Mining User Opinions To Support Requirement Engineering: An Empirical Study

Jacek Dąbrowski$^{1,2[0000-0003-3392-0690]}$, Emmanuel Letier$^{1[0000-0002-8935-343X]}$, Anna Perini$^{2[0000-0001-8818-6476]}$, and Angelo Susi$^{2[0000-0002-5026-7462]}$

[1] University College London, London, UK
`{j.dabrowski, e.letier}@cs.ucl.ac.uk`
[2] Fondazione Bruno Kessler, Trento, Italy
`{dabrowski, perini, susi}@fbk.eu`

Abstract. App reviews provide a rich source of user opinions that can support requirement engineering activities. Analysing them manually to find these opinions, however, is challenging due to their large quantity and noisy nature. To overcome the problem, automated approaches have been proposed for so-called opinion mining. These approaches facilitate the analysis by extracting features discussed in app reviews and identifying their associated sentiments. The effectiveness of these approaches has been evaluated using different methods and datasets. Unfortunately, replicating these studies to confirm their results and to provide benchmarks of different approaches is a challenging problem. We address the problem by extending previous evaluations and performing a comparison of these approaches. In this paper, we present an empirical study in which, we evaluated feature extraction and sentiment analysis approaches on the same dataset. The results show these approaches achieve lower effectiveness than reported originally, and raise an important question about their practical use.

Keywords: Mining User Reviews · Requirement Engineering · Feature Extraction · Sentiment Analysis · Empirical Study

1 Introduction

App reviews is a rich source of user opinions [1, 2, 14, 17]. These opinions can help developers to understand how users perceive their app, what are users' requirements, or what are users' preferences [1, 2, 14]. Not surprisingly, knowing user opinions is an important information need developers seek to satisfy [2, 4]. The information can affect different software engineering practices [1, 14].

Analysing app reviews to find user opinions, however, is challenging [14, 17]; Developers may receive thousands of reviews per day [1, 14, 17]. Moreover, these reviews contain mostly noise [17, 19]. Consequently, the possibility of using these opinions to support engineering activities is obstructed [1, 14].

To address the problem, studies in requirement engineering proposed a few opinion mining approaches [10, 12, 13, 16]. These approaches facilitate mining user opinions by performing two tasks: extracting features discussed in reviews and identifying their associated users' sentiments [12, 16]. In particular, two approaches have become adopted in the community [17], GuMa [12][3] and SAFE [11].

Unfortunately, replicating the studies to confirm their results and to compare their approaches is a challenging problem. In fact, different methods and datasets have been used. The unavailability of their annotated datasets and evaluation procedures challenge their replicability even more [11–13, 21, 22].

The aim of the study is to address the problem by extending previous evaluations and performing comparison of these app review analysis approaches. We consider the following research questions to answer:
RQ1: What is the effectiveness of feature extraction approaches?
RQ2: What is the effectiveness of feature-specific sentiment analysis approaches?

To answer them, we conducted an empirical study in which we evaluated three approaches: GuMa [12], SAFE [11] and ReUS [10]. We evaluated them in performing feature extraction and sentiment analysis tasks using our annotated dataset.

The primary contributions of the study are: (i) an empirical evaluation expanding previous evaluations of the opinion mining approaches, (ii) a comparison of the approaches performing feature extraction and feature-specific sentiment analysis, and (iii) a new dataset of 1,000 reviews annotated with 1,521 opinions [7].

The remainder of the paper is structured as follows: In Sect. 2, we introduce terminology and the problem, then we give an overview of the opinion mining approaches we evaluate. In Sect. 3, we present scenarios motivating opinion mining. In Sect. 4, we present our study design. The results are detailed in Sect. 5, and the findings are discussed in Sect. 6. In Sect. 7, we provide threats to validity, then we discuss related works in Sect. 8. Conclusion is given in Sect. 9.

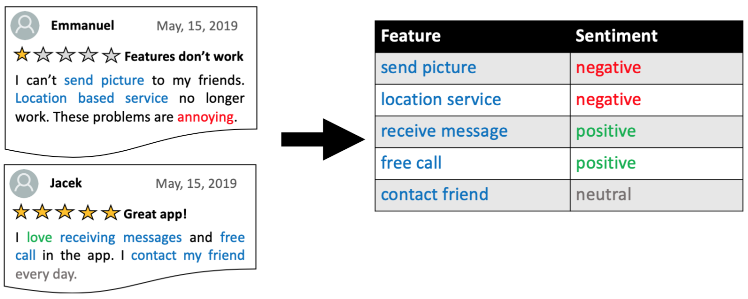
2 Background

This section introduces terminology and the formulation of opinion mining problem. It also provides an overview of approaches we evaluated.

2.1 Terminology and Problem Formulation

Definition 1 (Feature and Feature Expression) A feature is a user-visible functional attribute of an app that is intentionally provided. Attributes are typically functionalities (e.g., "send message"), modules providing functional capabilities (e.g., "user account") or design components (e.g., "UI") that can be utilized to perform tasks. A feature expression is a non-empty set of words $f = \{w_1, ..., w_m\}$ describing a feature in an app review. Further on in the text, we refer to a feature expression as a feature for the sake of simplicity.

[3] We refer to the approach using abbreviations derived from their authors' surnames.


Emmanuel May, 15, 2019

Features don't work

I can't send picture to my friends. Location based service no longer work. These problems are annoying.

Jacek May, 15, 2019

Great app!

I love receiving messages and free call in the app. I contact my friend every day.

Feature	Sentiment
send picture	negative
location service	negative
receive message	positive
free call	positive
contact friend	neutral

Fig. 1: Opinion Mining

Definition 2 (Sentiment) A sentiment s is a user attitude which can be either *positive*, *negative* or *neutral*.

Definition 3 (Opinion) An opinion is a tuple $o = (f, s)$, where f is a feature in a review r, s is a sentiment referencing to f in r.

Problem 1 (Opinion Mining) Given a set of reviews $R = \{r\}$ on an app a, the opinion mining problem is to find a multi-set of all the opinions $O = \{o\}$ in a set of reviews R.

Figure 1 illustrates opinion mining problem. The problem can be decomposed into two sub-problems, feature extraction and feature-specific sentiment analysis:

Problem 1.1 (Feature Extraction) Let $R = \{r\}$ be a set of reviews on an app a. Find a multi-set of all the features $F = \{f\}$ in a set of reviews R.

Problem 1.2 (Feature-specific Sentiment Analysis) Consider a set of pairs $\{(f, r)\}$ where f is a feature in a review r. Find a multi-set $S = \{s\}$ where s is a sentiment referring to f in r.

2.2 Approaches For Mining User Opinions

In our study, we selected three approaches: GuMa [12], SAFE [13] and ReUS [10]. We selected GuMa and SAFE as they are state-of-the-art approaches widely known in RE community [9, 17, 22]. We opted for ReUS [10] as the approach achieves a competitive performance in the context of opinion mining and sentiment analysis research [10, 16]. We also have its original implementation.
GuMa performs feature extraction and feature-specific sentiment analysis. These tasks are performed independently of each other. To extract features, the approach relies on a collocation finding algorithm. For predicting sentiment, the approach uses the SentiStrength tool [23]. First, the approach predicts the sentiment of a sentence, then assigns sentiments to features in the sentence. Unfortunately, GuMa's source code and evaluation data set are not available. We have therefore re-implemented GuMa's approach using SentiStrength for sentiment

analysis. We have tested that our implementation is consistent with GuMa's original implementation on examples in the original paper.
SAFE supports feature extraction, but not sentiment analysis. The approach extracts features based on linguistics patterns, including 18 part-of-speech patterns and 5 sentence patterns. These patterns have been identified through manual analysis of app descriptions. The approach conducts two main steps to extract features from a review: text preprocessing and the application of the patterns. Text preprocessing includes tokenizing a review into sentences, filtering-out noisy sentences, and removing unnecessary words. The final step concerns the application of linguistic patterns to each sentence to extract app features. We used the original implementation of the approach in our study.
ReUS exploits linguistics rules comprised of part-of-speech patterns and semantic dependency relations. These rules are used to parse a sentence and perform feature extraction and feature-specific sentiment analysis. Both tasks are performed at the same time. Given a sentence, the approach extracts a feature and an opinion word conveying a feature-specific sentiment. To determine the sentiment, the approach exploits lexical dictionaries. We used the original implementation of the approach, and set up it to identify one out of three sentiment polarities.

3 Motivating Scenarios

We describe three use cases in which the use of opinion mining can provide benefits. They are inspired by real-world scenarios, which were analysed in previous research [1,2,14].

Use Case 1 (Validation by Users) In any business endeavour, understanding customer opinions is an important aspect; app development is no exception [2,14]. Knowing what features users love or dislike can give project managers an idea about user acceptance of these features [1,2]. It can also help them draw a conclusion whether invested efforts were worth it [14]. As an example, imagine the development team changed core features in WhatsApp (e.g. video call). The team may want to know what users say about these features so that they can fix any glitches as soon as possible and refine these features. Mining user opinions could help them discover What are the most problematic features? or How many users do report negative opinions about a concrete feature (e.g. video call)?

Use Case 2 (Supporting Requirements Elicitation) Imagine now that WhatsApp receives negative comments about one of their features (e.g. group chat). It can be intimidating for developers to tackle a problem if they have to read through a thousand reviews. Using an opinion mining approach, developers could discover the issue within minutes. App mining tools could group reviews based on discussed features and their associated user's sentiment. Developers could then examine reviews that talk negatively about a specific feature (e.g. group chat). This could help developers understand user concerns about a problematic feature, and potentially help eliciting new requirements.

Use Case 3 (Supporting Requirements Prioritization) When added with statistics, user opinions can help developers prioritize their work [1,2,14]. Suppose the team is aware about problems with certain features which are commented negatively. Finding negative opinions mentioning these features could help them to compare how often these opinions appears, for how long these opinions have been made, and whether their frequency is increasing or decreasing. This information could provide an evidence of their relative importance from a users' perspective. Such information is not sufficient to prioritize issues, but it can provide useful evidence-based data to contribute to prioritization decisions.

For these scenarios having a tool that (i) mines user opinions and (ii) provides their summary with simple statistics could help the team to evolve their app.

4 Empirical Study Design

This section describes the empirical study design we used to evaluate the selected approaches. We provide the research questions we aimed to answer, the manually annotated dataset and evaluation metrics used to this end.

4.1 Research Questions

The objective of the study was to evaluate and compare approaches mining opinions from app reviews. To this end, we formulated two research questions:
RQ1: What is the effectiveness of feature extraction approaches?
RQ2: What is the effectiveness of feature-specific sentiment analysis approaches?

In RQ1, we evaluated the capability of the approaches in correctly extracting features from app reviews. In RQ2, we investigated the degree to which the approaches can correctly predict sentiments associated with specific features. A conclusive method of measuring the correctness of extracted features/predicted sentiments is by relying on human judgment. We used our dataset in which opinions (feature-sentiment pairs) have been annotated by human-coders (see Section 4.2). We compared extracted features/predicted sentiments to those annotated in ground truth using automatic matching methods (see Definition 4.3). In answering the questions, we report precision and recall.

4.2 Manually Annotated Dataset

This section describes the manually annotated dataset we created to answer RQ1 and RQ2 [7]. To create this datatset, we collected reviews from previously published datasets [18,24] and asked human-coders to annotate a selected samples of these reviews.

A) Data Collection

We have selected reviews from datasets used in previous review mining studies [18,24]. We selected these datasets because they include millions of English reviews from two popular app stores (i.e., Google Play and Amazon) for different apps, categories and period of times. We selected 8 apps from these datasets, 4

Table 1: The overview of the subject apps

App Name	**Category**	**Platform**	**#Reviews**
Evernote	Productivity	Amazon	4,832
Facebook	Social	Amazon	8,293
eBay	Shopping	Amazon	1,962
Netflix	Movies & TV	Amazon	14,310
Spotify Music	Audio & Music	Google Play	14,487
Photo Editor Pro	Photography	Google Play	7,690
Twitter	News & Magazines	Google Play	63,628
Whatsapp	Communication	Google Play	248,641

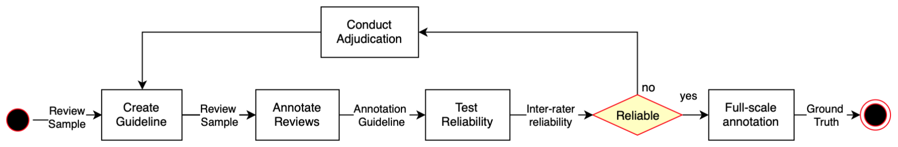


Fig. 2: The Method for Ground Truth Creation

apps from Google Play and 4 from Amazon app stores. For each subject app, we also collected their description from the app store. Table 1 illustrates the summary of apps and their reviews we used in our study. We selected subject apps from different categories to make our results more generalizable. We believe that the selection of popular apps could help annotators to understand their features, and to reduce their effort during the annotation.

B) Annotation Procedure

The objective of the procedure was to produce an annotated dataset that we use as ground truth to evaluate the quality of solutions produced by feature extraction and sentiment analysis approaches [20]. Figure 2 illustrates the overview of the procedure. Given a sample of reviews, the task of human-coders was to label each review with features and their associated sentiments.

We started by elaborating a guideline describing the annotation procedure, the definition of concepts and examples. We then asked two human-coders[4] to label a random sample of reviews using the guideline [7]. We evaluated the reliability of their annotation using the inter-rater agreement metrics F_1 and Fleiss' Kappa [5, 6]. F_1 is suitable for evaluating text spans' annotations such as feature expressions found in reviews; Fleiss' kappa is suitable to assess inter-rater reliability between two or more coders for categorical items' annotations such as users' sentiment (positive, negative, or neutral). We evaluated inter-rater agreement to ensure the annotation task was understandable, unambiguous, and could be replicated [20]. When disagreement was found, the annotators discussed to adjudicate their differences and refined the annotation guidelines. The process

[4] The first author and an external coder who has no relationship with this research. Both coders have an engineering background and programming experience.

Table 2: Statistics of the ground truth for 1,000 reviews for 8 subject apps.

		App Name								
		Evernote	Facebook	eBay	Netflix	Spotify	Photo Editor	Twitter	WhatsApp	**Overall**
Reviews	No. reviews	125	125	125	125	125	125	125	125	1,000
	Avg. review length	48.30	37.90	32.54	43.46	23.62	12.38	15.79	14.47	28.59
	No. sentences	367	327	294	341	227	154	183	169	2,062
	Avg. sentence length	16.45	14.49	13.84	15.93	13.00	10.05	10.79	10.70	13.85
	Sentence per review	2.94	2.62	2.35	2.73	1.82	1.23	1.46	1.35	2.06
Sentiment	No. sentiments	295	242	206	262	180	96	122	118	1,521
	No. positive	97	49	95	79	32	39	5	20	416
	No. neutral	189	168	102	159	122	47	93	84	964
	No. negative	9	25	9	24	26	10	24	14	141
Features	No. features	295	242	206	262	180	96	122	118	1,521
	No. distinct features	259	204	167	201	145	80	99	100	1,172
	No. single-word features	82	80	78	94	69	39	39	49	530
	No. multi-word features	213	162	128	168	111	57	83	69	991
	Feature per review	2.36	1.94	1.65	2.10	1.44	0.77	0.98	0.94	1,52
Agrmt.	F_1 measure	0.76	0.73	0.77	0.75	0.67	0.78	0.79	0.83	0.76
	Fleiss' Kappa	0.64	0.77	0.77	0.55	0.75	0.86	0.69	0.80	0.73

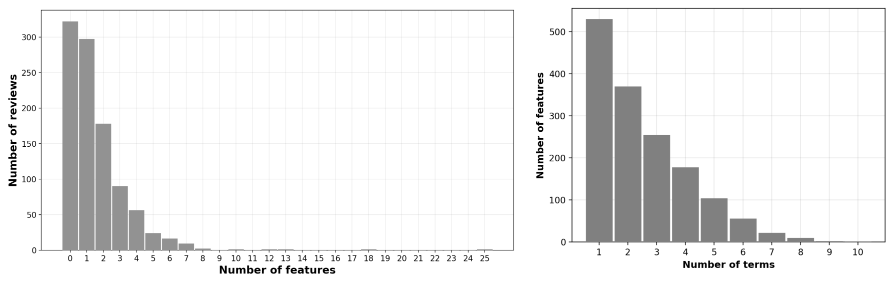
was performed iteratively, each time with a new sample of reviews until the quality of the annotation was at an acceptable level [6]. Once this was achieved, annotators conducted a full-scale annotation on a new sample of 1,000 reviews that resulted in our ground truth.

C) Ground truth

Table 2 reports statistics of our ground truth. These statistics concern subject app reviews, annotated opinions (feature-sentiment pairs) and inter-rater reliability measures. The average length of reviews and sentences is measured in words. Statistics of opinions are reported separately for features and sentiments. The number of features has been given for all the annotated features, distinct ones, and with respect to their length (in words). The number of sentiments has been described including their number per polarity.

The ground truth consists of 1,000 reviews for 8 subject apps. In total, 1,521 opinions (i.e., feature-sentiment pairs) have been annotated. Their sentiment distribution is unbalanced: most feature-sentiment pairs are neutral. Among 1,521 annotated features, 1,172 of them are distinct (i.e. mentioned only once).

The feature distribution in app reviews can be found in Figure 3a. A large number of reviews do not refer to any specific feature. 75% of reviews refers to



(a) Feature distribution in app reviews. (b) Distribution of feature length.

Fig. 3: Feature distribution in app reviews, and Feature length distribution.

no feature or to only one or two features. Figure 3b provides the feature length distribution. The median length for a feature is 2 words, 75% of features has between 1 and 3 words, and nearly 5% has more than 5 words.

4.3 Evaluation Metrics

We used precision and recall metrics [6] to answer RQ1 and RQ2. We used them because feature extraction is an instance of information extraction problem [6], whereas sentiment analysis can be seen as a classification problem [16].

A) Evaluation Metrics For Feature Extraction

In answering RQ1, precision indicates the percentage of extracted features that are true positives. Recall refers to the percentage of annotated features that were extracted. An extracted feature can be true or false positive. True positive features correspond to features that were both extracted and annotated; False positives are features that were extracted but not annotated; Annotated but not extracted features are called false negative. To determine whether an extracted feature is true or false positive, we compared them with annotated features in the ground truth. To this end, we used the following feature matching method:

Definition 4 (Feature Matching) Let Γ be the set of words in a review sentence and $f_i \subseteq \Gamma$ be the set of words used to refer to feature i in that sentence. Two features $f_1, f_2 \subseteq \Gamma$ match at level n (with $n \in \mathbb{N}$) if and only if (i) one of the feature is equal to or is a subset of the other, i.e. $f_1 \subseteq f_2$ or $f_2 \subseteq f_1$, and (ii) the absolute length difference between the features is at most n, i.e. $||f_1| - |f_2|| \leq n$.

B) Evaluation Metrics For Feature-Specific Sentiment Analysis

In answering RQ2, precision indicates the percentage of predicted sentiments that are correct. Recall refers to the percentage of annotated sentiments that are predicted correctly. To determine whether predicted sentiments are correct, we compared them with annotated ones in the ground truth.

We measured precision and recall for each polarity category (i.e. positive, neutral and negative). We also calculated the overall precision and recall of all

Table 3: RQ1. Results for feature extraction at varied levels of feature matching.

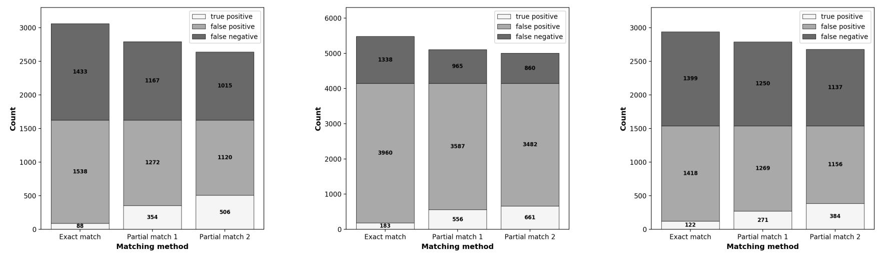
	Exact Match (n=0)						**Partial Match 1 (n=1)**						**Partial Match 2 (n=2)**					
	GuMa		**SAFE**		**ReUS**		**GuMa**		**SAFE**		**ReUS**		**GuMa**		**SAFE**		**ReUS**	
App Name	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R	P	R
Evernote	0.06	**0.13**	**0.07**	0.08	**0.07**	0.08	0.15	**0.35**	**0.22**	0.24	0.19	0.20	0.17	**0.39**	**0.32**	0.35	0.27	0.29
Facebook	0.03	0.07	0.03	0.03	**0.09**	**0.09**	0.10	**0.28**	**0.15**	0.17	**0.15**	0.14	0.13	**0.36**	**0.23**	0.26	0.20	0.19
eBay	0.04	**0.07**	0.04	0.05	**0.06**	0.06	0.14	**0.26**	**0.22**	**0.26**	0.14	0.14	0.17	0.32	**0.34**	**0.39**	0.22	0.21
Netflix	0.03	**0.13**	0.03	0.03	**0.06**	0.07	0.11	**0.45**	**0.19**	0.21	0.18	0.21	0.13	**0.55**	**0.27**	0.29	0.25	0.29
Spotify	0.05	0.10	0.05	0.04	**0.15**	**0.13**	0.18	**0.37**	**0.24**	0.23	0.23	0.20	0.21	**0.43**	**0.36**	0.34	0.29	0.26
Photo Editor	0.12	0.11	0.12	0.09	**0.14**	**0.13**	0.26	0.25	**0.34**	**0.27**	0.23	0.21	0.29	0.27	**0.38**	**0.30**	0.27	0.25
Twitter	**0.06**	**0.19**	**0.06**	0.07	0.02	0.02	0.16	**0.49**	**0.23**	0.24	0.11	0.11	0.18	**0.58**	**0.35**	0.36	0.27	0.26
WhatsApp	0.05	**0.21**	**0.11**	0.11	0.06	0.06	0.14	**0.56**	**0.32**	0.33	0.19	0.20	0.16	**0.64**	**0.39**	0.40	0.24	0.25
Mean	0.05	**0.13**	0.06	0.06	**0.08**	0.08	0.15	**0.37**	**0.24**	0.24	0.18	0.18	0.18	**0.44**	**0.33**	0.34	0.25	0.25

three sentiment polarities. To this end, we used the weighted average of precision and recall of each polarity category. The weight of a given polarity category was determined by the number of annotated sentiments with the sentiment polarity.

5 Results

RQ1: What is the effectiveness of feature extraction approaches?
To answer RQ1, we compared extracted features to our ground truth using feature matching at levels 0, 1 and 2 (see Definition 4). We selected these levels as extracted and annotated features may differ by a few words but still indicating the same app feature. We then computed precision and recall at these levels. Table 3 reports precision and recall for each approach at different matching levels (best in bold). The results show the approaches achieved low precision, recall given Exact Match. For all three approaches, precision and recall increase when we loosen the matching criteria to partial matching with $n = 1$ or 2. The growth can be attributed to the changed numbers of true positives (TPs), false positives (FPs) and false negatives (FNs) when n increases. Figures 4 shows their behavior as the matching level n increases; $\Delta TPs = -\Delta FPs = -\Delta FNs$ when n increases.

RQ2: What is the effectiveness of feature-specific sentiment analysis approaches?
In answering RQ2, we report the effectiveness of ReUS and GuMa in feature-specific sentiment (see Section 4.3). To this end, we compared predicted and annotated sentiments, and exploited a subset of the ground truth with opinions (feature-sentiment pairs) we used to answer RQ1. Indeed, since ReUS predicts sentiments only for extracted features, we considered only true positive features obtained in answering RQ1 and formed three datasets, each corresponding to true positive features (and their sentiment) from Exact Match, Partial Match$_1$ and Partial Match$_2$. Table 4 reports for each dataset the total number of opinions, and their breakdown to polarity categories. We also evaluated GuMa with these datasets and with all the annotated opinions in our ground truth.



(a) SAFE (b) GuMa (c) ReUS

Fig. 4: RQ1. No. TPs, FPs and FNs as the level of features matching changes.

Table 4: RQ2. Dataset used for evaluating feature-specific sentiment analysis.

Dataset	**# opinions**	**# positive**	**# neutral**	**# negative**
Exact Match	122	56	52	14
Partial Match 1	271	97	149	25
Partial Match 2	384	120	226	38
All Annotated	1521	416	964	141

The answer to RQ2 can be given at two levels of details, the overall effectiveness of predicting a sentiment, and the effectiveness of predicting a specific polarity (e.g., positive). We report our results at both levels of details.
Overall effectiveness. Table 5 reports the number of correct predictions, and weighted precision/recall for inferring overall sentiment (best in bold). We can observe that ReUS achieves higher precision and recall than GuMa for Exact Match dataset, whereas both approaches have similar performances on the Partial Match$_1$ and Partial Match$_2$ datasets.
Specific effectiveness. In Table 6, we report the metrics showing the effectiveness of the approaches in predicting specific polarities (best in bold). The results show that on positive opinions ReUS achieves higher precision while suffering from lower recall. Conversely, on neutral opinions GuMa provides better precision but lower recall than ReUS. When looking at the approaches, the analysis of the results revealed that none of the approaches was able to reliably assess the sentiment of negative options. Both approaches were good at discriminating between positive and negative opinions. Most incorrect predictions were caused by misclassifying positive/negative sentiment with neutral one and vice versa.

6 Discussion

The results indicate that the approaches have limited effectiveness in mining user opinions. Our findings bring into question their practical applications.

Table 5: RQ2. Results for feature-specific sentiment analysis (overall).

Dataset	**Approach**	**# correct prediction**	**Precision**	**Recall**
Exact Match	ReUS	**85**	**0.74**	**0.70**
	GuMa	77	0.65	0.63
Partial Match 1	ReUS	**184**	0.69	**0.68**
	GuMa	176	**0.72**	0.65
Partial Match 2	ReUS	**265**	0.69	**0.69**
	GuMa	252	**0.73**	0.66
All Annotated	ReUS	-	-	-
	GuMa	**958**	**0.73**	**0.63**

Table 6: RQ2. Results for feature-specific sentiment analysis (per each polarity).

		Positive			**Neutral**			**Negative**		
Dataset	**Approach**	**# correct prediction**	**Precision**	**Recall**	**# correct prediction**	**Precision**	**Recall**	**# correct prediction**	**Precision**	**Recall**
Exact Match	ReUS	35	**0.90**	0.62	**45**	0.60	**0.87**	5	**0.62**	0.36
	GuMa	**47**	0.68	**0.84**	21	**0.68**	0.40	**9**	0.41	**0.64**
Partial Match 1	ReUS	47	**0.80**	0.48	**131**	0.66	**0.88**	6	**0.43**	0.24
	GuMa	**86**	0.61	**0.89**	73	**0.85**	0.49	**17**	0.40	**0.68**
Partial Match 2	ReUS	53	**0.80**	0.44	**205**	0.68	**0.91**	7	**0.41**	0.18
	GuMa	**107**	0.59	**0.89**	122	**0.86**	0.54	**23**	0.38	**0.61**
All Annotated	ReUS	-	-	-	-	-	-	-	-	-
	GuMa	**355**	**0.52**	**0.85**	**510**	**0.87**	**0.53**	**93**	**0.36**	**0.66**

A) Feature Extraction
In our experiment, feature extraction methods have lower precision and recall than previously reported [10,12,13]. SAFE was reported with 0.71 recall [13]. Our results show the approach achieves 0.34 recall for the least rigorous evaluation strategy. The majority of features extracted by GuMa are incorrect. Although GuMa initially reported precision and recall of 0.58 and 0.52 [12], our experiment found lower figures of 0.18 precision and 0.44 recall.

Although the difference may be due to our re-implementation of the GuMa method, we have taken great care in implementing the method as described in the paper as rigorously as possible. Unfortunately, the original Guma implementation was not available for comparison. We believe ReUS suffered from low precision and recall because it was designed to extract features from product reviews in an online commerce website (Amazon) rather than from app reviews in app stores [16]. Our findings support a conjecture that the original evaluation procedures of SAFE and GuMa led to over-optimistic results. The limitations of these procedures have been questioned recently [21,22]. These procedures did not define a feature matching strategy [21], relied on a subjective judgment [13, 22], and used a biased dataset [12, 21, 22]. We hope our new annotated dataset and description of our evaluation method will contribute to improving the quality of feature extraction techniques and their evaluations.
B) Feature-Specific Sentiment Analysis
Our investigation of results (RQ2) concludes that the overall effectiveness of the approaches is promising (see Table 5). However, it reveals that their precision and

recall differ considerably by sentiment class (positive, negative, or neutral). The approaches provide satisfactory performance for predicting positive and neutral sentiments. But they suffer from inaccurate predictions for negative sentiments. Overall, we are surprised by the comparable effectiveness of both approaches. We expected ReUS to outperform GuMa. ReUS exploits a sophisticated technique to detect an opinion word in a sentence that carries a feature-specific sentiment; GuMa makes predictions based on a simplified premise that a feature-specific sentiment corresponds to the overall sentiment of a sentence.

C) Implication on Requirement Engineering Practices

Identifying what precision and recall app review mining techniques should have to be useful for requirements engineers in practice is an important open question [3]. In principle, a tool facilitating opinion mining should synthesize reviews so that the effort for their further manual analysis would be negligible or at least manageable. Clearly, this effort depends on a scenario the approach intends to support. Given a scenario of prioritizing problematic features, a developer may seek for information about the number of specific features that received negative comments, for example to understand their relevance. To this end, both information about extracted features and their predicted sentiments should be accurate and complete. Our results, however, show that feature extraction techniques generate many false positives. Given the large number of extracted features, filtering out false positives manually may not be cost-effective. We may imagine that the problem could be partially addressed using a searching tool [8]; Requirements engineers could use the tool to filter out uninteresting features (including false positives) and focus on those of their interest.

However, other issues remain unsolved. Feature extraction techniques fail to identify many references to features (they have low recall), and sentiment analysis techniques perform poorly for identifying feature-specific negative sentiments.

7 Threats to Validity

Internal Validity. The main threat is that the annotation of reviews was done manually with a certain level of subjectivity and reliability. To overcome the risk we followed a systematic procedure to create our ground truth. We prepared an annotation guideline with definitions and examples. We conducted several trial runs followed by resolutions of any conflicts. Finally, we evaluated the quality of the annotation using inter-rater agreement metrics.

External Validity. To mitigate the threat, we selected reviews for popular apps belonging to different categories and various app stores. These reviews are written using varied vocabulary. We, however, admit that the eight apps in our study represent a tiny proportion of all the apps in the app market. Although our dataset is comparable in size to datasets in previous studies [10, 12, 13], we are also exposed to sampling bias.

Table 7: The summarized differences between our study and related works.

	Criterion	Our Study	SAFE [13]	GuMa [12]	ReUS [10]
Evaluation	No. Approaches	**3**	2	1	1
	Feature Extraction	**Yes**	**Yes**	No	**Yes**
	Sentiment Analysis	**Yes**	-	**Yes**	**Yes**
	Method	**Automatic**	Manual	Manual	**Automatic**
Ground Truth	Released	**Yes**	No	No	No
	No. Apps	**8**	5	7	-
	No. Reviews	1000	80	**2800**	1000
	No. App Stores	**2**	1	**2**	-
	Dataset Analysis	**Yes**	No	No	**Yes**

Construct Validity. To mitigate the threat, we used precision and recall metrics that are extensively used for evaluating the effectiveness of information extraction and classification techniques.

8 Related Works

Previous work have proposed benchmarks for app review analytics (e.g. [15,21]) but with objective different than ours.

Table 7 shows the differences between our study and previous works, pointing out the different criteria that guided the evaluations, which are grouped into Evaluation and Ground Truth categories. The first includes criteria such as the number of evaluated approaches, evaluated tasks and a method type used for their evaluation. The latter includes characteristics of datasets.

In our study, we evaluated three approaches: SAFE, GuMa and ReUS. We assessed them in addressing problems of feature extraction and sentiment analysis. Johann et al [13] also compared SAFE to GuMa [12]. Our study extends their evaluation by including ReUS [10]. Unlike the original study [12], we evaluated GuMa in performing a feature extraction rather than modeling feature topics. We also compared the approach to ReUS in inferring a feature-specific sentiment.

We used a different methodology for evaluating SAFE and GuMa [12, 13]; The correctness of their solutions has been evaluated manually [12, 13]. The judgement criteria, however, has not been defined. Such a procedure suffered from several threats to validity such as human error, authors' bias and the lack of repeatability [22]. To address the limitations, we adopted automatic matching methods and defined explicit matching criteria.

The ground truth in our study differs from that used in previous works. Unlike Dragoni et al [10], we evaluated ReUS using app reviews. The authors used a dataset composed of comments for restaurant and laptops. As Johann [13] and Guzman [12], we created an annotated dataset for the evaluation. We, however, used a systematic procedure and assessed the quality of ground truth using acknowledged measures [16, 20]. Previous studies did not report a systematic annotation procedure [13] nor measured the quality of their annotation [12]. Their datasets were not analyzed nor made public [12, 13].

9 Conclusion

Mining user opinions from app reviews can be useful to guide requirement engineering activities such as user validation [1, 2, 19], requirements elicitation [1, 2], or requirement prioritization [1]. However, the performance of app review mining techniques and their ability to support these tasks in practice are still unknown.

We have presented an empirical study aimed at evaluating existing opinion mining techniques for app reviews. We have evaluated three approaches: SAFE [13] relying on part-of-speech parsing, GuMa [12] adopting a collocation-based algorithm, and ReUS [10] exploiting a syntactic dependency-based parser. We have created a new dataset of 1,000 reviews from which 1,521 opinions are specific features were manually annotated. We then used this dataset to evaluate the feature identification capabilities of all three approaches and the sentiment analysis capabilities of GuMa and ReUS.

Our study indicates that feature extraction techniques are not yet effective enough to be used in practice [9,21] and that have lower precision and recall than reported in their initial studies. Our study also indicates that feature-specific sentiment analysis techniques have limited precision and recall, particularly for negative sentiments. We hope our novel annotated dataset [7] and evaluation method will contribute to improving the quality of app review mining techniques.

References

1. AlSubaihin, A., Sarro, F., Black, S., Capra, L., and Harman, M. App store effects on software engineering practices. *IEEE Transactions on Software Engineering* (2019), 1–1.
2. Begel, A., and Zimmermann, T. Analyze this! 145 questions for data scientists in software engineering. In *36th International Conference on Software Engineering* (2014), pp. 12–13.
3. Berry, D. M., Cleland-Huang, J., Ferrari, A., Maalej, W., Mylopoulos, J., and Zowghi, D. Panel: Context-dependent evaluation of tools for nl re tasks: Recall vs. precision, and beyond. In *2017 IEEE 25th International Requirements Engineering Conference (RE)* (Sep. 2017), pp. 570–573.
4. Buse, R. P. L., and Zimmermann, T. Information needs for software development analytics. In *34th International Conference on Software Engineering* (2012), pp. 987–996.
5. Croft, B., Metzler, D., and Strohman, T. *Search Engines: Information Retrieval in Practice*, 1st ed. Addison-Wesley Publishing Company, USA, 2009.
6. Cunningham, H., Maynard, D., Tablan, V., Ursu, C., and Bontcheva, K. Developing Language Processing Components with GATE Version 8. *University of Sheffield Department of Computer Science* (Nov. 2014).

7. Dabrowski, J. Manually annotated dataset and an annotation guideline for CAiSE'20 paper. https://github.com/jsdabrowski/CAiSE-20/, Nov. 2019.
8. Dabrowski, J., Letier, E., Perini, A., and Susi, A. Finding and analyzing app reviews related to specific features: A research preview. In *25th International Working Conference, REFSQ 2019, Essen, Germany, March 18-21, 2019, Proceedings* (2019), pp. 183–189.
9. Dalpiaz, F., and Parente, M. RE-SWOT: from user feedback to requirements via competitor analysis. In *25th International Working Conference, REFSQ 2019, Essen, Germany, March 18-21, 2019, Proceedings* (2019), pp. 55–70.
10. Dragoni, M., Federici, M., and Rexha, A. An unsupervised aspect extraction strategy for monitoring real-time reviews stream. *Inf. Process. Manage. 56*, 3 (2019), 1103–1118.
11. Gu, X., and Kim, S. "what parts of your apps are loved by users?" (t). In *30th International Conference on Automated Software Engineering* (2015), pp. 760–770.
12. Guzman, E., and Maalej, W. How do users like this feature? a fine grained sentiment analysis of app reviews. In *RE* (2014), T. Gorschek and R. R. Lutz, Eds., IEEE Computer Society, pp. 153–162.
13. Johann, T., Stanik, C., B., A. M. A., and Maalej, W. Safe: A simple approach for feature extraction from app descriptions and app reviews. In *2017 IEEE 25th International Requirements Engineering Conference* (2017), pp. 21–30.
14. Johanssen, J. O., Kleebaum, A., Bruegge, B., and Paech, B. How do practitioners capture and utilize user feedback during continuous software engineering? *2019 IEEE 27th International Requirements Engineering Conference* (2019).
15. Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., and Oliveto, R. Sentiment analysis for software engineering: How far can we go? In *40th International Conference on Software Engineering* (2018), pp. 94–104.
16. Liu, B. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2012.
17. Martin, W., Sarro, F., Jia, Y., Zhang, Y., and Harman, M. A survey of app store analysis for software engineering. *IEEE Transactions on Software Engineering 43*, 9 (2017), 817–847.
18. McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. Image-based recommendations on styles and substitutes. In *38th International Conference on Research and Development in Information Retrieval* (2015), ACM, pp. 43–52.
19. Pagano, D., and Maalej, W. User feedback in the appstore: An empirical study. In *RE* (2013), IEEE Computer Society, pp. 125–134.
20. Pustejovsky, J., and Stubbs, A. *Natural Language Annotation for Machine Learning - a Guide to Corpus-Building for Applications*. O'Reilly, 2012.
21. Shah, F. A., Sirts, K., and Pfahl, D. Is the SAFE approach too simple for app feature extraction? A replication study. In *25th International Working Conference, REFSQ 2019, Essen, Germany, March 18-21, 2019, Proceedings* (2019), pp. 21–36.
22. Shah, F. A., Sirts, K., and Pfahl, D. Simulating the impact of annotation guidelines and annotated data on extracting app features from app reviews. In *International Conference on Software Technologies (ICSOFT)* (2019).
23. Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology 61*, 12 (2010), 2544–2558.
24. Vu, P. M., Nguyen, T. T., Pham, H. V., and Nguyen, T. T. Mining user opinions in mobile app reviews: A keyword-based approach (t). In *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (2015), ASE '15, pp. 749–759.