Requirements Eng (2016) 21:311–331
DOI 10.1007/s00766-016-0251-9

RE 2015

On the automatic classification of app reviews

Walid Maalej[1] · Zijad Kurtanović[1] · Hadeer Nabil[2] · Christoph Stanik[1]

Walid: please look at my highlightings and added stickies. Look in particular at the sticky I attached to highlighted text on page 321 (11 or 21).
Dan

Received: 14 November 2015 / Accepted: 26 April 2016 / Published online: 14 May 2016

Abstract App stores like Google Play and Apple AppStore have over 3 million apps covering nearly every kind of software and service. Billions of users regularly download, use, and review these apps. Recent studies have shown that reviews written by the users represent a rich source of information for the app vendors and the developers, as they include information about bugs, ideas for new features, or documentation of released features. The majority of the reviews, however, is rather non-informative just praising the app and repeating to the star ratings in words. This paper introduces several probabilistic techniques to classify app reviews into four types: bug reports, feature requests, user experiences, and text ratings. For this, we use review metadata such as the star rating and the tense, as well as, text classification, natural language processing, and sentiment analysis techniques. We conducted a series of experiments to compare the accuracy of the techniques and compared them with simple string matching. We found that metadata alone results in a poor classification accuracy. When combined with simple text classification and natural language preprocessing of the text—particularly with bigrams and lemmatization—the classification precision for all review types got up to 88–92 % and the recall up to 90–99 %. Multiple binary classifiers outperformed single multiclass classifiers. Our results inspired the design of a review analytics tool, which should help app vendors and developers deal with the large amount of reviews, filter critical reviews, and assign them to the appropriate stakeholders. We describe the tool main features and summarize nine interviews with practitioners on how review analytics tools including ours could be used in practice.

Keywords User feedback · Review analytics · Software analytics · Machine learning · Natural language processing · Data-driven requirements engineering

1 Introduction

Nowadays it is hard to imagine a business or a service that does not have any app support. In July 2014, leading app stores such as Google Play, Apple AppStore, and Windows Phone Store had over 3 million apps.[1] The app download numbers are astronomic with hundreds of billions of downloads over the last 5 years [9]. Smartphone, tablet, and more recently also desktop users can search the store for the apps, download, and install them with a few clicks. Users can also review the app by giving a star rating and a text feedback.

Studies highlighted the importance of the reviews for the app success [22]. Apps with better reviews get a better ranking in the store and with it a better visibility and higher sales and download numbers [6]. The reviews seem to help users navigate the jungle of apps and decide which one to use. Using free text and star rating, the users are able to express their satisfaction, dissatisfaction or ask for missing features. Moreover, recent research has pointed the potential importance of the reviews for the app developers and vendors as well. A significant amount of the reviews

✉ Walid Maalej
maalej@informatik.uni-hamburg.de

[1] Department of Informatics, University of Hamburg, Hamburg, Germany

[2] German University of Cairo, Cairo, Egypt

[1] http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

Table 2 Overview of the evaluation data

App(s)	Category	Platform	#Reviews	Sample
1100 apps	All iOS	Apple	1,126,453	1000
Dropbox	Productivity	Apple	2009	400
Evernote	Productivity	Apple	8878	400
TripAdvisor	Travel	Apple	3165	400
80 apps	Top four	Google	146,057	1000
PicsArt	Photography	Google	4438	400
Pinterest	Social	Google	4486	400
Whatsapp	Communication	Google	7696	400
Total			1,303,182	4400

From the collected data, we randomly sampled a subset for the manual labeling as shown in Table 2. We selected 1000 random reviews from the Apple store data and 1000 from the Google store data. To ensure that enough reviews with 1, 2, 3, 4, and 5 stars are sampled, we split the two 1000-review samples into 5 corresponding subsamples each of size 200. Moreover, we selected 3 random Android apps and 3 iOS apps from the top 100 and fetched their reviews between 2012 and 2014. From all reviews of each app, we randomly sampled 400. This led to additional 1200 iOS and 1200 Android app-specific reviews. In total, we had 4400 reviews in our sample.

For the truth set creation, we conducted a *peer, manual content analysis* for all the 4400 reviews. Every review in the sample was assigned randomly to 2 coders from a total of 10 people. The coders were computer science master students, who were paid for this task. Every coder read each review carefully and indicated its types: bug report, feature request, user experience, or rating. We briefed the coders in a meeting, introduced the task, the review types, and discussed several examples. We also developed a coding guide, which describes the coding task, defines precisely what each type is, and lists examples to reduce disagreements and increase the quality of the manual labeling. Finally, the coders were able to use a coding tool (shown on Fig. 1) that helps to concentrate on one review at once and to reduce coding errors. If both coders agreed on a review type, we used that label in our golden standard. A third coder checked each label and solved the disagreements for a review type by either accepting the proposed label for this type or rejecting it. This ensured that the golden set contained only peer-agreed labels.

In the third phase, we used the manually labeled reviews to train and to test the classifiers. A summary of the experiment data is shown in Table 3. We only used reviews, for which both coders *agreed* that they are of a certain type or not. This helped that a review in the corresponding evaluation sample (e.g., bug reports) is labeled correctly. Otherwise training and testing the classifiers on unclear data will lead to unreliable results. We evaluated the different techniques introduced in Sect. 2, while varying the classification features and the machine learning algorithms.

We evaluated the classification accuracy using the standard metrics precision and recall. Precision_i is the fraction of reviews that are classified correctly to belong to type i. Recall_i is the fraction of reviews of type i which are classified correctly. They were calculated as follows:

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \qquad \text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{1}$$

TP_i is the number of reviews that are classified as type i and actually are of type i. FP_i is the number of reviews that are classified as type i but actually belong to another type j where $j \neq i$. FN_i is the number of reviews that are classified to other type j where $j \neq i$ but actually belong to type i. We also calculated the F-measure ($F1$), which is the harmonic mean of precision and recall providing a single accuracy measure. We randomly split the truth set at a ratio of 70:30. That is, we randomly used 70 % of the data for the training set and 30 % for the test set. Based on the size of our truth set, we felt this ratio is a good trade-off for having large-enough training and test sets. Moreover, we experimented with other ratios and with the cross-validation method. We also calculated how informative the classification features are and ran paired t tests to check whether the differences of $F1$-scores are statistically significant.

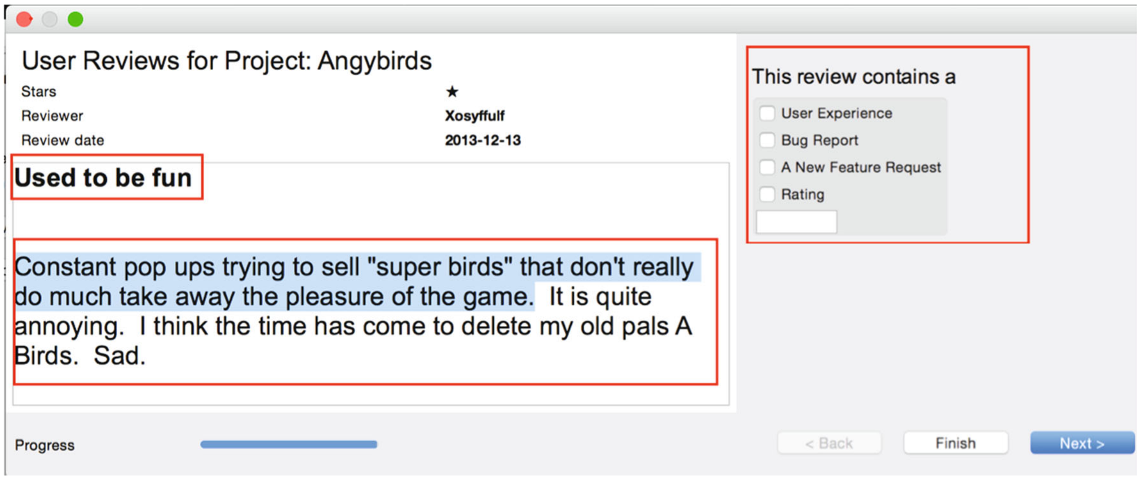
The results reported in Sect. 4 are obtained using the Monte Carlo cross-validation [38] method with 10 runs and random 70:30 split ratio. That is, for each run, 70 % of the truth set (e.g., for true positive bug reports) is randomly selected and used as a training set and the remaining 30 % is used as a test set. Additional experiments data, scripts, and results are available on the project Web site: http://mast.informatik.uni-hamburg.de/app-review-analysis/.

4 Research results

We report on the results of our experiments and compare the accuracy (i.e., precision, recall, and F-measures) as well as the performance of the various techniques.

4.1 Classification techniques

Table 4 summarizes the results of the classification techniques using Naive Bayes classifier on the whole data of the truth set (from the Apple AppStore and the Google Play Store). The results in Table 4 indicate the mean values obtained by the cross-validation for each single combination of classification techniques and a review type. The



User Reviews for Project: Angybirds

Stars ★

Reviewer Xosyffulf

Review date 2013-12-13

Used to be fun

Constant pop ups trying to sell "super birds" that don't really do much take away the pleasure of the game. It is quite annoying. I think the time has come to delete my old pals A Birds. Sad.

This review contains a

- [ ] User Experience
- [ ] Bug Report
- [ ] A New Feature Request
- [ ] Rating

Fig. 1 Tool for manual labeling of the reviews

Table 3 Number of manually analyzed and labeled reviews

Sample	Manually analyzed	Bug reports	Feature requests	User experiences	Ratings
Random apps Apple	1000	109	83	370	856
Selected apps Apple	1200	192	63	274	373
Random apps Google	1000	27	135	16	569
Selected apps Google	1200	50	18	77	923
Total	4400	378	299	737	2721

numbers in bold represent the highest scores for each column, which means the highest accuracy metric (precision, recall, and *F*-measure) for each classifier.

Table 5 shows the *p* values of paired *t* tests on whether the differences between the mean *F*1-scores of the baseline classifier and the various classification techniques are statistically significant. For Example: If one classifier result is 80 % for a specific combination of techniques and another result is 81 % for another combination, those two results could be statistically different or it could be by chance. If the *p* value calculated by the paired *t* test is very small, this means that the difference between the two values is statistically significant. We used Holm's step-down method [16] to control the family-wise error rate.

Overall, the precisions and recalls of all probabilistic techniques were clearly higher than 50 % except for three cases: the precision and recall of feature request classifiers based on rating only as well as the recall of the same technique (rating only) to predict ratings. Almost all probabilistic approaches outperformed the basic classifiers that use string matching with at least 10 % higher precisions and recalls.

The combination of text classifiers, metadata, NLP, and the sentiments extraction generally resulted in high precision and recall values (in most cases above 70 %). However, the combination of the techniques did not always rank best. Classifiers only using metadata generally had a rather low precision but a surprisingly high recall except for predicting ratings where we observed the opposite.

Concerning NLP techniques, there was no clear trend like "more language processing leads to better results." Overall, removing stopwords significantly increased the precision to predict bug reports, feature request, and user experience, while it decreased the precision for ratings. We observed the same when adding lemmatization. On the other hand, combining stop word removal and lemmatization did not had any significant effect on precision and recall.

We did not observe any significant difference between using one or two sentiment scores.

4.2 Review types

We achieved the highest precision for predicting user experience and ratings (92 %), the highest recall, and *F*-measure for user experience (respectively, 99 and 92 %).

For bug reports we found that the highest precision (89 %) was achieved with the bag of words, rating, and one sentiment, while the highest recall (98 %) with using bigrams, rating, and one score sentiment. For predicting bug reports the recall might be more important than precision. Bug reports are critical reviews, and app vendors would probably need to make sure that a review analytics

Table 4 Accuracy of the classification techniques using Naive Bayes on app reviews from Apple and Google stores (mean values of the 10 runs, random 70:30 splits for training:evaluation sets)

Classification techniques	Bug reports			Feature requests			User experiences			Ratings		
	Precision	Recall	*F*1	Precision	Recall	*F*1	Precision	Recall	*F*1	Precision	Recall	*F*1
Basic (string matching)	0.58	0.24	0.33	0.39	0.55	0.46	0.27	0.12	0.17	0.74	0.56	0.64
Document classification (&NLP)												
Bag of words (BOW)	0.79	0.65	0.71	0.76	0.54	0.63	0.82	0.59	0.68	0.67	0.85	0.75
FR Bigram	0.68	**0.98**	0.80	0.68	**0.97**	0.80	0.70	**0.99**	0.82	0.91	0.62	0.73
BOW + bigram	0.85	0.90	0.87	0.86	0.85	0.85	0.87	0.91	0.89	0.85	0.89	0.87
BOW + lemmatization	0.88	0.74	0.80	0.86	0.65	0.74	0.90	0.67	0.77	0.73	0.91	0.81
BOW − stopwords	0.86	0.69	0.76	0.86	0.65	0.74	0.91	0.67	0.77	0.74	0.91	0.81
BOW + lemmatization − stopwords	0.85	0.71	0.77	0.87	0.67	0.76	0.91	0.67	0.77	0.75	0.90	0.82
FR BOW + bigrams − stopwords + lemmatization	0.85	0.91	**0.88**	0.86	0.83	**0.85**	0.89	0.94	0.91	0.85	0.90	**0.87**
Metadata												
Rating	0.64	0.82	0.72	0.31	0.35	0.31	0.74	0.89	0.81	0.72	0.34	0.46
Rating + length	0.76	0.75	0.75	0.68	0.67	0.67	0.72	0.82	0.77	0.70	0.68	0.69
Rating + length + tense	0.74	0.73	0.74	0.64	0.71	0.67	0.74	0.80	0.77	0.70	0.68	0.69
Rating + length + tense + 1× sentiment	0.69	0.76	0.72	0.66	0.66	0.66	0.71	0.85	0.77	0.71	0.66	0.68
Rating + length + tense + 2× sentiments	0.66	0.78	0.71	0.65	0.72	0.68	0.67	0.88	0.76	0.69	0.67	0.68
Combined (text and metadata)												
BOW + rating + lemmatize	0.85	0.73	0.78	**0.89**	0.64	0.74	0.90	0.67	0.77	0.73	0.89	0.80
FR BOW + rating + 1× sentiment	**0.89**	0.72	0.79	**0.89**	0.60	0.71	**0.92**	0.73	0.81	0.75	**0.93**	0.83
BOW + rating + tense + 1 sentiment	0.87	0.71	0.78	0.87	0.60	0.70	**0.92**	0.69	0.79	0.74	0.90	0.81
UE BR Bigram + rating + 1× sentiment	0.73	**0.98**	0.83	0.71	0.96	0.81	0.75	**0.99**	0.85	0.92	0.69	0.79
Bigram − stopwords + lemmatization + rating + tense + 2× sentiment	0.72	0.97	0.82	0.70	**0.94**	0.80	0.75	0.98	0.85	0.92	0.72	0.81
BOW + bigram + tense + 1× sentiment	0.87	0.88	0.87	0.85	0.83	0.83	0.88	0.94	0.91	0.83	0.87	0.85
UE BR BOW + lemmatize + bigram + rating + tense	0.88	0.88	**0.88**	0.87	0.84	**0.85**	0.89	0.94	**0.92**	0.84	0.90	**0.87**
BOW − stopwords + bigram + rating + tense + 1× sentiment	0.88	0.89	**0.88**	0.86	0.84	**0.85**	0.87	0.93	0.90	0.83	0.89	0.86
BOW − stopwords + lemmatization + rating + 1× sentiment + tense	0.88	0.71	0.79	0.87	0.64	0.74	0.91	0.72	0.80	0.73	0.90	0.80
BOW − stopwords + lemmatization + rating + 2× sentiments + tense	0.87	0.71	0.78	0.86	0.68	0.76	0.91	0.73	0.81	0.75	0.90	0.82

Bold values represent the highest score for the corresponding accuracy metric per review type

Table 5 Results of the paired *t* test between the different techniques (one in each row) and the baseline BoW (using Naive Bayes on app reviews from Apple and Google stores)

Classification techniques	Bug reports		Feature requests		User experiences		Ratings	
	*F*1-score	*p* value	*F*1-score	*p* value	*F*1-score	*p* value	*F*1-score	*p* value
Document classification (&NLP)								
Bag of words (BOW)	0.71	Baseline	0.63	Baseline	0.68	Baseline	0.75	Baseline
Bigram	0.80	0.043	0.80	2.5e−06	0.82	0.00026	0.73	0.55
BOW + bigram	0.87	6.9e−05	0.85	2.6e−07	0.89	4.7e−06	0.87	2.9e−05
BOW + lemmatization	0.80	0.031	0.74	0.0022	0.77	0.0028	0.81	0.029
BOW − stopwords	0.76	0.09	0.74	0.0023	0.77	0.0017	0.81	0.0019
BOW − stopwords + lemmatization	0.77	0.051	0.76	0.0008	0.77	0.0021	0.82	0.0005
BOW − stopwords + lemmatization + bigram	0.88	6.6e−05	0.85	2.9e−07	0.91	4.3e−08	0.87	0.0009
Metadata								
Rating	0.72	1.0	0.31	0.04	0.81	7.1e−05	0.46	6.9e−06
Rating + length	0.75	0.09	0.67	0.04	0.77	0.0005	0.69	0.0098
Rating + length + tense	0.74	0.63	0.67	0.083	0.77	0.0029	0.69	0.029
Rating + length + tense + 1× sentiment	0.73	1.0	0.66	0.16	0.77	0.004	0.68	8.9e−05
Rating + length + tense + 2× sentiments	0.71	1.0	0.68	0.0002	0.76	0.028	0.68	0.029
Combined (text and metadata)								
BOW + rating + lemmatize	0.78	0.064	0.74	0.0005	0.77	0.0023	0.80	0.0044
BOW + rating + 1× sentiment	0.79	0.0027	0.71	0.039	0.81	0.0002	0.83	0.001
BOW + rating + 1 sentiment + tense	0.78	0.0097	0.70	0.039	0.79	0.0002	0.81	0.0012
Bigram + rating + 1 sentiment	0.83	0.0039	0.81	9.5e−06	0.85	2e−05	0.79	0.042
Bigram − stopwords + lemmatization + rating + tense + 2× sentiment	0.82	0.0019	0.80	1.7e−06	0.85	2.5e−05	0.81	0.029
BOW + bigram + tense + 1× sentiment	0.87	0.0001	0.83	1.2e−05	0.91	1.9e−07	0.85	0.0002
BOW + lemmatize + bigram + rating + tense	0.88	7.6e−06	0.85	7.6e−07	0.92	1.2e−07	0.87	1.6e−05
BOW − stopwords + bigram + rating + tense + 1× sentiment	0.88	1.6e−06	0.85	7.6e−07	0.90	4.8e−06	0.86	0.0002
BOW − stopwords + lemmatization + rating + tense + 1× sentiment	0.79	0.064	0.74	0.0008	0.80	0.0014	0.80	0.029
BOW − stopwords + lemmatization + rating + tense + 2× sentiments	0.78	0.051	0.76	0.0012	0.81	0.0003	0.82	0.0002

tool does not miss any of them, with the compromise that a few of the reviews predicted as bug reports are actually not (false positives). For a balance between precision and recall combining bag of words, lemmatization, bigram, rating, and tense seems to work best.

Concerning feature requests, using the bag of words, rating, and one sentiment resulted in the highest precision with 89 %. The best *F*-measure was 85 % with bag of words, lemmatization, bigram, rating, and tense as the classification features.

The results for predicting user experiences were surprisingly high. We expect those to be hard to predict as the basic technique for user experiences shows. The best option that balances precision and recall was to combine bag of words with bigrams, lemmatization, the rating, and the tense. This option achieved a balanced precision and recall with a *F*-measure of 92 %.

Predicting ratings with the bigram, rating, and one sentiment score leads to the top precision of 92 %. This result means that stakeholders can precisely select rating among many reviews. Even if not all ratings are selected (false negatives) due to average recall, those that are selected will be very likely ratings. A common use case would be to filter out reviews that only include ratings or to select another type of reviews with or without ratings.

Table 6 shows the ten most informative features of a combined classification technique for each review type.

4.3 Classification algorithms

Table 7 shows the results of comparing the different machine learning algorithms Naive Bayes, Decision Trees, and MaxEnt. We report on two classification techniques (bag of words and bag of words + metadata) since the other results are consistent and can be downloaded from the project Web site.[2] In all experiments, we found that binary

[2] http://mast.informatik.uni-hamburg.de/app-review-analysis/.

Table 6 Most informative features for the classification technique bigram − stop words + lemmatization + rating + 2× sentiment scores + tense

Bug report	Feature request	User experience	Rating
Rating (1)	Bigram (way to)	Rating (3)	Bigram (will not)
Rating (2)	Bigram (try to)	Rating (1)	Bigram (to download)
Bigram (every time)	Bigram (would like)	Bigram (use to)	Bigram (use to)
Bigram (last update)	Bigram (5 star)	Bigram (to find)	Bigram (new update)
Bigram (please fix)	Rating (1)	Bigram (easy to)	Bigram (fix this)
Sentiment (−4)	Bigram (new update)	Bigram (go to)	Bigram (can get)
Bigram (new update)	Bigram (back)	Bigram (great to)	Bigram (to go)
Bigram (to load)	Rating (2)	Bigram (app to)	Rating (1)
Bigram (it can)	Present cont. (1)	Bigram (this great)	Bigram (great app)
Bigram (can and)	Bigram (please fix)	Sentiment (−3)	Present simple (1)

Table 7 *F*-measures of the evaluated machine learning algorithms (B = binary classifier, MC = multiclass classifiers) on app reviews from Apple and Google stores

Type	Technique	Bug R.	*F* req.	*U* exp.	Rat.	Avg.
Naive Bayes						
B	Bag of words (BOW)	0.71	0.63	0.68	0.75	0.70
MC	Bag of words	0.66	0.31	0.43	0.59	**0.50**
B	BOW + metadata	0.79	0.71	0.81	0.83	**0.79**
MC	BOW + metadata	0.62	0.42	0.50	0.58	0.53
Decision Tree						
B	Bag of words	0.81	0.77	0.82	0.79	**0.79**
MC	Bag of words	0.49	0.32	0.44	0.52	0.44
B	BOW + metadata	0.73	0.68	0.78	0.78	0.72
MC	BOW + metadata	0.62	0.47	0.53	0.54	0.54
MaxEnt						
B	Bag of words	0.66	0.65	0.58	0.67	0.65
MC	Bag of words	0.26	0.00	0.12	0.22	0.15
B	BOW + metadata	0.66	0.65	0.60	0.69	0.65
MC	BOW + metadata	0.14	0.00	0.29	0.04	0.12

classifiers are more accurate for predicting the review types than multiclass classifiers. One possible reason is that each binary classifier uses two training sets: one set where the corresponding type is observed (e.g., bug report) and one set where it is not (e.g., not bug report). Concerning the binary classifiers Naive Bayes outperformed the other algorithms. In Table 7, the numbers in bold represent the highest average scores for the binary (B) and multiclass (MC) case.

4.4 Performance and data

The more data are used to train a classifier the more time the classifier would need to create its prediction model. This is depicted in Fig. 2 where we normalized the *mean time* needed for the four classifiers depending on the size of the training set.

In this case, we used a consistent size for the test set of 50 randomly selected reviews to allow a comparison of the results.

We found that when using more than 200 reviews to train the classifiers the time curve gets much more steep with a rather exponential than a linear shape. For instance, the time needed for training almost doubles when the training size grows from 200 to 300 reviews. We also found that MaxEnt needed much more time to build its model compared to all other algorithms for binary classification. Using the classification technique BoW and Metadata, MaxEnt took on average ∼40 times more than Naive Bayes and ∼1.36 times more than Decision Tree learning.

These numbers exclude the overhead introduced by the sentiment analysis, the lemmatization, and the tense detection (part-of-speech tagging). The performance of these techniques is studied well in the literature [4], and their overhead is rather exponential to the text length. However, the preprocessing can be conducted once on each review and stored separately for later usages by the classifiers. Finally, stopword removal introduces a minimal overhead that is linear to the text length.

Figure 3 shows how the accuracy changes when the classifiers use larger training sets. The precision curves are

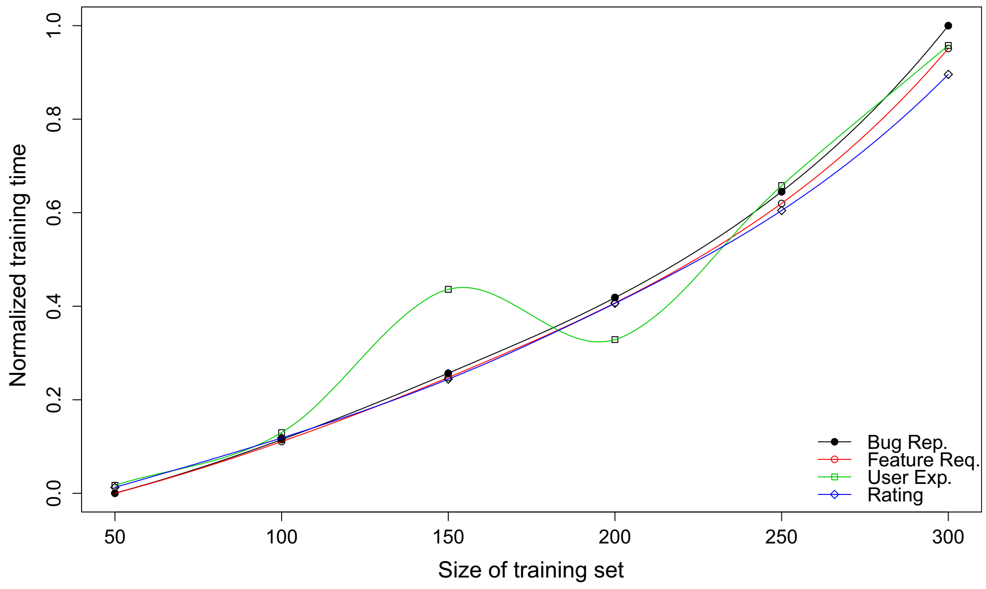


Fig. 2 How the size of the training set influences the time to build the classification model (Naive Bayes using BoW + rating + lemmatization (see Table 4))

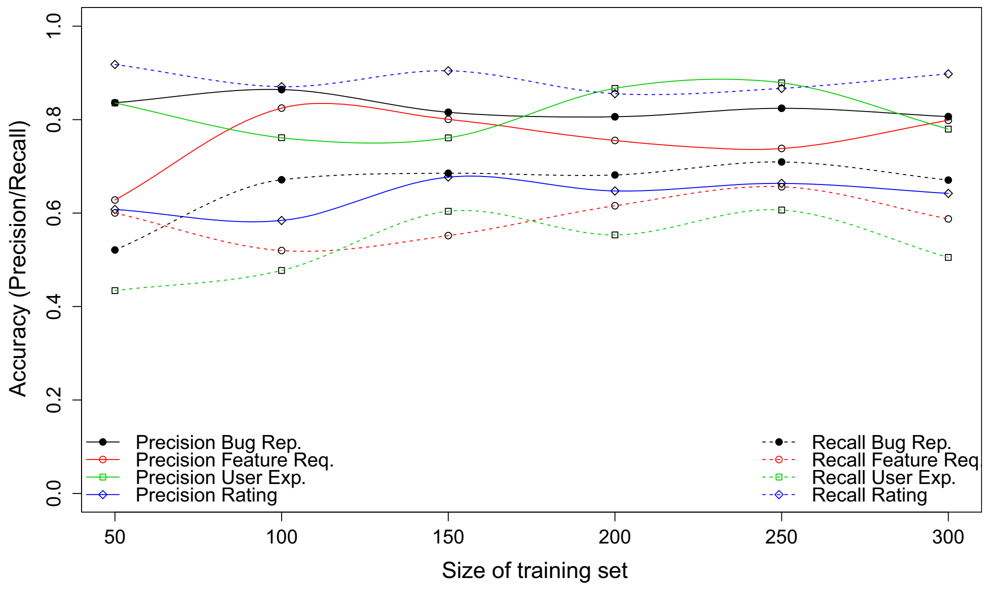


Fig. 3 How the size of the training set influences the classifier accuracy (Naive Bayes using BoW + rating + lemmatization (see Table 4))

represented with continuous lines, while the recall curves are dotted. From Figs. 2 and 3 it seems that 100–150 reviews are a good size of the training sets for each review type, allowing for a high accuracy while saving resources. With an equal ratio of candidate and non-candidate reviews the expected size of the training set doubles leading to 200–300 reviews per classifier recommended for training.

Finally, we also compared the accuracy of predicting the Apple AppStore reviews with the Google Play Store reviews. We found that there are differences in predicting the review types between both app stores as shown in Tables 8 and 9. The highest values of a metric are emphasized as bold for each review type. The biggest difference in both stores is in predicting bug reports. While the top value for *F*-measure for predicting bugs in the Apple AppStore is 90 %, the *F*-measure for the Google Play Store is 80 %. A reason for this difference might be that we had less labeled reviews for bug reports in the Google Play Store. On the other hand, feature requests in the Google Play Store have a promising precision of 96 % with a recall of 88 %, while the precision in the Apple AppStore is 88 % with a respective recall of 84 %, by