IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 1, 2021, accepted March 13, 2021, date of publication March 18, 2021, date of current version March 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3067000

Extracting Arguments Based on User Decisions in App Reviews

ANANG KUNAEFI[1] AND MASAYOSHI ARITSUGI[2], (Member, IEEE)
[1]Computer Science and Electrical Engineering, Graduate School of Science and Technology, Kumamoto University, Kumamoto 860-8555, Japan
[2]Big Data Science and Technology, Faculty of Advanced Science and Technology, Kumamoto University, Kumamoto 860-8555, Japan

Corresponding author: Anang Kunaefi (akunaefi@dbms.cs.kumamoto-u.ac.jp)

ABSTRACT Review mining from app marketplaces has gained immense popularity from researchers in recent years. Most studies in this area, however, tend to focus on improving the performance of classification prediction. In this study, we consider review mining from a different perspective, that is, mining user actions/decisions along with their respective arguments/reasons. Our motivation is to obtain a deeper understanding of users' decisions regarding applications and their underlying justifications, e.g., why users give ratings or recommendations. These information abstractions can benefit app developers, especially in planning app updates, by providing data-driven requirements from users' points of view. We utilized a supervised learning approach and built a machine-based annotator to set the ground truth. Seven classifiers and different feature configurations were trained and evaluated on two app review datasets. We then extracted relations between user decisions and arguments based on functional and nonfunctional requirement attributes. The results show an improved performance over the results of the baselines and favorably acceptable performance compared to the results from a human assessment.

INDEX TERMS Argument mining, data-driven requirement, review mining, software requirement.

I. INTRODUCTION

The last decade marks the explosion of user-generated content as web and mobile application (app) technologies have progressively developed. User-generated content, such as product reviews and mobile app reviews, is created every day on a unimaginable scale. This development has given rise to a new business model that is more open and user-oriented [1], [2].

The usefulness of customer/user reviews can be seen from two perspectives. First, for other users, app reviews can provide initial thoughts on whether an app is worth buying and installing. In fact, 86% of users in 2017 trust online customer reviews as much as personal recommendations.[1] Second, for developers, user reviews are an invaluable basis for building user-based requirements, e.g., planning new features for the next release [3]–[5]. In fact, a one-star increase in a Yelp rating indicates a 5-9% increase in revenue [6]. For these reasons, developers are eager to obtain positive reviews from their customers/users. These reviews in turn can be analyzed to develop better application updates [7]–[10]. Despite its great advantages, mining user reviews from app marketplaces can potentially be misleading due to bad review practices such as opinion spam [11] and deceptive reviews [12].

TABLE 1. Review mining studies from different perspectives.

No.	Perspective	Studies
1	Sentiment analysis	[3], [13], [14]
2	App development	[17], [4], [18], [19], [21], [22]
3	Review spam	[11], [12]
4	Surveys	[15], [16]

Studies on review mining have been carried out from many different perspectives (see Table 1). From a sentiment analysis perspective, some efforts have been undertaken to extract user assessments of apps [3], perform polarity detection [13], and perform intensity measurements [14]. For further reading, Nayebi *et al.* [15] and Martin *et al.* [16] provided comprehensive surveys on the study of review mining. Moreover, for app development purposes, some studies conducted app improvements based on user feedback, such

The associate editor coordinating the review of this manuscript and approving it for publication was Porfirio Tramontana.

[1]https://www.statista.com/statistics/315755/online-custmer-review-trust/

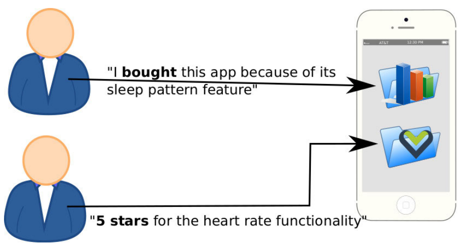

"I **bought** this app because of its sleep pattern feature"
"**5 stars** for the heart rate functionality"

FIGURE 1. User decisions and justifications.

as detecting bug reports and requests [4], [17], [18], extracting features favored by users [19], [20], extracting complaints [21], and identifying emerging issues in reviews [22].

However, most of the previous studies approached review mining as text classification problems, which caused researchers to tend to focus only on improving classification performance. This development is marked by the emergence of new deep-learning-based algorithms that allow classification predictions to achieve a rate of 95% or more. In addition, extracting deep and meaningful information from reviews, such as why users gave a certain rating, bought, or recommended an app, is still not sufficiently explored.

To fill this gap, in this study, we consider learning user reviews from another perspective, that is, mining user actions/decisions along with their respective arguments/reasons. For instance, a review *"The registration process is seamless, highly recommended"* contains user decisions and arguments. The phrase *"highly recommended"* depicts the user's action where the user would recommend the app to others, while the phrase *"The registration process is seamless"* refers to the user's argument regarding why he/she would recommend the app.

Mining user decisions and the underlying arguments, as studied in this paper, can bring about several advantages. First, a user's decision can become an indicator of sentiment intensity, e.g., a buying user (i.e., a user who purchases an app) arguably is more satisfied (has higher sentiment) than a rating user. By nature, user sentiments are subjective. Knowing the degree of sentiments can play a key role in understanding the exact feelings of users, as pointed out by Akhtar *et al.* [14]. Second, extracting users' arguments/justifications can provide answers regarding why users perform certain actions or make certain decisions, such as giving a rating, acquiring an app, or recommending an app. In this regard, user rationales can provide insight into what motivates users to take certain actions, as studied by Kurtanovic and Maalej [23]. Finally, extracting the correlation between user decisions and their justification can provide insights into which app functionalities/features lead to certain user decisions. For example, in a fitness tracking app, some users might purchase the app because of its sleep pattern feature, while other users rate the app because of its heart rate detection feature [24] (see Fig. 1).

In general, this study aims to extract user arguments that underlie decisions from app reviews. To achieve this objective, several challenges should be addressed, such as unstructured and extremely noisy review texts, the unavailability of labeled user decision datasets, unexplained classification predictions, the broad spectrum of user argument expressions, and hidden correlation between user decisions and arguments. In summary, our paper makes the following contributions:

1) We mine app reviews with an emphasis on extracting user arguments that underlie their decisions;
2) We propose an automatic framework based on weakly supervised learning, from handcrafted rule-based annotators and feature configurations to relation extractors, to model user decisions and arguments based on functional/nonfunctional requirement attributes;
3) We evaluate, discuss, and compare our results with human-based judgments. Additionally, we give some recommendations for the research community.

The rest of the paper is organized as follows: Section II describes the research question and problem definition used throughout this paper. We present our method in detail in Section III, followed by an explanation of the dataset in Section IV. Section V gives the empirical results of the experiments followed by the discussion in Section VI. We describe limitations and possible risks to the validity of this study in Section VII. Section VIII discusses related work in the field of review mining. Finally, we conclude this work with further research directions in Section IX.

II. RESEARCH DESIGN

The main objective of this study is to exploit user arguments concerning their actions toward an app (e.g., a rating and whether the app is purchased and recommended). In this section, we explain the problem definition and research questions (RQ), which are used throughout this paper.

A. PROBLEM DEFINITION

Suppose a set of reviews is denoted by R_n, where n denotes the number of reviews in the corpus. The tasks are to predict user decision class D_{nk} expressed in the review and to extract correlated arguments A_{nki}; k denotes the index of a user decision, and i denotes the index of user arguments for decision D_{nk} and review R_n.

To determine the user decision classes, we refer to two previous studies that discuss the variety of user actions in app reviews. Kurtanovic and Maalej [23] conducted a grounded theory study involving experts in the field of software engineering and described the decision concept as an action that has already been taken or will be taken by a user; the decision concept comprises four actions, i.e., acquiring, updating, switching and relinquishing an app. Kunaefi and Aritsugi [24], on the other hand, utilized an unsupervised learning approach with latent dirichlet allocation (LDA) [25] on argumentative reviews to specify various user decisions (i.e., acquiring, buying, recommendation, requesting, and rating decisions). We combine both results and define the user decision classes as follows:

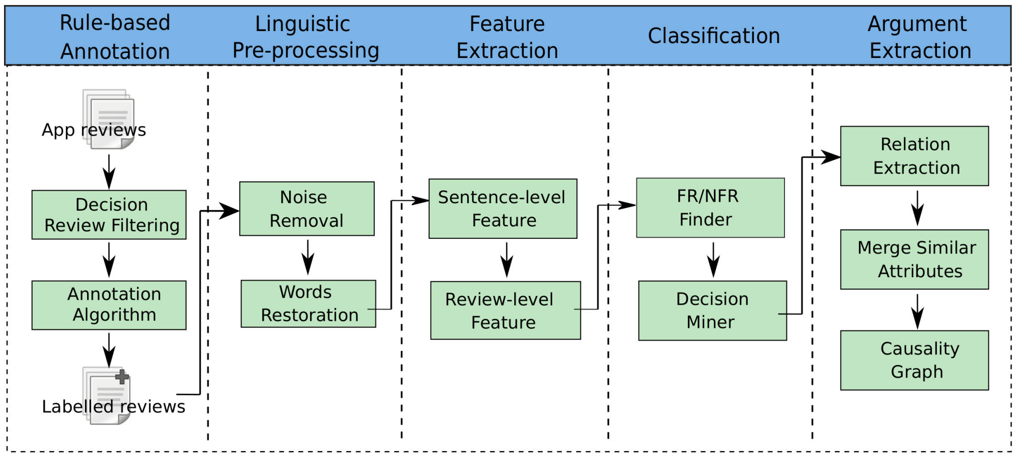


FIGURE 2. Our framework.

1) **Acquiring Decision** is an action expressed by users indicating that the users are willing to acquire, purchase, and use the app. For example, a review *"The app works without any glitches. Will purchase it very soon"* is considered a review with an acquiring decision.
2) **Recommending Decision** represents user action that indicates a user's commitment to recommend the app to others. For instance, *"I have this on my kindle and my phone. Would recommend to everyone"* could be classified as a recommending decision.
3) **Requesting Decision** is a user decision that denotes user requests for certain app features or app updates. For example, *"My only request would be in finding other friends that play, searching didn't seem to work for me."* could be considered a requesting decision.
4) **Rating Decision** is user behavior that describes a user assessment by giving ratings in the marketplace. For example, the review *"I'm glad it's over, so I give it 5 stars for the addiction until the end"* explains a user decision on giving a rating for the app.
5) **Relinquishing Decision** denotes a user's decision to uninstall, reject, or cancel the app for several reasons. For instance, a review *"You have to use real debit cards to start the game. So I deleted it right away"* can be classified as a relinquishing decision.

B. RESEARCH QUESTION

To support the objective of this study, we formulate the following research questions.

1) RQ1. WHAT FEATURES ARE DISCRIMINANT FOR CLASSIFYING USER DECISIONS AND HOW GOOD IS THE PERFORMANCE OF THE CLASSIFIERS?

This research question investigates which of the utilized classifiers perform well in classifying user decisions. We employ seven machine learning algorithms, namely, naive Bayes (NB), linear regression (LR), support vector machine (SVM), random forest (RF), AdaBoost, XGBoost, and multilayer perceptron (MLP), to evaluate features that are discriminant and nondiscriminant for the decision classification task. The reason is that those algorithms show notable performance in previous studies [23], [37].

2) RQ2. WHAT KINDS OF ARGUMENTS ARE USUALLY EXPRESSED BY USERS IN FAVOR OF THEIR DECISION CONCERNING AN APP?

This research question aims to reveal various user justifications behind their decisions. In other words, we are interested in determining whether certain app functionalities drive certain user decisions. We base this question on the assumption that for every decision made by a user, there exists an app's feature that influenced it.

3) RQ3. TO WHAT DEGREE IS THE PROPOSED APPROACH's RESULT ALIGNED WITH A HUMAN-BASED ASSESSMENT?

This research question is designed to seek evidence concerning the efficacy of our approach. We are interested in measuring the degree of alignment by comparing our results with information extracted from another feedback channel, e.g., an online forum.

III. METHOD AND PROCEDURE

Our proposed framework consists of 5 main modules: a) rule-based annotation, b) linguistic preprocessing, c) feature extraction, d) model training, and e) relation extraction, as depicted in Fig. 2. Each of the modules is described in the following subsections.

A. RULE-BASED ANNOTATION

In supervised learning, relying on human-based labeled data is time-consuming and labor-intensive. Our aim in this study is to develop an automated analysis tool that can generate results within a single run with minimum human supervision. Therefore, we developed a rule-based review annotator to mimic how human annotators work. Since we are interested in user arguments and decisions in text reviews, we define the following rules:

1) The text review needs to be long enough. We base this assumption on the study by Palau and Moens [26], which stressed that to be considered argumentative, a review must contain at least one claim and one premise. We set a 5-word limit, meaning that reviews with a word length of less than or equal to 5 are removed. In this way, we filter out very short reviews such as *"great app," "5 stars,"* and *"nice two thumbs up,"* since these reviews do not contain meaningful information.
2) The text review needs to contain causality links such as *because*, *since*, and *therefore*, as described by Khoo *et al.* [27]. For example, *"I like this app because it supports NFC payment."* We called this review an argumentative review.
3) The review needs to contain decision words, such as *purchase*, *recommend*, and *rate*, which represent the five user decisions described in subsection II-A. We called these reviews decision reviews. We explain how we compose the list of decision words in the following section.

Applying the previous rules removes noninformative reviews and leaves only informative reviews, namely, argumentative and decision reviews. The next crucial task is to annotate the reviews. Maalej *et al.* [28] utilized a set of keywords compiled from the literature and conducted a string matching classifier to automatically categorize the reviews. The Sentiment 140 dataset[2] was built with a machine-tagged approach by evaluating the number of emoticons in tweets (i.e., ☺ for positive and ☹ for negative) [29], [30]. In this study, we perform a similar approach to automatically label the review dataset by employing a rule-based mechanism.

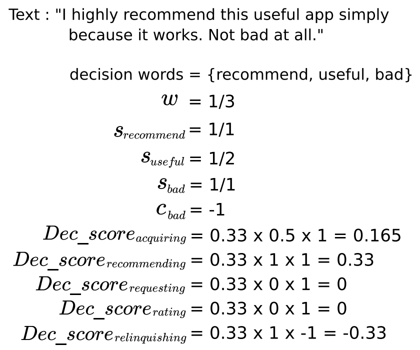
TABLE 2. Example of decision vocabularies.

Acquiring	Recommending	Requesting	Rating	Relinquishing
purchase	**recommend**	**request**	**rate**	**uninstall**
buy	must have	wish	star	delete
subscribe	must try	update	give	abandoned
pay	download	upgrade	five star	refund
contract	suggest	repair	four star	trash
useful	advice	improve		remove
enjoy		feature		bad

We list signal words that might represent user decisions/actions and become a clue for a human annotator to decide whether a user would acquire (e.g., *buy*, *purchase*, or *pay*) or relinquish (e.g., *uninstall*, *delete*, or *garbage*) the app, as shown in Table 2. To compose the decision wordlist, we select one keyword that most represents each decision class and extract similar words with a word vector representation based on the word2vec[3] algorithm pretrained on

[2]https://www.sentiment140.com

[3]https://code.google.com/archive/p/word2vec/


Text : "I highly recommend this useful app simply because it works. Not bad at all."

decision words = {recommend, useful, bad}

w = 1/3

$s_{recommend}$ = 1/1

s_{useful} = 1/2

s_{bad} = 1/1

c_{bad} = -1

$Dec\_score_{acquiring}$ = 0.33 x 0.5 x 1 = 0.165

$Dec\_score_{recommending}$ = 0.33 x 1 x 1 = 0.33

$Dec\_score_{requesting}$ = 0.33 x 0 x 1 = 0

Dec_score_{rating} = 0.33 x 0 x 1 = 0

$Dec\_score_{relinquishing}$ = 0.33 x 1 x -1 = -0.33

FIGURE 3. Example of applying the annotator formula.

our datasets. The result is shown in Table 2, where words in bold are the keyword for each decision. The keywords listed in Table 2 are not meant to be exhaustive and are likely to be expanded.

$$w = 1/sum(found) \tag{1}$$

$$s_j = \frac{1}{root\_distance() + 1} \tag{2}$$

$$c_j = \begin{cases} -1, & \text{if negation exist.} \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

$$Dec\_score_i = \sum_{j=0}^{n} w \times s_j \times c_j \tag{4}$$

We compute the value of label candidate Dec_score_i using (4), where i (i.e., $i = 0..4$) represents each decision (i.e., acquiring, recommending, requesting, rating, and relinquishing). Variable w denotes the weight obtained from (1). This is performed to normalize the final computation result between -1 and 1.

Next, for each decision word found, we investigate the word significance s in the sentence. We make use of a generated dependency tree that describes the role of each word in the sentence [31]. Fig. 4 shows an example of a dependency tree for the review *"I highly recommend this useful app simply because it works. Not bad at all."* Variable s is obtained by computing the distance of the current word to the ROOT word with (2). A ROOT word in the dependency tree marks the main verbs used in the sentence. When the currently evaluated word is a ROOT word, the value of s is equal to 1 (most important).

Then, we also take into account the verb context by detecting whether there is a negation sign associated with the current word or not. Context variable c is computed with (3). We make use of an NEG tag in the dependency tree using Spacy[4] to detect whether a word has a child word containing negation. For example, the word *bad* in Fig. 4 has a child *not* with an NEG marker. We assign -1 to c to punish the verb that indicates the opposite.

[4]https://spacy.io

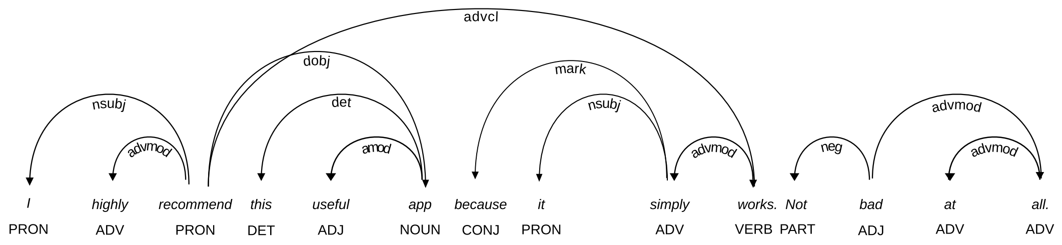


FIGURE 4. Example of a dependency parser tree.

Algorithm 1 User Decision Annotator

```
1: Input: Unlabeled review dataset (previously filtered
   reviews containing argumentative and decision words)
2: Output: Labeled review dataset
3: while not end of review $R_n$ do
4:     $tokens = tokenize(R_n)$
5:     $lemmas = lemmatize(tokens)$
6:     for each $lemma$ in $lemmas$ do
7:         if $lemma$ in decision vocab $V_d$ then
8:             $Nd \leftarrow Nd + 1$
9:         end if
10:    end for
11:    $w \leftarrow 1/sum(Nd)$                              ▷ Eq. (1)
12:    for each decision word $dw$ found do
13:        $s_j \leftarrow 1/(root\_distance() + 1)$          ▷ Eq. (2)
14:        $c_j \leftarrow get\_context(dw)$                  ▷ Eq. (3)
15:        $Dec\_score_i \leftarrow Dec\_score_i + (w * s_j * c_j)$   ▷ Eq. (4)
16:    end for
17:    $Label_n \leftarrow max(Dec\_score_i)$                 ▷ Eq. (5)
18: end while
```

We briefly give an example of how to apply the algorithm to the review *"I highly recommend this useful app simply because it works. Not bad at all"*, as depicted in Fig. 3. From the review, the algorithm captures three decision words based on Table 2, i.e., *recommend*, *useful*, and *bad*, which are associated with $Dec\_score_{recommending}$, $Dec\_score_{acquiring}$, and $Dec\_score_{relinquishing}$, respectively. Based on the generated dependency tree in Fig. 4, the significance values are $s_{recommend} = 1$, $s_{useful} = 0.5$, and $s_{bad} = 1$.

To determine the final decision label for the review, we use the *max* function in (5) to obtain the highest value from Dec_score_i. Hence, the decision ground truth label for this review example is a recommending decision. We evaluate the effectiveness of this algorithm in Section V-A.

$$Label_n = max(Dec\_score_i) \tag{5}$$

B. LINGUISTIC PREPROCESSING

App review is unique because it contains messy unstructured text, such as slang, elongated words, and typos [32]. These characteristics are quite different when compared to product reviews, which often have to go through an inspection process before being posted on the marketplace [33]. Hence, several preprocessing steps should be carefully performed to cope with these traits.

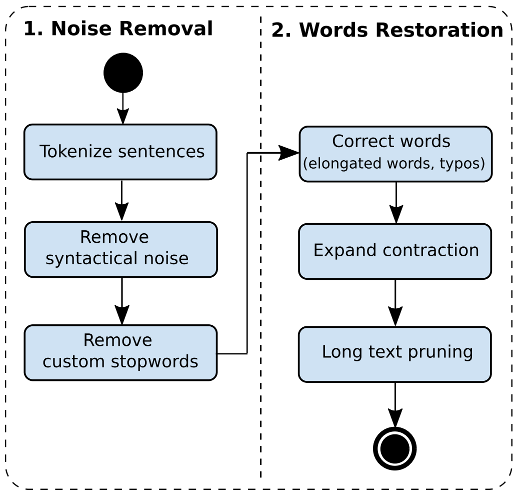


FIGURE 5. Linguistic preprocessing steps.

One common practice in the work of natural language processing (NLP) is to clean sentences from noise by performing stopword removal, stemming and lemmatization [34]–[37]. In this study, we employ two consecutive preprocessing steps, namely, noise removal and word restoration, as depicted in Fig. 5. In brief, we aim to not only clean the text review of lexical noise but also restore words so that the review can be meaningful to the reader. We believe that this refinement of the review sentence is important, as it can provide a strong foundation for the next process (i.e., feature extraction).

In terms of the noise removal step, we remove syntactical noise containing emoticons, symbols, punctuations, and stopwords. For the word restoration step, we perform the following:

1) ELONGATED WORD RECTIFICATION

We target hyperbolic words expressed by users in the text review and convert them to their original form. For example, the word *looooove* is converted to *love*. For this purpose, we utilized an open-source library by zed.[5]

[5]https://gist.github.com/zed/9616954

2) WORD CORRECTION

We also aim to correct typos in a review sentence, and we make corrections based on dictionary words by using character distance. For example, the word *onlin* is corrected to *online*. For this purpose, we utilized the pyspellchecker[6] library for Python.

3) EXPAND CONTRACTION

App reviews often contain slang words or informal abbreviations. We therefore target contracted words in reviews and convert them into dictionary form. For example, we expand not only the word *"I've"*, which is converted to *"I have"*, but also words such as *"osm"* and *"coz"*, which are converted to *"awesome"* and *"because,"* respectively.

4) LONG TEXT PRUNING

A review text can sometimes be extremely long, especially when users explain their experiences that are not related to the app or possibly a spam review. One common approach is pruning the text review to a certain length of words. Jha and Mahmoud [38] suggested pruning text length to a certain n value ($n = 12, 13, 14$), as it gives the optimal classification performance based on their experience. We adopted their suggestion to limit long text reviews to a maximum of 14 words. Our goal in this step is to turn a long review sentence into a more straightforward sentence while maintaining users' expressions regarding their decisions and arguments.

C. FEATURE EXTRACTION

In this phase, we employ two-level feature extraction, namely, sentence-level extraction and review-level extraction.

1) SENTENCE-LEVEL FEATURES

We decompose a review into sentences using a sentence tokenizer.[7] For each sentence, we extract POS tags to differentiate the role of each word in the sentence (e.g., noun, verb, adverb, and adjective). We then employ named entity recognition (NER), which is used for detecting concept entities such as organizations, people, and monetary values. We also take into account some indicative keywords for detecting argumentative phrases in the sentence, such as *because*, *so that*, and *since*.

Additionally, we extract lexical features in terms of n-grams ($n = 1, 2, 3$) and word combinations. An n-gram feature captures a sequence of n words in a sentence that are important for extracting meaningful phrases such as "New York" instead of the individual words "New" and "York." Word combinations, on the other hand, extracts combinations of every word in the sentence that might not be captured by n-grams (Fig. 6). Table 3 shows the differences in the results obtained by n-grams and word combinations for the sentence *"fun simple control highly recommend."* As seen in Table 3, the word combination feature gives more results than the n-gram feature.

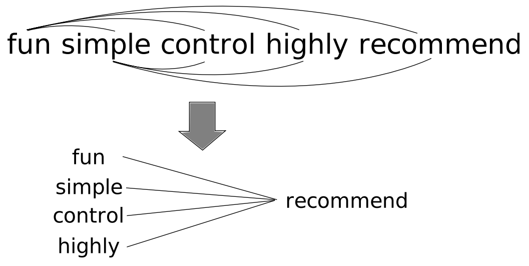


FIGURE 6. Word combination example.

TABLE 3. Word combination example.

Feature	Result
n-grams (n = 2)	"fun simple," "simple control," "control highly," "highly recommend"
word combination	"fun simple," "fun control," "fun highly," "fun recommend," "simple control," "simple highly," "simple recommend," "control highly," "control recommend," "highly recommend"

TABLE 4. Features used for decision prediction.

Feature	Description
Token statistic	Length of the text review
POS tag	Part of speech (POS) tags from review body {Noun, Verb, Adverb, Adjective, Modal, Coordinating Conjunction, Subordinating Conjunction}
NER	The number of named entities found in the review, such as the person, organization, location, event, currency, date, time, and cardinality
N-grams	n-grams of words from the review body for n = 1,2,3
Word Combination	All possible two-words combinations in the sentence
Rating	The rating score determined by the number of stars given by a user
Senti_pos	The positive sentiment value
Senti_neg	The negative sentiment value
Senti_neu	The neutral sentiment value
Senti_com	The compound sentiment value (aggregation of positive, negative, and neutral sentiment)

2) REVIEW-LEVEL FEATURES

For the document or review-level features, we extract rating and sentiment scores and use the results as added features. This is done because we are interested in acknowledging whether user ratings and sentiment can be used to classify user decisions. For rating value, we convert the number of stars given by users in each review into an integer value (a scale of 1 to 5). For the sentiment score, we utilized VADER, a sentiment analysis system by Hutto and Gilbert [39] that was developed based on social media text and is arguably similar to the app review studied in this paper. In addition, VADER returns a more detailed score for each sentiment class, namely, positive sentiment score, negative sentiment score, neutral sentiment score, and compound sentiment score (an aggregation of other scores). Table 4 lists all features that are used for classification tasks.

D. CLASSIFICATION

In this phase, we train machine learning algorithms to predict user decisions in the datasets. Seven classifiers and various

[6]https://pypi.org/project/pyspellchecker/
[7]https://nltk.org

feature configurations described in the previous section are utilized to examine which classifier and configuration yielded the best result. We selected three classifiers from the single methods (i.e., NB, LR, and SVM), three classifiers from the ensemble methods (i.e., RF, AdaBoost, and XGBoost) and a multilayer perceptron (MLP) classifier that represents a neural network method. This selection is based on the notable performances of these classifiers within previous studies [23], [37]. The reason is that we want to compare the performances of different machine learning methods (i.e., single, ensemble, and neural network) with regard to user decision classification tasks.

For NB and LR, we utilize multinomial and liblinear algorithms, respectively. For SVM, we use linear kernel, set $degree = 3$ and $gamma = auto$. For RF, AdaBoost, and XGBoost, we set $n\_estimators = 100$ and use a decision tree as the base estimator. For MLP, we set $max\_iter = 500$. We measure the F1-score on a 10-fold cross-validated dataset to evaluate the performance of the classifiers. For classifier implementation, the Scikit-Learn library[8] is used in a Python development environment.

E. ARGUMENT EXTRACTION

Predicting user decisions and their arguments can be categorized as a cause-effect problem or causality relation extraction. For example, the cause-effect pair is (A_i, D_i, r_i), where $(A_i \rightarrow D_i)$ represents the argument A_i that leads to user decision D_i and r_i denotes the relation. Causality relations can be helpful for discovering unknown relationships among entities [40].

Since we are interested in extracting user arguments in the context of software improvement, we focus on mining software attribute-related arguments in the form of functional requirements (FRs) and nonfunctional requirements (NFRs) (see Fig. 7). FRs relate to the functionalities/features of an application and are mostly expressed explicitly, such as in the review *"I like its super-fast fingerprint reader."* In this review, the phrase *"super-fast fingerprint reader"* refers to an app feature or a function (functional requirement) provided by the app. An NFR-based review, on the other hand, relates to software quality attributes in the form of -ilities (e.g., usability, dependability, supportability, and performance), which are generally implicit [38], [41], [42]. For example, in the review *"this app uses a lot of battery,"* the phrase *"uses a lot of battery"* indicates the user's evaluation concerning a performance aspect of the app. Table 5 describes review examples for both FRs and NFRs.

Next, we explain how values are assigned to nodes and edges of a graph. To extract user arguments from a review, we make use of the explainability module of machine learning algorithms by Ribeiro *et al.* [43]. Fig. 8 shows an example of explainer results obtained with Ribeiro *et al.*'s approach. It shows the most likely contributive words learned by the ML algorithm during training for each prediction.

[8]https://scikit-learn.org/stable/

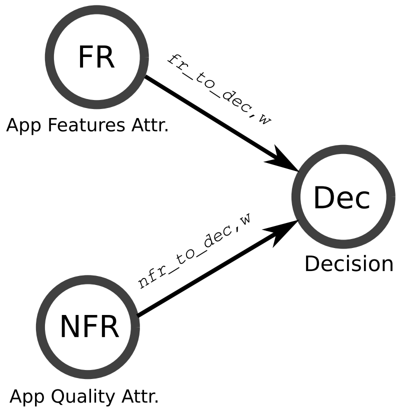
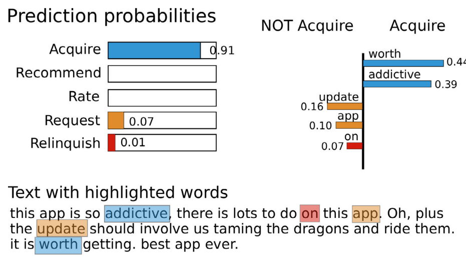


FIGURE 7. Argument to decision representation.



this app is so addictive, there is lots to do on this app. Oh, plus the update should involve us taming the dragons and ride them. it is worth getting. best app ever.

FIGURE 8. Explanation of the classification result.

For example, the review in Fig. 8 is predicted to be an acquiring decision with a prediction probability of 0.91, and the most contributive words are *worth* and *addictive*.

We combine the results of this interpretability with the app's feature extractor using the SAFE pattern [44] because this approach produces only one word that might become biased and ambiguous. SAFE is a pattern-based approach for the purpose of extracting app features from app reviews. Johann *et al.* [44] utilized 18 handcrafted part-of-speech (POS) patterns to extract app features mentioned by users (e.g., "noun-noun-noun" to capture features such as "email chat history" or "adjective-adjective-noun" to capture "super bright flashlight") from a review. The benefit of this approach is that it does not require training or statistical models [45], [46]. We accommodate this approach to extract functional requirements and nonfunctional requirements from reviews.

However, for the results to be valid, several processing steps are required. These steps are summarized in Fig. 9. For each review in the dataset, we predict the user decision using one of the classifiers, as explained in the previous subsection. For each prediction, we extract the contributive keywords from reviews that support the prediction. Then, we compare these keywords with the result of the SAFE pattern, which results in a list of feature phrases. We used dictionary-based software quality attributes by Jha and Mahmoud [38] to determine whether a certain feature can be categorized as an FR or NFR (i.e., dependability, performance, supportability, and

TABLE 5. Descriptions and examples of FRs and NFRs.

Type of Requirement	Description	Example
Functional Requirement (FR)	Aspects related to app-specific features and functionalities	*"I like the app because of the pdf converter feature."*
Usability	An NFR attribute to which an app or software achieves user satisfaction or goals effectively and efficiently.	*"I love the app because it connects to my smartwatch seamlessly."*
Dependability	An NFR attribute to which an app or software performs successfully (failure-free) under the specified period of time and conditions.	*"I couldn't login with my email after I made a purchase."*
Supportability	NFR attribute related to aspects of maintainability and interoperability with other apps or devices.	*"The improvement on the latest update is amazing change to five star."*
Performance	NFR attribute related to software run time, speed and scalability.	*"The loading time is too long only to show the first page."*

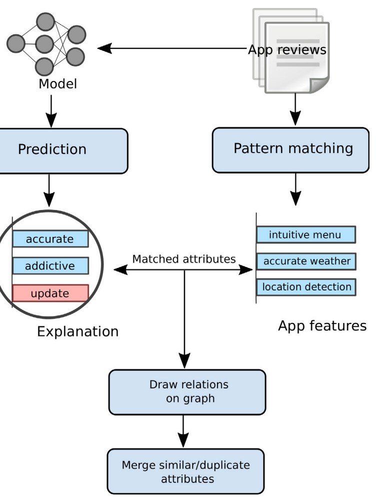


FIGURE 9. Find matching attributes based on the explanation.

usability). If we found a match, then a relation is populated in the causality graph, and the weight is set to 1 (see Fig. 7).

After we draw every relation found in the review, we then find and merge similar attributes based on their context using the combination of two semantic similarity measures, namely, global-context similarity and local-context similarity. For global-context similarity, we employ WordNet [47], an online lexical database that provides synset-based vocabularies. In (6), t_1 and t_2 represent the terms, *depth* is the depth of the term from the root, and *lcs* is the least common subsumer of both terms [48].

To accommodate domain-dependent words, we utilized cosine similarity based on a word2vec vector trained on our app review datasets (7). In (7), V_{t_1} and V_{t_2} represent vectors of t_1 (term 1) and t_2 (term 2), respectively. The rationale behind this is that text reviews contain many domain-dependent words that might not be recognized by general lexical databases. For example, the words *screen* and *loading* have higher similarity in the latter approach than in the prior approach. Thus, we can merge similar arguments such as *"work weather app"* and *"accurate weather app"* into one argument ($sim = 0.839$). For every merging process, we increment the weight of the edge by 1 to indicate the

TABLE 6. Semantic similarity example for merging feature attributes.

Feature 1	Feature 2	Semantic Similarity
the game freeze	the game hang	0.885
app weather work	accurate weather app	0.705
need speed improvement	great speed app	0.651
use this app	useless this app	0.508
fix the bug	upgrade the app	0.353

TABLE 7. Details of acronyms.

Acronyms	Details
PRON	Pronoun
DET	Determinant
ADV	Adverb
ADJ	Adjective
CONJ	Conjugation
FR	Functional Requirement
NFR	Nonfunctional Requirement
NB	Naive Bayes classifier
LR	Logistic Regression classifier
SVM	Support Vector Machine classifier
RF	Random Forest classifier
AdaBoost	Adaptive Boosting classifier
XGBoost	Extreme Gradient Boosting classifier
MLP	Multilayer Perceptron classifier
POS	Part-of-speech tags
NER	Named Entity Recognition
BoW	Bag-of-words
RT	Rating feature
WC	Word Combination feature
SC	Sentiment score feature

importance of the attribute. To give a better context, Table 6 shows some similarity results on several different arguments. Based on the experiment, we set the similarity threshold to 0.7 to determine whether two arguments should be merged or not. The final attributes, as a result of the merger process, are then populated into a causality graph with the Pyvis library [49]. As a reference for readers, we display details of acronyms used throughout the paper in Table 7.

$$sim_{global}(t_1, t_2) = 2 * \frac{depth(lcs(t_1, t_2))}{depth(t_1) + depth(t_2)} \tag{6}$$

$$sim_{local}(t_1, t_2) = \frac{V_{t_1} * V_{t_2}}{\|V_{t_1}\|\|V_{t_2}\|} \tag{7}$$

$$sim_{aggregate} = \frac{sim_{global} + sim_{local}}{2} \tag{8}$$

IV. DATASET

We conduct the experiments using two app review datasets as follows:

1) Android app and feedback from F-droid[9] (288k). This dataset is used because it provides a large amount of Android app reviews specifically for software evolution and quality improvement [50], which is similar to the objective of this study.
2) Amazon reviews for android[10] (752k) [51]. In addition to the size of the user reviews it provides, this dataset is used as a comparison for the F-droid dataset since Amazon has different user characteristics, especially in terms of style and user expression.

The F-droid dataset contains 395 different apps with an average number of 730 reviews for each app, while the Amazon dataset contains 750 apps with an average number of 500 reviews for each app. Both datasets have different characteristics since they contain reviews from different marketplaces. As Table 8 shows, in the F-droid dataset, users are more concise in expressing their reviews, with an average number of 13.482 words in a review compared to the Amazon dataset with an average number of 48.720 words in a review. This is in line with Amazon's policy in which reviews are filtered before they are posted to the website.

TABLE 8. Dataset statistics.

Information	F-droid	Amazon
number of reviews	288,065	752,928
number of sentences	438,238	2,481,148
number of words	3,883,631	36,682,897
avg sentences in a review	1.521	3.295
avg words in a review	13.482	48.720
number of apps	395	750

V. RESULT

In this section, we present our results obtained by the following three experiments, which consist of examining the effectiveness of our machine-based annotator, evaluating the performance of the classifiers and feature configurations, and assessing the efficacy of the argument extractor.

A. MACHINE-BASED ANNOTATOR

Based on our rule-based annotation algorithm, we obtained an identical label distribution for both the F-droid (Fig. 10(a)) and Amazon (Fig. 10(b)) reviews. Both figures describe a similar pattern where users express more acquirements and requests in their reviews compared to recommendations.

We compared this result with that of the human annotators by randomly selecting 500 reviews and asking the annotators to label the reviews. The human-based annotation process involved 3 annotators (different from the authors of this paper), consisting of 1 Ph.D. student and 2 Masters students in the field of computer science. All annotators were

[9]http://f-droid.org
[10]http://jmcauley.ucsd.edu/data/amazon/index.html

TABLE 9. Example of the machine-labelled review results.

Label	Review
Acquiring	Amazing A very useful app. I originally downloaded the free version from github but decided to buy it here as the dev deserves the money.
	just downloaded this app today and when you open the game when you get to the home page it says pixel mall has stop I'm so disappointed now if this was pay I want my money back
Recommending	Great and Simple ... Simple but moderate interface and great features like compression exclude from media scan etc. And best part..... its free....!!! Recommend it to everyone....
	Not a very bad app, but not the best. I recommend TuneIn app which has better function to find local stations, (the TuneIn app made this easy). This app did not have as comprehensive catagory functions.
Requesting	Needs option to check if BT is connected as the Headset on doesn't seem relevant to my GS4 unless I have a wired headset plugged in. Very promising app though.
	I can play at any level as many times a day as I want. Love the hints, and the explanations. I have improved my sudoku playing because of this program.
Rating	Good app. This app is really good. I only gave it 4 stars cause the menu options blur badly while scrolling. Other than that it's perfect and very helpful.
	I don't know what to rate it as it is not working I guess. Can you tell me if it will work on ASUS ZenFone?
Relinquishing	Poor Not working anymore. Tried uninstalling n installing but still not working. Worked well for two months but now it's just occupying space n nothing else. So bye bye VN. Deleted.
	So well integrated it could be part of the OS. No crashes or hangups. Love this app.

Italic reviews indicate incorrectly assigned labels compared to the human-annotator.

TABLE 10. Interagreement with human coders.

Decisions	Coder 1	Coder 2	Coder 3	Avg
Acquiring	0.84	0.78	0.76	0.79
Recommending	0.62	0.58	0.60	0.60
Requesting	0.81	0.69	0.77	0.75
Rating	0.69	0.66	0.74	0.70
Relinquishing	0.73	0.78	0.72	0.74
Total	0.77	0.70	0.72	0.72

briefed before performing the coding process. The result yields a favorably acceptable interagreement ratio between 0.58-0.84, as depicted in Table 10. The lowest average percentage belongs to the recommending decision (0.60), while the highest is the acquiring decision (0.79), with a total average percentage of 0.72 for all five decisions. Based on our observations, the recommending decision obtains the lowest agreement because users often express their recommendation together with other decisions, such as buying or rating. In other words, some reviews contain more than one user action, yielding a dispute in the annotation result.

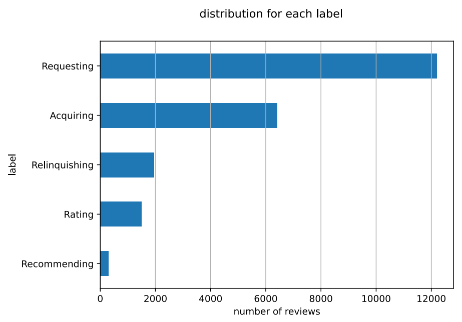
Acquiring, requesting, and relinquishing decisions have a fairly good interagreement value when compared to the others, with average percentage values of 0.79, 0.75, and 0.74, respectively. This is because all three decisions have a relatively specific usage of words that make our machine-based annotator work properly. For example, in acquiring decisions, users mostly used words such as useful, download,

TABLE 11. **Classification Results on F-Droid Reviews (F1-score).**

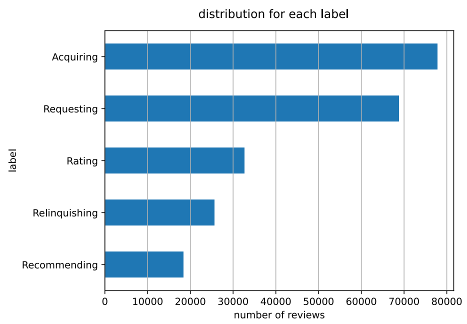
Features	NB	LR	SVM	RF	AdaBoost	XGBoost	MLP
Baseline (BoW)	0.84 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.92 ± 0.01	0.91 ± 0.02	0.94 ± 0.01	0.91 ± 0.01
Baseline (Rating)	0.18 ± 0.01	0.25 ± 0.02	0.21 ± 0.01	0.23 ± 0.02	0.24 ± 0.02	0.25 ± 0.01	0.23 ± 0.02
Baseline (Sentiment)	0.25 ± 0.01	0.26 ± 0.02	0.24 ± 0.01	0.24 ± 0.02	0.29 ± 0.01	0.29 ± 0.02	0.31 ± 0.02
BoW+WC	0.84 ± 0.01	$\mathbf{0.93 \pm 0.02}$	$\mathbf{0.94 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.95 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$
BoW+RT	0.84 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.92 ± 0.01	0.90 ± 0.01	0.94 ± 0.01	0.91 ± 0.01
BoW+SC	0.83 ± 0.02	0.93 ± 0.01	0.93 ± 0.01	0.91 ± 0.01	0.91 ± 0.02	0.94 ± 0.01	0.91 ± 0.01
BoW+NER	0.84 ± 0.01	0.93 ± 0.01	0.93 ± 0.01	0.92 ± 0.01	0.91 ± 0.02	0.94 ± 0.01	0.91 ± 0.01
BoW+WC+RT+SC	$\mathbf{0.85 \pm 0.01}$	$\mathbf{0.93 \pm 0.02}$	$\mathbf{0.94 \pm 0.01}$	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{0.95 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$

TABLE 12. **Classification Results on Amazon Reviews (F1-score).**

Features	NB	LR	SVM	RF	AdaBoost	XGBoost	MLP
Baseline (BoW)	0.70 ± 0.02	0.80 ± 0.02	0.80 ± 0.03	0.84 ± 0.04	0.72 ± 0.01	0.86 ± 0.03	0.75 ± 0.04
Baseline (Rating)	0.20 ± 0.01	0.23 ± 0.01	0.25 ± 0.07	0.24 ± 0.03	0.24 ± 0.02	0.24 ± 0.02	0.24 ± 0.03
Baseline (Sentiment)	0.27 ± 0.02	0.30 ± 0.02	0.30 ± 0.06	0.29 ± 0.04	0.30 ± 0.03	0.30 ± 0.03	0.29 ± 0.04
BoW+WC	0.70 ± 0.03	$\mathbf{0.81 \pm 0.04}$	$\mathbf{0.82 \pm 0.04}$	$\mathbf{0.85 \pm 0.04}$	$\mathbf{0.73 \pm 0.02}$	$\mathbf{0.87 \pm 0.02}$	$\mathbf{0.76 \pm 0.03}$
BoW+RT	0.70 ± 0.02	0.78 ± 0.02	0.80 ± 0.04	0.84 ± 0.05	0.72 ± 0.02	0.85 ± 0.03	0.75 ± 0.04
BoW+SC	0.70 ± 0.03	0.77 ± 0.02	0.80 ± 0.05	0.81 ± 0.05	0.72 ± 0.02	0.86 ± 0.03	0.75 ± 0.04
BoW+NER	0.71 ± 0.02	0.80 ± 0.02	0.81 ± 0.03	0.84 ± 0.04	0.72 ± 0.02	0.86 ± 0.03	0.75 ± 0.03
BoW+WC+RT+SC	$\mathbf{0.71 \pm 0.02}$	$\mathbf{0.81 \pm 0.04}$	$\mathbf{0.82 \pm 0.04}$	$\mathbf{0.85 \pm 0.04}$	$\mathbf{0.73 \pm 0.02}$	$\mathbf{0.87 \pm 0.02}$	$\mathbf{0.76 \pm 0.03}$



(a) Decision distribution on F-Droid dataset



(b) Decision distribution on Amazon dataset

FIGURE 10. **Decision distribution.**

and buy, while for expressing rejection, users mostly used words such as poor, uninstall, and delete.

To provide more context, we display some examples from our annotator for both correct and incorrect samples in Table 9. According to the experiment, our machine-labeling algorithm is effective for straightforward reviews but has the opposite result for longer reviews.

Although some results are not yet on par with the results obtained by human-based labeling, this approach has a significant impact on increasing the processing speed because it is done automatically by machines and only takes minutes compared to weeks when human labor is used. Additionally, the agreement results in some decision classes that reached more than 0.80 points have the potential for improvement in future research.

B. CLASSIFIER PERFORMANCE

We evaluate the performance of the classifiers with respect to feature configurations. Table 11 and Table 12 show the results of the experiments with different feature configurations.

We set three baselines, namely, bag-of-words (BoW), sentiment score (SC), and user rating (RT), for each of the features. Out of the three baselines, BoW performed quite well on both datasets and obtained the highest F1-score of 0.94 (XGBoost) on the F-droid dataset and 0.86 (XGBoost) on the Amazon dataset.

Moreover, the sentiment and rating features are not effective for classifying the decision reviews. When the sentiment score is used as a feature, the highest F1-scores are 0.31 (F-droid) and 0.30 (Amazon); however, the rating feature obtains the highest F1-score of 0.25 for both datasets. Even the combination of BoW+RT (BoW and rating) and

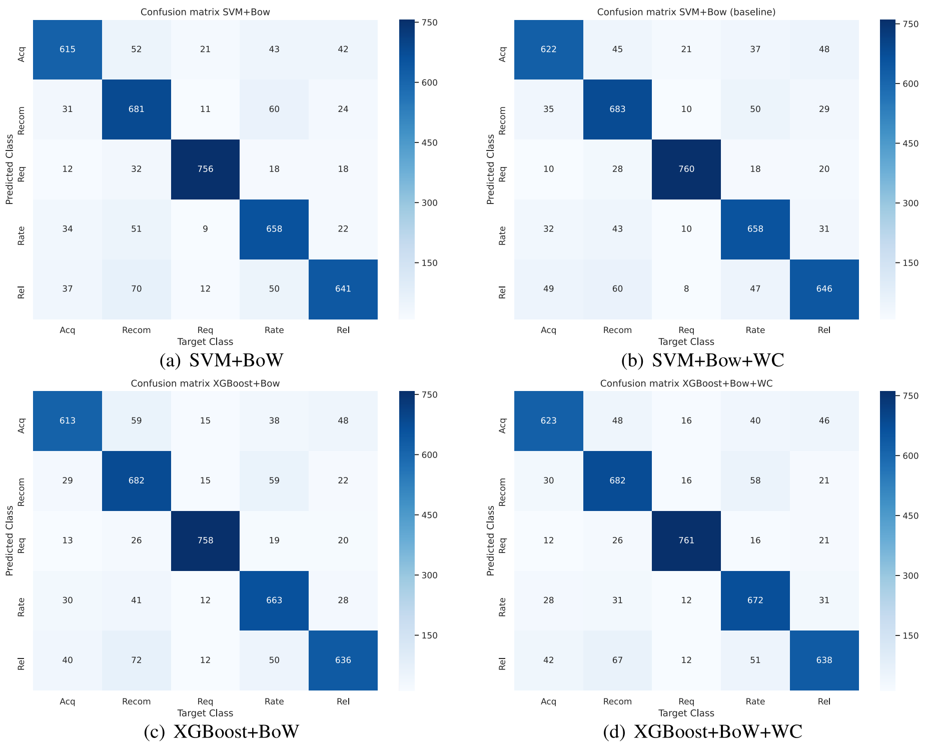


FIGURE 11. **Confusion matrix for Amazon reviews (n_{test} = 4000 or 20% of the dataset) with previously undersampled data targeting equal review distribution for each class.**

BoW+SC (BoW and sentiment) cannot improve the performance of the classifiers. It can be concluded that the rating and sentiment are not discriminant for predicting user decisions.

In general, the best F1-score is obtained by the combination of all features (BoW+WC+RT+SC) as well as the BoW+WC (BoW and word combination) feature combination with F1-scores of 0.95 (F-droid) and 0.87 (Amazon). In terms of classifier performance, XGBoost, SVM, RF, and MLP performed very well compared to the other classifiers.

To further examine the classifiers' performance results against the classification result in each class, we display the confusion matrices of the experiments in Fig. 11. As seen in Fig. 11, the BoW+WC feature combination slightly increases the correct prediction for almost all decision categories. The requesting class experiences the highest correct prediction, while the acquiring class shows the lowest. In addition, recommendation decisions and rating decisions receive the highest misprediction rates compared with other classes. Based on our observation, the explanation for this is that many of the acquiring users also wrote their recommendations and ratings to express their satisfaction, such as in the review *"I bought this app last month and didn't regret at all, highly recommended for everyone."* In this review, the user acquires (i.e., purchases) and recommends the app at the same time, which makes the classifier incorrectly predict the decision.

To evaluate the time performance, we plot the time required (i.e., training and testing) by all classifiers to finish the

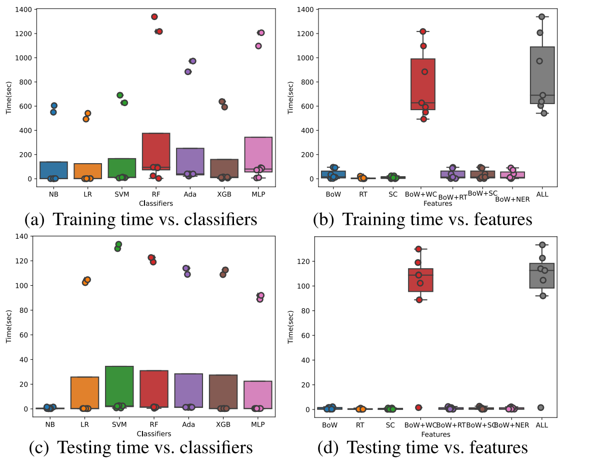


FIGURE 12. **Computation time for all classifiers and feature configurations with $n = 20,000$, which has an 80% and 20% data review split for training and testing, respectively.**

classification task with a boxplot, as depicted in Fig. 12. For the training time, RF and MLP obtained a slightly higher median value than the other classifiers, followed by AdaBoost and XGBoost (Fig. 12(a)). From the aspect of feature configuration, BoW+WC and the combination of all features (ALL) require significant training and testing times, indicated by their high median and interquartile range values compared to those of the other features (Fig. 12(b) and Fig. 12(d)). The reason for this is that the WC feature composes all word-pair combinations of the review text. Despite the increased performance, WC might require quite expensive computations, especially if the review data become large. This poses potential future research on how to reduce the dimensionality of the WC feature. Furthermore, the SVM classifier requires the longest testing time, while NB is the fastest (Fig. 12(c)). Combining the trade-off between F1-score and runtime, XGBoost achieves the best overall result compared to the other classifiers.

C. ARGUMENT EXTRACTION

We display the result of the causality graph for the subset of the Amazon dataset containing 5000 reviews (Fig. 13) based on the algorithm described in subsection III-E. As shown in the graph, there are five epicenters that represent the five user decisions. Each epicenter is surrounded by nodes with different colors representing different types of user arguments (i.e., FRs or NFRs).

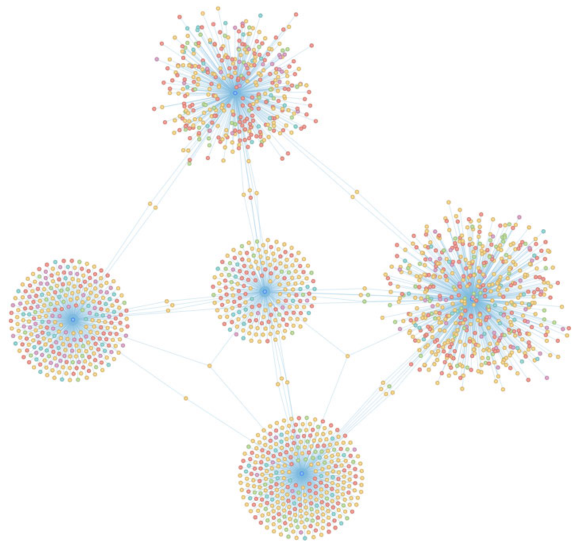


FIGURE 13. Causality graph (Amazon reviews).

A graph that represents user arguments and decisions can reveal unknown relationships in the data, for example, an argument (node) that connects to more than one decision. Additionally, we can easily acknowledge higher priority attributes based on the weight of the edge/relation. Table 13 shows the top arguments (highest weights) for each decision for the F-droid dataset. As seen in the table, the arguments arguably make sense for most of the decisions. For example, for the recommending decision, users argue that the app has easy control, many options (usability category), and supports direct synchronization (supportability category).

Fig. 14 summarizes the distribution of the argument types (FRs and NFRs) for each decision class. From the figure, app features and usability are arguments most written by users for expressing acquiring, rating, and relinquishing decisions. In addition, users decide to abandon the apps based on dependability and usability reasons. Performance reasons (e.g., battery, speed, and memory) were least expressed by users in any decision.

VI. DISCUSSION

In this section, we discuss important findings from our experiments by revisiting the previous research questions described in subsection II-B.

A. RQ1. DISCRIMINANT AND NONDISCRRIMINANT FEATURES FOR CLASSIFYING USER DECISIONS

Based on our results, lexical features (i.e., n-grams and word combinations) are good predictors for classifying user decisions. By contrast, statistical features, such as user ratings, sentiment scores, and NERs, are nondiscriminant for the classification task.

To strengthen this conclusion, we plot user sentiments and ratings for each decision class with a bar plot and a scatter plot (see Fig. 15). In both figures (Fig. 15(a) and Fig. 15(b)), the user rating for all decisions has the same pattern (except for relinquishing decision), where most users give 4- to 5-star ratings regardless of their decision in the reviews (i.e., acquiring, requesting, rating, or recommending). By contrast, for relinquishing decisions, most users give a 1-star rating.

Similarly, the sentiment and the decision scatter plots, as depicted in Fig. 15(c) and Fig. 15(d), do not show different patterns than the five decisions. From the plots, the sentiment score measurement for each decision is fairly even from the best to the worst sentiments (-1 indicates very bad and 1 indicates very good sentiment). This can be caused by the characteristics of users who make reviews containing two perspectives, namely, the pros and cons of an application.

These findings should make us aware that rating or sentiment alone is not enough to assess whether users like the application or not. In other words, from a practitioner prospective, we need to further ask what the actual meaning of a user rating is in our application. Behind the rating values, there are users who are willing to buy and even recommend an application, but on the other hand, there are users who merely like the app without any intention to acquire the app. This is where learning user decisions/behaviors comes into play, from which we can more deeply analyze the actual value of our application.

B. RQ2. VARIETY OF USER ARGUMENTS THAT DRIVE DECISIONS

From the experiments, user arguments related to app functionalities/features are arguments most expressed by users for all five decisions (see Fig. 14). From a decision perspective, each user argument represents a fairly specific rationale. For the acquiring decisions, the most common user reasons were acquiring the app's free version, enjoying the app, and connectivity with other systems. For recommendation decisions, users encourage other users for reasons such as the app's simple control, synchronization support, and battery performance. Arguments, such as asking for new updates, adding more options, and fixing issues, support

TABLE 13. Top arguments for each decision.

Decision	FR	Usability	Supportability	Performance	Dependability
Acquiring	enjoy the game, fun free game, useful simple news, great useful app, be nice game	touch the screen, ease of access, offer lock screen, control music playback, be interface design	play free version, update the game, specific bluetooth device, have good service, support future development	fast download speed, freeze up game play, conserve battery, problem with lag	login with facebook, reset the game, do restart, have an issue, fix the connection
Recommending	recommend the app, like this game, download this game, decent gallery app, recommend to people	have easy control, nice clean interface, lot of option, good home screen, be an option	support direct sync, download version for offline, wifi network signal, scanning of wifi, support language user	great battery monitoring, need speed improvement, amount of memory, have app for battery, lot of battery	resolve the bug, get this error, incorrect password, be an error, think the glitch
Requesting	need this app, add wordpress site, need more app, more easy level, add more character	select the option, add an option, update the view, offer the option, give this option	need an update, web page version, pro version review, make an update, awesome customer service	memory management issue, memory via otg, drain battery issue, tolerate this memory, have a load	fix this problem, fix a bug, keep some privacy, fix the compatibility, perform the fix
Rating	deserve a star, give the app, final fifth star, fan of star, star puzzle entertainment	nifty manual control, increase frame rate, clear playing screen, rotate the screen	download this version, contact customer support, like this version, high definition version, contact customer service	game to load, little battery drain, improve since last update, play offline, get stuck time	fix the issue, brief error message, fix this user, crash the app, get error message
Relinquishing	reinstall the app, waste of money, waste of time, get a hang, have no idea	get black screen, whole website crash, change the entry, read the review, have many ad	poor customer support, do the update, ruin old version, update the app, connect with friend	memory on phone, load on phone, get stuck hint, need the battery, battery life information	delete the app, restart each time, day feature crash, cause the crash, crash every time

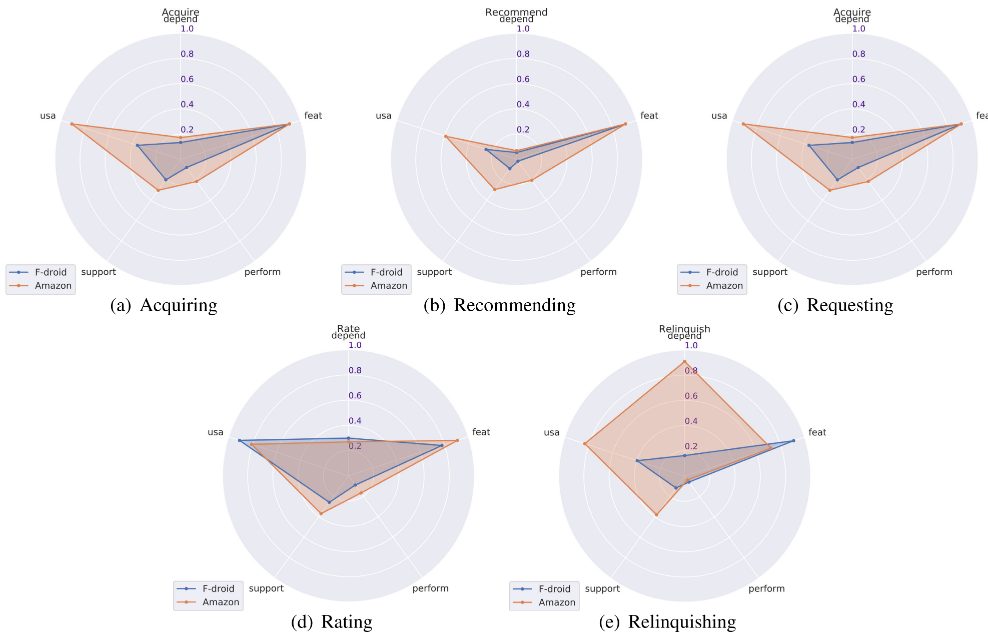
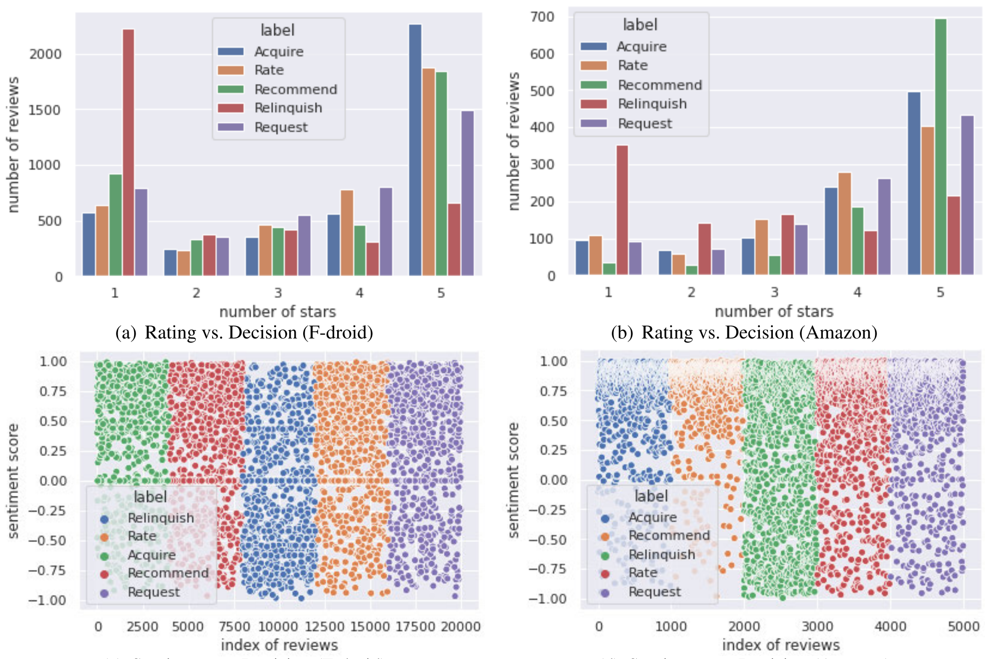


FIGURE 14. Distribution of FR/NFR attributes among 5 decision categories.

users' requesting decisions. For rating decisions, users based their decisions on justifications, such as camera function, easy app control, and an app's user interface. Dependability reasons, such as an app having bugs, an app showing errors, and a crashed app, were the most expressed arguments supporting relinquishing decisions.



(a) Rating vs. Decision (F-droid) (b) Rating vs. Decision (Amazon)

(c) Sentiment vs. Decision (F-droid) (d) Sentiment vs. Decision (Amazon)

FIGURE 15. Decision vs. Rating and sentiment.

From the argument's perspective, both FR- and NFR-based arguments present interesting results. For example, in terms of usability, users based their decisions mostly on an app's user interface, app control, and available app options. These results are in contrast with performance-related arguments, where users mainly mentioned memory management, battery consumption, and app loading time. These findings stress that nonfunctional requirements are pivotal to user considerations in making decisions.

In general, our results are consistent with the study by Lim *et al.* [52], which conducted a massive survey concerning user behavior involving 4,824 respondents across 15 countries. For example, Lim *et al.* reported that most app users did not pay for apps (57%) and chose the free version of the app. They also pointed out that NFR-based requirements were important considerations for app users, which is similar to the results of this study.

A slightly different result was shown in terms of the relinquishing decision. Lim *et al.* stated that users abandon an app because they no longer need the app (44%) and found better alternatives (38%), while our result indicates that users abandon an app for common reasons such as the presence of bugs, errors, and crashes. This result, however, is in line with the result of another study by Khalid *et al.* [21].

C. RQ3. COMPLIANCE WITH HUMAN-ASSESSMENT

Table 13 shows the extraction results, which are difficult to evaluate, from hundreds of apps, as described in subsection IV. To answer RQ3, we focus and analyze reviews from a single app and compare the results with the results of a human assessment to seek evidence on the efficacy of our approach.

To accomplish this, we utilized reviews from the Bitdefender app. This app was chosen because it has an active community forum on its website.[11] Among the many pages on the Bitdefender website, we focused on the page entitled *sent_to_devel*, which compiles all user feedback for the next app release development, which is similar to the purpose of this study. We list the top 10 threads from the forum from 2018 until 2020 based on the number of user views and interactions. On the other hand, we crawled Bitdefender app reviews from the Google Play Store and performed argument and decision extraction.

TABLE 14. Comparison with the Bitdefender user forum.

No.	Our approach (Requesting decision)	Bitdefender user forums
1.	Antitheft feature	**Excessive memory utilization**
2.	Add fingerprint usage	Blocking Nvidia Geforce
3.	**Lock trusted network**	**Password manager my wallet password**
4.	**Wish VPN service**	**Firewall blocks remote connections**
5.	Autopilot option help	Installer hangs
6.	**Free up memory**	**Block access to network shares**
7.	**Add password wallet**	Bitdefender blocking blizzard products
8.	**Upgrade for firewall**	**VPN DNS**
9.	Report battery usage	**Parental control not working**
10.	**Parental control feature**	Accessing user app data folder

Table 14 shows the comparison of our result (i.e., the extracted arguments from the requesting decision) and the Bitdefender user forum, where our result captured 6 out

[11] https://community.bitdefender.com/en/discussions/tagged/sent-to-devel/, accessed: Nov 29th, 2020

of 10 requests from user feedback in the forum. This result is arguably acceptable considering that we use an automatic machine-based labeling approach. In addition to this result, our approach captures richer information concerning app features from several user decision perspectives (i.e., acquiring, recommending, rating, and relinquishing) rather than only one perspective (request/suggestion). For example, in terms of acquirement, our results reveal that some of the common arguments expressed by users are that the apps has an excellent security package, reduced prices, and great technical support.

VII. LIMITATIONS & VALIDITY RISKS

Our approach in this study has several limitations; i.e., it merely employs statistical machine learning algorithms that rely on a text-based feature engineering extraction process and does not include a deep learning algorithm. Additionally, this study omits the temporal aspect of the review, such as users who updated their reviews and changed their decisions over time (e.g., added more stars).

Internal Validity. This risk concerns factors that might influence the results from an internal perspective. One risk might come from the annotation process performed by human coders, which can be subjective. To counteract this issue, we selected coders with a computer science background and briefed the coders prior to the annotation process.

External Validity. This risk is related to factors from an external aspect, e.g., the validity of user reviews in the corpus. We employ a relatively small dataset (1 million in total out of billions of reviews in an app marketplace), which might not be representative. To alleviate this issue, we employ two different datasets (from two different marketplaces) that represent different user characteristics and behaviors.

VIII. RELATED STUDIES

Data-driven requirement. The world of software engineering has moved toward data-driven requirement analysis, which utilizes explosively user-generated content, in recent years [1]. Chen *et al.* [32] extracted informative reviews from a whole review corpus to help app developers understand user requests. Different studies tried to reveal aspects that users dislike [53] and like [19] within an app. Mining information about bugs and feature requests [17], as well as app maintenance [4], has also attracted attention from researchers. While these works explore what users like, dislike, and want, we focus on arguments or why users perform certain actions/decisions. Our study targets two aspects from the reviews, namely, user decisions (e.g., purchasing, ratings, and recommendations) that could describe user intensity in supporting apps and user arguments that reflect an app's features/functionalities in favor of their decision.

Argument mining. Argument mining has recently gained popularity from researchers because it can reveal reasoning comprehension in argumentative text. Earlier studies of argument mining targeted text essays and legal text [26], [35], [36]. More recent studies conducted argument mining on app reviews to learn user justifications over certain user behaviors [23], [37]. A recent survey by Cabrio and Villata reported weaknesses and challenges in the study of argument mining within the last decade [54]. In this study, we had a different point of view because we highlight specific user arguments targeting software quality attributes (i.e., FRs and NFRs) in correlation with their decisions expressed in the reviews.

Explainable AI. As the performance of machine learning increases every day, the research community and users demand a more transparent system that can provide interpretations, at some level of detail, of the machine learning results. Ribeiro *et al.* [43] developed a proxy model out of a full deep neural network model to identify inputs that are most influential on a decision. Zilke *et al.* [56] introduced a decision tree model to decompose a deep neural network model to easily explain predictions. Gilpin *et al.* performed a survey of explanatory artificial intelligence (XAI) and pointed out that current approaches are insufficient [55]. They concluded that machine learning models are hard to explain; hence, our study uses an interpretation module and combines it with other techniques (i.e., semantic similarity) to extract informative features from the classification results.

IX. CONCLUSION & FUTURE WORK

In this paper, we performed user argument extraction based on user decisions regarding an app. We developed a rule-based user decision annotation algorithm for building annotated datasets for training purposes that reduced the time consumed when human annotators were used. We extracted sentence-level and review-level features from text and trained two datasets using various feature configurations. We built a causality graph by extracting argument-to-decision relation representations from reviews. We evaluated and discussed the results of our framework and compared them with the results of the human annotators, which were favorably acceptable.

Based on our experiments, the sentiment score and rating cannot be used as user decision predictors. In other words, the sentiment score and rating alone are not enough to measure the intensity of users when expressing their satisfaction (i.e., rate, recommend, and purchase) or dissatisfaction (i.e., complaint, request, and relinquish) with an app. In contrast, lexical features (i.e., n-grams and word combinations) produce the best results for predicting user decisions. In terms of the classifiers, most of the classifiers utilized perform well for the user decision classification task. However, XGBoost outperformed other classifiers based on the F1-score and runtime assessment.

From the user argument perspective, we found that both FRs and NFRs play a significant role for app users in driving their decisions. Users based their decisions mostly on app features, such as easy levels, multiple game characters, and decent galleries. From an NFR point of view, usability aspect (e.g., clean user interface, easy application control, and availability of options), supportability (e.g., version update, direct synchronization, and customer support), and performance aspect (e.g., memory management, battery consumption, and loading time) greatly influence many users in making their decisions. Users abandoned apps based on common arguments, such as the existence of bugs, errors, and crashes.

In the future, we will extend our study in two directions to learn user decisions and justifications. First, we are interested in enhancing our approach by employing dimensionality reduction on word combination features and utilizing language model generation with an unsupervised deep learning approach. Second, we will investigate more user actions to address multilabel classification, that is, users who perform more than one decision in a review.

ACKNOWLEDGMENT

Anang Kunaefi thanks the Ministry of Religious Affairs, Republic of Indonesia, 5000 Doktor program who has given the opportunity and support for him to pursue the Ph.D. degree at Kumamoto University, Japan.

REFERENCES

[1] D. Pagano and B. Bruegge, "User involvement in software evolution practice: A case study," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, San Fransisco, CA, USA, May 2013, pp. 953–962.

[2] W. Maalej, M. Nayebi, T. Johann, and G. Ruhe, "Toward data-driven requirements engineering," *IEEE Softw.*, vol. 33, no. 1, pp. 48–54, Jan. 2016.

[3] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: How far can we go?" in *Proc. 40th Int. Conf. Softw. Eng.*, Gothenburg, Sweden, May 2018, pp. 94–104, doi: 10.1145/3180155.3180195.

[4] S. Panichella, A. Di Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can i improve my app? Classifying user reviews for software maintenance and evolution," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Bremen, Germany, Sep. 2015, pp. 281–290.

[5] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proc. 38th Int. Conf. Softw. Eng.*, Austin, TX, USA, May 2016, pp. 14–24, doi: 10.1145/2884781.2884818.

[6] M. Luca, "Reviews, reputation, and revenue: The case of Yelp.com," Harvard Univ., Boston, MA, USA, Harvard Bus. School Work. Paper 12-016, Sep. 2011.

[7] R. P. L. Buse and T. Zimmermann, "Analytics for software development," in *Proc. FSE/SDP workshop Future Softw. Eng. Res. (FoSER)*, Santa Fe, NM, USA, 2010, pp. 77–80.

[8] T. Menzies and T. Zimmermann, "Software analytics: So what?" *IEEE Softw.*, vol. 30, no. 4, pp. 31–37, Jul. 2013.

[9] R. Jongeling, S. Datta, and A. Serebrenik, "Choosing your weapons: On sentiment analysis tools for software engineering research," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Bremen, Germany, Sep. 2015, pp. 531–535.

[10] M. Kim, T. Zimmermann, R. DeLine, and A. Begel, "The emerging role of data scientists on software development teams," in *Proc. 38th Int. Conf. Softw. Eng.*, Austin, TX, USA, May 2016, pp. 96–107.

[11] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Sydneym NSW, Australia, Aug. 2015, pp. 985–994, doi: 10.1145/2783258.2783370.

[12] A. H. A. M. Siagian and M. Aritsugi, "Robustness of word and character N-gram combinations in detecting deceptive and truthful opinions," *J. Data Inf. Qual.*, vol. 12, no. 1, pp. 1–24, Jan. 2020, doi: 10.1145/3349536.

[13] E. Cambria, S. Poria, A. Gelbukh, and M. Thelwall, "Sentiment analysis is a big suitcase," *IEEE Intell. Syst.*, vol. 32, no. 6, pp. 74–80, Nov. 2017, doi: 10.1109/MIS.2017.4531228.

[14] M. S. Akhtar, A. Ekbal, and E. Cambria, "How intense are you? Predicting intensities of emotions and sentiments using stacked ensemble [application notes]," *IEEE Comput. Intell. Mag.*, vol. 15, no. 1, pp. 64–75, Feb. 2020, doi: 10.1109/MCI.2019.2954667.

[15] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *J. Syst. Softw.*, vol. 125, pp. 207–219, Mar. 2017.

[16] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE Trans. Softw. Eng.*, vol. 43, no. 9, pp. 817–847, Sep. 2017.

[17] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. IEEE 23rd Int. Requirements Eng. Conf. (RE)*, Ottawa, ON, Canada, Aug. 2015, pp. 116–125.

[18] M. Gomez, B. Adams, W. Maalej, M. Monperrus, and R. Rouvoy, "App store 2.0: From crowdsourced information to actionable feedback in mobile ecosystems," *IEEE Softw.*, vol. 34, no. 2, pp. 81–89, Mar. 2017, doi: 10.1109/MS.2017.46.

[19] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. IEEE 22nd Int. Requirements Eng. Conf. (RE)*, Karlskrona, Sweden, Aug. 2014, pp. 153–162.

[20] X. Gu and S. Kim, "'What parts of your apps are loved by users?' (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Lincoln, NE, USA, Nov. 2015, pp. 760–770.

[21] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?" *IEEE Softw.*, vol. 32, no. 3, pp. 70–77, May 2015.

[22] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proc. 40th Int. Conf. Softw. Eng.*, Gothenburg, Sweden, May 2018, pp. 48–58.

[23] Z. Kurtanović and W. Maalej, "On user rationale in software engineering," *Requirements Eng.*, vol. 23, no. 3, pp. 357–379, Sep. 2018, doi: 10.1007/s00766-018-0293-2.

[24] A. Kunaefi and M. Aritsugi, "Characterizing user decision based on argumentative reviews," in *Proc. IEEE/ACM Int. Conf. Big Data Comput., Appl. Technol. (BDCAT)*, Leicester, U.K., Dec. 2020, pp. 161–170, doi: 10.1109/BDCAT50828.2020.00002.

[25] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003.

[26] R. M. Palau and M. F. Moens, "Argumentation mining: The detection, classification and structure of arguments in text," in *Proc. 12th Int. Conf. Artif. Intell. Law (ICAIL)*, Barcelona, Spain, 2009, pp. 98–107.

[27] C. S. G. Khoo, J. Kornfilt, R. N. Oddy, and S. H. Myaeng, "Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing," *Literary Linguistic Comput.*, vol. 13, no. 4, pp. 177–186, Dec. 1998, doi: 10.1093/llc/13.4.177.

[28] W. Maalej, Z. Kurtanović, H. Nabil, and C. Stanik, "On the automatic classification of app reviews," *Requirements Eng.*, vol. 21, no. 3, pp. 311–331, Sep. 2016, doi: 10.1007/s00766-016-0251-9.

[29] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," Stanford Univ., Project Rep. CS224N, Dec. 2009, vol. 1, no. 12.

[30] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *J. Artif. Intell. Res.*, vol. 50, pp. 723–762, Aug. 2014, doi: 10.1613/jair.4272.

[31] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 740–750.

[32] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, "AR-miner: Mining informative reviews for developers from mobile app marketplace," in *Proc. 36th Int. Conf. Softw. Eng.*, Hyderabad, India, May 2014, pp. 767–778.

[33] X. Fang and J. Zhan, "Sentiment analysis using product review data," *J. Big Data*, vol. 2, no. 1, Dec. 2015, doi: 10.1186/s40537-015-0015-2.

[34] A. Minnich, N. Abu-El-Rub, M. Gokhale, R. Minnich, and A. Mueen, "ClearView: Data cleaning for online review mining," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, San Fransisco, CA, USA, Aug. 2016, pp. 555–558.

[35] C. Stab and I. Gurevych, "Identifying argumentative discourse structures in persuasive essays," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 46–56.

[36] H. Nguyen and D. Litman, "Extracting argument and domain words for identifying argument components in texts," in *Proc. 2nd Workshop Argumentation Mining*, Denver, CO, USA, 2015, pp. 22–28.

[37] A. Aker, A. Sliwa, Y. Ma, R. Lui, N. Borad, S. Ziyaei, and M. Ghobadi, ''What works and what does not: Classifier and feature analysis for argument mining,'' in *Proc. 4th Workshop Argument Mining*, Copenhagen, Denmark, 2017, pp. 91–96.
[38] N. Jha and A. Mahmoud, ''Mining non-functional requirements from app store reviews,'' *Empirical Softw. Eng.*, vol. 24, no. 6, pp. 3659–3695, Dec. 2019.
[39] C. J. Hutto and E. Gilbert, ''VADER: A parsimonious rule-based model for sentiment analysis of social media text,'' in *Proc. 8th Int. Conf. Weblogs Social Media (ICWSM)*, Ann Arbor, MI, USA, Jun. 2014, p. 10.
[40] N. Asghar, ''Automatic extraction of causal relations from natural language texts: A comprehensive survey,'' 2016, *arXiv:1605.07895*. Accessed: Sep. 4, 2020. [Online]. Available: http://arxiv.org/abs/1605.07895
[41] M. Lu and P. Liang, ''Automatic classification of non-functional requirements from augmented app user reviews,'' in *Proc. 21st Int. Conf. Eval. Assessment Softw. Eng.*, Karlskrona, Sweden, Jun. 2017, pp. 344–353, doi: 10.1145/3084226.3084241.
[42] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, ''The detection and classification of non-functional requirements with application to early aspects,'' in *Proc. 14th IEEE Int. Requirements Eng. Conf. (RE)*, Minneapolis/St. Paul, MN, USA, Sep. 2006, pp. 39–48, doi: 10.1109/RE.2006.65.
[43] M. Tulio Ribeiro, S. Singh, and C. Guestrin, '''Why should i trust you?': Explaining the predictions of any classifier,'' 2016, *arXiv:1602.04938*. Accessed: Nov. 18, 2020. [Online]. Available: http://arxiv.org/abs/1602.04938
[44] T. Johann, C. Stanik, A. M. Alizadeh B., and W. Maalej, ''SAFE: A simple approach for feature extraction from app descriptions and app reviews,'' in *Proc. IEEE 25th Int. Requirements Eng. Conf. (RE)*, Lisbon, Portugal, Sep. 2017, pp. 21–30, doi: 10.1109/RE.2017.71.
[45] M. A. Hearst, ''Automatic acquisition of hyponyms from large text corpora,'' presented at the COLING, 1992.
[46] S. Roller, D. Kiela, and M. Nickel, ''Hearst patterns revisited: Automatic hypernym detection from large text corpora,'' 2018, *arXiv:1806.03191*. [Online]. Available: http://arxiv.org/abs/1806.03191
[47] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller, ''Introduction to WordNet: An on-line lexical database,'' *Int. J. Lexicogr.*, vol. 3, no. 4, pp. 235–244, 1990.
[48] Z. Wu and M. Palmer, ''Verb semantics and lexical selection,'' in *Proc. 32nd Annu. Meeting Assoc. Comput. Linguistics*, Santa Fe, NM, USA, 1994, pp. 133–138.
[49] G. Perrone, J. Unpingco, and H.-m. Lu, ''Network visualizations with pyvis and VisJS,'' 2020, *arXiv:2006.04951*. Accessed: Oct. 13, 2020. [Online]. Available: http://arxiv.org/abs/2006.04951
[50] G. Grano, A. Di Sorbo, F. Mercaldo, C. A. Visaggio, G. Canfora, and S. Panichella, ''Android apps and user feedback: A dataset for software evolution and quality improvement,'' in *Proc. 2nd ACM SIGSOFT Int. Workshop App Market Anal.*, Paderborn, Germany, Sep. 2017, pp. 8–11.
[51] R. He and J. McAuley, ''Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering,'' in *Proc. 25th Int. Conf. World Wide Web (WWW)*, Montreal, QC, Canada, 2016, pp. 507–517.
[52] S. L. Lim, P. J. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden, ''Investigating country differences in mobile app user behavior and challenges for software engineering,'' *IEEE Trans. Softw. Eng.*, vol. 41, no. 1, pp. 40–64, Jan. 2015, doi: 10.1109/TSE.2014.2360674.
[53] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, ''Why people hate your app: Making sense of user feedback in a mobile app store,'' in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, Chicago, IL, USA, 2013, pp. 1276–1284.
[54] E. Cabrio and S. Villata, ''Five years of argument mining: A data-driven analysis,'' in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 5427–5433, doi: 10.24963/ijcai.2018/766.
[55] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, ''Explaining explanations: An overview of interpretability of machine learning,'' 2018, *arXiv:1806.00069*. Accessed: Dec. 7, 2020. [Online]. Available: http://arxiv.org/abs/1806.00069
[56] J. R. Zilke, E. Loza Mencia, and F. Janssen, ''DeepRED—Rule extraction from deep neural networks,'' in *Discovery Science*, vol. 9956, T. Calders, M. Ceci, and D. Malerba, Eds. Cham, Switzerland: Springer, 2016, pp. 457–473.

ANANG KUNAEFI received the B.C. and M.C. degrees in informatics engineering from the Sepuluh Nopember Institute of Technology, Surabaya, Indonesia, in 2004 and 2013, respectively. He is currently pursuing the Ph.D. degree in computer science and electrical engineering with Kumamoto University, Kumamoto, Japan.

From 2003 to 2010, he was a Software Engineer and focused on the development of a web-based application system using the Java programming language, PHP, and JavaScript. Since 2014, he has been with the Department of Information System, Sunan Ampel State Islamic University Surabaya, Indonesia. He wrote two books about the analysis and implementation of information systems for mapping students' interest in Indonesia's educational institutions. His research interests include software engineering, data-driven requirements, business process automation, semantic web services, and the implementation of information systems for educational institutions.

MASAYOSHI ARITSUGI (Member, IEEE) received the B.E. and D.E. degrees in computer science and communication engineering from Kyushu University, Japan, in 1991 and 1996, respectively.

From 1996 to 2007, he was with the Department of Computer Science, Gunma University, Japan. Since 2007, he has been a Professor with Kumamoto University, Japan. His research interests include database systems and parallel/distributed data processing. He is currently a member of ACM and DBSJ and a Senior Member of IPSJ and IEICE. He was a recipient of the COMPSAC 2015 Best Paper Award, the Best Paper Award in Image Processing and Understanding in 13th IEEE International Conference on Signal Processing (ICSP2016), and the Best Paper Award in the 2019 IEEE International Cyber Science and Technology Congress (CyberSciTech).