Is the SAFE Approach Too Simple for App Feature Extraction? A Replication Study

Faiz Ali Shah(✉), Kairit Sirts, and Dietmar Pfahl

Institute of Computer Science, University of Tartu, Tartu, Estonia
{faiz.ali.shah,kairit.sirts,dietmar.pfahl}@ut.ee

Abstract. [**Context and motivation**] Automatic extraction and analysis of app features from user reviews is helpful for software developers to better understand users perceptions of delivered app features. Recently, a rule-based approach called SAFE was proposed to automatically extract app features from user reviews. SAFE was reported to obtain superior performance in terms of precision and recall over previously proposed techniques. However, the procedure used to evaluate SAFE was in part subjective and not repeatable and thus the whole evaluation might not be reliable. [**Question/problem**] The goal of our study is to perform an external replication of the SAFE evaluation using an objective and repeatable approach. [**Principal ideas/results**] To this end, we first implemented SAFE and checked the correctness of our implementation on the set of app descriptions that were used and published by the authors of the original study. We applied our SAFE implementation to eight review datasets (six app review datasets, one laptop review dataset, one restaurant review dataset) and evaluated its performance against manually annotated feature terms. Our results suggest that the precision of the SAFE approach is strongly influenced by the density of the annotated app features in a review dataset. Overall, we obtained an average precision and recall of 0.120 and 0.539, respectively which is lower than the performance reported in the original SAFE study. [**Contribution**] We performed an unbiased and reproducible evaluation of the SAFE approach for user reviews. We make our implementation and all datasets used for the evaluation available for replication by others.

Keywords: App feature extraction · SAFE approach · App review mining · Review summarization

1 Introduction

User feedback is an important source of information for software developers to enhance software quality [1]. In the context of mobile applications, i.e., *apps*, app marketplaces such as AppStore and PlayStore have become useful channels distributing millions of apps to their users. These marketplaces allow users to

E. Knauss and M. Goedicke (Eds.): REFSQ 2019, LNCS 11412, pp. 21–36, 2019.
https://doi.org/10.1007/978-3-030-15538-4_2

submit feedback in the form of reviews. User reviews contain valuable information such as feature evaluation, feature requests or bug reports that is helpful for developers for improving their apps [9]. However, the enormous volume of reviews received every day for a popular app make the manual analysis of these reviews impractical. Earlier studies have performed automatic analysis of user reviews to find out new app features requested by users [7] or to discover the sentiment of app features extracted from user reviews [2,3,13]. One major challenge in these studies has been the automatic extraction of app features from user reviews which is difficult for several reasons. First, there is great variability in how users express identical features and secondly, review texts often contain non-standard language such as slang, typos, and incorrect grammar.

Several approaches have been proposed to extract app features automatically from app user reviews. These approaches include topic models [3], set of patterns/rules [2,5], and supervised sequence tagging models [12]. The recently proposed rule-based approach SAFE [5] uses 18 Part-of-Speech (POS) patterns and five sentence patterns for automatically extracting app features from app descriptions and user reviews. Johann et al. reported the precision and recall of SAFE to be 0.559 and 0.434, respectively, for app feature extraction from app descriptions which is superior over the technique of Harman et al. [4]—an earlier rule-based approach developed for the same purpose. Moreover, SAFE was also reported to outperform the topic-modeling approach of Guzman et al. [3] for the extraction of app features from user reviews with a reported precision and recall of 0.239 and 0.709, respectively [5].

To evaluate the SAFE performance for app descriptions in their original study, Johann et al. created a labeled dataset, in which app features have been manually annotated. However, they did not create such a dataset to evaluate the performance of extracting app features from user reviews. Instead, the authors of the original SAFE study used a coding tool that showed a review text along with a list of SAFE-extracted app feature terms to coders who then had to decide whether the extracted app features were true or false. In case any true app features had not been extracted by SAFE (i.e., false negatives) from a user review, coders had to add them manually by writing them in a corresponding text box. This procedure to spot false negatives (FNs) is subjective and could introduce researcher bias because coders might have accidentally skipped entering some true app features not extracted by SAFE, thus lowering the number of false negatives and thus boosting performance. In summary, the evaluation of the SAFE approach for user reviews as conducted in the original study has the following two issues: **(a)** the evaluation is not repeatable because the true app features in the user reviews were not reported and **(b)** the evaluation procedure is potentially biased as it bases the identification of true and false positives on subjective decisions of coders after the list of SAFE-extracted app features has been shown to them. In order to validate the performance of the SAFE, we conducted an external replication [6] of the SAFE evaluation on user reviews, using an unbiased and repeatable procedure. Our goal is to answer the following research question:

RQ: What is the expected performance of SAFE on user reviews?

This research question had to be answered in two steps. Since exact implementation of the SAFE approach has not been published, we first implemented the SAFE method and validated our implementation using the annotated app description dataset made publicly available by the authors of the original SAFE study. This lead us to the first sub-question of RQ.

RQ-A: Does our implementation of the SAFE approach have the same performance as the original implementation of the SAFE approach when applied to app descriptions?

After confirming that our SAFE implementation on the app description dataset achieves a performance close to the one reported in the original SAFE study, we applied SAFE to the following eight annotated review datasets: GUZMAN dataset[1], GUZMAN+ dataset (an extension of the GUZMAN dataset), and four dataset variants derived from the SHAH dataset [14], and LAPTOP and RESTAURANT review datasets[2]. In the rest of this paper, we use the word "features" to collectively refer the features of a software app, laptop product, or restaurant services. Features contained in these review datasets have been manually annotated by humans. The application of our SAFE implementation to these datasets answered the second sub-question of RQ.

RQ-B: Does our implementation of the SAFE approach have the same performance as the original implementation of the SAFE approach when applied to review datasets?
The evaluation results show that the SAFE performance in terms of f1-score for all review datasets is lower than the performance reported in the original SAFE study. Our analyses further reveal that the precision of the SAFE approach is influenced by the density of true features in a review dataset.

The rest of the paper is structured as follows. In Sect. 2, we provide a brief introduction of SAFE approach. Section 3 describes our methodology that include details of our SAFE implementation and its validation, followed by the description of the evaluation method and characteristics of four annotated review datasets. Section 4 discusses the results. In Sect. 5, threats to validity are examined. Section 6 summarizes the previous work related to our study. Conclusions are presented in Sect. 7.

2 SAFE Approach

The SAFE approach is a rule-based method recently proposed by Johann et al. [5] for the extraction of app features from both app descriptions and user reviews. The authors of the SAFE approach performed a manual analysis of descriptions of 100 apps in Google Play Store and identified frequent textual patterns which are used to denote app features of these apps. The 18 Part-of-Speech (POS)

[1] The dataset was obtained from the authors of study [3].
[2] http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools.

patterns found in the app descriptions are shown in Table 1 together with their frequencies. In addition, the authors also identified five sentence patterns where the app features are mentioned as enumeration, conjunctions and feature identifiers [5]. The exact specification of the five sentence patterns was not presented in the original study. We will describe our interpretation of these patterns in Subsect. 3.1 where we describe our implementation of the SAFE approach.

SAFE first applies a number of pre-processing steps that remove sentences containing URLs, quotations, email addresses, and explanations (text between brackets). Then some parts of the remaining sentences are removed, including subordinate clauses, stop words, bullet points, and symbols such as "*" or"#". Then SAFE patterns are applied to sentences for the extraction of 2-to-4-word candidate app features. In the final step, the list of candidate app features is cleaned by removing duplicates and noise such as identical words pairs, e.g., "document document", which may be extracted using a POS pattern ⟨NOUN-NOUN⟩.

Table 1. List of SAFE POS patterns with frequency of occurrence [5]

#	POS pattern	Freq	#	POS pattern	Freq
1	⟨Noun-Noun⟩	183	10	⟨Adjective-Adjective-Noun⟩	20
2	⟨Verb-Noun⟩	122	11	⟨Noun-Preposition-Noun⟩	18
3	⟨Adjective-Noun⟩	119	12	⟨Verb-Determiner-Noun⟩	14
4	⟨Noun-Conjunction-Noun⟩	98	13	⟨Verb-Noun-Preposition-Noun⟩	14
5	⟨Adjective-Noun-Noun⟩	70	14	⟨Adjective-Noun-Noun-Noun⟩	12
6	⟨Noun-Noun-Noun⟩	35	15	⟨Adjective-Conjunction-Adjective⟩	12
7	⟨Verb-Pronoun-Noun⟩	29	16	⟨Verb-Preposition-Adjective-Noun⟩	11
8	⟨Verb-Pronoun-Noun⟩	29	17	⟨Verb-Pronoun-Adjective-Noun⟩	11
9	⟨Verb-Adjective-Noun⟩	26	18	⟨Noun-Conjunction-Noun-Noun⟩	10

3 Research Method

In this section, we present the main elements of the research method of our replication study. In Subsect. 3.1, we describe the details of our SAFE implementation. In Subsect. 3.2, we describe how we match SAFE-extracted features with true features. In Subsect. 3.3, we present the characteristics of the annotated review datasets that we used for our unbiased and repeatable evaluation of the SAFE approach. Finally, in Subsect. 3.4, we present the experimental setup of our study.

3.1 SAFE Implementation

Since the SAFE implementation used by the authors of the original study is not publicly available, we created our own implementation of the SAFE approach[3]

[3] https://github.com/faizalishah/SAFE_REPLICATION.

based on the information detailed in [5]. Like the original study, we used the Python programming language and the Natural Language ToolKit (NLTK)[4] for SAFE implementation. However, since not all details of the implementation of the SAFE approach have been published in the original study, we had to make some decisions on our own. The details of those decisions are discussed in the following paragraphs.

Table 2. List of SAFE sentence patterns [5]

#	Sentence pattern
1	⟨Noun-Conj-Noun: Noun⟩
2	⟨Verb|Noun: (Noun-Comma)$^+$-Conj-Noun⟩
3	⟨Verb|Noun-Conj-Noun|Verb: Noun-Conj-Noun⟩
4	⟨Verb-Noun-Noun-to-Adv-Verb-Conj-Verb-on-Noun-of-Noun-including : (Noun-Noun-Comma)$^+$-Noun-Conj-Noun⟩
5	⟨Verb-(Comma-Verb)$^+$-Conj-Verb-Noun: IN (Noun-Comma)$^+$-Conj-Noun-Noun⟩

After performing the pre-processing steps as described in the original study, the SAFE implementation applies linguistic patterns to extract the candidate app features. Following the original study, we first apply the sentence patterns and then the POS patterns. Since the original study does not state in which order the individual POS patterns shall be applied, we decided to apply them in the order in which they are presented in Table 1 (see Sect. 2). Also, the original SAFE study does not explicitly state the format of the sentence patterns. In Table 2, we present the list of sentence patterns used in our SAFE implementation for extracting app features.

The syntax of the patterns is following that of regular expressions. Once a sentence pattern finds a match in the analyzed text, it extracts the app features and represents them using one of the POS patterns. This might require deletion of words found in the matching pattern. For example, conjunctions and commas are always dropped. We indicate in Table 2 the words that are deleted with an underscore.

All patterns have the format ⟨LeftTerm1-Conj1-RightTerm1 : LeftTerm2-Conj2-RightTerm2⟩. The colon symbol ":" denotes where the right-hand side of the first conjunction ends and the left-hand side of the subsequent conjunction begins. Based on the sentence pattern, the following POS patterns are then generated by taking the cross-product of the left-hand and right-hand terms of each conjunction, i.e., the following set of POS patterns will be generated: ⟨LeftTerm1, LeftTerm2⟩, ⟨LeftTerm1, RightTerm2⟩, ⟨RightTerm1, LeftTerm2⟩, and ⟨RightTerm1, LeftTerm2⟩. In the first two sentence patterns Conj1 and Conj2 are empty, respectively. In those cases, the left-hand and right-hand

[4] https://www.nltk.org/.

terms of the missing conjunction fall together and the cross-product is simplified accordingly.

An additional complication is introduced by the fact that several of the 18 POS patterns are overlapping. For instance, the shorter POS pattern ⟨Verb-Noun⟩ may overlap with some of the longer POS patterns such as ⟨Verb-Noun-Noun⟩ or ⟨Verb-Noun-Preposition-Noun⟩. Thus, applying these patterns in a sequential order would extract overlapping candidate app features. Since we do not know how this is handled in the original SAFE study, in our implementation, when the overlapping features are extracted from a review sentence, only the longest feature term is preserved. Since we only preserve the longest feature terms, the results of feature extraction would not depend on the order in which POS patterns were applied. Moreover, the original version of the SAFE implementation uses a custom list of stop words which is not publicly available. Therefore, we use our own list of custom stop words for our implementation[5].

3.2 Strategy for Matching SAFE-Extracted and True Features

To compute the performance (precision and recall) of our SAFE implementation on an evaluation set, the number of true positives (TPs), false positives (FPs), and false negatives (FNs) must be counted by matching the SAFE-extracted features against the true features (i.e., those features that were labeled by humans). However, the original SAFE study does not give information about how exactly the extracted and true features were matched to count TPs, FPs, and FNs.

In our study we adopted the token-based subset matching strategy for evaluating our SAFE implementation. In token-based subset matching strategy, an extracted feature is counted as true positive (TP) either when the extracted feature words are a subset of the true feature words or the words of a true feature are a subset of the extracted feature words. In addition, the extracted feature must appear in the same review sentence in which the true feature was annotated. For instance, when the extracted app feature is "create document" and the true app feature annotated in a review text is "create new document" then the extracted app feature "create document" would be counted as a TP because the extracted app feature word-set {create, document} is a subset of the true app feature word-set {create, new, document}. In contrast to this, when the extracted app feature is "create document" from a review sentence but the app feature "create document" has not been annotated in the same review sentence then the extracted app feature "create document" would be counted as a false positive (FP). Finally, the true features, which were not matched with any extracted features will be counted as false negatives (FNs).

We consider this matching strategy justified, because the annotation of true app features in a review text is to a certain degree subjective and it would be too demanding to expect from an extraction method to identify the exact same words as app features as were annotated in the evaluation dataset. The

[5] https://github.com/faizalishah/SAFE_REPLICATION/blob/master/SAFE/List_StopWords.

difficulty to annotate identical app features in one and the same user review by two or more annotators has been observed, for example by Guzman et al. [3] who reported an agreement of 53% between two coders. We had a similar experience when annotating our SHAH user review dataset [14]. We should not assume that automatic app feature extraction works better than human annotators do.

3.3 User Review Datasets

In the original SAFE study, no evaluation set was created for the evaluation of SAFE on user reviews. This makes the evaluation of SAFE on user reviews not reproducible even if the original SAFE implementation would be available. Thus, to be able to perform a reproducible evaluation of the SAFE approach, we had to find user reviews in which app features have been annotated. We use four English review datasets that are publicly available and have been used in previous studies, i.e., GUZMAN, SHAH, LAPTOP, and RESTAURANT. The review datasets vary with regards to several characteristics, i.e., domain, annotation guidelines used, the number of annotators, the number of review sentences, and the number of annotated features. This diversity of datasets enables us to analyse the performance of the SAFE approach under different viewpoints and, hence, to obtain a more reliable evaluation for user reviews. We should point out that two of the review datasets, LAPTOP and RESTAURANT, do not contain reviews from app users. We included those review datasets because the SAFE patterns are purely syntactic and thus should not be sensitive to the choice of domain – be it software apps (GUZMAN and SHAH datasets), products (LAPTOP), or services (RESTAURANT).

In Table 3, we characterize each review dataset based on the following information:

(a) the total number of reviews;
(b) the total number of sentences in all reviews;
(c) the total number of 2-to-4-word annotated features;
(d) the density of 2-to-4-word annotated features over review sentences;
(e) the total number of annotated features;
(f) the density of all annotated features over review sentences.

GUZMAN REVIEW DATASETS. The original GUZMAN dataset (See footnote 1) was used as an evaluation set in the study conducted by Guzman et al. [3]. It contains annotated app reviews of seven apps belonging to six different categories: Angry Birds (Games category), DropBox and EverNote (Productivity category), TripAdvisor (Travel category), PicsArt (Photography category), Pinterest (Social category) and WhatsApp (Communication category).[6] In Table 3, we do not show the data of individual app categories but the aggregated summary of the GUZMAN dataset.

According to Guzman et al., the dataset initially consisted of 2800 user reviews (i.e., 400 user reviews per app). After annotation by human coders it

[6] Review titles with their annotated app features were removed for our study.

Table 3. Characteristics of the annotated review datasets

Dataset	#Reviews	#Sentences	#2-4-word features	2-4-word features density	#All features	All features density
GUZMAN	1479	4367	1421	.325	2350	.538
GUZMAN+	2800	8267	1421	.172	2350	.284
SHAH-I	3500	5970	352	.059	644	.108
SHAH-II	3500	5970	441	.074	756	.127
SHAH-I ∪ SHAH-II	3500	5970	575	.096	1017	.170
SHAH-I ∩ SHAH-II	3500	5970	242	.041	419	.070
LAPTOP	-	3845	1134	.295	3012	.783
RESTAURANT	-	3841	1157	.301	4827	1.25

turned out that there were 1321 user reviews left without annotation of a single app feature. Only those 1479 user reviews containing at least one annotated app feature were included in the published GUZMAN dataset and used for evaluation. The removed 1321 user reviews were not made publicly available.

In the context of Guzman et al.'s original study, it might have made sense to only use reviews containing annotated app features for evaluation purposes but in a real-world setting, taking a random sample of user reviews from App Store would normally be a mix of reviews mentioning app features (related to specific app features) and reviews that are praising or criticizing the app/versions/updates as a whole but not mentioning any specific app features. In order to also capture the real-world situation in our analysis, we artificially created a new version of the GUZMAN dataset which we named GUZMAN+. The GUZMAN+ dataset contains both types of reviews, i.e., with and without app features, and is thus comparable to other review datasets used in our analysis. Since we did not know which reviews were removed from the original GUZMAN dataset, we simply randomly sampled 1321 reviews without app features from the annotated SHAH dataset and added them to the annotated GUZMAN reviews. As expected, the ratio between number of app features and number of sentences in GUZMAN+ (see Table 3) goes down by almost 50% as compared to the original GUZMAN dataset.

SHAH REVIEW DATASETS. In the context of a previous study we created the SHAH dataset [14]. All reviews in the SHAH dataset were independently annotated by two coders.[7] The Dice coefficient score between the two annotation sets was low (i.e., 0.28), indicating a low agreement between the two coders. Because of that, we decided to use four different versions of the SHAH dataset in this study, i.e., **(1)** SHAH-I, **(2)** SHAH-II, **(3)** SHAH-I ∪ SHAH-II, and **(4)** SHAH-I ∩ SHAH-II. Among the four versions of the SHAH dataset, SHAH-I and SHAH-II contain the annotations of only the first and only the second coder, respectively. The

[7] Both coders were software engineering bachelors students at the University of Tartu.

SHAH-I ∪ SHAH-II dataset contains the annotations of both coder 1 and coder 2. In the case of overlapping annotations, only the longer annotation was retained. Finally, the SHAH-I ∩ SHAH-II dataset only contains the annotations annotated by both coders. As we did for the SHAH-I ∪ SHAH-II dataset, when annotations were overlapping we only retained the longer annotation. From all SHAH datasets we removed all app features that were referring to the app itself [14].

The summary statistics of all four versions of the SHAH dataset are shown in Table 3. Overall, in comparison to the GUZMAN, LAPTOP, and RESTAURANT datasets, the SHAH dataset contains a smaller number of app features. Among the four versions of the SHAH dataset, as expected, the SHAH-I ∪ SHAH-II dataset contains the highest number of app features. However, even in this dataset the ratio between the number of app features and the number of sentences (i.e., the features density) is clearly lower than in the other review datasets.

LAPTOP AND RESTAURANT REVIEW DATASETS. The LAPTOP and RESTAURANT review datasets (See footnote 2) are standard benchmark datasets contributed by the SEMEVAL research community.[8] Both datasets have been used in studies that aimed at performing the task of feature extraction (called "aspect terms") from user reviews and its evaluation [8,11]. Both datasets are distributed in predefined training and test splits, which is relevant in the context of machine learning based methods. For our purpose, we merged the training and test sets into single LAPTOP and RESTAURANT datasets, respectively.

The characteristics of the LAPTOP and RESTAURANT datasets in Table 3 show that the ratio between the number of all annotated features and the number of sentences is clearly higher than for the app review datasets. The ratio between the number of 2-to-4-word features and the number of sentences, however, follows the same pattern as most app review datasets with an exception of the GUZMAN dataset which has a comparable ratio.

3.4 Experimental Setup

This section explains the settings used for the SAFE approach evaluation. To answer sub-question RQ-A of our research question RQ, we analyse the performance of our SAFE implementation (as described in Sect. 3.1) when applied to the ten annotated app descriptions made available in the original SAFE study. If the performance of our SAFE implementation in terms of precision, recall, and f1-score is comparable to that reported in the original study, we consider our SAFE implementation to be suitable for tackling sub-question RQ-B of our research question RQ. To answer RQ-B, we apply our SAFE implementation to eight annotated review datasets (see Table 3). The performance measures (precision, recall and F1-score) of SAFE are computed on each review dataset for the annotated 2-to-4-word features and for all annotated features using the token-based subset matching strategy (see Sect. 3.2).

[8] http://alt.qcri.org/semeval2018/.

4 Results and Discussion

In this section, we present the results to our research question RQ in two steps. First we present and discuss the results related to sub-question RQ-A, then we present and discuss the results to sub-question RQ-B.

4.1 Validation of SAFE Implementation (RQ-A)

The correctness of our SAFE implementation can be validated by applying it on the same evaluation set used in the original SAFE study. We contacted the main author of the original study and learned that in the original study, only the dataset containing the app descriptions had annotated app features but not the dataset containing the app reviews. Since the authors of the original study shared their annotated dataset of app descriptions, we were at least able to apply our SAFE implementation to the same app description dataset and thus validate our implementation.

Table 4 shows the evaluation results on the annotated app description dataset of our SAFE implementation (on the right) as well as the evaluation results reported by Johann et al. (on the left). Our SAFE implementation achieves exactly the same precision and recall as the original SAFE implementation only for one app description (Google Docs). On two app descriptions (Forest and Dropbox), we achieve higher precision and recall than the original SAFE implementation. For Google Drive app description, we achieve identical recall but higher precision compared to the original SAFE implementation. On the rest of the six app descriptions, we obtain lower precision and recall than the original implementation of SAFE. These differences in performance between the two implementations might be related to the unspecified details brought out in Sect. 3.1. Additionally, there could be differences in matching the extracted app features with true app features that can lead to different results (see Sect. 3.2).

Based on the results of individual app descriptions we cannot claim that our SAFE implementation is the same as the original SAFE method. However, on average over all app descriptions, our SAFE implementation achieves only slightly lower precision and recall than the original SAFE implementation. Since based on the average f1-score the difference between the two implementations is only 0.011, we believe that we can still perform useful analyses with our implementation.

4.2 Evaluation of SAFE Approach (RQ-B)

In this section, we answer the sub-question RQ-B of our research question RQ by comparing the performance reported in the original SAFE study with the performance achieved with our implementation of the SAFE approach on the eight annotated datasets described in Sect. 3.3.

The performance of our implementation of the SAFE approach is presented in Table 5. We evaluated the SAFE approach separately against 2-to-4-word features and against all features. The left-hand side of the table shows the SAFE

Table 4. Comparison of results obtained with the original SAFE implementation and our SAFE implementation on app description dataset.

App name	Original SAFE implementation			Our SAFE implementation		
	Precision	Recall	F1 score	Precision	Recall	F1 score
Forest: Stay focused, be present	.462	.400	.429	.636	.467	.538
Yahoo Mail	.737	.389	.509	.680	.436	.531
Printer Pro	.214	.250	.231	.190	.333	.242
Gmail	.714	.400	.513	.611	.524	.564
Google Drive	.875	.389	.538	1.0	.389	.560
CloudApp Mobile	.722	.481	.578	.478	.423	.449
Google Docs	.667	.462	.545	.667	.462	.545
Dropbox	.300	.300	.300	.400	.333	.364
Fantastical 2 for iPhone	.500	.697	.582	.302	.500	.377
iTranslate Voice	.500	.278	.357	.316	.286	.300
Average	**.559**	**.434**	**.458**	**.528**	**.415**	**.447**

performance evaluated for 2-to-4 word features. The right-hand side of the table presents the SAFE performance evaluated for all features.

The original SAFE study used only 2-to-4-word app features for evaluation since the POS and sentence patterns defined in the SAFE approach can only extract app features composed of two to four words. The original study reported precision and recall of 0.239 and 0.709, respectively, for the SAFE approach [5]. As shown in Table 5, the performance of our SAFE implementation on each of our evaluation datasets when evaluating on 2-to-4-word features varies but is consistently lower than the performance reported in the original study (average precision is 0.120, average recall is 0.539, and average f1-score is 0.184).

When comparing the precision of our SAFE implementation with that reported in the original study, one observes that the evaluation on three of our datasets, i.e., GUZMAN, LAPTOP, and RESTAURANT, is relatively close to the reported precision of 0.239 in the original study. The reason for this phenomenon could be that the density score of the annotated 2-to-4-word features is clearly higher for these three review datasets as compared to the other five review datasets. The sensitivity of SAFE precision to features density is also clearly visible when we look at the evaluation results using all annotated features (right-hand side of Table 5). Also, the fact that the evaluation results when using all features has consistently higher precision values supports the hypothesis that higher features density yields higher precision when using the SAFE approach.

When looking at the recall values, the interpretation is less straightforward than for precision. The highest recall of 0.624 when evaluating on 2-to-4-word

Table 5. Evaluation of SAFE extracted features on annotated review datasets

Dataset	2-4 word features			All features		
	Precision	Recall	F1-score	Precision	Recall	F1-score
GUZMAN	.201	.462	.280	.317	.426	.363
GUZMAN+	.096	.462	.159	.151	.426	.223
SHAH-I	.056	.624	.103	.080	.463	.136
SHAH-II	.064	.544	.115	.090	.443	.149
SHAH-I ∪ SHAH-II	.084	.550	.146	.118	.433	.185
SHAH-I ∩ SHAH-II	.040	.612	.074	.055	.522	.099
LAPTOP	.208	.490	.292	.359	.319	.337
RESTAURANT	.211	.569	.308	.492	.318	.386
Average	**.120**	**.539**	**.184**	**.207**	**.419**	**.235**

app features is obtained for the SHAH-II dataset but it is still considerably lower than the recall of 0.709 reported in the original study. Also, when comparing the recall values across the app review datasets it seems that whenever precision is low (correlating with low app features density) recall is respectively higher. However, this observation can neither be made for the LAPTOP and RESTAURANT datasets nor for the GUZMAN+ dataset. While the obvious explanation for the capped GUZMAN+ recall of 0.462 is that due to the construction of GUZMAN+ it has exactly the same set of annotated app features as GUZMAN, it is less clear why the recall values for the LAPTOP and RESTAURANT datasets are still relatively high. We speculate that other factors than features density have an impact on recall, e.g., the nature of the annotation guidelines used and the subjective interpretation of the annotation guidelines by the coders.

When comparing the precision of 2-to-4-word features with the precision of all features, Table 5 shows that the precision values consistently improve while the recall values go down. This happens because in each dataset the set of 2-to-4-word features is a strict subset of all features. As a consequence, some of the extracted features counted as false positives (FPs) when evaluated against 2-to-4-word features might be counted as true positives (TPs) due to the subset matching strategy that we use to match the extracted features with the true features. The impact is stronger on review datasets where the number of annotated features is higher, such as RESTAURANT, LAPTOP, and GUZMAN.

Based on our analysis of the performance of the SAFE approach we can make several observations about its usefulness to developers who might wish to analyze reviews in order to better understand user needs. The first observation is that due to the purely syntactic-based extraction patterns defined in the SAFE approach, its applicability is not restricted to a specific domain. We have demonstrated this by including review datasets from other domains such as those represented by the LAPTOP and RESTAURANT datasets. Interestingly, the performance of the SAFE approach in terms of f1-score is better on the LAPTOP and RESTAURANT datasets when compared to five realistic app review datasets (i.e., GUZMAN+

and all SHAH datasets). As mentioned before, this seems to be due to the higher density of features in the LAPTOP and RESTAURANT datasets.

Johann et al. [5] comment their evaluation by writing:

> As for the accuracy and benchmark values, we refrain from claiming that these are exact values and we think that they are rather indicative. We think that the order of magnitude of precisions and recalls calculated, as well as the differences between the approaches is significant.

Although we were not able to demonstrate that our implementation exactly matches the one used in the original study, our evaluation results give a reason to suspect that the true estimates of precision and recall of the SAFE approach are in fact lower than suggested by Johann et al. This fact again raises the question of how useful can SAFE approach be for the developers just as it is. The problem with low precision even when the recall is relatively high is that the extracted features contain a lot of noise and if the system does not provide any ranking of "usefulness" over the extracted features, it will be very difficult to spot the useful info from the noise. As Johann et al. [5] themselves say when discussing their results:

> Nevertheless, we think that the achieved accuracy of SAFE—even if it outperforms other research approaches—is not good enough to be applied in practice. We think that a hybrid approach (a simple, pattern and similarity based as SAFE together with a machine learning approach) is probably the most appropriate. For instance, machine learning can be used to pre-filter and classify reviews before applying SAFE on them.

In addition to the idea of first classifying reviews or sentences before applying SAFE we would also propose another way that could potentially improve the usefulness of the SAFE method via machine learning. Assuming that the SAFE approach obtains reasonably high recall when extracting app features from app reviews, one could imagine training a classifier to learn to discriminate between correctly (TPs) and incorrectly (FPs) extracted app features. In such a way it might be possible to retain the high recall while improving the precision.

5 Threats to Validity

The main threat to the validity of our study is that we were not able to exactly replicate the evaluation results of our SAFE implementation on the app description dataset provided by the authors of [5]. This means that although we have carefully checked our implementation but our implementation of the SAFE approach is not exactly the same as used in the original study.

One likely reason for the differences in the performance measures is that we might have decided certain implementation details, which were not specified in the SAFE paper (described in Sect. 3.1), differently than the original authors. For instance, we might have interpreted the sentence patterns differently than intended by the original authors and thus implemented them differently. Similarly, the proposers of the SAFE approach use a custom list of stop words in their

SAFE implementation. This list has not been published. Thus, we had to define our own list of custom stop words and the impact of our choice on the achieved performance values is not known. We intend to make our implementation as well as the custom list of stop words publicly available so that others could replicate and validate our results.

The differences in performance measures might also stem from a different way of counting TPs, FPs and FNs. The authors of the original SAFE study do not explain the matching strategy (exact match or partial match) used to match the SAFE extracted app features against the true app features. In our study, we adopted token-based subset matching strategy for the evaluation of SAFE on user reviews. It is possible that in the original study, the matching was performed differently.

The validity of our results depends partly on the reliability of the annotations of the review datasets. Since we not only used our own annotations (i.e., datasets SHAH-I and SHAH-II) but applied SAFE implementation to other review datasets published in the literature; so we believe that the existing limitations of reliability for the mentioned tasks is not a major threat to validity of our results.

6 Related Work

Recently, Johann et al. proposed a rule-based approach called SAFE that uses POS and sentence patterns for extracting app features from app descriptions and user reviews [5]. The SAFE approach has achieved better performance over the technique of Guzman et al. [3]. However, some aspects of the implementation of the SAFE approach as well as some aspects of its evaluation on user reviews are nor precisely described in the original study. Therefore, we decided to conduct an external replication with a fully published replication package allowing others to reproduce our results.

Several other approaches to extract app features from app reviews have been proposed. We list some of them in the following.

The study of Guzman et al. [3] used an unsupervised LDA topic modeling approach for automatic extraction of app features from user reviews of seven apps (three from App Store and four from Play Store). The performance of the approach is evaluated by matching the extracted app features against the human labeled app features in their labeled dataset. In our study, we used the same labeled dataset (i.e., GUZMAN dataset) for evaluation purpose.

The study of Gu et al. [2] classifies review sentences into categories, such as feature evaluation, praise, feature requests, bug reports and others, and then app features are extracted using 26 manually designed rules only from those sentences that belong to the feature evaluation category. In comparison to the approach of Gu et al., the SAFE approach for app feature extraction is not limited to feature evaluation sentences and it can extract app features from sentences mentioning feature requests, opinions related to features, and bug reports related to features alike.

Keertipati et al. extracted nouns as candidate app features from app review sentences but they did not perform an evaluation to check whether the extracted

app features actually represent true app features [7]. On the other hand, Vu et al.'s study [15] instead of directly extracting app features, extracted all potential keywords from user reviews and rank them based on the review rating and occurrence frequency.

In one of our own previous studies, we developed the prototype of a web-based tool to identify competing apps and to compare them based on the users' sentiments mentioned on the common set of app features [13]. This tool extracts two-word collocations as candidate app features without evaluating the extracted app features against true app features. Similar to the original study on the SAFE approach, the evaluation of the performance of the tool prototype with regards to app feature extraction performance was partly biased and subjective and thus not reproducible.

A recent study of Malik et al. [10] used syntactic relations between the features and opinion words for identification of "hot" app features from user review but the dataset used for the evaluation is not publicly available.

7 Conclusion

The SAFE approach is a recently proposed simple rule-based method for automatic extraction of app features from app descriptions and app reviews. For the evaluation of SAFE on app descriptions, the authors of the original SAFE study created and publicly shared an evaluation dataset. However, for evaluation on user reviews no evaluation dataset exists and the evaluation was instead performed using a coding tool. The procedure adopted for the evaluation of the SAFE approach on user reviews is subjective and might have suffered from researcher bias. Due to its subjective nature it is also not reproducible. Therefore, in this study, we performed an unbiased and reproducible evaluation of the SAFE approach with the goal to investigate the true performance of the SAFE approach when applied to user reviews.

We implemented the SAFE approach and once we had confirmed that our implementation achieves comparable average performance when applied to app descriptions as reported in the original study, we applied SAFE to eight different review datasets. The results show that the performance of the SAFE approach when evaluated against 2-to-4-word app features is clearly lower than the performance reported in the original SAFE study. Inspecting the characteristics of the used review datasets it became clear that the precision of the SAFE approach is strongly sensitive to the density of app features in the review datasets.

We conclude that due to very low precision and only moderate recall, SAFE is too simple to be useful in practice for extracting app features from app reviews. In order to make it usable in practice, methods, potentially involving machine learning, for improving the precision while retaining the recall should be studied.

Acknowledgment. We are grateful to Emitza Guzman and Christoph Stanik for sharing the datasets. This research was supported by the institutional research grant IUT20-55 of the Estonian Research Council and the Estonian Center of Excellence in ICT research (EXCITE).

References

1. Groen, E.C., et al.: The crowd in requirements engineering: the landscape and challenges. IEEE Softw. **34**(2), 44–52 (2017). https://doi.org/10.1109/MS.2017.33
2. Gu, X., Kim, S.: What parts of your apps are loved by users? In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 760–770, November 2015. https://doi.org/10.1109/ASE.2015.57
3. Guzman, E., Maalej, W.: How do users like this feature? A fine grained sentiment analysis of app reviews. In: 2014 IEEE 22nd International Requirements Engineering Conference (RE), pp. 153–162. IEEE (2014)
4. Harman, M., Jia, Y., Zhang, Y.: App store mining and analysis: MSR for app stores. In: Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, MSR 2012, pp. 108–111. IEEE Press, Piscataway (2012). http://dl.acm.org/citation.cfm?id=2664446.2664461
5. Johann, T., Stanik, C., Maalej, W.: SAFE: a simple approach for feature extraction from app descriptions and app reviews. In: 2017 IEEE 25th International Requirements Engineering Conference (RE), pp. 21–30. IEEE, September 2017. https://doi.org/10.1109/RE.2017.71
6. Juristo, N., Gómez, O.S.: Replication of software engineering experiments. In: Meyer, B., Nordio, M. (eds.) LASER 2008-2010. LNCS, vol. 7007, pp. 60–88. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25231-0_2
7. Keertipati, S., Savarimuthu, B.T.R., Licorish, S.A.: Approaches for prioritizing feature improvements extracted from app reviews. In: Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, p. 33. ACM (2016)
8. Liu, P., Joty, S., Meng, H.: Fine-grained opinion mining with recurrent neural networks and word embeddings. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1433–1443 (2015)
9. Maalej, W., Nabil, H.: Bug report, feature request, or simply praise? On automatically classifying app reviews. In: Proceedings of RE 2015, pp. 116–125. IEEE, August 2015
10. Malik, H., Shakshuki, E.M., Yoo, W.S.: Comparing mobile apps by identifying 'Hot' features. Futur. Gener. Comput. Syst. (2018)
11. Poria, S., Cambria, E., Gelbukh, A.: Aspect extraction for opinion mining with a deep convolutional neural network. Knowl.-Based Syst. **108**, 42–49 (2016)
12. Sänger, M., et al.: Scare–the sentiment corpus of app reviews with fine-grained annotations in German. In: LREC (2016)
13. Shah, F.A., Sabanin, Y., Pfahl, D.: Feature-based evaluation of competing apps. In: Proceedings of the International Workshop on App Market Analytics - WAMA 2016. pp. 15–21. ACM Press, New York (2016). https://doi.org/10.1145/2993259.2993267
14. Shah, F.A., Sirts, K., Pfahl, D.: The impact of annotation guidelines and annotated data on extracting app features from app reviews. arXiv preprint arXiv:1810.05187 (2018)
15. Vu, P.M., Nguyen, T.T., Pham, H.V., Nguyen, T.T.: Mining user opinions in mobile app reviews: a keyword-based approach. In: Proceedings of ASE 2015, pp. 749–759. IEEE (2015)