2017 IEEE/ACM 39th IEEE International Conference on Software Engineering Companion

SURF: Summarizer of User Reviews Feedback

Andrea Di Sorbo*, Sebastiano Panichella†, Carol V. Alexandru†, Corrado A. Visaggio*, Gerardo Canfora*
*University of Sannio, Department of Engineering, Italy
†University of Zurich, Department of Informatics, Switzerland
disorbo@unisannio.it, {panichella, alexandru}@ifi.uzh.ch, {visaggio, canfora}@unisannio.it

***Abstract*—Continuous Delivery (CD) enables mobile developers to release small, high quality chunks of working software in a rapid manner. However, faster delivery and a higher software quality do neither guarantee user satisfaction nor positive business outcomes. Previous work demonstrates that app reviews may contain crucial information that can guide developer's software maintenance efforts to obtain higher customer satisfaction. However, previous work also proves the difficulties encountered by developers in manually analyzing this rich source of data, namely (i) the huge amount of reviews an app may receive on a daily basis and (ii) the unstructured nature of their content. In this paper, we propose *SURF* (Summarizer of User Reviews Feedback), a tool able to (i) analyze and classify the information contained in app reviews and (ii) distill actionable change tasks for improving mobile applications. Specifically, SURF performs a systematic summarization of thousands of user reviews through the generation of an interactive, structured and condensed agenda of recommended software changes. An end-to-end evaluation of SURF, involving 2622 reviews related to 12 different mobile applications, demonstrates the high accuracy of SURF in summarizing user reviews content. In evaluating our approach we also involve the original developers of some apps, who confirm the practical usefulness of the software change recommendations made by SURF.**
Demo URL: https://youtu.be/Yf-U5ylJXvo
Demo webpage:
http://www.ifi.uzh.ch/en/seal/people/panichella/tools/SURFTool.html

***Keywords*-Natural Language Processing, Mobile Applications, Summarization, Software Maintenance**

I. Introduction

Today, with the advent of agile development and Continuous Delivery (CD), developers design and develop apps in a faster and more lightweight manner. However, faster delivery and a higher perceived software quality alone do not guarantee user satisfaction or positive business outcomes. Therefore, user involvement during the software evolution process is a crucial aspect for satisfying user needs and building successful products: user feedback contains important information for developers, which helps to identify bugs and point out missing features [11]. Consequently, development teams are interested in exploiting opinions and feedback of end-users during throughout the lifecycle of their software [8]. App distribution platforms (widely known as app stores) such as the Apple App Store, Google Play and the Windows Phone Store allow users to *review* downloaded apps. Thus, app reviews constitute a sort of communication infrastructure between users and developers [14]. Through this channel, users can post their overall experience, report bugs and request features. However, manually analyzing user reviews to become aware of user needs is a challenging and time consuming task, due to the huge amount of reviews each app receives on a daily basis as well as their unstructured nature [11], [12]. In order to reduce the effort required to analyze app reviews, automated approaches aimed at classifying (or prioritizing) reviews according to specific topics (e.g., bugs, enhancements, etc.) have been proposed in the literature [1], [9], [15]. However, a problematic aspect of these approaches resides in the fact that they are not able to *reduce* the amount of reviews and information developers have to deal with, which is very large for popular apps [3], [12].

In this paper, we introduce **SURF** (**S**ummarizer of **U**ser **R**eviews **F**eedback), a tool, based on our previous work [3], able to automatically generate summaries of users feedback with the aim of helping developers in better understanding user needs and coping with the potentially large volume of reviews. In particular, we introduce some improvements over the previous version: (i) tool outputs are in XML format, making it easier to integrate them in third party frameworks, (ii) a report-viewer utility with a totally redesigned GUI allows to easily browse the summaries, and (iii) a reviews-downloader utility allows to directly download reviews from Google Play and store them in the required XML format. Similarly to previous work [13], [6], [1] SURF's purpose is to make information contained in user reviews more manageable and actionable for developers. However, differently from these previous approaches, our tool is able to simultaneously (i) determine for a large number of reviews the specific topics discussed in the review (e.g., UI improvements, security/licensing issues, etc.), (ii) identify the maintenance task to perform for addressing the request stated in the review (e.g., bug fixing, feature enhancement, etc.), and (iii) present such information in the form of a *condensed, interactive and structured agenda of recommended software changes*. We argue that the combination of topic extraction and the ability of synthesizing reviews into well defined maintenance tasks regarding specific aspects of an app can concretely help developers in planning further software changes and meeting market requirements. To support this conjecture, we conduct an empirical study involving 12 developers from companies and research institutions in Switzerland, Italy and the Netherlands to investigate the practical usefulness of the summaries generated by *SURF*.

II. Approach

`SURF` summarizes user reviews by performing the following steps: (i) extract the topics of a review, (ii) classify the intention of the author to suggest the specific kind of

978-1-5386-1589-8/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSE-C.2017.5

TABLE I
INTENTION CATEGORIES DEFINITION

Information Giving	Sentences that inform other users or developers about some aspect of the app.
Information Seeking	Sentences describing attempts to obtain information or help from other users or developers.
Feature Request	Sentences expressing ideas, suggestions or needs for enhancing the app.
Problem Discovery	Sentences reporting unexpected behavior or issues.
Other	Sentences not belonging to any of the previous categories.

maintenance task required to act on the review, and (iii) group together sentences covering the same topic. In particular, SURF is built on top of the User Reviews Model (URM) which aim is to describe informative paragraphs contained in app reviews along two orthogonal dimensions: (i) the *user intention*, modeling the user's goals when writing a review (Table I depicts the *intentions categories* modeled by URM), and (ii) the *review topics*, capturing the specific topics covered by the review (Table II shows the *topics* covered by URM). More details about URM can be found in our previous work [3].

```
<reviews>
  <review>
      <date>2014-12-04</date>
      <star_rating>1</star_rating>
      <user>Jmbrowne</user>
      <app_version>2.0.1</app_version>
      <review_title>Painfully slow</review_title>
      <review_text>
          This puzzle would be enjoyable if it was not
          so painfully slow.
      </review_text>
  </review>
  . . .
</reviews>
```

Fig. 1. The XML reviews file

Since most common app distribution platforms collect reviews in different kinds of data structures, SURF relies on an XML *data exchange schema* for representing review data from multiple sources exemplified in figure 1. Thus, the input of the SURF process consists of an XML file containing the reviews under analysis. SURF comprises the following steps:

1) **Intention Classification**: For each review, the text is divided into sentences. Each sentence is passed through the Stanford Typed Dependencies (STD) parser [2]. On top of the STD representation we defined a set of 246 NLP heuristics[1] to detect the presence of structural patterns associated with one of the *intention* categories in Table I. Since previous work demonstrates that sentiment analysis helps to improve the accuracy of intent classification [14], [15], each sentence is also analyzed through the sentiment annotator provided by the Stanford CoreNLP [10], which returns a sentiment value from 1 (strong negative) to 5 (strong positive). The previously extracted features (structural patterns and sentiment features) are combined together trough a pre-trained Machine Learning (ML) model for predicting the *intention* category of the sentence. To integrate ML algorithms in our code, we used the Weka API [7]. More information about the intention classification step can be found in our previous papers [14], [15].
2) **Topic Classification**: Each sentence of a review is passed to an automated topics classifier. Specifically, for each of the topics in Table II we define a list of n-gram keywords that could be clues for the assignment of a sentence to that specific topic. As mentioned in [3], both URM and keyword lists have been defined through the manual analysis of reviews selected from a different dataset from the one used for evaluating the tool. On top of these *topic-related dictionaries* we built an NLP classifier to automatically assign every sentence in a review to one or more topics. Each sentence is stemmed (*i.e.*, reduced to its root form) using the Snowball Stemmer Algorithm [16]. Given a sentence S and a topic C, let W_C be the number of tokens in S that also appear in the list of C-related keywords (*i.e.*, *the topic-related dictionary* of C); let W_S be the total number of token in S, the NLP classifier computes the probability that S falls in the topic C as:

$$P_{(S,C)} = \frac{W_C}{W_S} \quad (1)$$

3) **Sentence Scoring**: All the sentences are preprocessed by applying Snowball Stemming [16] and doing stopword removal. We assign a global score $GS_{(S,C)}$ to each sentence S with respect to a topic C using the equation:

$$GS_{(S,C)} = IRS_S * P_{(S,C)} * [L_S * (1 + MFWR_{(S,C)})] \quad (2)$$

The first partial score IRS_S assigns to each sentence S an initial score on the basis of its *intention* category assigned during the *Intention Classification* step. $P_{(S,C)}$ is the probability of a sentence S to be relevant for a given topic C (defined in the *Topics Classification* step). The third partial score L_S, computes the total number of characters constituting the sentence S. Finally, for each sentence S belonging to the topic C, we compute $MFWR_{(S,C)}$, which measures the ratio of frequent words appearing in the sentence ($MFR_{(S,C)}$) with respect of the number of total words in the sentence (TW_S). For computing frequent words in each topic we rely on Classifier4J[2] java library. The proposed scoring function is able to simultaneously reward: (i) sentences containing feature requests or bug reports with respect to other kinds

TABLE II
TOPIC CLUSTERS

Cluster	Description
App	sentences related to the entire app, *e.g.*, generic crash reports, ratings, or general feedback
GUI	sentences related to the Graphical User Interface or the look and feel of the app
Contents	sentences related to the content of the app
Pricing	sentences related to app pricing
Feature or Functionality	sentences related to specific features or functionality of the app
Improvement	sentences related to explicit enhancement requests
Updates/ Versions	sentences related to specific versions or the update process of the app
Resources	sentences dealing with device resources such as battery consumption, storage, etc.
Security	sentences related to the security of the app or to personal data privacy
Download	sentences containing feedback about the app download
Model	sentences reporting feedback about specific devices or OS versions
Company	sentences containing feedback related to the company/team which develops the app

[1]http://www.ifi.uzh.ch/seal/people/panichella/Appendix.pdf

[2]http://classifier4j.sourceforge.net/

of feedback, (ii) sentences that are likely to relate to specific topics, (iii) longer sentences (that are usually more informative than shorter ones), and (iv) sentences concerning frequently discussed features. Only sentences in the top positions of the ranked list are selected. More formally, let NS_C be the number of total scored sentences for the topic C, the sentences occupying the first $0.7 * NS_C$ positions in the ranked list (*i.e.*, about 2/3 of total sentences) are extracted for further processing, while the remaining sentences, occupying the last $0.3 * NS_C$ positions, are discarded.

4) **Summary Generation**: SURF provides output reports in XML format. XML summaries can be easily browsed through the provided visualization utility, as well as used by developers for further analysis. Reports are presented and can be filtered along a two-level hierarchy: (i) the sentences are grouped together according to their topics (*e.g.*, App, GUI, etc.) leveraging the $GS_{(S,C)}$ scores computed in the previous step; (ii) then, sentences in each topic are grouped by *intention* categories, which were assigned during the *Intention Classification* step.

III. How to SURF App Review Data

SURF consists of a command line tool which requires to specify the path of the XML file containing the user reviews to summarize and the path of the output file which will contain the generated summary. The generated summary will consist of a single XML file. To use SURF it is sufficient to download and unzip `SURF.zip` from the tool's webpage and follow the running instructions provided in the `README.txt` file. Before running SURF, developers interested in summarizing app reviews have to collect target reviews in a XML file following the structure showed in Figure 1. For this purpose, we provide in the SURF package a side utility (`reviews-downloader.jar`) with a user friendly GUI for automatically downloading reviews from Google Play and collecting downloaded reviews in the required XML.

Finally, for generating the reviews summary the `path of the XML file` containing the reviews and the `path and filename of the output file` (where will be stored the reviews summary). will be passed to SURF (i.e., `SURF.jar`) as command-line arguments.

Once the reviews summarization is finished, users can view the output report using a simple web-based browsing utility we provide as part of the SURF package (`report-viewer.html`). The viewer groups reviews by topic and shows the proportion of review intentions as an integrated bar chart for each topic. This allows developers to see the distribution of reviews across topics and intentions at first glance. The developer can now click on a topic to show all related reviews or on a bar-chart segment of a particular intent to show only those reviews for a given topic that correspond to the selected intent. For each review, the most relevant sentence is shown, and clicking on a sentence expands the review text and shows additional review metadata, such that the sentence can be viewed in context of the original review.

IV. Evaluation

In order to assess the usefulness of summaries generated by SURF, as reported in our previous work [3], we conducted an empirical evaluation involving 17 apps. In this paper we discuss results obtained on a subset of such set of apps (12 apps). Specifically, the context of the study consisted of 2622 app reviews, mined from 3 different app stores and relating to 12 apps. These apps belong to 7 different app categories in the stores. Thus, for evaluating SURF we (i) generated the summaries of reviews for each app in our dataset; (ii) involved 3 original app developers, 3 postdoc researchers in Software Engineering, 2 software testers, 3 industrial software engineers, and 1 software engineering master student; (iii) randomly assigned (with the exception of the three original app developers involved in the study) an app to each participant; (iv) provided the subjects with the summaries generated by SURF; (v) asked them to validate the summaries (*i.e.*, report a list of reviews incorrectly classified in the summary), and finally (vi) invited participants to answer the questions of a short survey. Table III depicts the questions of the survey, as well as the aggregated data regarding the answers provided by participants.

The majority of participants (9 out of 12) judged the provided summaries highly useful and comprehensible, and the majority of them (11 out of 12) partially or totally agreed with the affirmation that, without summaries, evaluating user feedback is tedious and difficult. SURF summaries ease the difficulty of analyzing reviews: 8 out of 12 participants said that analyzing reviews is easy or very easy when the SURF summary is provided; conversely 9 out of 12 participants declared that the same process would be hard or very hard without consulting the summary. Moreover, all the survey respondents perceived SURF summaries to be time-saving: 2/3 of subjects declared that SURF summaries allow to save at least 50% of the time otherwise spent manually collecting and analyzing user reviews. Regarding the quality of the summaries, they asserted an accuracy of 92%: only 107 out of 1260 total sentences in summaries have been labeled as incorrectly classified by survey participants. Furthermore, SURF summaries demonstrated to present a reasonably good content adequacy, as well as a high conciseness and expressiveness. Indeed, (i) 10 out of 12 participants declared that summaries are not missing any information or that the missing information is not relevant for understanding user needs, (ii) 11 out of 12 participants declared that summaries do not contain unnecessary information, and (iii) 10 out of 12 participants declared that SURF summaries are easy to read and understand. Last, but not least, 11 out of 12 participants perceived summaries highly useful for better understanding user reviews feedback of mobile apps.

V. Conclusions

In this paper we present SURF, a summarization tool able to summarize app reviews and generate an interactive, structured and condensed agenda of recommended software changes. We also present the results of an end-to-end evaluation of

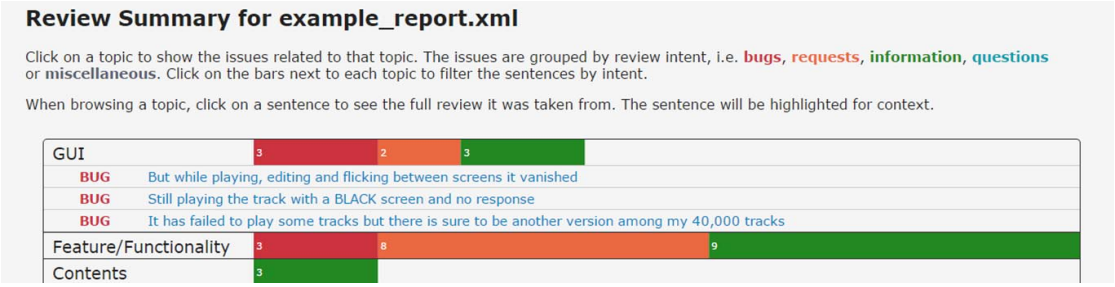

Fig. 2. Extract of a summary

TABLE III
SURVEY DATA

Question	Answer	
How do you judge the usefulness and comprehensibility of the provided summary?	VERY LOW	0 (0%)
	LOW	1 (8%)
	MEDIUM	2 (17%)
	HIGH	5 (42%)
	VERY HIGH	4 (33%)
How difficult is to analyze user feedback contained in reviews WITH the summary?	VERY LOW	3 (25%)
	LOW	5 (42%)
	MEDIUM	4 (33%)
	HIGH	0 (0%)
	VERY HIGH	0 (0%)
How difficult is to analyze user feedback contained in reviews WITHOUT the summary?	VERY LOW	0 (0%)
	LOW	0 (0%)
	MEDIUM	3 (25%)
	HIGH	8 (67%)
	VERY HIGH	1 (8%)
Proportionally, how much time you can save by analyzing feedback contained in user reviews WITH the proposed summary (if compared with the time required WITHOUT the summary)?	NOTHING	0 (0%)
	10% <= TIME GAIN <= 20%	1 (8%)
	33% <= TIME GAIN < 50%	3 (25%)
	TIME GAIN >= 50%	8 (67%)
WHITOUT the proposed summary, evaluating the user feedback contained in reviews is prohibitively difficult and/or tedious	TOTALLY DISAGREE	0 (0%)
	PARTIALLY DISAGREE	1 (8%)
	PARTIALLY AGREE	3 (25%)
	TOTALLY AGREE	8 (67%)
Considering only the content of the summary of user feedback and not the way it is presented, do you think that the report?	A. Is missing any information	7 (59%)
	B. Is missing some information but the missing information are not necessary	3 (25%)
	C. Is missing some very important information	1 (8%)
	D. Not sure	1 (8%)
Considering only the content of the summary and not the way it is presented, do you think that the report?	A. Has no unnecessary information	11 (92%)
	B. Has some unnecessary information	1 (8%)
	C. Has a lot of unnecessary information	0 (0%)
Considering only the way the summary of user feedback is presented and not its content, do you think that the report?	A. Is easy to read and understand	10 (84%)
	B. Is somewhat readable and understandable	1 (8%)
	C. Is hard to read and understand	1 (8%)
Are the summaries useful for understanding user reviews feedback of mobile apps?	VERY LOW	0 (0%)
	LOW	0 (0%)
	MEDIUM	1 (8%)
	HIGH	5 (42%)
	VERY HIGH	6 (50%)

summaries, demonstrating the high quality of generated summaries. In future work, we are interested in implementing, on top of SURF, a mechanism able to recognize which part of the source code needs to be changed in the app to perform the change tasks suggested by the tool. The review extraction, summarization and browsing tools could at this point be integrated with an issue tracker, such that developers could directly create new issues from the review browsing utility. We are also interested in providing a feature for customizing topics and topics-related dictionaries, with the aim of making SURF more adaptable to different working contexts and summarizing additional sources of information [5], [4].

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*, ICSE 2014, pages 767–778, New York, NY, USA, 2014. ACM.

[2] M.-C. de Marneffe and C. D. Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics.

[3] A. Di Sorbo, S. Panichella, C. Alexandru, J. Shimagaki, C. Visaggio, G. Canfora, and H. Gall. What would users change in my app? summarizing app reviews for recommending software changes. In *Foundations of Software Engineering (FSE), 2016 ACM SIGSOFT International Symposium on the*, pages 499–510, 2016.

[4] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. C. Gall. Development emails content analyzer: Intention mining in developer discussions (T). In *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*, pages 12–23, 2015.

[5] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. C. Gall. DECA: development emails content analyzer. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume*, pages 641–644, 2016.

[6] X. Gu and S. Kim. What parts of your apps are loved by users? (T). In *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*, pages 760–770, 2015.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[8] S. Krusche and B. Bruegge. User feedback in mobile development. In *Proceedings of the 2Nd International Workshop on Mobile Development Lifecycle*, MobileDeLi '14, pages 25–26, New York, NY, USA, 2014. ACM.

[9] W. Maalej and H. Nabil. Bug report, feature request, or simply praise? on automatically classifying app reviews. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, Aug 2015.

[10] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[11] D. Pagano and B. Brügge. User involvement in software evolution practice: A case study. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 953–962, Piscataway, NJ, USA, 2013. IEEE Press.

[12] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *Proceedings of the 21st IEEE International Requirements Engineering Conference (RE 2013)*. IEEE Computer Society, 2013.

[13] F. Palomba, M. Linares-Vasquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia. User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 291–300, Sept 2015.

[14] S. Panichella, A. Di Sorbo, E. Guzman, C. Visaggio, G. Canfora, and H. Gall. How can I improve my app? classifying user reviews for software maintenance and evolution. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 281–290, Sept 2015.

[15] S. Panichella, A. Di Sorbo, E. Guzman, C. Visaggio, G. Canfora, and H. Gall. Ardoc: App reviews development oriented classifier. In *Foundations of Software Engineering (FSE), 2016 ACM SIGSOFT International Symposium on the*, pages 1023–1027, 2016.

[16] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.