How Should Developers Respond to App Reviews? Features Predicting the Success of Developer Responses

Kamonphop Srisopha
University of Southern California
Los Angeles, California, USA
srisopha@usc.edu

Daniel Link
University of Southern California
Los Angeles, California, USA
dlink@usc.edu

Barry Boehm
University of Southern California
Los Angeles, California, USA
boehm@usc.edu

ABSTRACT

Context: The Google Play Store allows app developers to respond to user reviews. Existing research shows that response strategies vary considerably. In addition, while responding to reviews can lead to several types of favorable outcomes, not every response leads to *success*, which we define as increased user ratings.
Aims: This work has two objectives. The first is to investigate the potential to predict early whether a developer response to a review is likely to be successful. The second is to pinpoint how developers can increase the chance of their responses to achieve success.
Method: We track changes in user reviews of the 1,600 top free apps over a ten-week period, and find that in 11,034 out of 228,274 one- to four-star reviews, the ratings increase after a response. We extract three groups of features, namely time, presentation and tone, from the responses given to these reviews. We apply the extreme gradient boosting (XGBoost) algorithm to model the success of developer responses using these features. We employ model interpretation techniques to derive insights from the model.
Results: Our model can achieve an AUC of 0.69, thus demonstrating that feature engineering and machine learning have the potential to enable developers to estimate the probability of success of their responses at composition time. We learn from it that the ratio between the length of the review and response, the textual similarity between the review and response, and the timeliness and the politeness of the response have the highest predictive power for distinguishing successful and unsuccessful developer responses.
Conclusions: Based on our findings, we provide recommendations that developers can follow to increase the chance of success of their responses. Tools may also leverage our findings to support developers in writing more effective responses to reviews on the app store.

CCS CONCEPTS

• **Information systems → Content analysis and feature selection**; **Retrieval models and ranking**; • **Computing methodologies → Model development and analysis**.

KEYWORDS

Developer Response, User Reviews, Feature Importance, Partial Dependence Plot, Extreme Gradient Boosting, App Store Mining

ACM Reference Format:
Kamonphop Srisopha, Daniel Link, and Barry Boehm. 2021. How Should Developers Respond to App Reviews? Features Predicting the Success of Developer Responses. In *Evaluation and Assessment in Software Engineering (EASE 2021), June 21–23, 2021, Trondheim, Norway.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3463274.3463311

1 INTRODUCTION

Since 2013, the Google Play Store has allowed developers to respond to reviews and establish two-way communications with their users. However, only a small fraction of reviews are read and garner a response from developers, and there is a considerable variation in response strategies [20, 38, 40]. Despite the sizable body of literature on mobile app reviews [29], there have been no findings in literature on how developers should respond to app reviews.

Nonetheless, recent studies agree that developers are more likely to respond to reviews with negative ratings than those with positive ones. One reason why mobile apps are particularly vulnerable to negative rating reviews is that the more positive ratings and reviews an app has, the higher it will rank in the search results, making it more visible and encouraging downloads [19, 26]. For this reason, reviews with negative ratings present great opportunities for developers to respond for potential ratings increase [20, 40]. Additionally, most research in this area has treated user reviews and developer responses as static when in fact they can change over time. This inspired us to compare the attributes of a review before and immediately after a response in order to determine the effectiveness of the response.

We are interested in two objectives. The first is investigating the potential to predict the probability of a successful response early (manifested in a higher rating). The second is to pinpoint how developers can increase that probability. To achieve our objectives, we tracked changes to user review ratings and developer responses of the 1,600 top free-to-download apps in the Google Play Store for ten weeks. We then identified the features of successful developer responses and explored how they differ from the unsuccessful ones. Specifically, we focused on features that developers can control when composing a response. We extracted three groups of features, namely Time, Presentation and Tone. We then utilized the extreme gradient boosting algorithm (XGBoost) to model the success of developer responses from these features. A ranking of features by their importance and a set of important features were extracted from the model. Partial dependence plots were generated for the important features to uncover how the probability of success of responses changes with these features. We believe that our findings could be of practical significance to a broad range of practitioners.

Our primary contributions can be summarized as follows:

- We conduct an empirical investigation of features that predict the success of developer responses.
- We demonstrate that feature engineering and machine learning have the potential to enable developers to estimate the probability of success of their responses.
- We provide a set of evidence-based recommendations that developers can follow to write effective responses and increase the chance of success of the responses.

The rest of the paper is structured as follows. Section 2 describes the research setup. Section 3 presents the experiment results. Section 4 explores the implications for practical use of the findings. Section 5 identifies limitations and threats to validity. Section 6 provides a summary of the related literature. Section 7 concludes the paper and proposes future research work.

2 RESEARCH SETUP

This work has two objectives. The first is to investigate the potential to predict early whether a developer response is likely to be successful. The second is to pinpoint what and how features that developers can control influence the success of the responses. Through our analysis, we seek to derive a set of recommendations that developers can follow to increase the probability of success of their responses.

2.1 Terms

While developers can respond to reviews indirectly, such as through the deletion of features [33], our focus is on how their written responses impact the behavior of individual users. Therefore, in this paper, a *developer response* refers to a written response a developer posts in an app store in reply to a user review in the same app store. In reaction to the response, a user has the option of doing nothing, or changing their rating, their review, or both. In this study, we are particularly interested in a developer response that after receiving it, a user increases his or her rating. We consider such a response *successful*. Figure 1 shows an example of a successful developer response.

2.2 Research Questions

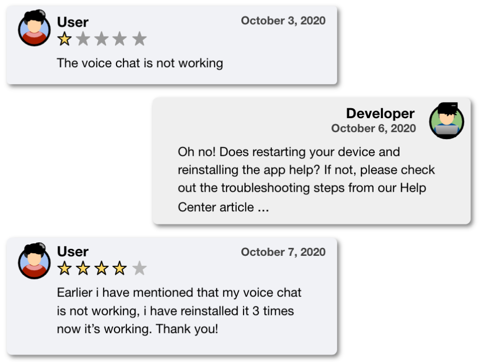
We break the aforementioned high-level objective into four concrete research questions.

- **RQ1**: How often do users change their original ratings with and without a developer response?
- **RQ2**: Can we effectively predict which developer responses will be successful?
- **RQ3**: Which features play roles in predicting the success of developer responses, and how?

The research method we employed in this study includes four major steps: data collection, feature extraction, model construction, and model interpretation.

2.3 Data Collection

We based our app selection on the following three criteria.


User October 3, 2020
★☆☆☆☆
The voice chat is not working

Developer October 6, 2020
Oh no! Does restarting your device and reinstalling the app help? If not, please check out the troubleshooting steps from our Help Center article ...

User October 7, 2020
★★★★☆
Earlier i have mentioned that my voice chat is not working, i have reinstalled it 3 times now it's working. Thank you!

Figure 1: An example of a successful developer response on the Google Play Store

- **Popularity:** We focused on top free-to-download apps as we believe that developers would want to maintain the popularity of their apps by providing support, including responses, to their users. We did not consider paid apps because we want to avoid the influence of app prices on our results.
- **Category:** We focused on apps across all app categories in the Google Play Store to avoid selection bias and masking disparities that different app categories may exhibit. There are 32 categories for apps in the Google Play Store[1].
- **Maturity:** We focused on apps that have been in the app store for at least 3 months before our study period. This decision is to avoid an influx of reviews for newly released apps [35] and developers being more active than otherwise (e.g., since they would still be trying to establish their apps).

We selected the top 50 apps in each app category on July 2, 2020. Consequently, we collected data from 1600 top apps in the Google Play Store. The app store does not provide an easy way to collect the change histories of reviews and such data is only available to the developers of apps. Therefore, to circumvent this and avoid overwhelming the app store with requests, for each app, we set up cron jobs (timed program execution) to crawl the store every hour to collect new data and detect changes to the already crawled data. The changes include a change to review contents, ratings, and developer responses. The crawler was programmed to act like a smartphone device and interface with the APIs of the Google Play Store. The crawler ran every hour for ten weeks, starting on October 4, 2020 and ending on December 13, 2020. Consequently, our dataset contains each app's complete change history of its user reviews during that period.

In total, we collected 9,352,326 reviews, of which 676,311 reviews (7.23%) had changed at least once. Among these reviews, the changes in 494,298 reviews include receiving a developer response (73.09%).

2.4 Feature Extraction

We purposely focused only on features that can be controlled by developers when composing a response because our ultimate goal

[1]https://support.google.com/googleplay/android-developer/answer/113475

is to derive a set of recommendations for writing effective responses that developers can follow. Previous work on effective complaint handling, communications on public forums, and mobile app review analysis have pointed to a number of textual and non-textual features which we anticipate collectively distinguish successful from not successful responses. We extracted three categories of features from developer responses. The three categories are *Time*, *Presentation*, and *Tone*.

2.4.1 ***Time***.

Upon a developer's response to their review, a user who wrote the review will receive an email and a push notification. Research shows that timing aspects affect recipients' responses to such notifications [49]. Timing aspects also have contributed to the successes of users with their requests for responses from developers on the iOS store [40]. For this category, we derived features from the posting time of a response and a review. The times are denoted in Greenwich Mean Time.

(1) *Timeliness* – The time difference between the posting time of the review and the response.
(2) *Day of Week* – The posting day of week of the response (Monday, Tuesday, etc.) and whether it is the same as that of the review.
(3) *Weekday Weekend* – Whether the posting of the response occurred on a weekday or a weekend and whether the same applies to the review.
(4) *Time of Day* — The posting time of day of the response and whether it is the same as that of the review. We partitioned the time into four slices: Morning [6-12), Afternoon [12-18), Evening [18-24/0), and Night [24/0-6).

2.4.2 ***Presentation***.

Research shows that several features of text, such as its readability and length, can influence its perceived value [24, 39]. The use of the user's name in text has important implications for organizational communication [48]. The presence of URLs and email addresses inform users that additional help resources are available. High similarity between the review and response may indicate that developers have thoroughly read and understood the review. These features can collectively make responses less generic and more personal, which are important factors in effective complaints handling [9, 31]. We extracted the following text features from a response.

(1) *Length* — The number of characters.
(2) *Length Ratio* — The ratio between the number of characters of the review and that of the response.
(3) *Readability* — The readability score obtained from the Flesch Reading Ease test [13].
(4) *Reviewer's Name* – Whether the response includes the name of the reviewer.
(5) *URL* – The count of links.
(6) *Email Address* – The count of email addresses.
(7) *Phone Number* – The count of phone numbers.
(8) *Number* – The count of numbers, excluding phone numbers.
(9) *Capitalized Word* – The percentage of capitalized words.
(10) *Non-AN Char* – The percentage of non-alphanumeric characters, excluding white spaces.
(11) *Mention Rating* – Whether the response include words about star rating (e.g., stars, ratings).
(12) *Shannon's Entropy* – Shannon's entropy of the response.
(13) *RR Similarity* — The degree of similarity between the review and the response. We utilized a pre-trained Universal Sentence Encoder (USE) model [6] to generate dense vector representations for each review and response pair. The model has shown to be efficient and achieve state-of-the-art performance on the semantic textual similarity task. Although the model does not require input texts to be pre-processed, we removed URLs, emails, numbers, and phone numbers to reduce noise in the responses. We then used cosine similarity to measure the similarity between the vector representations of the pair.

2.4.3 ***Tone***.

The group of features that make up the tone of a developer's response is perhaps the one that is most intuitively matched with the success of that response. When actual or perceived defects of an app have lead to negative emotions on the part of the user, it is important for the developer to mitigate them so the user does not perceive the interaction as negative and learns this as a negative impression. In line with this, research shows that the manner in which a complaint is handled is crucial for an organization to generate a successful service recovery, noting that responses should be polite and empathetic [9, 31]. We extracted the following features from a response:

(1) *Sentiment* — The sentiment polarity (positive, negative, or neutral), computed using SentiStrength [47], a publicly available tool for sentiment analysis.
(2) *Emotion* — The six different emotions (anger, fear, joy, love, sadness, and surprise), obtained from the emotion module in the EmoTxt tookit [5]. The module contains six binary classifiers, each detects a specific emotion in text.
(3) *Persistence* — The number of times developers have edited their response before the current state.
(4) *Mood and Modality* – The response's mood and modality, measured by the Pattern tool of De Smedt et al. [10].
(5) *Politeness* — The politeness score, measured using a tool by Danescu et al. [8], and the count of 36 syntactic and social markers of politeness, extracted using the politeness R package by Yeomans et al. [50]. Examples of these markers include request, greeting, gratitude, and apology.

2.5 Model Construction

The two main goals for building a machine learning model are (1) to investigate how well we can determine the success of a developer response from its features and (2) to assess which features play roles in determining the success of the response and how they do. Here we formalize our task as a binary classification problem, i.e., given a developer response, predict whether a user will increase their rating after the response.

To achieve our goals, we experimented with the XGBoost (eXtreme Gradient Boosting) model [7], a widely used tree-based ensemble machine learning model for this type of task [34]. The model is based on gradient tree boosting where it trains many decision trees iteratively, where each new tree is built to reduce the errors of previous trees. The model can handle heterogeneous feature types, does not require feature scaling, and has been shown to work well on imbalanced dataset [32]. More importantly, it has several measures to combat over-fitting, mainly accomplished through hyperparameter optimization. For these reasons, we chose to experiment with XGBoost.

To build the model, we used the XGBoost python package [7]. The dataset is split into 90% training and 10% testing (hold-out) set using stratified sampling. We used the training set to tune the model's hyperparameters.

2.5.1 ***Hyperparameter Optimization.*** Hyperparameters play a critical role to a model performance [37, 44]. For this, we constructed a hyperparameter search grid for the model to sample from. The range of values used for different hyperparameters in the search space are as follows:

- *max_depth* — 6 to 12 with step size of 1.
- *learning_rate* — draw uniformly between 0.01 and 0.1.
- *gamma* — draws uniformly between 0 and 0.5.
- *min_child_weight* — draws uniformly between 1 and 15.
- *subsample* — draws uniformly between 0.1 and 1.0.
- *colsample_bytree*: — draws uniformly between 0.1 and 1.0.
- *n_estimators* — always set to 10000.
- *early_stopping_rounds* — always set to 50.

We refer to the official documentation of XGBoost[2] for the description of each parameter. As it is too computationally expensive to try every hyperparameter setting, we adopted a Bayesian Optimization approach [3] to try out a total of 100 settings from the search space with the goal of maximizing the model's performance. For each setting, we applied a stratified 10-fold cross validation to ensure performance stability. We took the hyperparameter setting from the best-performing model to train the final model on entire training set. We then evaluated the final model on the hold-out set, to assess its performance on unseen new data.

2.5.2 ***Performance Measure.*** We selected Area Under the Receiver Operating Characteristic Curve (AUC-ROC) as the performance measure. The ROC curve plots the false positive rate (false alarm) against the true positive rate (recall) at different classification thresholds. The area under this curve (AUC) provides a scalar measure of performance regardless of what classification threshold is chosen. AUC value lies between 0.50 to 1.00. A classifier that predicts a random or a constant class has an AUC of 0.50, while a perfect classifier has an AUC of 1.00. In general, the higher the AUC, the better the model is at distinguishing between classes.

2.6 Model Interpretation

To derive insights from the model, we adopted two widely used model-agnostic interpretation methods: permutation feature importance and partial dependence plot.

2.6.1 ***Permutation Feature Importance.*** To determine which features are predictive of success, we adopted a method called *permutation feature importance* [4]. The key intuition is that a feature is important if permuting or shuffling its values drops the classification performance. We refer to this drop in performance as the **Importance**. Hence, the greater the drop, the more important the feature is. The method has several advantages. For example, it takes into account feature interactions, can be used on a single feature or a group of features (by permuting features in the same group together), and does not require retraining the model. More importantly, the results are intuitive and easy to interpret. Nonetheless, there are two pitfalls that need to be addressed to properly use the method. First, in the presence of highly correlated features, the method can decrease the importance of such features [43]. One way handle this is to identify and discard correlated features before training the model. To do so, we generated a correlation matrix of features based on the Spearman's correlation. We used a correlation ($|\rho|$) threshold of 0.70 to mark a pair of features as highly correlated. For each pair, we kept the feature with the highest mutual information [25] and discarded the other. Second, due to the stochastic nature of the method, different permutations of the same feature can give slightly different importance estimates. To obtain a more reliable estimate for each feature, we ran the method 20 times using different permutations. The mean and the standard deviation across these runs are reported as the importance estimate of a given feature.

2.6.2 ***Partial Dependence Plot.*** While permutation feature importance reveals which features are predictive of success, it does not determine how these important features affect the predictions. To answer this, we turned to a partial dependence plot (PDP). A PDP is a visualization method that plots the change in the average prediction probability for a class as an independent variable varies over its marginal distribution [14]. More specifically, a PDP shows exactly how the average prediction probability that a developer response will be successful varies across a range of possible values of a feature of interest, while keeping other features constant. Thus, insights from PDPs can shed light on how developers can increase the chance of success of their responses.

3 RESULTS

RQ1: How often do users change their original ratings with and without a developer response?

It is important to first make explicit whether responding to reviews is worthwhile. To answer this question correctly required us to filter out the developer responses that were given to reviews and the reviews without a response up to 24 days prior to the final data collection date. This step was necessary to avoid introducing bias because most users may not have had enough time to make changes to their ratings before that date. We specifically chose 24 days because we observed that 90% of users made changes to their ratings within 24 days after a response.

After filtering the data in this way, in 13,647 out of 320,274 reviews (4.26%), the rating changed after a response. Without a response, the changes in ratings happened in 68,382 out of 5,843,802 reviews (1.17%). Tables 1A and 1B break down the changes in ratings[3] with and without a response. The diagonal values indicate that the ratings remain unchanged. The values above and below the diagonal indicate that the ratings increase and decrease, respectively.

[2]https://xgboost.readthedocs.io/en/latest/parameter.html

[3]In cases where a user changed their ratings multiple times after a response, we only considered the earliest change because it is more likely to have been caused by the response than the later ones. Similarly, when multiple rounds of a developer response followed by a rating change occurred, we only considered the earliest instance because the changes in later rounds may be confounded by the earlier rounds.

Table 1: The changes in ratings with and without a response

(A) Reviews with a developer response

Rating before	Rating after 1	2	3	4	5
1	122,211	686	1,109	1,336	2,998
2	1,091	30,487	448	562	834
3	391	378	34,333	769	1,220
4	95	59	137	28,058	1,072
5	170	57	100	135	91,538

(B) Reviews without a developer response

Start rating	End rating 1	2	3	4	5
1	835,389	1,885	2,150	2,421	10,826
2	5,586	185,131	1,060	993	2,406
3	3,394	2,125	285,186	1,808	3,175
4	1,984	1,121	1,895	560,112	5,507
5	9,963	2,360	3,203	4,520	3,909,602

For our comparison of ratings increases and decreases, we only considered those cases in which increases or decreases are possible and consequently excluded 1-star ratings for decreases and 5-star ratings for increases, causing us to start from a different base for each respective comparison. We found that in 11,034 out of 228,274 1-4 star reviews (4.83%), the ratings increase after a response. On the other hand, the ratings increase without a response in 32,231 out of 1,914,154 such reviews (1.68%). This indicates that users who left a 1-4 star rating review are 2.9 times as likely to increase their ratings after a response as without one.

In the case of rating decreases, we found that in 2,613 out of 191,934 reviews with a 2-5 star rating (1.36%), the ratings decrease after a response. On the other hand, the ratings decrease without a response in 36,151 out of 4,991,131 such reviews (0.72%). This indicates that users who left a 2-5 star rating review are 0.9 times more likely to decrease their ratings after a developer response than without it.

Overall, the findings suggest that responding to reviews is worthwhile and can have a favorable outcome, although not every response will increase user ratings. Our next step is to employ a modeling technique to identify features within a response that influence users to *increase* their ratings. Consequently, the dataset to be used for the later parts ended up containing 228,274 developer responses given to 1-4 star reviews, 11,034 of which were successful.

RQ2: Can we effectively predict which developer responses will be successful?

Figure 2 shows the ROC curve of our model[4] (the green line) and the ROC curve of a classifier that predicts a random or a constant class, i.e., a baseline classifier (the dashed red line). The area between the two curves measures how well our model performs over a baseline classifier. The model is able to achieve an AUC of 0.69.

[4]The hyperparameters setting we used for the model is as follows: *learning_rate* = 0.02, *gamma* = 0.27, *max_depth* = 11, *min_child_weight* = 9.0, *subsample* = 0.90, and *col_sample_bytree* = 0.30.

Although the model is far from perfect (AUC = 1.00), it outperforms a baseline classifier (AUC = 0.50) by a considerable margin. This indicates the feasibility to predict early the responses that are more likely to be successful using only features that developers can control during response composition. It also suggests that the selected features have some association with the success of responses. Hence, we demonstrated that feature engineering and machine learning have the potential to enable developers to estimate the probability of success of their responses at composition time.

Note that due to the imbalanced nature of the dataset, one may apply a technique to balance the dataset, such as random undersampling. However, when the purpose of building a model is to derive insights and understandings, one should avoid using such a technique [45].

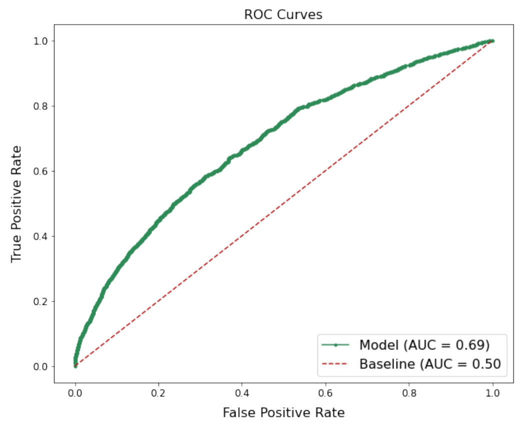


Figure 2: The ROC curves of the model and the baseline

RQ3: Which features play roles in predicting the success of developer responses, and how?

To answer this question, we apply the two model-agnostic interpretation methods described in Section 2.6 on the hold-out set to derive insights from the model regarding which features contribute to the generalization power of the model, and how.

Table 2 shows the importance of the three categories of features: Time, Presentation, and Tone. We observed that the Presentation category is the most important in predicting the success of developer responses, followed by the Tone category and then the Time category.

Table 2: The importance of the three categories of features

Feature	Importance
Presentation	0.1004 ± 0.0072
Tone	0.0802 ± 0.0046
Time	0.0183 ± 0.0034

The Presentation category:

Table 3 shows the importance value of each feature in the presentation category. Our analysis shows that the ratio between the number of characters of the review and that of the response is one of the most dominating features that distinguish successful from unsuccessful responses. When generating a PDP for the feature (Fig 3a), we observed that responses that are shorter than reviews are more likely to be successful. This, however, could also mean that users who posted a longer length review are more likely to increase their rating after a response than those who posted a short length review. This is the reason why the length feature, though important for prediction, rank lower than the length ratio feature.

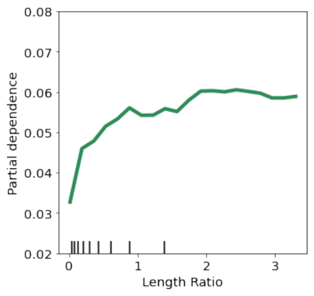
Our results show that the textual similarity between the review and response pairs helps distinguishing successful developer responses. The PDP of the feature (Fig 3b) depicts that the probability of success increases as the similarity between the review and response pair increases. This finding is congruent with research that says that less generic and more personal responses are key to generate a successful service recovery, noting that high similarity between the complaint and response demonstrates a grasp of customers' situation [18, 31].

Another important predictor is the number of emails sent. However, we observed that responses that do not include an email are more likely to be successful. This may indicate that users want to have their issues resolved on the app store and do not want to contact developers through additional means of communication.

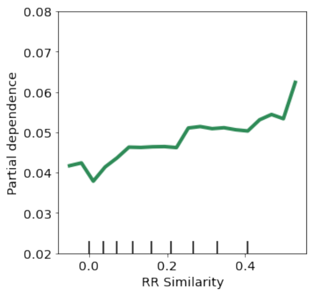
The next important feature for prediction in this category worth noting is Mention Rating. We observed that responses are more likely to be successful if developers mentioned about star ratings in the response. Below shows actual examples of such a response:

- "*. . . Is there something else that we could do to deserve a better rating from you? . . .*"
- "*. . . We hope you'll reconsider the rating once we fix the issue. . . .*"

Interestingly, whether or not developers include the name of the user in the response, numbers, phone numbers, URLs, do not influence the success of developer responses as these features are among the least important features in this group. Albeit low importances, we found that responses that contains URL, phone numbers, have slightly lower probability of success.



(a) Length Ratio



(b) RR Similarity

Figure 3: PDPs of the top 2 features in the Presentation category, ordered left to right from the top.

Table 3: The importance of features in the Presentation category

Feature	Importance
Length Ratio	0.0266 ± 0.0023
RR similarity	0.0163 ± 0.0020
Email	0.0051 ± 0.0011
Mention Rating	0.0032 ± 0.0010
Length	0.0031 ± 0.0013
Non-AN Char	0.0023 ± 0.0013
Readability	0.0018 ± 0.0011
Shannon's Entropy	0.0017 ± 0.0017
Capitalized Word	0.0012 ± 0.0010
URL	0.0012 ± 0.0005
Reviewer's Name	0.0004 ± 0.0004
Number	0.0003 ± 0.0011
Phone	0.0000 ± 0.0001

The Tone category:

Table 4 lists the importance of each feature in the Tone category. It shows that politeness is one of the most important features in distinguishing successful from unsuccessful responses. Figure 4a shows the PDP of the politeness score, as measured by the tool of Danescu et al. [8]. It shows that the more polite the responses are, the more they are likely to be successful. Our results are congruent with research that shows that the manner in which a complaint is handled is crucial for an organization to generate a successful service recovery, noting that responses should be polite and empathetic (part of the politeness markers) [9, 31].

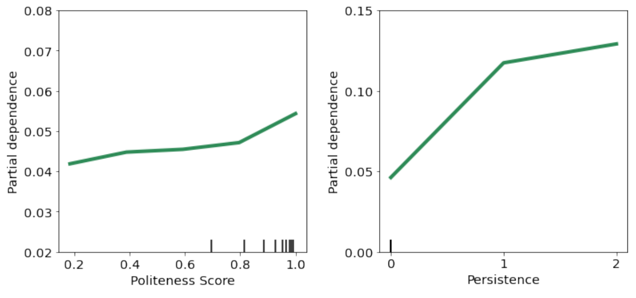
Regarding the 36 syntactic and social markers of politeness, when observed each in isolation, we did not find them to be important. However, when combined, these markers did interact and were found to be important for the prediction. Among these 36 markers, the use of plural first-person pronouns (e.g., we, us, our), showing gratitude (e.g., "thanks for reaching out"), the absence of the use of negation (e.g., never, not), the use of "please", and the use of indirect requests (e.g., "could you") are the top 5 markers for the prediction.

Our results show that persistence enhances responses' success. More specifically, the more times the developers edit their responses, the more likely they will be successful. As shown in Fig 4b, keeping all other features constant, the probability of success of responses increases as the number of edits increases.

Interestingly, we found that sentiment and the group of features for emotions (anger, fear, joy, love, sadness, and surprise) as a whole did not contribute to the prediction.

Table 4: The importance of features in the Tone category

Feature	Importance
Politeness	0.0365 ± 0.0032
Persistence	0.0118 ± 0.0015
Mode and Modality	0.0052 ± 0.0026
Sentiment	0.0009 ± 0.0006
Emotion	0.0005 ± 0.0009



(a) Politeness Score (b) Persistence

Figure 4: PDPs of the top 2 features in the Tone category.

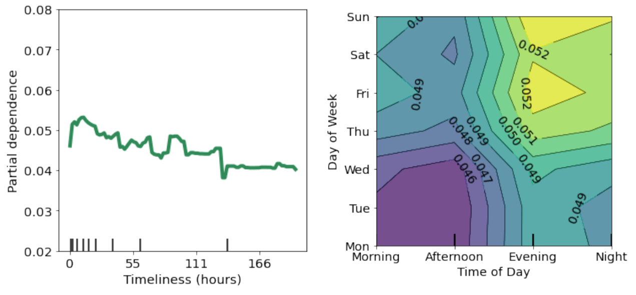
The Time category:

Table 5 shows the importance of each feature in the Time category. The timeliness has shown to be the most important feature in the Time category. We found that responses that are posted too quickly or too slow are less likely to be successful (Fig 5b).

We also found that responses posted during some time windows are more successful than others. We observed that responses that were posted on Friday, Saturday, and Sunday, as well as in the evening and night in GMT zone, which correspond to the morning and afternoon in US Central Time, are more likely to be successful than responses that were posted on the other days of week and times of day (Fig 5b). Regarding whether developers should respond to reviews on the same time of day, day of week, or weekday/weekend as the reviews', our results indicate that this does not affect the success of the responses.

Table 5: The importance of features in the Time category

Feature	Importance
Timeliness	0.0129 ± 0.0021
DayOfWeek	0.0019 ± 0.0012
TimeOfDay	0.0017 ± 0.0011
Weekday Weekend	0.0000 ± 0.0005



(a) Timeliness (b) TimeOfDay vs. DayOfWeek

Figure 5: PDPs of the top features in the Time category.

4 IMPLICATIONS

This section outlines the most promising prospects and practical implications of our work.

4.1 Recommendations

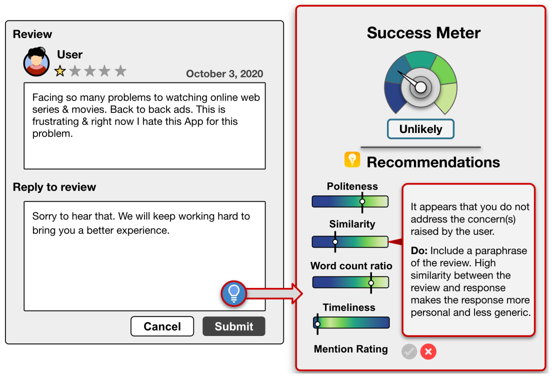
We believe that our findings are of interest as we have examined the predictive power of a wide range of features that developers can control in terms of distinguishing successful from unsuccessful developer responses. Therefore, based on our findings, in responding to reviews with a one- to four-star rating, we recommend that developers do the following to their responses to increase the probability of success:

- Include a paraphrase of the review. This is to demonstrate genuine attention restating the concern raised by the users. The response will be less generic and more personal.
- Be polite and possess different syntactic and social markers of politeness (e.g., show gratitude, use indirect request). Responses that are polite can ease customer's anger and dissatisfaction.
- Write a response that is long, but, at the same time, shorter than the review.
- Try to resolve the issues within the app store and refrain from asking users to contact you through additional means of communication.
- Include a sentence mentioning the star rating. This is to remind users about the ratings they gave or to suggest they reconsider their rating.
- Post in a timely fashion, but not too soon or too late. Responses that come in too fast may indicate that developers have not thoroughly read and understood the review. As a result, such responses may be viewed as automatic and not be as valuable as those that come later. However, at the same time, delayed responses may indicate a lack of concern for the users. Still, legitimately urgent requests need to be processed with urgency.
- Utilize the 'more successful' posting time windows. In other words, responses should be posted in the evening and the night in the GMT zone, which correspond to the morning and afternoon in US Central Time, of Fridays, Saturdays, and Sundays.
- Be persistent by following up if you do not get a reaction or response back from the users. Editing the responses shows the users that you still care and would like to resolve user's dissatisfaction and issues. This increases the chance of users to react to the response.

4.2 Response Writing Assistant

By modeling the success of responses, we can integrate machine intelligence into a developers' app management portal (e.g., the Google Play Console) to provide value for developers. Figure 6 shows a user interface (UI) prototype of a response writing assistant tool. In the scenarios we envisage, the tool is not intended to write or automatically generate a response for developers, but rather to assist developers when they write a response. More specifically, while developers compose their response to a review, the tool 1) collects metadata, 2) scans the response body for the presence and counts of important features, 3) evaluates the probability of success of the response in its current state, and 4) makes recommendations for developers to increase that probability. For example, in Figure 6, the tool has determined that the response is unlikely to be successful and shows that the response's similarity level is not in the optimal spot. It then recommends the developers modify their response to address the user's concern(s) more. It is easy to see that, indirectly through its recommendations, such a tool has a potential of increasing the quality and the effectiveness of developer responses on the app store. Hence, better response strategies could also lead to improve user-developer communication efficiency.

Additionally, we investigated whether we could build a lightweight model using only the top features identified in RQ3. For this, we repeated the entire model construction process and found that the final model achieves a lower performance than what we could achieve using all features. This indicates that the interaction of multiple important and unimportant features reflects the success of developer responses. Hence, in practice, we recommend practitioners consider all features when building a prediction model for a response writing assistant tool to achieve the best performance.



Facing so many problems to watching online web series & movies. Back to back ads. This is frustrating & right now I hate this App for this problem.

Sorry to hear that. We will keep working hard to bring you a better experience.

It appears that you do not address the concern(s) raised by the user.

Do: Include a paraphrase of the review. High similarity between the review and response makes the response more personal and less generic.

Figure 6: A UI prototype of a response writing assistant tool

5 LIMITATIONS AND THREATS TO VALIDITY

An investigation of features influencing the success of developer responses can be conducted in at least one other way, that is, through a user survey. However, this may require the participants to have received at least one developer response for the survey to yield reliable findings. Due to the limitations of time and accessibility to such a population of users, we did not pursue this route. Furthermore, we consider a response to be *successful* if a user increases their rating after receiving it. This definition of success may be too strict. For example, the success of a response could also be measured by the change in the sentiment of a review after a response (from negative to positive sentiment). It could even be argued that editing a review while keeping the rating and sentiment unchanged constitutes some success in that the response intensifies the engagement of the user. Additionally, users may or may not change their reviews for many reasons that are not necessarily in reaction to the responses they received. For instance, users may just simply forget to update their ratings, or they decrease the ratings because they find new issues with the app and become more dissatisfied. We acknowledge that this may affect the conclusions of our study and encourage future work to expand the definition of success and triangulate our findings through user surveys.

While the tools we employed to extract the tone and text similarity features have been shown to be effective in literature, it is possible that the intrinsic inaccuracies in them can affect our results. We noted that we also experimented with sentiment analysis tools that were built specifically for the software engineering domain such as SentistrengthSE [22] and observed a comparable performance for our model. Next, the method we used to extract each of the presentation features may not be fully accurate due to the informal and unstructured nature of the text data. Furthermore, we cannot claim that our set of features are exhaustive as we might have overlooked other important features or features that are external to developers that may play role in the success of responses, such as the starting star rating or the number of times the user has edited their review prior to receiving a response. In line with our objectives, we purposely focused only on features that developers can control because our goal was to derive a set of recommendations for writing a successful response. We noted that evidence in our analysis has lead us to believe that incorporating more features from the reviews may further help improve the performance of the model. Investigation on this in currently in the work.

It is still unclear whether our findings hold true for apps that have different characteristics (e.g., other top apps, paid apps, unpopular apps, or apps from other app stores), or for a study in a different time frame. As a result, our study is subject to the app sampling problem [28]. Furthermore, we only experimented with one machine learning model. Different models can lead to different results [44], and thus, we cannot claim that our findings will hold true for other modeling algorithms.

6 RELATED WORK

In this section, we summarize related work dealing with user reviews and developer responses on the app stores.

6.1 Analysis of App User Reviews

App reviews contain a rich source of information [35] that can be used for various software engineering activities [1]. For instance, in maintenance and requirement engineering, analyzing app reviews help developers elicit new requirements and identify unexpected app behaviours [15, 21, 23, 27]. In addition, research has also pointed out that users' perception of the same apps can very across different platforms [2] and countries [42].

Nonetheless, apps can receive over thousands of reviews in a day. Manually reading and extracting information from each review is impracticable. Hence, a sizeable body of literature has been devoting to find ways to extract useful information from user reviews automatically. For example, Iacob and Harrison [21] proposed MARA, a framework for retrieving app feature requests based on linguistic rules. Gao et al. [15] adapted the topic modeling technique to detect emerging issues across different time slices. Maalej and Nabil [27] evaluated several techniques to classify reviews. Srisopha et al. [41] detailed a method to identifying country-specific app feature requests. Sorbo et al. [11] proposed a method to categorize reviews into different intentions and topics. Palomba et al. [36] proposed a tool that recommends changes to software artifacts. For more research in this area, please refer to a survey by Martin et al. [29], Tavakoli et al. [46], and Nayebi et al. [17].

Different from these existing analysis of review research, our work contribute towards improving the effectiveness of the communications between users and developers on the app store instead of focusing only on extracting information from user reviews.

6.2 Analysis of Developer Responses

Lim et al. [26] have shown that reviews and ratings are among the most important factors that people consider when choosing apps to download. Additionally, the more positive ratings and reviews an app has, the higher it will rank in search results, and the more visible it is to potential users. Research shows that responding to reviews can have a positive impact on user ratings. As a result, user support and maintaining an app's reputation are important activities for app developers to perform as part of the software lifecycle.

McIlroy et al. [30] analyzed review and response pairs from 10,713 top apps on the Google Play Store. They found evidence that responding to reviews has a positive impact on user ratings. Hassan et al. [20] further studied the topic and found the chance of ratings increase with a developer response to be 6 times as high as without. While we did not observe a similar likelihood, we argue that our findings provide a more realistic estimate because we avoided bias by filtering out reviews with a 5 star rating (as users cannot increase their rating further than 5 stars) and reviews that were posted towards the end of our data collection period. Srisopha et al. [40] investigated a wide range of features within user reviews that spur developers' responses on the iOS app store. They found that rating, review body length, and the proportions of positive and negative words are the most important features to predict developer responses. They presented a feasibility evidence that machine learning can be used to help predict reviews that developers are most likely to respond to. Savarimuthu et al. [38] studied developer responses from a social norm perspective. They manually analyzed 2,668 developer responses and identified 12 norms in the responses. They found, for example, that 80% of responses contain appreciation, 30% of responses did not have any personalization. Importantly, these studies observed that only a small fraction of reviews are read and garner a response from developers, and there is a considerable variation in response strategies.

Due to the large number of user reviews an app can receive, many studies have also started to investigate ways to generate developer responses automatically. Gao et al. [16] proposed RRGen, an attention-based sequence-to-sequence neural model to generate responses. RRGen has shown to create satisfactory responses to reviews that are common among many apps, but did not create responses that are app-specific well. Farooq et al. [12] attempted to solve this problem. They proposed AARSYNTH, a method that augmented the similar technique used in RRGen with information specific to a given app, such as app description.

Despite the sizable and growing body of literature on analysis of app reviews and developer responses, there have been no findings in literature on how developers should respond to app reviews. Hence, we take a step to fill this gap in research as we believe that examining the predictive power of a wide range of features within a response in terms of distinguishing successful from unsuccessful responses carries useful implications.

7 CONCLUSIONS AND FUTURE WORK

While responding to reviews can have a favorable outcome, not every response leads to the increased user ratings we define as success. By focusing on the state of a review before and immediately after a response, we can determine whether the response is successful. This paper has two objectives. The first is to investigate the potential to predict early the success of developer responses. The second is to pinpoint which features in a response influence the success of it, and how they do. Three categories of features were considered: Time, Presentation, and Tone. The XGBoost algorithm was adopted to model the success of developer responses based on these features. We achieve an AUC of 0.69, outperforming a baseline classifier. The use of two model-agnostic interpretation techniques furthers the understanding of what and how features influence the success of developer responses. We believe our findings could be of practice significance to a broad range of stakeholders.

There are several interesting avenues for future research. The next step of our work could be to develop a functional prototype of the response writing assistant tool. It could also be to investigate how developers should response based on ratings or types of reviews. Last but not least, it would make for an interesting comparison to replicate this study for the iOS App Store, which has recently allowed developers to respond to reviews.

REFERENCES

[1] Afnan A. Al-Subaihin, Federica Sarro, Sue Black, Licia Capra, and Mark Harman. 2021. App Store Effects on Software Engineering Practices. *IEEE Transactions on Software Engineering* 47, 2 (2021), 300–319. https://doi.org/10.1109/TSE.2019.2891715

[2] Mohamed Ali, Mona Erfani Joorabchi, and Ali Mesbah. 2017. Same App, Different App Stores: A Comparative Study. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. IEEE Press, Buenos Aires, Argentina, 79–90. https://doi.org/10.1109/MOBILESoft.2017.3

[3] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. *Computational Science & Discovery* 8, 1 (jul 2015), 014008. https://doi.org/10.1088/1749-4699/8/1/014008

[4] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[5] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2017. EmoTxt: A Toolkit for Emotion Recognition from Text. In *Proc. of 7th Int'l Conf. on Affective Computing and Intelligent Interaction Workshops and Demos* (San Antonio, TX, USA) *(ACII '17)*. 79–80. https://doi.org/10.1109/ACIIW.2017.8272591

[6] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder. arXiv:1803.11175 [cs.CL]

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[8] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 250–259. https://www.aclweb.org/anthology/P13-1025

[9] Moshe Davidow. 2003. Organizational responses to customer complaints: What works and what doesn't. *Journal of service research* 5, 3 (2003), 225–250.

[10] Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *The Journal of Machine Learning Research* 13, 1 (2012), 2063–2067.

[11] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2016. What Would

Users Change in My App? Summarizing App Reviews for Recommending Software Changes. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering* (Seattle, WA, USA) *(FSE 2016)*. Association for Computing Machinery, New York, NY, USA, 499–510. https://doi.org/10.1145/2950290.2950299

[12] Umar Farooq, AB Siddique, Fuad Jamour, Zhijia Zhao, and Vagelis Hristidis. 2020. App-Aware Response Synthesis for User Reviews. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE Computer Society, Los Alamitos, CA, USA, 699–708. https://doi.org/10.1109/BigData50022.2020.9377983

[13] Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology* 32, 3 (1948), 221.

[14] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29, 5 (2001), 1189 – 1232. https://doi.org/10.1214/aos/1013203451

[15] Cuiyun Gao, Jichuan Zeng, Michael R. Lyu, and Irwin King. 2018. Online App Review Analysis for Identifying Emerging Issues. In *Proceedings of the 40th International Conference on Software Engineering* (Gothenburg, Sweden) *(ICSE '18)*. Association for Computing Machinery, New York, NY, USA, 48–58. https://doi.org/10.1145/3180155.3180218

[16] Cuiyun Gao, Jichuan Zeng, Xin Xia, David Lo, Michael R. Lyu, and Irwin King. 2019. Automating App Review Response Generation. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering* (San Diego, California) *(ASE '19)*. IEEE Press, 163–175. https://doi.org/10.1109/ASE.2019.00025

[17] Necmiye Genc-Nayebi and Alain Abran. 2017. A systematic literature review: Opinion mining studies from mobile app store user reviews. *Journal of Systems and Software* 125 (2017), 207–219.

[18] Thorsten Gruber. 2011. I want to believe they really care: How complaining customers want to be treated by frontline employees. *Journal of Service Management* 22 (2011), 85–110.

[19] Mark Harman, Yue Jia, and Yuanyuan Zhang. 2012. App store mining and analysis: MSR for app stores. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE Press, 108–111. https://doi.org/10.1109/MSR.2012.6224306

[20] Safwat Hassan, Chakkrit Tantithamthavorn, Cor-Paul Bezemer, and Ahmed E Hassan. 2018. Studying the dialogue between users and developers of free apps in the google play store. *Empirical Software Engineering* 23, 3 (2018), 1275–1312.

[21] Claudia Iacob and Rachel Harrison. 2013. Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories* (San Francisco, CA, USA) *(MSR '13)*. IEEE Press, 41–44.

[22] Md Rakibul Islam and Minhaz F Zibran. 2018. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software* 145 (2018), 125–146.

[23] Hammad Khalid, Emad Shihab, Meiyappan Nagappan, and Ahmed E Hassan. 2015. What do mobile app users complain about? *IEEE Software* 32, 3 (2015), 70–77.

[24] Nikolaos Korfiatis, Elena GarcíA-Bariocanal, and Salvador SáNchez-Alonso. 2012. Evaluating content quality and helpfulness of online product reviews: The interplay of review helpfulness vs. review content. *Electronic Commerce Research and Applications* 11, 3 (2012), 205–217.

[25] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. 2004. Estimating mutual information. *Physical review E* 69, 6 (2004), 066138.

[26] Soo Ling Lim, Peter J Bentley, Natalie Kanakam, Fuyuki Ishikawa, and Shinichi Honiden. 2014. Investigating country differences in mobile app user behavior and challenges for software engineering. *IEEE Transactions on Software Engineering* 41, 1 (2014), 40–64.

[27] Walid Maalej and Hadeer Nabil. 2015. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*. 116–125. https://doi.org/10.1109/RE.2015.7320414

[28] William Martin, Mark Harman, Yue Jia, Federica Sarro, and Yuanyuan Zhang. 2015. The App Sampling Problem for App Store Mining. In *Proceedings of the 12th Working Conference on Mining Software Repositories* (Florence, Italy) *(MSR '15)*. IEEE Press, 123–133.

[29] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. 2016. A survey of app store analysis for software engineering. *IEEE transactions on software engineering* 43, 9 (2016), 817–847.

[30] Stuart McIlroy, Weiyi Shang, Nasir Ali, and Ahmed E Hassan. 2015. Is it worth responding to reviews? studying the top free apps in google play. *IEEE Software* 34, 3 (2015), 64–71.

[31] Hyounae Min, Yumi Lim, and Vincent P Magnini. 2015. Factors affecting customer satisfaction in responses to negative online hotel reviews: The impact of empathy, paraphrasing, and speed. *Cornell Hospitality Quarterly* 56, 2 (2015), 223–231.

[32] Nuno Moniz, Paula Branco, and Luís Torgo. 2017. Evaluation of Ensemble Methods in Imbalanced Regression Tasks. In *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*. PMLR, PMLR, Skopje, Macedonia, 129–140.

[33] Maleknaz Nayebi, Konstantin Kuznetsov, Paul Chen, Andreas Zeller, and Guenther Ruhe. 2018. Anatomy of Functionality Deletion: An Exploratory Study on Mobile Apps. In *Proceedings of the 15th International Conference on Mining Software Repositories* (Gothenburg, Sweden) *(MSR '18)*. Association for Computing Machinery, New York, NY, USA, 243–253. https://doi.org/10.1145/3196398.3196410

[34] Didrik Nielsen. 2016. *Tree boosting with xgboost-why does xgboost win" every" machine learning competition?* Master's thesis. NTNU.

[35] Dennis Pagano and Walid Maalej. 2013. User feedback in the appstore: An empirical study. In *2013 21st IEEE International Requirements Engineering Conference (RE)*. 125–134. https://doi.org/10.1109/RE.2013.6636712

[36] Fabio Palomba, Pasquale Salza, Adelina Ciurumelea, Sebastiano Panichella, Harald Gall, Filomena Ferrucci, and Andrea De Lucia. 2017. Recommending and Localizing Change Requests for Mobile Apps Based on User Reviews. In *Proceedings of the 39th International Conference on Software Engineering* (Buenos Aires, Argentina) *(ICSE '17)*. IEEE Press, 106–117. https://doi.org/10.1109/ICSE.2017.18

[37] Philipp Probst, Anne-Laure Boulesteix, and Bernd Bischl. 2019. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *The Journal of Machine Learning Research* 20, 53 (2019), 1–32.

[38] Bastin Tony Roy Savarimuthu, Sherlock Licorish, Manjula Devananda, Georgia Greenheld, Virginia Dignum, and Frank Dignum. 2017. Developers' responses to app review feedback-A study of communication norms in app development. In *Proc. of the COIN 2017 workshop@ AAMAS (2017)*.

[39] Robert M Schindler and Barbara Bickart. 2012. Perceived helpfulness of online consumer reviews: The role of message content and style. *Journal of Consumer Behaviour* 11, 3 (2012), 234–243.

[40] Kamonphop Srisopha, Daniel Link, Devendra Swami, and Barry Boehm. 2020. Learning Features That Predict Developer Responses for IOS App Store Reviews. In *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (Bari, Italy) *(ESEM '20)*. Association for Computing Machinery, New York, NY, USA, Article 12, 11 pages. https://doi.org/10.1145/3382494.3410686

[41] Kamonphop Srisopha, Chukiat Phonsom, Mingzhe Li, Daniel Link, and Barry Boehm. 2020. On Building an Automatic Identification of Country-Specific Feature Requests in Mobile App Reviews: Possibilities and Challenges. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (Seoul, Republic of Korea) *(ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 494–498. https://doi.org/10.1145/3387940.3391492

[42] Kamonphop Srisopha, Chukiat Phonsom, Keng Lin, and Barry Boehm. 2019. Same App, Different Countries: A Preliminary User Reviews Study on Most Downloaded iOS Apps. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE Computer Society, Los Alamitos, CA, USA, 76–80. https://doi.org/10.1109/ICSME.2019.00017

[43] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. Conditional variable importance for random forests. *BMC bioinformatics* 9, 1 (2008), 307.

[44] Chakkrit Tantithamthavorn and Ahmed E. Hassan. 2018. An Experience Report on Defect Modelling in Practice: Pitfalls and Challenges. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice* (Gothenburg, Sweden) *(ICSE-SEIP '18)*. Association for Computing Machinery, New York, NY, USA, 286–295. https://doi.org/10.1145/3183519.3183547

[45] Chakkrit Tantithamthavorn, Ahmed E. Hassan, and Kenichi Matsumoto. 2020. The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models. *IEEE Transactions on Software Engineering* 46, 11 (2020), 1200–1219. https://doi.org/10.1109/TSE.2018.2876537

[46] Mohammadali Tavakoli, Liping Zhao, Atefeh Heydari, and Goran Nenadić. 2018. Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools. *Expert Systems with Applications* 113 (2018), 186–199.

[47] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology* 61, 12 (2010), 2544–2558.

[48] Joan Waldvogel. 2007. Greetings and closings in workplace email. *Journal of Computer-Mediated Communication* 12, 2 (2007), 456–477.

[49] Liu Yang, Susan T. Dumais, Paul N. Bennett, and Ahmed Hassan Awadallah. 2017. Characterizing and Predicting Enterprise Email Reply Behavior. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 235–244. https://doi.org/10.1145/3077136.3080782

[50] Michael Yeomans, Alejandro Kantor, and Dustin Tingley. 2018. The politeness Package: Detecting Politeness in Natural Language. *R Journal* 10, 2 (2018), 489–502.