How Can I Improve My App? Classifying User Reviews for Software Maintenance and Evolution

ICSME 2015

Jaymin Desai
jddesai2@ncsu.edu

ABSTRACT

App Stores, such as Google Play or the Apple Store, allow users to provide feedback on apps by posting review comments, which previous research showed contains usage scenarios, bug reports and feature requests, that can help app developers to accomplish software maintenance and evolution tasks. However, in the case of the most popular apps, the large amount of received feedback, its unstructured nature and varying quality can make the identification of useful user feedback a very challenging task.

This paper presents a taxonomy to classify app reviews into categories relevant to software maintenance and evolution, as well as an approach that merges three techniques: (1) Natural Language Processing, (2) Text Analysis and (3) Sentiment Analysis to automatically classify app reviews into the proposed categories; which allows to achieve better results (a precision of 75% and a recall of 74%) than results obtained using each technique individually in isolation (precision of 70% and a recall of 67%).

KEYWORDS

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

1 INTRODUCTION

Previous work [3], [4] has shown that approximately one third of the information contained in user reviews is helpful for developers. However, processing, analyzing and selecting useful user feedback involves several challenges. To handle this problem Chen *et al.* [3] proposed AR-Miner, an approach that uses (i) Text Analysis and Machine Learning to filter out non-informative reviews and (ii) Topic Analysis to recognize topics in reviews classified as informative.

In this paper, authors argue that text content represents just one of the possible dimensions that can be explored to detect informative reviews. In particular, topic analysis techniques are useful to discover topics treated in the review texts, but they are not able to reveal the reviewer‘s intentions. For example:

- *"The awful button in the page doesn‘t work"*
- *"A button in the page should be added"*

Topic analysis will reveal that these two reviews are likely to discuss the same topics: *"button"* and *"page"*. However, these reviews have different intentions: in review *(1)* the user has exposed a problem related to the app, while in the review *(2)* the user asks for the implementation of a new feature; which illustrates the importance of understanding the *intention* in user reviews.

Foundations of Software Science, Fall 2018, Computer Science

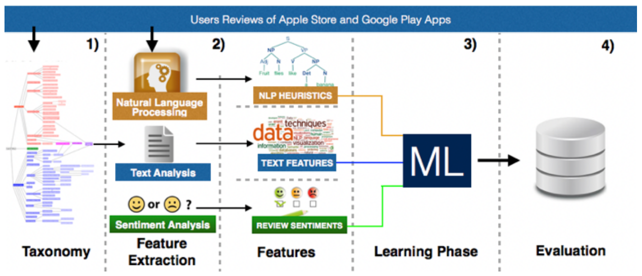


Figure 1: Overview Research Approach

This paper emphasizes on following research questions:

- **RQ1:** Are the language structure *(NLP)*, content *(Text Analysis)* and sentiment information *(Sentiment Analysis)* able to identify user reviews that could help developers in accomplishing software maintenance and evolution tasks?
- **RQ2:** Does the combination of language structure, content and sentiment information produce better results than individual techniques used in isolation?

2 APPROACH

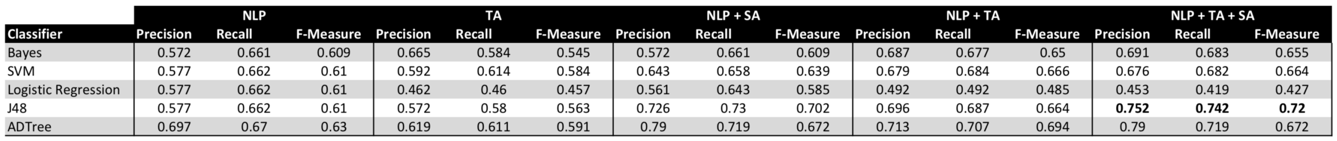
2.1 Taxonomy for User Review Categories

Users reviews data of seven Apple Store and Google Play apps viz. *AngryBirds, Evernote, TripAdvisor, Pintrest, WhatsApp, PicsArt, Dropbox*; belonging to six different categories was rigorously analyzed at a **sentence-level granularity**. The reviews were then classified into six categories in context of application development realm. Two categories were discarded after applying Topic Modeling on the categories as previously proposed by Pagano *et al.* [4].

- Feature Request
- Opinion Asking (discarded)
- Problem Discovery
- Solution Proposal (discarded)
- Information Seeking
- Information Giving

2.2 Text Analysis

Preprocessing: Stop-word removal and stemming were performed on all the terms in the user reviews.
Text Feature Weighing: BagOfWords or simple Term Frequency transformation was used to vectorize the test data.
Output of this phase was a Matrix with columns as sentences and rows as the weight of a term across the corpus comprised of all the reviews.



	NLP			TA			NLP + SA			NLP + TA			NLP + TA + SA		
Classifier	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Bayes	0.572	0.661	0.609	0.665	0.584	0.545	0.572	0.661	0.609	0.687	0.677	0.65	0.691	0.683	0.655
SVM	0.577	0.662	0.61	0.592	0.614	0.584	0.643	0.658	0.639	0.679	0.684	0.666	0.676	0.682	0.664
Logistic Regression	0.577	0.662	0.61	0.462	0.46	0.457	0.561	0.643	0.585	0.492	0.492	0.485	0.453	0.419	0.427
J48	0.577	0.662	0.61	0.572	0.58	0.563	0.726	0.73	0.702	0.696	0.687	0.664	**0.752**	**0.742**	**0.72**
ADTree	0.697	0.67	0.63	0.619	0.611	0.591	0.79	0.719	0.672	0.713	0.707	0.694	0.79	0.719	0.672

Figure 2: Results

2.3 Natural Language Processing

Based on the assumption that users write app reviews in recurrent linguistic patterns, generalized NLP Heuristics comprising of 246 patterns was prepared after manually inspecting 500 reviews which enables the automatic detection of a sentence which matches a specific structure, for example:

- *"[someone] should add [something]"*
- *"[someone] would love to see/use [something]"*

Stanford Typed Dependencies (STD) parser [2], a widely acknowledged and tested tool that resolves dependencies between individual words contained in sentences and to label each of them with a specific grammatical relation was used to assign each sentence in the input to its corresponding NLP Heuristic.

2.4 Sentiment Analysis

The classes for this research are defined as three different levels of sentiment intensity: positive, negative and neutral. Naive Bayes was used for predicting the sentiment in the user reviews as previous work [1] found that Naive Bayes performed better than other machine learning algorithms traditionally used for text classification when analyzing the sentiment in movie reviews.

The Sentiment Analysis task was performed using the Weka tool, generating as output of this step an integer value such as, 1 determines positive sentiments, whereas 0 and -1 denote neutral and negative sentiments respectively.

2.5 Learning Classifiers

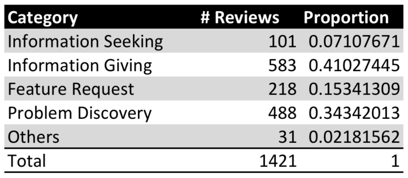
Five classifiers listed below, were used along with 10 fold cross-validation to predict the categories as per the taxonomy presented in Section 2.1 for software maintenance and evolution:

- The standard probabilistic Naive Bayes Classifier
- Logistic Regression
- Support Vector Machines
- J48
- Alternating Decision tree (ADTree)

The choice of these techniques is not random since they have been successfully used for bug reports classification and for defect prediction in many previous works, thus allowing to increase the generalisability of the findings.

3 DATASET

The unprocessed raw data was reduced of non-informative reviews using AR-Miner [3], and then a sample of dataset sentences selected using stratified random sampling strategy was manually labeled. During the sampling it was verified that the percentage of the number of extracted sentences per app was the same as the percentage of reviews per app in the original set. In total 1421 sentences out of 7696 reviews (18.46%) were sampled as shown in Figure 3.

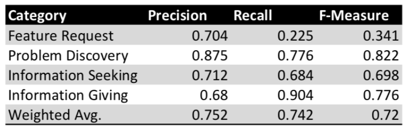


Category	# Reviews	Proportion
Information Seeking	101	0.07107671
Information Giving	583	0.41027445
Feature Request	218	0.15341309
Problem Discovery	488	0.34342013
Others	31	0.02181562
Total	1421	1

Figure 3: Labeled Trust Set Instances

4 RESULTS

The results in Figure 2 show that the NLP + TA + SA configuration had the best results with the J48 algorithm, among all possible feature inputs and classifiers with 75% precision and 74% recall as shown in Figure 4. In particular, *problem discovery* was the category with the highest F-measure and *feature request category was the category with the lowest F-measure (with precision of 70% but very low recall of 23%).*



Category	Precision	Recall	F-Measure
Feature Request	0.704	0.225	0.341
Problem Discovery	0.875	0.776	0.822
Information Seeking	0.712	0.684	0.698
Information Giving	0.68	0.904	0.776
Weighted Avg.	0.752	0.742	0.72

Figure 4: Results of J48 Algorithm

The results of Friedman test, followed by a post-hoc Nemenyi test revealed that the difference in performance among the classifiers is not statistical significant in terms of F-Measure.

5 CONCLUSIONS AND FUTURE WORK

Results show that some configurations substantially improve both precision and recall when increasing the size of the training set (changing the size of training set (60%) with J48). Additionally, it was also deduced that a classifier trained with structure (with NLP) and sentiment (with SA) features performs significantly better than when only trained with text (with TA) features.

REFERENCES

[1] L. Lee B. Pang and S. Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *ACL-02, Conference on Empirical Methods in Natural Language Processing*. 79–86.

[2] B. MacCartney M.C. de Marneffe and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*. 449–454.

[3] S.C.H. Hoi X. Xiao N. Chen, J. Lin and B. Zhang. 2014. AR-miner: mining informative reviews for developers from mobile app marketplace. In *36th International Conference on Software Engineering (ICSE)*. 767–778.

[4] D. Pagano and W. Maalej. 2013. User Feedback in the AppStore: An Empirical Study. In *21st IEEE International Requirements Engineering Conference (RE)*. 125–134.