Requirements Intelligence with OpenReq Analytics

Christoph Stanik and Walid Maalej
University of Hamburg
Hamburg, Germany
{stanik, maalej}@informatik.uni-hamburg.de

***Abstract*—With the rise of social media like Twitter and distribution platforms like app stores, users have various ways to express their opinions about software products. Popular software vendors get user feedback thousandfold per day. Research has shown that such feedback contains valuable information for software development teams. However, a manual analysis of user feedback is cumbersome and hard to manage. We present *OpenReq Analytics*, a software requirements intelligence service, that collects, processes, analyzes, and visualizes user feedback.**

***Index Terms*—Requirements Intelligence, Data-Driven Requirements, Data Mining, Social Media Analytics, App Store Analytics**

I. Introduction

Software users share a large amount of feedback, which can be valuable to software development teams to better understand user needs. Feedback shared in platforms such as app stores and Twitter contain insights like problems/bugs, feature requests, inquiries, or experience reports [6]. Research showed that vendors considering user feedback are more successful in terms of download numbers and ratings, but a manual analysis is cumbersome [7]. Further, interviews with industry practitioners highlighted the need for developers and managers for tool support to monitor user feedback continuously [3].

II. Requirements Intelligence

Inspired by Maalej et al. [4], we define (Software) Requirements Intelligence as a data-driven concept to explore requirements in the masses of implicit feedback (i.e., usage data) and explicit feedback (i.e., written feedback). As a step toward requirements intelligence, we are developing the web-based tool *OpenReq Analytics*. *OpenReq*[1] is a European Horizon 2020 open source project, which aims at creating methods, tools, and APIs toward achieving an intelligent recommendation and decision support for community-driven requirements engineering. *OpenReq Analytics* is the OpenReq component that collects, processes, analyzes, and visualizes implicit and explicit user feedback from app stores and Twitter. *OpenReq Analytics* targets developers, as well as product and innovation managers. To get started, the users only need to configure the accounts they want to analyze, e.g. in Twitter.

A. Architecture

OpenReq Analytics is currently based on ten microservices. Figure 1 shows the communication flow of the microservices for monitoring tweets from Twitter. The single access point

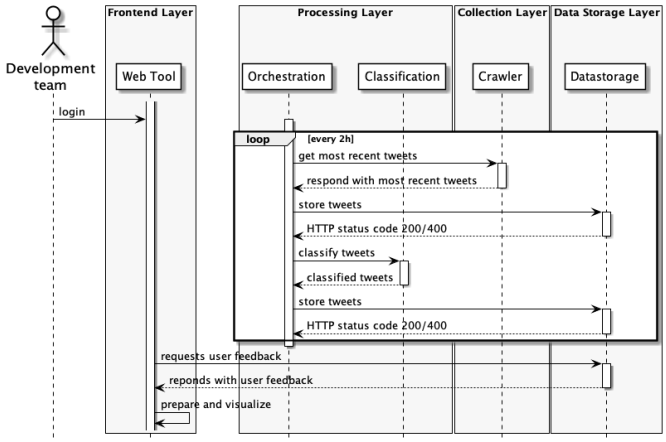

Fig. 1: Microservice Architecture of *OpenReq Analytics*.

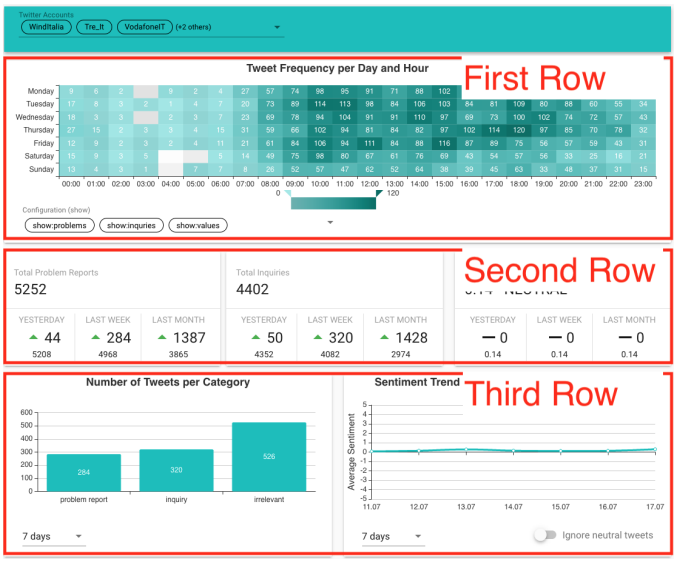

Fig. 2: *OpenReq Analytics*: The Dashboard.

is the web-tool (see Fig. 2), which loads all data needed for visualization purposes from the *Datastorage* microservice. To ensure that the data presented is up-to-date, the orchestration microservices continuously crawl, classify, and store user feedback in a two-hour interval.

B. Dashboard

The main view of *OpenReq Analytics* is the dashboard shown on Figure 2. It consists of three rows, each explained

[1]**Website:** www.openreq.eu, **GitHub:** https://github.com/OpenReqEU

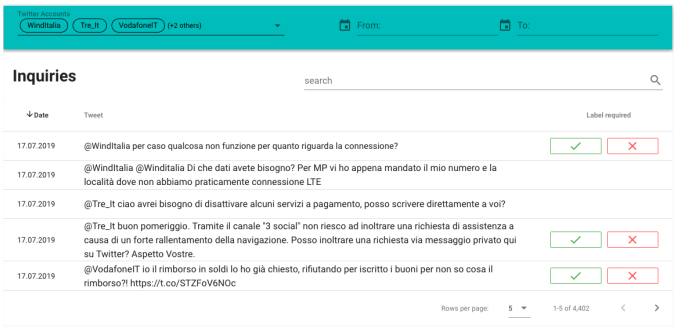

Fig. 3: *OpenReq Analytics*: Focus View for Inquiries.

in the following paragraphs.

When do users write? The first row shows a heat map that gives insights on when users are giving feedback. The heat map is an helpful indicator of when users face issues or think about new functionality. It assists vendors to understand when customer support is needed. When this explicit feedback analysis is combined with implicit feedback, detailed conclusions about usage patterns such as which steps caused the issue can be drawn. *OpenReq Analytics* allows filtering for *relevant feedback entries*, which are only those that are either a problem report, feature request, or inquiry.

Trend Reports. In the second row, three trend reports are showing information regarding predefined time windows. The goal is to give an overview of three particular interests: 1) how many users are facing problems, 2) how many inquiries (incl. feature requests) are being received, and 3) what is the overall sentiment of the users when giving feedback? This view helps to understand the general performance of the vendor over time.

Historical Analysis. The third row enables the user to analyze custom time frames such as specific releases. If, e.g., the software got a new release, the vendor can understand if bug reports are peaking or if less are being received. On the other hand, whenever problem reports are peaking, users might get impatient and write rather negative reviews [5]. The sentiment chart on the right side helps to understand when the overall sentiment is back to normal, and if the strategy followed by the customer support is effective.

C. Focus Views

OpenReq Analytics has two focus views (see Figure 3)—each aggregating information related to either problem reports or inquiries. The goals of the focus views are to aggregate feedback and to enable the development team to explore the concrete feedback given. In this view, one can search for keywords of interest, filter the source of the feedback and its language, and defined time frames. A machine learning model performs the separation into problem reports and inquiries following best practices from related work [3], [1]. However, as the performance of machine learning models may decay over time, e.g., because software introduced new features or the product portfolio of a vendor changed, it is important to keep the models up-to-date. For this, we included an active learning component. On the right side of Figure 3, two buttons are shown—one for agreeing with the classification and one for disagreeing. When the software development team disagrees with the classification, *OpenReq Analytics* shows other categories the user feedback should belong to. In case the problem report focus view is open and the development team disagrees with the classification, two buttons appear: one for labeling that feedback as an inquiry and one for labeling it as irrelevant. It is important to note, that only user feedback can be labeled for which the classification model is uncertain. The rationale behind that decision is twofold. First, those cases have a higher impact on the improvement of the model. Second, software development teams are not overwhelmed with a labeling task of potentially thousands of labels per day.

III. Conclusion and Future Work

OpenReq Analytics is still under development and will get additional features in the future. One feature to come is topic modeling as at the moment the information presented is only separated in the high-level categories *problem reports*, *inquiries*, and *irrelevant*. But this leaves out details such as "what particular feature causes the problem?" or "what concrete feature our user wish?" [2].

In addition to the historical analysis, we will include a release-based perspective which helps to understand better how each software/service update impacts user feedback.

Another import feature is clustering of similar user feedback to get a better understanding of how many users are facing a problem or requesting specific information or features.

Acknowledgement

The work presented in this paper has been conducted within the scope of the Horizon 2020 project OpenReq, which is supported by the European Union under the Grant Nr. 732463

References

[1] E. Guzman, M. Ibrahim, and M. Glinz. A little bird told me: mining tweets for requirements and software evolution. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 11–20. IEEE, 2017.

[2] T. Johann, C. Stanik, A. M. A. B., and W. Maalej. Safe: A simple approach for feature extraction from app descriptions and app reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 21–30, Sep. 2017.

[3] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik. On the automatic classification of app reviews. *Requirements Engineering*, 21(3):311–331, Sep 2016.

[4] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54, 2016.

[5] D. Martens and T. Johann. On the emotion of users in app reviews. In *2017 IEEE/ACM 2nd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pages 8–14. IEEE, 2017.

[6] D. Pagano and W. Maalej. User feedback in the appstore: An empirical study. In *Requirements Engineering Conference (RE), 2013 21st IEEE International*, pages 125–134. IEEE, 2013.

[7] 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 *2015 IEEE international conference on software maintenance and evolution (ICSME)*, pages 291–300. IEEE, 2015.