ARdoc: App Reviews Development Oriented Classifier

Panichella, Sebastiano; Di Sorbo, Andrea; Guzman Ortega, Emitza; Visaggio, Corrado Aaron; Canfora, Gerardo; Gall, Harald

Abstract: Google Play, Apple App Store and Windows Phone Store are well known distribution platforms where users can download mobile apps, rate them and write review comments about the apps they are using. Previous research studies demonstrated that these reviews contain important information to help developers improve their apps. However, analyzing reviews is challenging due to the large amount of reviews posted every day, the unstructured nature of reviews and its varying quality. In this demo we present ARdoc, a tool which combines three techniques: (1) Natural Language Parsing (NLP), (2) Text Analysis (TA) and (3) Sentiment Analysis (SA) to automatically classify useful feedback contained in app reviews important for performing software maintenance and evolution tasks. Our quantitative and qualitative analysis (involving mobile professional developers) demonstrate that ARdoc correctly classi es feedback useful for maintenance perspectives in user reviews with high precision (ranging between 84% and 89%), recall (ranging between 84% and 89%), and an F-Measure (ranging between 84% and 89%). While evaluating our tool we also found that ARdoc substantially helps to extract important maintenance tasks for real world applications. Demo URL: https://youtu.be/Baf18V6sN8E Demo Web Page: http://www.ifi.uzh.ch/seal/people/panichella/tools/ARdoc.html

ARdoc: App Reviews Development Oriented Classifier

Sebastiano Panichella[1], Andrea Di Sorbo[2], Emitza Guzman[1],
Corrado A. Visaggio[2], Gerardo Canfora[2], Harald Gall[1]
[1]University of Zurich, Department of Informatics, Switzerland
[2]University of Sannio, Department of Engineering, Italy
panichella@ifi.uzh.ch, disorbo@unisannio.it, guzman@ifi.uzh.ch,
{visaggio,canfora}@unisannio.it, gall@ifi.uzh.ch

ABSTRACT

Google Play, Apple App Store and Windows Phone Store are well known distribution platforms where users can download mobile apps, rate them and write review comments about the apps they are using. Previous research studies demonstrated that these reviews contain important information to help developers improve their apps. However, analyzing reviews is challenging due to the large amount of reviews posted every day, the unstructured nature of reviews and its varying quality. In this demo we present `ARdoc`, a tool which combines three techniques: (1) Natural Language Parsing (NLP), (2) Text Analysis (TA) and (3) Sentiment Analysis (SA) to automatically classify *useful feedback* contained in app reviews important for performing software maintenance and evolution tasks. Our quantitative and qualitative analysis (involving mobile professional developers) demonstrate that `ARdoc` correctly classifies feedback useful for maintenance perspectives in user reviews with high precision (ranging between 84% and 89%), recall (ranging between 84% and 89%), and an F-Measure (ranging between 84% and 89%). While evaluating our tool we also found that `ARdoc` substantially helps to extract important maintenance tasks for real world applications.
Demo URL: https://youtu.be/Baf18V6sN8E
Demo Web Page:
`http://www.ifi.uzh.ch/seal/people/panichella/tools/ARdoc.html`

CCS Concepts

•Software and its engineering → Software maintenance tools;

Keywords

User Reviews, Mobile Applications, Natural Language Processing, Sentiment Analysis, Text Classification

1. INTRODUCTION

Mobile users can download mobile applications from the app stores (e.g. Google Play and Apple Store). These platforms, besides the download service, offer to the users the possibility to rate the apps and write reviews about them, in the form of unstructured text. Recent work [1–3] demonstrated that approximately one third of the information contained in user reviews is relevant to guide app developers in accomplishing software maintenance and evolution tasks (e.g. requests of implementation of new features, descriptions of bugs, users' feedback about specific features, etc.) [4, 5].

However, the manual inspection of feedback contained in user reviews is a challenging task for three main reasons: (i) *apps receive a lot of reviews every day*, for example Pagano *et al.* [3] found that iOS apps, receive approximately 22 reviews per day, while popular apps, such as Facebook, receive more than 4000 reviews per day; (ii) *the unstructured nature of reviews makes them hard to parse and analyze*; (iii) *the quality of reviews varies greatly*, from useful reviews providing ideas for improvement or describing specific issues to generic praises and complaints [1].

To handle this problem, several approaches have been proposed in the literature to automatically select and discover useful reviews from a developer's perspective [1,2,6–8]. However, the proposed techniques rely on traditional approaches that treat text as bags of words [9]. Such techniques are useful for discovering text fragments (i) sharing several concepts (or words), or (ii) are likely to treat the same topics but are not able to reveal anything about purposes or intentions of humans-written text [10]. In a previous work [11] we demonstrated that, in order to (i) enable the mining of writer's intentions and, consequently, (ii) detect in an automated way useful feedback contained in user reviews, three dimensions of texts can be investigated: *lexicon* (i.e., the specific words used in the review), *structures* (i.e., the grammatical frames constituting the reviews) and *sentiment* (i.e., the writer's intrinsic attitude (or mood) towards the topics treated in the text). Thus, we combined three techniques: (1) Natural Language Parsing (NLP), (2) Text Analysis (TA) and (3) Sentiment Analysis (SA) for the automatic classification of useful feedback contained in app reviews.

In this paper we present `ARdoc` (App Reviews Development Oriented Classifier), an all-in-one tool that automatically classifies useful sentences in user reviews from a software maintenance and evolution perspective. Specifically, the proposed approach classifies user reviews content according to a taxonomy designed to model developers' information needs when performing software maintenance and evolution tasks [11]. As shown in our study `ARdoc` substantially helps to extract important maintenance tasks for real world applications.

Paper structure. Section 2 briefly describes the mining

FSE 2016 Seattle, WA, USA
© 2016 ACM. ISBN 123-4567-24-567/08/06. . . $15.00
DOI: 10.475/123_4

approach we proposed to automatically classify user reviews, the technologies we employed and how we combined them. Section 3 shows how `ARdoc` works, while Section 4 details the studies we performed to evaluate `ARdoc` performance. Finally, Section 5 concludes the paper outlining future directions.

2. THE APPROACH

This section briefly describes the approach and technologies we employed.

Table 1: Categories Definition

Category	Description	User Feedback Example
Information Giving	Sentences that inform or update users or developers about an aspect related to the app	`“This app runs so smoothly and I rarely have issues with it anymore"`
Information Seeking	Sentences related to attempts to obtain information or help from other users or developers	`“Is there a way of getting the last version back?"`
Feature Request	Sentences expressing ideas, suggestions or needs for improving or enhancing the app or its functionalities	`“‘Please restore a way to open links in external browser or let us save photos"`
Problem Discovery	Sentences describing issues with the app or unexpected behaviours	`“App crashes when new power up notice pops up"`
Other	Sentences do not providing any useful feedback to developers	`“What a fun app"`

`ARdoc` classifies sentences contained in user reviews, that are useful for maintenance perspective, in five categories: *feature request, problem discovery, information seeking, information giving* and *other*. Table 1 shows, for each category: (i) the category name, (ii) the category description and (iii) an example sentence belonging to category. As described in [11], these categories emerged from a systematic mapping between the taxonomy of topics occurring in app reviews described by Pagano et al. [3] and the taxonomy of categories of sentences occurring in developers' discussions over development-specific communication means [10,12]. Specifically, such taxonomy is defined to model feedback from user reviews that are important from a maintenance perspective.

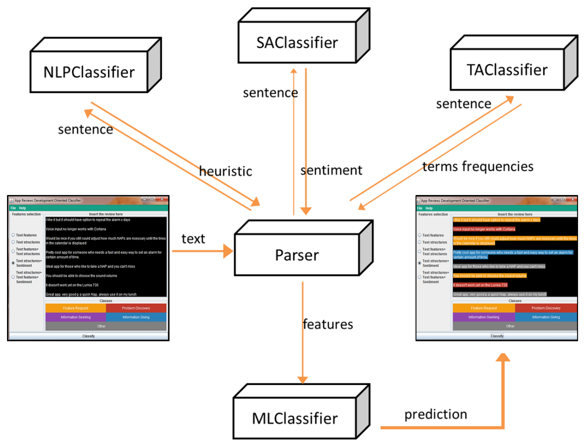


Figure 1: ARdoc's architecture overview

Figure 1 depicts `ARdoc`'s architecture. The main module of the tool is represented by the `Parser`, which prepares the text for the analysis (i.e., text cleaning, sentence splitting, tokenization, etc.). Our `Parser` exploits the functionalities provided by the Stanford CoreNLP API [13], which annotates the natural text with a set of meaningful tags. Specifically it instantiates a pipeline with annotations for tokenization and sentences splitting. The tokenizer divides text into a sequence of tokens, which roughly correspond to “words”.

Once the text is divided into sentences ARdoc extracts from each of these sentences three kinds of features: (i) the lexicon (i.e., the words used in the sentence) through the TAClassifier, the structure (i.e., grammatical frame of the sentence) through the NLPClassifier, and (iii) the sentiment (i.e., a quantitative value assigned to the sentence expressing an affect or mood) through the SA Classifier. Finally, in the last step the `MLClassifier` uses the NLP, TA and SA information extracted in the previous phase of the approach to classify app reviews according to the taxonomy reported in Table 1 by exploiting a Machine Learning (ML) algorithm. In the following we briefly describe the information extracted by our tool (Section 2.1) from app reviews and the classification techniques we adopted (Section 2.2).

2.1 Features Extraction

The `NLPClassifier` implements a set of NLP heuristics to automatically detect recurrent linguistic patterns present in user reviews. Through a manual inspection of 500 reviews from different kinds of apps we identified 246 recurrent linguistic patterns[1] often occurring in app reviews, and for each of these patterns we implemented an NLP heuristic in order to automatically recognize it (more details about the process performed for the definition of the heuristics are available in our previous work [11]). The `NLP classifier` uses the Stanford Typed Dependencies (STD) parser [14], a natural language parser which represents dependencies between individual words contained in sentences and labels each dependency with a specific grammatical relation (e.g., subject or direct/indirect object).
Through the analysis of the typed dependencies, each NLP heuristic tries to detect the presence of a text structure that may be connected to one of the categories in Table 1, looking for the occurrences of specific keywords in precise grammatical roles and/or specific grammatical structures. For each sentence in input, the `NLPClassifier` returns the corresponding linguistic pattern. If the sentence does not match any of the patterns we defined, the classifier simply returns the label “No patterns found”.

The `SAClassifier` analyzes the sentences trough the sentiment annotator provided by the Stanford CoreNLP [13] and for each sentence in input returns a sentiment value from 1 (strong negative) to 5 (strong positive).

The `TAClassifier` exploits the functionalities provided by the Apache Lucene API[2] for analyzing text content in user reviews. Specifically, this classifier performs a stop-words removal (i.e., words not containing important information) through the `StopFilter` and normalizes the input sentences (i.e., reduces the inflected words in the root form) through the `EnglishStemmer` in combination with the `SnowballFilter` in order to extract a set of meaningful terms that are weighted using the *tf* (term frequency), which weights each word i in a review j as:

$$tf_{i,j} = \frac{rf_{i,j}}{\sum_{k=1}^{m} rf_{k,j}}$$

where $\mathbf{rf}_{i,j}$ is the raw frequency (number of occurrences) of word i in review j.

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

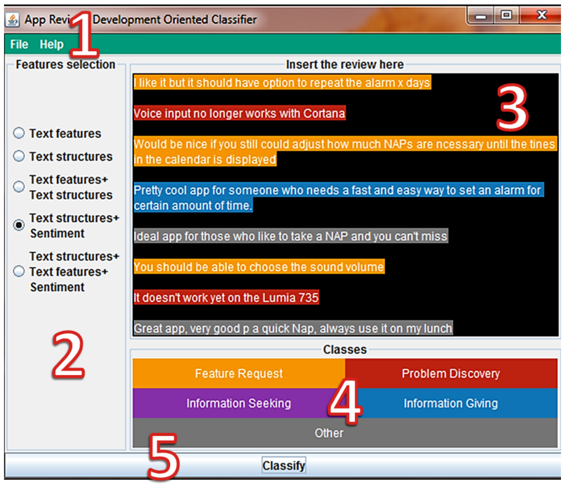
2.2 Automatic Classification via ML Techniques

We used the NLP, TA and SA features extracted in the previous phase of the approach to train ML techniques and classify app reviews according to the taxonomy in Table 1. To integrate ML algorithms in our code, we used the Weka API [15]. The `MLClassifier` module provides a set of java methods for prediction, each of them exploits a different pre-trained ML model and uses a specific combination of the three kinds of extracted features: (i) text features (extracted through the `TAClassifier`), (ii) structures (extracted through the `NLPClassifier`) and (iii) sentiment features (extracted through the `SAClassifier`). Specifically, methods implemented in the `MLClassifier` may use the following combinations of features (as shown in Figure 2): (i) only text features, (ii) only text structures, (iii) text structures + text features, (iv) text structures + sentiment, and (v) text structures + text features + sentiment. We do not provide (i) sentiment and (ii) text features + sentiment combinations, because, as discussed in our previous work [11], they proved very poor effectiveness in classifying sentences into the defined categories.

All the prediction methods provided by the `MLClassifier` class create a new `Instance` using a combination of the extracted features to learn a specific ML model and classify the `Instance` according to the categories showed in Table 1. Among all the available ML algorithms we use the J48 algorithm since in our previous work it was the algorithm that achieved the best results [11]. We trained all the ML models using as training data a set of 852 manually labeled sentences randomly selected from the user reviews of seven popular apps (more details can be found in [11]).

3. USING ARDOC

This section describes how the tool works.



I like it but it should have option to repeat the alarm x days

Voice input no longer works with Cortana

Would be nice if you still could adjust how much NAPs are ncessary until the tines in the calendar is displayed

Pretty cool app for someone who needs a fast and easy way to set an alarm for certain amount of time.

Ideal app for those who like to take a NAP and you can't miss

You should be able to choose the sound volume

It doesn't work yet on the Lumia 735

Great app, very good p a quick Nap, always use it on my lunch

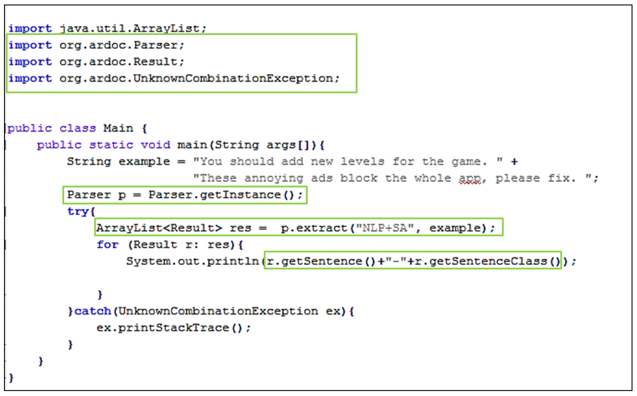
Features selection: ( ) Text features ( ) Text structures ( ) Text features+ Text structures [✓ Text structures+ Sentiment] ( ) Text structures+ Text features+ Sentiment

Figure 2: ARdoc Graphic Interface

We provide two versions of `ARdoc`. The first version provides a practical and intuitive Graphic User Interface. Users simply have to download the zipped file `ARDOC.zip`, unzip the downloaded file and follow the running instructions provided in the `README.txt` file. Figure 2 shows the tool's interface.

The tool's window is divided into the following sections: (i) the *menu bar* (point 1 in Figure 2) provides functions for creating a new blank window, loading the text to classify from an existing text file, **importing the reviews for classification from Google Play**, or exporting the classified data for further analysis; (ii) the *features selection panel* (point 2 in Figure 2) allows users to choose the desired combination of features for reviews classification; (iii) the *input text area* (point 3 in Figure 2) allows users to write (or copy and paste) reviews to classify and visualize the classification results; (iv) the *panel with the legend* (point 4 in Figure 2) reports the categories and their associated colors; (v) the *button Classify* (point 5 in Figure 2) allows to start the classification and produces the classification results.

To analyze the reviews the user can simply (i) paste the reviews in the input text area of the GUI; (ii) load them from a text file, or **import them directly from Google Play** (specifying the url of the app as reported in the instructions of the provided `README.txt` file); (iii) select the desired combination of features she wants to exploit for the classification, and press the `Classify` button. For classifying multiple reviews, users can insert blank lines to separate the reviews to each other, as showed in Figure 2. At the end of the recognition process, all the recognized sentences will be highlighted with different colors depending on the categories the tool assigned to them.



```java
import java.util.ArrayList;
import org.ardoc.Parser;
import org.ardoc.Result;
import org.ardoc.UnknownCombinationException;


public class Main {
    public static void main(String args[]){
        String example = "You should add new levels for the game. " +
                         "These annoying ads block the whole app, please fix. ";
        Parser p = Parser.getInstance();
        try{
            ArrayList<Result> res =  p.extract("NLP+SA", example);
            for (Result r: res){
                System.out.println(r.getSentence()+"-"+r.getSentenceClass());

            }
        }catch(UnknownCombinationException ex){
            ex.printStackTrace();
        }
    }
}
```

Figure 3: ARdoc java API usage

The second version of ARdoc is a Java API that provides an easy way to integrate our classifier in other Java projects. Figure 3 shows an example of Java code that integrates the ARdoc's capabilities. To use it, it is necessary to download the `ARdoc_API.zip` from the tool's Web page, unzip it, and import the library `ARdoc_API.jar`, as well as the jars contained in the lib folder of `ARdoc_API.zip`, in the build path of the project. To use ARdoc it is sufficient to import the classes `org.ardoc.Parser` and `org.ardoc.Result` and instantiate the Parser through the method `getInstance`. The method `extract` of the class Parser represents the entry point to access to the tool's classification. This method accepts in input a String representing the combination of features the user wants to exploit, and a String containing the text to classify. The extract method returns a list of `Result` objects, providing all the methods to access to ARdoc's classification results.

4. EVALUATION

This section describes the methodology we used to evaluate the performance achieved by `ARdoc` and reports the obtained results. We evaluated the performance of our tool for three real-life applications. The original app developers shared with us the user reviews related to three real-life mobile apps: Minesweeper Reloaded[3], PowernAPP[4] and Picturex[5].

For the tuning of the tool we performed a first experiment using the user reviews related to Minesweeper Reloaded. In particular, we asked an external validator (a software engineer with experience in mobile development) to manually assign each sentence to one of the categories described in Table 1. We separately launched `ARdoc` on the same set of sentences and compared the labels assigned by the tool with the labels assigned by the human rater (a run for each possible features' combination). Table 2 reports (i) true positives, (ii) false positives, (iii) false negatives, (iv) precision, (v) recall, and (vi) F-Measure achieved for the different features' configurations.

Table 2: Classification results

Features Configuration	TP	FP	FN	PREC	REC	F-MEASURE
TEXT FEATURES (LEXICON)	32	89	89	0.264	0.264	0.264
TEXT STRUCTURES	102	19	19	0.843	0.843	0.843
TEXT FEATURES (LEXICON)+ TEXT STRUCTURES	94	27	27	0.777	0.777	0.777
TEXT STRUCTURES + SENTIMENT	**105**	**16**	**16**	**0.868**	**0.868**	**0.868**
TEXT STRUCTURES + TEXT FEATURES (LEXICON) + SENTIMENT	94	27	27	0.777	0.777	0.777

Outcomes in Table 2 are in line with results obtained in our previous work [11], in which we demonstrated that, among all the classification models we investigated, the best performing one employs the J48 machine learning algorithm and relies on the *structure* (i.e., extracted through the `NLP-Classifier`) and the *sentiment* (i.e., extracted through the `SAClassifier`) of sentences. These results also confirm the importance of text structures and sentiment features over the text features when classifying reviews into categories relevant to maintenance and evolution tasks.

Table 3: Classification results for PowernAPP

CATEGORY	TP	FP	FN	PREC.	RECALL	F-MEASURE
FEATURE REQUEST	46	6	16	0.885	0.742	0.807
PROBLEM DISCOVERY	29	4	7	0.879	0.806	0.841
INFORMATION SEEKING	11	1	3	0.917	0.786	0.846
INFORMATION GIVING	209	42	92	0.833	0.694	0.757
OTHER	999	110	45	0.901	0.957	0.928
TOTAL	**1294**	**163**	**163**	**0.888**	**0.888**	**0.888**

Table 4: Classification results for Picturex

CATEGORY	TP	FP	FN	PREC.	RECALL	F-MEASURE
FEATURE REQUEST	2	0	1	1.000	0.667	0.800
PROBLEM DISCOVERY	1	0	2	1.000	0.333	0.500
INFORMATION SEEKING	1	0	0	1.000	1.000	1.000
INFORMATION GIVING	25	5	20	0.833	0.556	0.667
OTHER	119	23	5	0.838	0.960	0.895
TOTAL	**148**	**28**	**28**	**0.841**	**0.841**	**0.841**

Then we performed a second experiment involving the user reviews related to the remaining apps. Specifically, we classified such reviews by using the best performing configuration (i.e., text structures + sentiment, which in our previous experiment achieved the best results) of `ARdoc`. We then asked the original developers of the apps to manually validate the classification performed by the tool, by reporting all the sentences having the wrong labels and assigning the right category to such sentences. Tables 3 and 4 report the results achieved by `ARdoc` in classifying the reviews related to PowernAPP and Picturex respectively. In particular, these tables show the amounts of (i) true positives, (ii) false positives, (iii) false negatives, (iv) precision, (v) recall, and (vi) F-Measure achieved for each category of sentences. For both apps, the `ARdoc` achieved a global classification accuracy ranging from 84.1% to 88.8%. For the two mobile apps `ARdoc` is able to classify with an high precision (i.e. 88.5% and 100%, respectively) and substantial high recall (i.e. 74.2% and 66.7%, respectively) the *Feature Requests*. `ARdoc` also classifies with a good accuracy (i.e. 84.1% and 50% respectively) sentences related to bug reports (i.e. *Problem Discovery*). Also in *Information Seeking* and *Information Giving* categories `ARdoc` achieves quite good classification results (i.e., 84.6% and 100% for *Information Seeking*, 75.7% and 66.7% for *Information Giving*). Thus, `ARdoc` classifies with an high accuracy (i.e., 92.8% and 89.5% respectively) sentences with irrelevant contents for developers (classified as *Other*). These results are also in line with previous literature [1–3] which demonstrated that approximately one third of the information contained in user reviews is helpful for developers. Indeed, useless sentences for developers (classified as *Other*) constitute the 71.7% and 70.5%, respectively, of all the sentences contained in the user reviews. Finally, the original developers of the selected apps considered `ARdoc` very useful for extracting useful feedback from app reviews, which is a very important task for meeting market requirements[6].

5. CONCLUSIONS

In this paper we presented `ARdoc` a novel tool able to extract structures, sentiment and lexicon features from app user reviews and combining them through ML techniques in order to extract important maintenance tasks for real world applications which is consider very important for developers. Experiments involving real-life applications demonstrated that our tool, by analyzing text structures in combination with sentiment features, is able to correctly classify useful feedback (from a maintenance perspective) contained in app reviews with a precision ranging between 84% and 89%, a recall ranging between 84% and 89%, and an F-Measure ranging between 84% and 89%. As first future work we plan to enhance `ARdoc` by improving the preprocessing part of the approach which combines text, sentiment and structure features, in order to achieve even better classification results. We plan to use `ARdoc` as a preprocessing support for summarization techniques in order to generate summaries of app reviews. Finally, the classification operated by `ARdoc` could be also used in combination with topic modeling techniques. Such a combination could be used, for example, to cluster all the feature requests (or bug reports) involving the same functionalities, in order to plan a set of code change tasks.

Acknowledgments

We thank Benjamin Sautermeister and André Meyer for helping us to evaluate the accuracy of ARdoc validating the results of the automatic classification on user reviews of their mobile apps. Sebastiano Panichella gratefully acknowledges the Swiss National Science foundation's support for the project *"Essentials"* (SNF Project No. 200020−153129).

[3]https://itunes.apple.com/us/app/minesweeper-reloaded/id477031499?mt=8

[4]http://www.bsautermeister.de/powernapp/

[5]www.picturexapp.com

[6]http://bsautermeister.blogspot.it/2015/12/app-reviews-zu-powernapp-dienen-als.html

6. 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, (New York, NY, USA), pp. 767–778, ACM, 2014.

[2] L. V. Galvis Carreño and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, (Piscataway, NJ, USA), pp. 582–591, IEEE Press, 2013.

[3] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study.," in *In Proceedings of the 21st IEEE International Requirements Engineering Conference (RE 2013)*, pp. 125–134, IEEE Computer Society, 2013.

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

[5] T. Vithani, "Modeling the mobile application development lifecycle," in *Proceedings of the International MultiConference of Engineers and Computer Scientists 2014, Vol. I*, IMECS 2014, pp. 596–600, 2014.

[6] H. Yang and P. Liang, "Identification and classification of requirements from app user reviews," in *Proceedings of the 27th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 7–12, Knowledge Systems Institute, 2015.

[7] E. Guzman and W. Maalej, "How do users like this feature? a fine grained sentiment analysis of app reviews," in *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pp. 153–162, Aug 2014.

[8] C. Iacob, R. Harrison, and S. Faily, "Online reviews as first class artifacts in mobile app development," in *Mobile Computing, Applications, and Services* (G. Memmi and U. Blanke, eds.), vol. 130 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 47–53, Springer International Publishing, 2014.

[9] 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*, pp. 760–770, 2015.

[10] 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 *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pp. 12–23, Nov 2015.

[11] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. 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*, pp. 281–290, Sept 2015.

[12] 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*, pp. 641–644, 2016.

[13] 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*, pp. 55–60, 2014.

[14] 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, (Stroudsburg, PA, USA), pp. 1–8, Association for Computational Linguistics, 2008.

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