2015 30th IEEE/ACM International Conference on Automated Software Engineering

Tool Support for Analyzing Mobile App Reviews

Phong Minh Vu, Hung Viet Pham, Tam The Nguyen, Tung Thanh Nguyen
Computer Science Department
Utah State University
{phong.vu, hung.pham, tam.nguyen}@aggiemail.usu.edu
tung.nguyen@usu.edu

***Abstract*—Mobile app reviews often contain useful user opinions for app developers. However, manual analysis of those reviews is challenging due to their large volume and noisy-nature. This paper introduces MARK, a supporting tool for review analysis of mobile apps. With MARK, an analyst can describe her interests of one or more apps via a set of keywords. MARK then lists the reviews most relevant to those keywords for further analyses. It can also draw the trends over time of the selected keywords, which might help the analyst to detect sudden changes in the related user reviews. To help the analyst describe her interests more effectively, MARK can automatically extract and rank the keywords by their associations with negative reviews, divide a large set of keywords into more cohesive subgroups, or expand a small set into a broader one.**

Keywords*—*App Review, Opinion Mining, Keyword

I. Introduction

Online reviews of mobile apps often contain useful user opinions, e.g. complaints or suggestions, which app developers can address to improve user satisfaction. However, analyzing those reviews manually for such opinions is challenging due to their large volume and noisy-nature. For example, a popular app like Facebook often gets thousands of reviews each day. In addition, a prior research reports that more than 60% of reviews do not contain useful opinions [1].

We have developed MARK (*Mining and Analyzing Reviews by Keywords*), a semi-automated tool to support the analysis of apps' reviews. As its fullname suggests, MARK operates based on keywords. That is, when using MARK, an analyst can describes his interests in one or more apps via a set of keywords. MARK then lists the reviews most relevant to those keywords for further analysis. It can also draw the trends over time of those selected keywords, which might help the analyst to detect sudden changes in the related user reviews.

Figure 1 illustrates MARK's internal architecture and processing pipeline. First, it crawls raw reviews from designated app stores[1] and only keeps reviews written in English. Then it extracts keywords from those reviews and pre-processes them. Because apps reviews often have a lot of typos, slangs, acronyms, and abbreviations, MARK stems and sanitizes those keywords with customized stemming rules and dictionaries. It also learns vector-based representation of keywords using word2vec [2], which will be used to calculate word similarity. The resulted keyword data is stored in a common database for future analysis tasks.

[1]The current version of MARK works with Google Play. We plan to connect other online app stores in the future.

The next three components of MARK are for keyword recommendation which help analysts describe their interests more effectively. MARK ranks keywords based on review ratings, i.e. top-ranked keywords are the ones occur most frequently in negative reviews. With that ranking, the analysts can focus on the issues the users complain or dislike. If the analyst already has a few keywords to start with, MARK can recommend additional similar keywords. In contrast, if the chosen keywords are too broad, MARK can narrow them down by dividing them into smaller, more cohesive subsets. Both of these functions are based on the semantic similarity of the keywords, which MARK calculates based on their vector-based representation learned with word2vec.

MARK's last two functions work when a set of keywords is chosen. Using the standard Vector Space Model and *tf.idf* term weighting scheme [3], MARK computes the relevance between each review and the chosen keyword set before reporting to analysts the most relevant reviews. It can also compute the trends of those keywords overtime (based on their occurrence frequencies in the reviews and their simple moving averages). MARK plots those trends and marks the sudden changes, which often indicate when users report wide-spread errors or major issues [4].

MARK is developed and deployed as a web-based tool. Readers interested in MARK are encouraged to access it online at http://useal.cs.usu.edu/mark and send feedback to ususeal@gmail.com. The full technical details of MARK are presented in [5]. In the rest of this paper, we will demonstrate the usefulness of MARK via two usage scenarios.

II. Running Scenarios

A. Facebook Messenger

Let us demonstrate MARK via the following example. Assume that we interested in negative user opinions about Facebook Messenger (one of the most popular apps on Google Play with hundreds of millions of users). Figure 2 shows the Launching screen, where we could choose this app for our analysis. To do that, we type in the word Messenger and apps with names most similar to that word are listed in the App Selection screen shown in Figure 3. This screen also shows basic information about the apps such as description, average ratings, common keywords, etc. We then can click on the link of Facebook Messenger to add it to the list of selected apps showed in the right and click on the Analyze button to go the next screen, the General Analyses screen.

Initially, we are interested in all aspects of this app that get negative opinions. Thus, MARK lists all potential keywords

978-1-5090-0025-8/15 $31.00 © 2015 IEEE
DOI 10.1109/ASE.2015.101

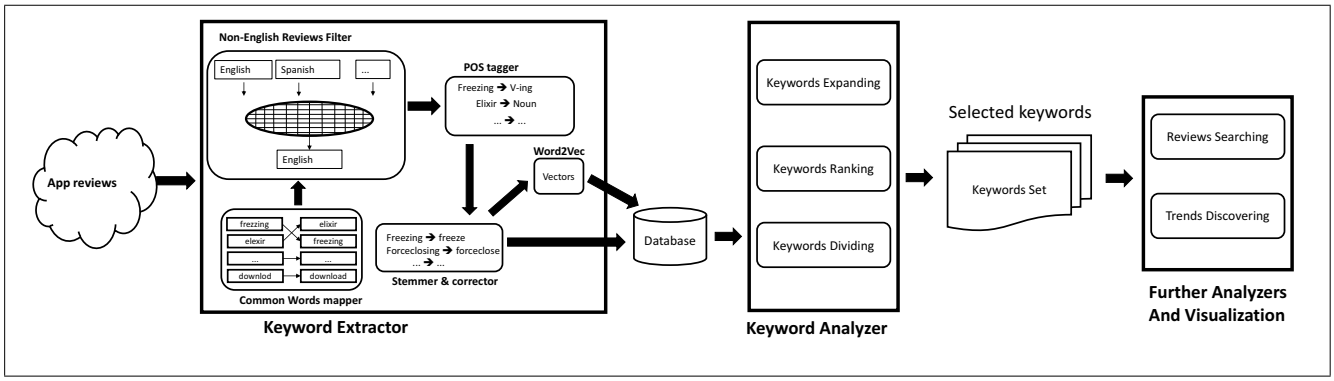

Fig. 1: System overview

Fig. 2: Launching screen for app selection

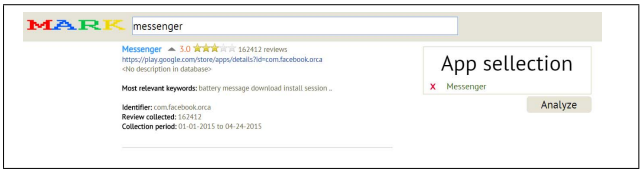

Fig. 3: Selecting Facebook Messenger for analysis

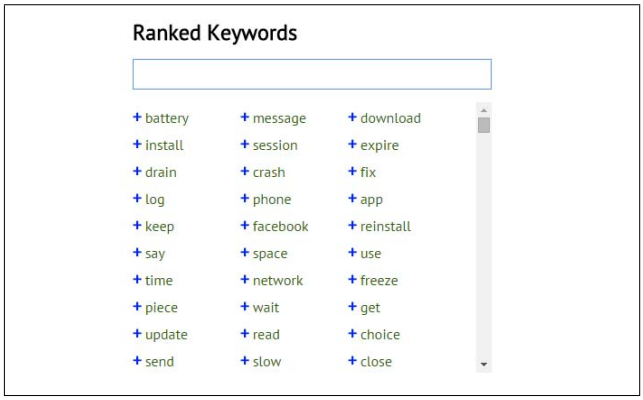

Fig. 4: Ranked negative keywords for Facebook Messenger

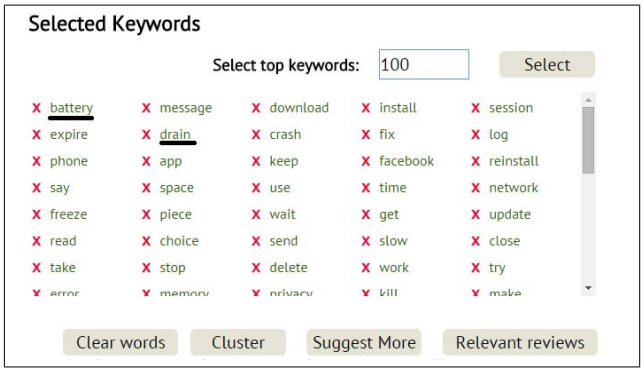

Fig. 5: Selection of top 100 keywords

from raw reviews of Facebook Messenger and ranks those keywords based on their associations with negative ratings (e.g. top-ranked words like update or login occur most frequent in 1 or 2-star reviews) like in Figure 4. Because the list contains all possible keywords, to narrow down our analysis, we select the top 100 (i.e. most negative) out of it. Figure 5 shows the listed and selected keywords.

As seen in the figure, several selected keywords are related and indicate a more general concern/issue. For example both keywords crash and freeze could be used to describe the app's status when an *"unrecoverable error"* occurs. Or, battery and drain often go together to describe the bad *"energy consumption"* of the app. Therefore, we use the Cluster function of MARK to divide the 100 selected keywords into smaller groups, each potentially for a more general concern. Figure 6 shows the clustering results produced for Facebook Messenger.

This clustering task is based on Word2Vec, a distributed, vector-based representation of words [2]. Word2Vec represents each word in a vocabulary as a high dimensional vector learned from a large corpus of text. Words having similar or related syntactic roles or semantic meanings often have similar vectors. Thus, MARK divides a keyword set into smaller subsets of related ones by applying K-mean [6], a similarity-based clustering algorithm on their vectors. It should be noted that, because K-mean algorithm initializes its clusters randomly, the clustering results might be slightly different between runs of the same analysis!

Browsing the clusters, we find one containing keywords like battery and drain, i.e. possibly related to the user opinions about *"energy consumption"*. We select this cluster and remove some keywords that seem to be non-related like data and phone. We suspect that users might use some other keywords

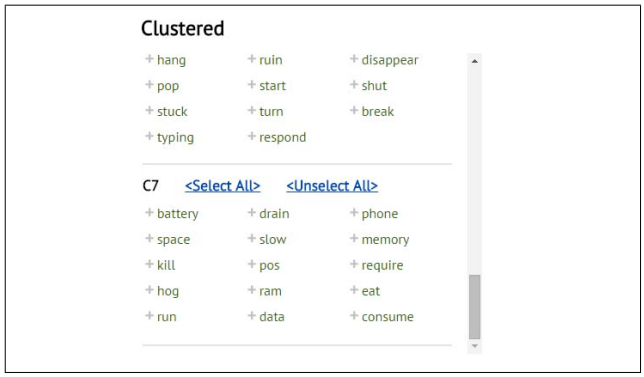

Fig. 6: Clustered keywords for Facebook Messenger

Fig. 7: Suggested keywords for Facebook Messenger

for this topic, thus, we ask MARK to suggest more (which is also performed based on the vector-based similarity of keywords). Figure 7 shows the suggested keywords, containing new ones like usage, deplete, and consumption.

Once those keywords are selected, MARK visualizes the trends of their occurrences overtime which can be analyzed for abnormal patterns. Figure 8 shows the trends for the keywords related to *"energy consumption"*. We could see an unusual peak in occurrences of those keywords in Feb 2015, which was after the release of a new version of Facebook Messenger. Prior works [1], [4] suggest that those sudden changes often occur when a newly released version of an app contains some defects or issues that make many users unsatisfied. To detect such abnormalities, MARK analyzes the keyword occurrence counts as a time-series, computes its simple moving average (SMA) and the differences between actual counts and SMA values. If the difference value is significant higher (e.g. two times) than the standard deviation of those SMA values, it indicates a sudden change in the corresponding occurrence counts.

We investigated further into this observation by asking MARK to query its review database and to return reviews created in the selected time (Feb 2015) which are most relevant to the selected keywords. This querying task is based on the standard Vector Space Model. That is, MARK applies the *tf.idf* weighting scheme on the keywords and measures the relevance between the selected keywords to a review as the cosine similarity of their *tf.idf* feature vectors. To help analysts reading the reviews more effectively, in the Review Search screen, MARK allows users to sorts the reviews by relevance (i.e. most related to keywords first), by time (e.g. most recent reviews first), or by rating (e.g. most negatively rated first). Users can also filter the listed reviews by their ratings (e.g. showing only 1-star reviews) or by using full-text search.

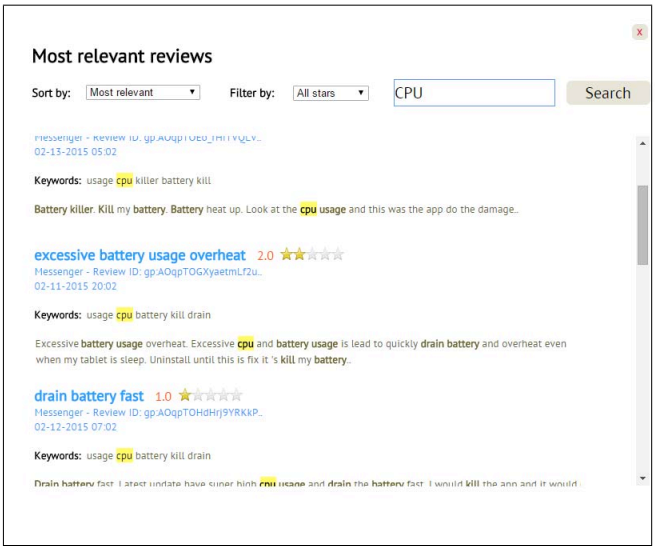

Fig. 9: Reviews of keywords for "energy consumption" in Facebook Messenger

Excessive **CPU** and **battery usage** is leading to quickly **drained battery** and overheating even when my tablet is sleeping. Uninstalling until this is fixed; it's killing my **battery**.

Fig. 10: A negative review on "energy consumption" for Facebook Messenger

Figure 9 shows the reviews listed in the Review Search screen. As seen, those reviews contain (mostly negative) user opinions about the *energy consumption* aspect of this app. Figure 10 lists one among them. In this review, the user complains that this app drains his tablet's battery and makes it overheated. The issue is so severe that he has to uninstall the app. According to a confirmation from a developer at Facebook[2], this problem was caused by a syncing error on Android (the app keeps syncing between the phone and the messaging server, thus utilizes a lot of CPU time which leads to high power consumption and overheat). A newly update version of Facebook Messenger has been released on February 13th to fix this problem.

B. Login issues of several messaging apps

In the previous example, we demonstrated an analysis on Facebook Messenger. In this section, we present another analysis on two similar apps: Whatsapp and Viber. Both of them are mobile apps for SMS and VoIP (i.e. sending text messages and calling). They both have a very large userbase and many daily activities. Their functionalities are almost identical and thus, they could be considered as direct competitors. Thus, we wonder *what are their common problems and what can we learn from them?* In this section, we will demonstrate how to answer that question with MARK.

We start with the Launching screen and select them for analysis. After that, we go to the Keyword Selection screen as showed in Figure 11. In this screen, we can examine

[2]http://androidforums.com/threads/facebook-messenger-battery-drain.902687/

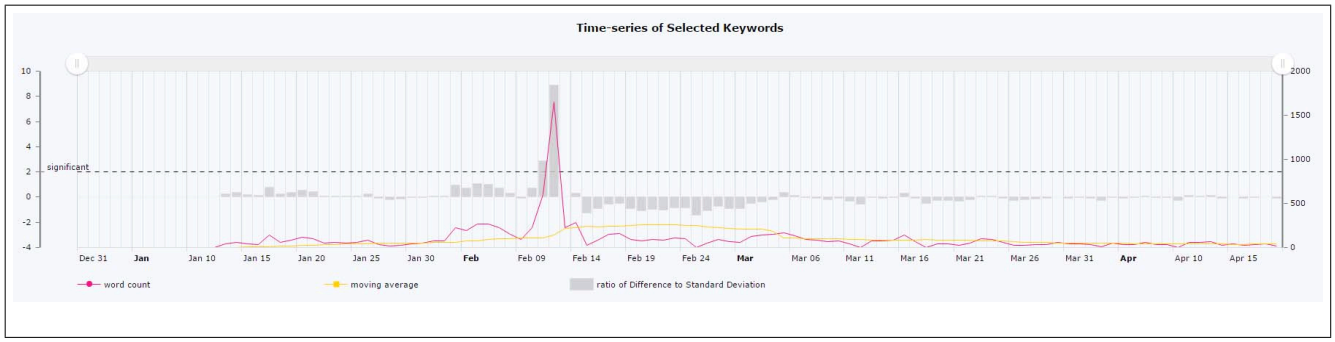

Fig. 8: Trends of keywords for "energy consumption" in Facebook Messenger

Fig. 11: Ranked negative keywords for Whatsapp and Viber

each keyword individually. For example, Figure 12 shows the analysis for keyword activation. As seen, MARK draws the trends of this keyword overtime, suggests related keywords, like login and email, and also lists the relevant reviews.

A quick skim through the reviews suggests that the users were having trouble with activating their accounts. Thus, we wonder if this is the only problem they have, or there are more regarding the topic of *"login and authentication"*. To investigate further, we choose the keywords that might relate to this problem. In this list, it seems like enter is the only keyword that does not really belong to our case. After that, we go back to the Keyword Selection screen. At this screen, we ask MARK to suggest more keywords (e.g. see Figure 13). We add some of them and search for the reviews. Figure 14 shows the two most relevant reviews. Much to our surprise, users also talked about the problem of not being able to sign in using Google email service for both apps.

In this analysis, we have discovered at least two common problems for two mass messaging apps which made users rather unsatisfied and frustrated. Thus, developers of these apps or similar ones can learn about these common problems to fix them when they happen, or prevent them in advance.

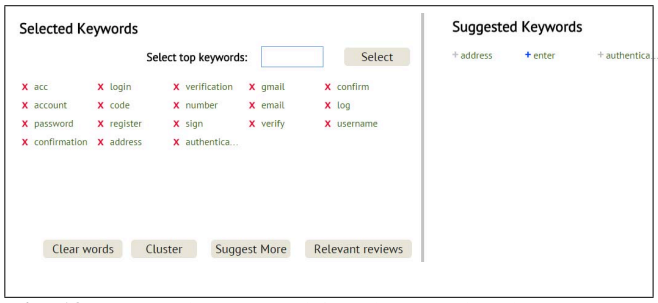

Fig. 13: Keywords suggested for *"login and authentication"*

Two presented examples have shown the usefulness of our tool. For example, it can help app developers to quickly identify users' complaints about their apps and the severity of such complaints (e.g. via the trends and counts of the related keywords and reviews). Additionally, our tool can help app developers examine multiple apps (e.g. including competitors of their apps), identify the common problems and thus, provide them more information to improve their apps.

III. Related Work

Studies on user issue reports and feedbacks of traditional software are popular in the literature [7]–[9]. However, to the best of our knowledge, there are very few tools or systems that focus on analyzing apps reviews. One of them is Wiscom [4]. It introduces several level of user reviews analysis, including micro level (for a single review), meso level (for reviews of an app) and macro level (for reviews of all apps in the market). The authors do sentiment analysis of words using Linear Regression Model is comparable to our keyword negative ranking scheme but with a different intention. MARK tries to address the impact of keyword's concerns to users while Wiscom wants to address the impact on sentiment of a keywords to discover inconsistent review.

On "meso level" Wiscom uses a LDA model to analyze topics of user reviews based on their distribution. Similarly, we group the keywords using K-mean clustering on vector-space representation of words, but our approach focus on exploiting different layers of semantic meaning for words inside the corpus, which give us a different perspective of user opinions. To the best of our knowledge, their work is also the first work to mention the use of timeseries on reviews for analysis, however, their approach was to use root cause analysis based

Fig. 12: Ranked keywords for Whatsapp and Viber

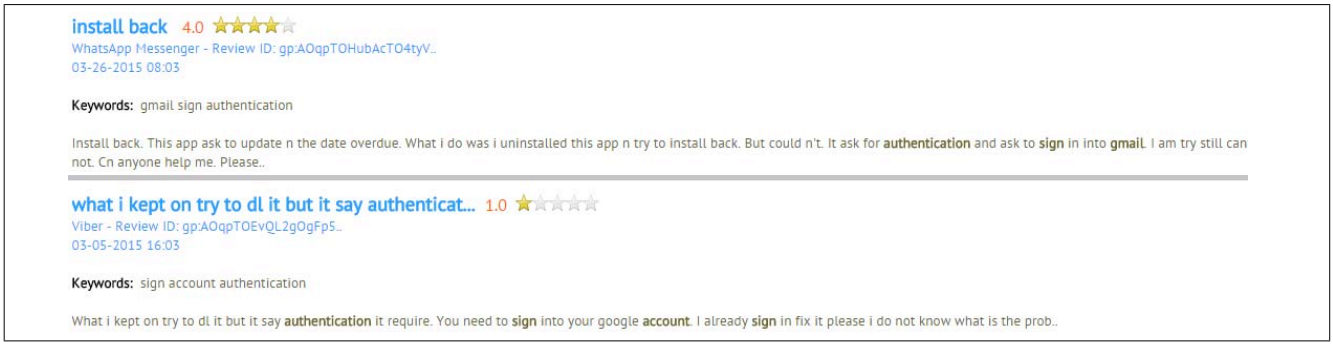

Fig. 14: Reviews of keyword for *login and authentication* problems

on the observed busts in negative or positive comments, which does not address the problems that may lie inside normal stream of comments as ours does.

Another interesting tool is the prototype MARA developed by Iacob et al [10]. It is designed to retrieve app feature requests from comments using a set of linguistic rules. These rules were manually derived from the reviews' text. This helps MARA analyzes feature requests and identify their common topics. Our approach of using keywords also can be used to find feature requests in addition to user complaints, but in a different dimension of understanding. More over, MARK allows users to analyze app reviews in a more comprehensive manner, including time, rating and semantic meaning of words.

Gomez et al. [11] developed a static error-proneness checker for app based on permissions, based on an empirical study suggested that there are error-prone permissions reported in user reviews. As other works, it is different from our main purpose and approach of mining keyword from reviews.

AR-Miner [1] is another approach in extracting information from reviews. They proposed a computational framework to extract and rank informative reviews at sentence level. We want to discover user's concerns, and provided a comprehensive approach to do that, while they want to find and rank most informative reviews, so the benefit for developers is different. Moreover, we focus on keywords level because their distributed representations can discover more detailed semantic meanings of user's reviews. Our tool is the first approach to provide a reliable way to search for relevant reviews using keywords, which is yet to be mentioned by any prior work.

IV. Conclusions

Apps reviews analysis is a new research field in mobile apps development. However, its impact can be very wide as the information extracted from users reviews can be used for many purpose, including widen the understanding of developers of their apps, and strategic development of competitor apps. However, reading millions of reviews could be an unfeasible option for them, thus the need of having a tool capable of analyzing reviews has risen. In this paper, we introduce MARK, a tool that can easily and comprehensively help developers to understand and discover user's concerns of chosen apps. We have demonstrated several examples the use cases of our tool and proved that it can find useful information from reviews for developers and reduce their effort in the process.

REFERENCES

[1] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "Ar-miner: Mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 767–778. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568263

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[3] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1. [Online]. Available: http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[4] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 1276–1284. [Online]. Available: http://doi.acm.org/10.1145/2487575.2488202

[5] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach," in *Proceedings of the 30th ACM/IEEE International Conference on Automated Software Engineering*. ACM, 2015.

[6] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[7] A. Ko, "Mining whining in support forums with frictionary," in *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '12. New York, NY, USA: ACM, 2012, pp. 191–200. [Online]. Available: http://doi.acm.org/10.1145/2212776.2212797

[8] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '10. New York, NY, USA: ACM, 2010, pp. 301–310. [Online]. Available: http://doi.acm.org/10.1145/1718918.1718973

[9] S. Mani, R. Catherine, V. S. Sinha, and A. Dubey, "Ausum: Approach for unsupervised bug report summarization," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, ser. FSE '12. New York, NY, USA: ACM, 2012, pp. 11:1–11:11. [Online]. Available: http://doi.acm.org/10.1145/2393596.2393607

[10] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 41–44. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487094

[11] M. Gomez, R. Rouvoy, M. Monperrus, and L. Seinturier, "A recommender system of buggy app checkers for app store moderators," Ph.D. dissertation, Inria Lille, 2014.