshop on mining unstructured data (MUD). IEEE, pp 1–5

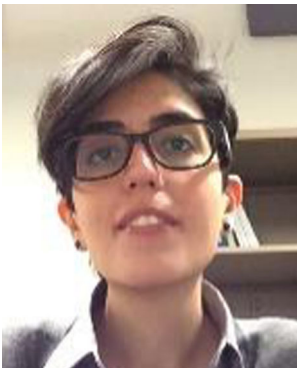
Villarroel L, Bavota G, Russo B, Oliveto R, Di Penta M (2016) Release planning of mobile apps based on user reviews. In: Proceedings of the 38th international conference on software engineering. ACM, pp 14–24

Wang X, Kuzmickaja I, Abrahamsson P (2014) Microblogging in open source software development, 8–12

Wiese IS, da Silva JT, Steinmacher I, Treude C, Gerosa MA (2016) Who is who in the mailing list? Comparing six disambiguation heuristics to identify multiple addresses of a participant. In: 2016 IEEE International conference on software maintenance and evolution (ICSME). IEEE, pp 345–355

Williams G, Mahmoud A (2017) Mining twitter data for a more responsive software engineering process. In: Proceedings of the 39th international conference on software engineering companion. IEEE Press, pp 280–282

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer Science & Business Media



Maleknaz Nayebi is currently a Post Doctoral fellow at the University of Toronto. Maleknaz got her PhD at the University of Calgary's Laboratory for Software Engineering Decision Support. Her research interests include app store mining, mining patterns from data, mining software repositories, release engineering, open innovation, and empirical software engineering. Nayebi received a BSc in computer science from Tehran Polytechnic. She's a member of IEEE and ACM.

Henry Cho is currently an undergraduate student at the University of Toronto pursuing a B.A.Sc. in Engineering Science. Henry worked at the University of Calgary's Laboratory for Software Engineering Decision Support as research assistant. His research interests lie in the fields of big data mining, machine learning and empirical software engineering. He is a student member of IEEE, and the electronics chair of IEEE University of Toronto.

Guenther Ruhe is the Industrial Research Chair in Software Engineering at the University of Calgary. His research focuses on product release planning, software project management, decision support, data analytics, empirical software engineering, and search-based software engineering. Ruhe received a habilitation in computer science from the University of Kaiserslautern. He's the editor in chief of Information and Software Technology. He's a senior member of IEEE and a member of ACM.