



Article

# Enhancing Software Feature Extraction Results Using Sentiment Analysis to Aid Requirements Reuse

Indra Kharisma Raharjana <sup>1,\*</sup> , Via Aprillya <sup>1</sup> , Badrus Zaman <sup>1</sup>, Army Justitia <sup>1</sup>   
and Shukor Sanim Mohd Fauzi <sup>2</sup>

<sup>1</sup> Information Systems, Universitas Airlangga, Surabaya 60115, Indonesia;

via.aprillya-2016@fst.unair.ac.id (V.A.); badruszaman@fst.unair.ac.id (B.Z.); army-j@fst.unair.ac.id (A.J.)

<sup>2</sup> Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Perlis Branch, Arau Campus, Arau, Perlis 02600, Malaysia; shukorsanim@uitm.edu.my

\* Correspondence: indra.kharisma@fst.unair.ac.id; Tel.: +62-31-5936501

**Abstract:** Recently, feature extraction from user reviews has been used for requirements reuse to improve the software development process. However, research has yet to use sentiment analysis in the extraction for it to be well understood. The aim of this study is to improve software feature extraction results by using sentiment analysis. Our study's novelty focuses on the correlation between feature extraction from user reviews and results of sentiment analysis for requirement reuse. This study can inform system analysis in the requirements elicitation process. Our proposal uses user reviews for the software feature extraction and incorporates sentiment analysis and similarity measures in the process. Experimental results show that the extracted features used to expand existing requirements may come from positive and negative sentiments. However, extracted features with positive sentiment overall have better values than negative sentiments, namely 90% compared to 63% for the relevance value, 74–47% for prompting new features, and 55–26% for verbatim reuse as new requirements.

**Keywords:** software feature extraction; sentiment analysis; requirements reuse; requirements elicitation; user reviews



**Citation:** Raharjana, I.K.; Aprillya, V.; Zaman, B.; Justitia, A.; Fauzi, S.S.M. Enhancing Software Feature Extraction Results Using Sentiment Analysis to Aid Requirements Reuse. *Computers* **2021**, *10*, 36. <https://doi.org/10.3390/computers10030036>

Academic Editor:  
Yevgeniya Kovalchuk

Received: 30 January 2021  
Accepted: 15 March 2021  
Published: 19 March 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Requirement elicitation is the process of searching and acquiring stakeholders' interests in a software development [1–3]. This aims to help produce new requirements from relevant stakeholders [4], which can be taken from various sources and are available in a range of formats. Sources of requirements can be from any relevant parties, experts, current systems, and competitor systems [1]. Stakeholders may comprise system end-users, top management, or external interested parties, such as regulators [5]. Techniques that are commonly used in requirement elicitation are open/closed interviews, scenarios, use cases, observations, questionnaires, brainstorming, prototyping, focus group discussions, and joint application development (JAD) workshops [6]. It is widely recognized among researchers and industry practitioners that software projects are very prone to failure when the requirements elicitation process is not carried out properly [7,8]. In the software development life cycle (SDLC), this undertaking is challenging because it involves socio-technical characteristics and changing requirements [9,10], and because there may be problems of ambiguity and tacit knowledge transfer [11,12]. Such problems, if found too late, can lead to high rework costs because they necessitate modifications in other processes [4,13].

Different software products in the same area often have similar requirements. The reuse of these requirements can reduce costs in the elicitation process [14]. Requirements reuse could be from formal [15] or informal sources [16], product description sources [17], user reviews [18–20], expert reviews [18], social media [21,22], and online news [5,23].

Aside from developing the mobile application ecosystem, user reviews on Google Play are becoming increasingly popular to gather requirement reuse [18,24], requests for improvements [25], and feedback on specific features [26]. App users continually post a large number of reviews per day, addressed to both developers and user communities. When addressed to developers, reviews could be either appreciations, complaints, bug reports, or new feature requests [27]. User reviews are written in natural language and so natural language preprocessing (NLP) is used to extract software features from user reviews [28].

Although extensive research has been carried out to analyze user reviews, only limited studies use sentiment analysis in the software feature extraction. Sentiment analysis can understand the interests and emotional responses from product users [29,30]. Zang et al. [31] used sentiment analysis in analyzing reviews of mobile users, whereas Panichella et al. [32] classified application reviews relevant to software maintenance and evolution. Chen et al. [33] mined mobile app reviews to extract the most informative information from users. Guzman and Maalej [26] assisted developers in assessing user reviews about application features in order to identify new requirements or plan for future releases.

However, none of the research above looked into the use of sentiment analysis in examining the extracted features' relevance and its influence on the decision-making of new requirements. System analysts' domain knowledge vary in levels and this may jeopardize the requirements' completeness. This information is vital for system analysts in the requirements elicitation process as they can use the results to verify the requirements and improve the completeness.

Using user reviews data, this study aims to use sentiment analysis to obtain a comprehensive point of view of requirement reuse. The research questions are as follows RQ1: What is the correlation between feature extraction and sentiment analysis in terms of acquiring relevant requirements? RQ2: What is the correlation between feature extraction and sentiment analysis in informing system analysts in the decision-making of new requirements? RQ3: What is the correlation between feature extraction and sentiment analysis pertaining to verbatim reuse of extracted features as new requirements?

## 2. Materials and Methods

This study aims to provide a comprehensive requirement report by using sentiment analysis based on user reviews on Google Play. We use the initial requirements made by the system analyst as the system input. The requirements are then optimized by using the outcomes of feature extraction from user reviews available on Google Play.

### 2.1. Research Framework

The research procedure is shown in Figure 1. The pre-processing steps for handling the initial requirements are carried out by tokenizing, case folding, and removing stop words. The pre-processing stage for preparing user review data includes data cleaning, spelling normalization, sentence tokenization, and part-of-speech (POS) tagging. Software feature extraction with POS chunking is carried out based on the POS tag pattern. Subsequent to this, the data were treated with case folding, lemmatization, and stop words elimination. Sentiment analysis is conducted on the data by calculating the polarization and subjectivity. The similarities between features from the initial requirements list and user reviews are then calculated to produce the requirements report. This report provides a general outline of the new relevant features and their sentiment analysis. The system analysts could then use this report as a basis for the decision-making.

### 2.2. Data

The data required in this study consisted of the initial requirements created by the system analysts and the extracted features from user reviews on the Google Play. We used the earthquake and the e-marketplace apps as case studies. Data were collected from the requirements document and user reviews.

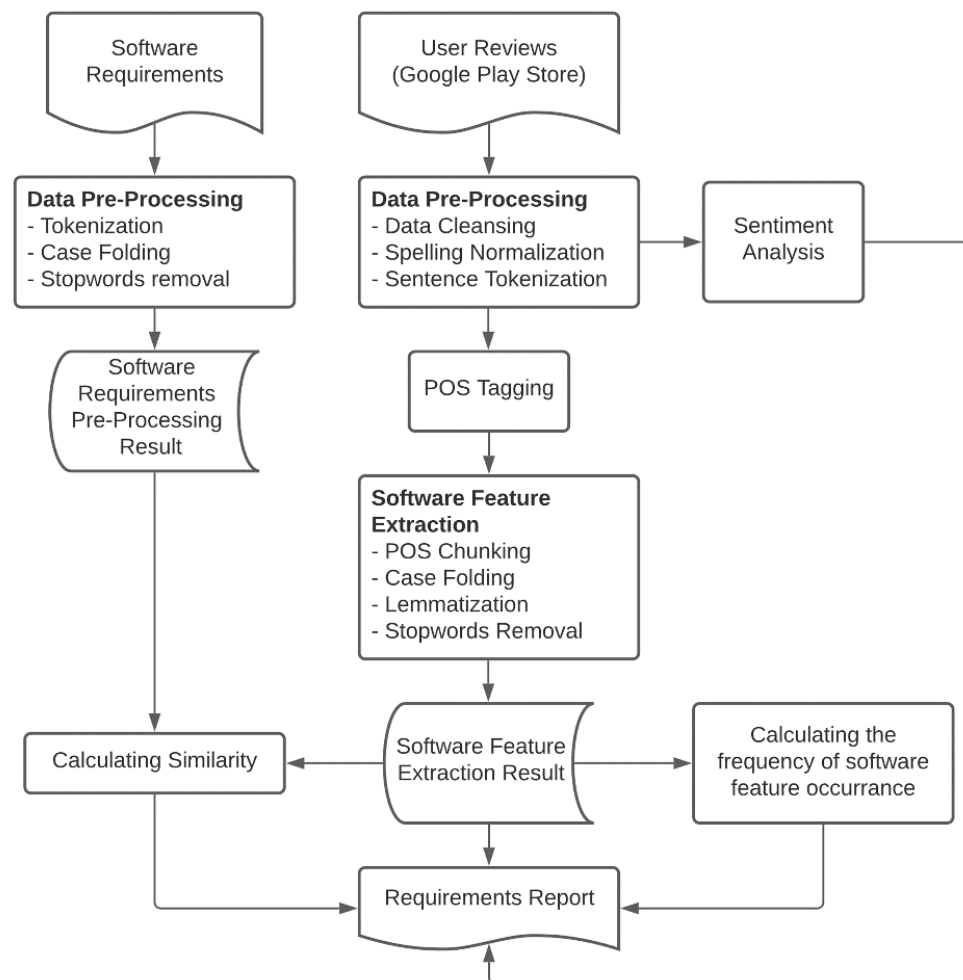


Figure 1. The research framework.

### 2.2.1. Software Requirements Documents

A list of requirements was used as an initial input to the system. Ten system analysts composed the requirements document using specific cases, which resulted in a list of functional requirements. The information used to make this document was obtained from several sources. Table 1 shows an example of a list of requirements in the requirements document.

Table 1. List of requirements.

#	Software Feature for Earthquake App	Software Feature for E-Marketplace App
1	Notification when the earthquake happened.	Users can sign up for a new account.
2	Notification of the aftershocks	Users can filter search results.
3	Information for disaster relief.	Users can sort search results.
4	Information about the earthquake location.	Users can add a product to the shopping cart.
5	Information of the earthquake magnitude.	Users can see the total amount to pay before checkout.
6	Display the time of the earthquake.	Users can track delivery status.
7	Educational about actions to be taken when an earthquake occurs.	Users can interact with the seller using the chat room.
8	Display data of the previous earthquakes.	Users can see purchase history.
9	Information about first aid.	Users can give a product review.
10	Information on how to deal with post-earthquake trauma.	Users can report frauds or scams.
11	Latest news related to the earthquake.	
12	Display the gathering point/safe zone in the user's area.	
13	Shows victim data (lost and found).	
14	Displays emergency numbers.	
15	Fundraising and donation information.	

### 2.2.2. User Reviews on Google Play

User reviews were sourced from similar applications relevant to the case study. Keywords for the query were ‘earthquake’ and ‘e-marketplace’, entered into the search feature on the Google Play. From the list of applications that were most relevant to these keywords, the top 10–15 applications for each case study were selected. This search was conducted on 8 July 2020 for the earthquake app, and 18 November 2020 for the e-marketplace app. The list of applications that were used in this study is shown in Table 2.

**Table 2.** List of Apps.

#	List of Application Related to Earthquake	List of Application Related to the E-Marketplace
1	My Earthquake Alerts—US & Worldwide Earthquakes by jRustonApps B.V.	Amazon Shopping—Search, Find, Ship, and Save by Amazon Mobile LLC
2	Earthquake Tracker—Latest quakes, Alerts & Map by trinitytech.com.tr	eBay—Buy, sell, and save money on your shopping by eBay Mobile
3	LastQuake by EMSC-CSEM	Shopee by Shopee
4	Info BMKG by Badan Meteorologi, Klimatologi, dan Geofisika	Lazada—Online Shopping App by Lazada Mobile
5	Earthquake Alert! by Josh Clemm	AliExpress by Alibaba Mobile
6	Volcanoes & Earthquakes by VolcanoDiscovery	ZALORA—Fashion shopping by Zalora South East Asia Pte Ltd.
7	Earthquakes Tracker by DoubleR Software	Tokopedia—Bebas Ongkir by Tokopedia
8	Earthquakes by KB@	Wish—Shopping Made Fun by Wish Inc.
9	Earthquake Network—Realtime alerts by Francesco Finazzi	Carousell: Snap-Sell, Chat-Buy by Carousell
10	EarthQuake—Map & Alerts by Francesk Xhaferri	Bukalapak—Online Marketplace by PT Bukalapak.com
11	PH Weather And Earthquakes by droidgox	
12	Earthquake Network Pro—Realtime alerts by Francesco Finazzi	
13	Earthquake Plus—Map, Info, Alerts & News by Slava Barouline	
14	EQInfo—Global Earthquakes by gempa GmbH	
15	Earthquake by Appendroid	

User reviews from 15 applications on Google Play were collected using the Python library `google-play-scraper` 0.1.1 (<https://pypi.org/project/google-play-scraper/>, (accessed on 1 March 2021)). Data extracted from user reviews in each application was stored in plain text format. From the 15 earthquake-related applications extracted, and 17,133 user reviews were gathered.

### 2.3. Pre-Processing

The data collected were raw data that need processing. Since the requirements document and user reviews had different data characteristics, it was necessary to carry out the data pre-processing separately.

#### 2.3.1. The Requirement Document

The pre-processing for the requirement document involved sentence tokenizing, case folding, and stop word eliminating. A sentence tokenizer was used to separate functional requirements written in the requirement document. Case folding was used to change all letters in the requirement list data into lowercase. Stop words are the most common words in languages, such as ‘the’, ‘a’, ‘on’, ‘is’, and ‘all’. These words do not have any significance and are removed from the text. We used the stop word list provided by the

Natural Language Toolkit (NLTK) library [34]. Examples of the pre-processing for the requirement document are shown in Table 3.






**Table 3.** Examples of implementing pre-processing in the requirements document.

#	Requirements	Case Folding	The Stop Words Removal
1.	Notification when earthquake happened.	Notification when earthquake happened.	Notification earthquake happened
2.	Notification of the aftershocks	Notification of the aftershocks	Notification aftershocks
3.	Information of disaster relief.	Information of disaster relief.	Information disaster relief
4.	Information of the earthquake location.	Information of the earthquake location.	Display map earthquake location.
5.	Information of the earthquake magnitude.	Information of the earthquake magnitude.	Information earthquake magnitude

### 2.3.2. User Reviews

The pre-processing of user reviews data was performed through data cleaning, spelling normalization, tokenization, POS tagging, and stop word removal. Data cleaning was performed to remove unnecessary characters to reduce noise, such as hashtags (#), URLs, numbers, certain punctuation marks, and other symbols. Data cleaning was performed because many user reviews contain irrelevant characters, such as emojis or excessive dots. This was done by using the string encoding function with the parameters ‘ignore’ and ‘ascii’ so that characters outside of ASCII were eliminated from the text. Table 4 shows an example of the data cleaning implementation.

**Table 4.** Example of data cleaning.

No	Original Sentences	After Data Cleaning
1.	Love this app  very accurate!	Love this app very accurate!
2.	This is an excellent app. Clear presentation and easy to set up. Notifications are good. Plenty of information about each quake. Recommended  .	This is an excellent app. Clear presentation and easy to set up. Notifications are good. Plenty of information about each quake. Recommended.
3.	Excellent! I downloaded this right after my local news announced an earthquake, and sure enough, it showed up as the most recent quake plus all the pertinent details!! Clear, concise, vivid colors, super easy to use. Many thanks Developers, important in my area as we have had several small ones the past few years. Highly recommended!!  .	Excellent! I downloaded this right after my local news announced an earthquake, and sure enough, it showed up as the most recent quake plus all the pertinent details!! Clear, concise, vivid colors, super easy to use. Many thanks Developers, important in my area as we have had several small ones the past few years. Highly recommended!!
4.	Good for visual people like me . . . . gives you a good idea of energy movement   .	Good for visual people like me gives you a good idea of energy movement.

Spelling normalization is the process of correcting words that are nonetheless incorrectly written or abbreviated. For example, the words ‘notification’ and ‘notif’ have the same meaning. However, because ‘notif’ is an abbreviation that is not recorded in dictionaries, the word ‘notif’ is considered non-valuable. Therefore, the word ‘notif’ needs to be rewritten as ‘notification.’ Abbreviations such as ‘aren’t’ also need to be normalized into ‘are not.’ Spelling normalization is done using the spell function in the NLTK library. The example of spelling normalization is shown in Table 5.

In this study, the sentiment value was calculated in each sentence. The sentence tokenization divided user reviews into several sentences using the sent\_tokenize function from the NLTK library, which splits texts into sentences. Meanwhile, POS tagging functions in NLP automatically labeled words into their parts of speech. For example, in the sentence “I build applications”, there is a label PP = pronoun, VV = verb, and NN = noun. The system receives the input in the form of a sentence, and the output is be “I/PP build/VV applications/NN.” POS tagging was performed using the pos\_tag function in the NLTK library. Table 6 shows an example of POS tagging process in the sentences.

**Table 5.** Example of spelling normalization.

No	Original Sentences	After Spelling Normalization
1.	Nice application it have good info for earthquake and tsunami and updates regularly.	Nice application it have good info for earthquake and tsunami and updates regularly.
2.	Great notiification on my alert sound Has good data on earthquakes!	Great notification on my alert sound Has good data on earthquakes!
3.	I check my earthquake alerts constanly.	I check my earthquake alerts constantly.
4.	Helpfull for Disaster Management.	Helpful for Disaster Management.

**Table 6.** POS Tagging in sentences

No	Sentence	POS Tag	Description
1.	Good for basic notifications only.	[('Good', 'JJ'), ('basic', 'JJ'), ('notifications', 'NNS'), ('.', '.')] ]	JJ: Adjective NNS: Noun, plural
2.	Needs to have ability to set multiple alarms.	[('Needs', 'NNP'), ('ability', 'NN'), ('set', 'VBN'), ('multiple', 'JJ'), ('alarms', 'NNS'), ('.', '.')] ]	NNP: Proper noun, singular NN: Noun, singular or mass VBN: Verb, past participle JJ: Adjective NNS: Noun, plural
3.	I want to be notified of lesser magnitude quakes in my area and greater worldwide.	[('I', 'PRP'), ('want', 'VBP'), ('notified', 'VBN'), ('lesser', 'JJR'), ('magnitude', 'NN'), ('quakes', 'NNS'), ('area', 'NN'), ('greater', 'JJR'), ('worldwide', 'NN'), ('.', '.')] ]	PRP: Personal pronoun VBP: Verb, non-3rd person singular present VBN: Verb, past participle JJR: Adjective, comparative NN: Noun, singular or mass NNS: Noun, plural
4.	Also a setting for specific radius for near me	[('Also', 'RB'), ('setting', 'VBG'), ('specific', 'JJ'), ('radius', 'NN'), ('near', 'IN'), ('.', '.')] ]	RB: Adverb VBG: Verb, gerund or present participle JJ: Adjective NN: Noun, singular or mass IN: Preposition or subordinating conjunction.

#### 2.4. Sentiment Analysis

Sentiment analysis gives the quantitative values (positive, negative, and neutral) to a text representing the authors' emotions. User reviews are calculated for polarity and subjectivity using the sentiment feature. We employed the TextBlob library in the analysis sentiment to determine the value. The polarity value lies in the range of  $[-1, 1]$ , where 1 means a positive sentiment and  $-1$  means a negative sentiment. Meanwhile, the value of subjectivity lies in the range of  $[0, 1]$ , where 1 means an absolute subjective statement or in the form of opinion, and 0 means an absolute objective statement or in the form of facts [35]. An example of value assignment can be seen in Table 7.

#### 2.5. Software Feature Extraction

Software feature extraction was performed to obtain the requirements/features from user reviews. This was done by performing POS chunking. The results were then processed again by case folding, lemmatization, and stop word elimination.

**Table 7.** Example of sentiment analysis results on user reviews.

No	User Reviews	Polarity/Sentiment	Subjectivity
1.	Great notifications on my alert sound Has good data on earthquakes!	0.692 Positive	0.583 Subjective
2.	Nice application it has good info for earthquake and tsunami and updates regularly.	0.65 Positive	0.8 Subjective
3.	That is not good for immediate reporting, you can turn off that function, but would allow people to get info for recent quakes.	-0.175 Negative	0.425 Objective
4.	My only problem with this application is the type of font used is difficult to read on my android phone.	-0.25 Negative	1 Subjective

### 2.5.1. POS Chunking

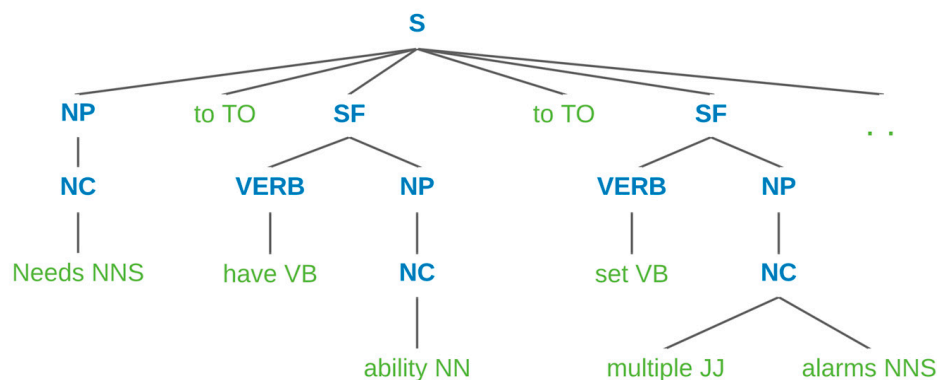
POS chunking is a phrase in a sentence extracted using the POS tag pattern with a regular expression. The POS tag pattern containing nouns, verbs, and adjectives is extracted into a phrase to bring out the features. For example, the terms ‘voice alerts’ and ‘satellite maps’ represent a software feature. In this study, the POS tag pattern used to extract the software features (SFs) are shown in Figure 2. After SFs were extracted, we also needed to define VERB (a group of verbs), NC (noun phrases), and NP (a group of NC). Explanation of the POS tag set, and regular expression code used in this study is shown in Table 8. Penn Treebank POS [36] was used in this study.

```

SF:  {( <VERB|(JJ|JJS|JJR)|NP>+ <DT>* <NP|(RB|RBR|RBS)|(JJ|JJS|JJR)>+ )+ }
VERB: { <VB|VBG|VBZ|VBD|VBN|VBP>+ <IN>* }
NP :  { ( ( <NC>+ <,> ) * <NC>* <CC>+ ) * <NC|IN>+ }
NC:  { ( <DT>* <JJ|JJS|JJR>* ) ? ( <IN>* <NN|NNP|NNS|NNPS>* ) }
    
```

**Figure 2.** POS chunking pattern.

An example of a parse tree from POS chunking results can be seen in Figure 3. We considered all phrases with SF tags. In the example, there are two phrases with SF tags, namely ‘have the ability’ and ‘set multiple alarms’. The examples of software feature extraction are shown in Table 9.



**Figure 3.** Example of parse tree generated from POS chunking pattern.

**Table 8.** Regular expression and part-of-speech tags syntax.

POS Tag/Expression	Abbreviation
*	Zero or more occurrences
?	Zero or one occurrences
+	One or more occurrences
	Or
< >	Word boundaries
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
DT	Determiner
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
CC	Coordinating conjunction
IN	Preposition or subordinating conjunction
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural

**Table 9.** Examples of software feature extraction.

No	Original Sentences	Software Features
1.	Needs to have ability to set multiple alarms.	Have ability Set multiple alarms
2.	I want to be notified of lesser magnitude quakes in my area and greater worldwide.	Be notified of lesser magnitude quakes
3.	Good for basic notifications only.	Good for basic notifications only.
4.	This application helps us identify the ferocity of these quakes and puts us at ease knowing the aftershocks are lower than the initial quake.	Identify the ferocity of these quakes At ease knowing the aftershocks are lower than the initial quake

### 2.5.2. Case Folding, Lemmatization, and Stop Words

The feature extraction results from the POS chunking process were then processed by case folding, lemmatization, and stop word elimination. Case folding aims to change all letters in software feature data to lowercase, by calling the lower() function. Lemmatization aims to group words with different forms into one item. In this process, word affixes are removed to convert words into the basic word forms. For example, the words 'mapping', 'maps', and 'mapped' are changed to their root form 'map'. This process uses the lemmatize function in the WordNet Lemmatizer from the NLTK library. Examples of the application of case folding, lemmatization, and stop word elimination are shown in Table 10. The frequency of software features is recorded to know how many times a feature appears. This determines the importance of a software feature.

### 2.6. Software Feature Similarity

Similarity calculation aims to determine which software features extracted from user reviews are related to the features in the requirement document. The similarity has a value range of 0–1, indicating the similarity or closeness of two words, phrases, or sentences. The



similarity was assessed using the similarity function from the spaCy library [37]. SpaCy uses the word2vec approach to calculate the semantic similarity of words [38]. This process is done by finding the similarity between word vectors in the vector space. The ‘en-core-web-md’ vector model from the spaCy library was used in this process. If the similarity value reaches the threshold of 0.5, then the extracted software feature is considered related to the existing feature. Please note that this threshold may need to be adjusted based on the data characteristics and the real-world conditions.

**Table 10.** POS Tagging in Sentences.

No	SF	Case Folding	Lemmatization	Stopwords
1.	have ability	have ability	have ability	ability
2.	set multiple alarms	set multiple alarms	set multiple alarm	set multiple alarm
3.	be notified of lesser magnitude quakes	be notified of lesser magnitude quakes	be notified of lesser magnitude quake	notified lesser magnitude quake
4.	Good for basic notifications only.	good for basic notifications only	good for basic notification only	good basic notification
5.	identify the ferocity of these quakes	identify the ferocity of these quakes	identify the ferocity of these quake	identify ferocity quake
6.	at ease knowing the aftershocks are lower than the initial quake	at ease knowing the aftershocks are lower than the initial quake	at ease knowing the after shock are lower than the initial quake	ease knowing after shock lower initial quake

### 2.7. Requirement Report

Requirement report displays the results of the extraction in a structured format, generated using a data frame from the Pandas library. The requirement report shows the software features, the polarity values, the subjectivity values, the similarity values, and the software feature frequencies. Polarity and subjectivity values were further divided into minimum, maximum, and average. The collected software features may come from different user review texts and may have different sentiment values. The sentiment scores were distributed by groups by function on the data frame. The argument’s minimum, maximum, average values are displayed in the polarity and subjectivity columns. An example of a requirement report can be seen in Table 11.

### 2.8. Evaluation

The evaluation seeks to analyze the extracted software features’ characteristics based on the polarity and subjectivity values. We also assessed the similarity value as a comparison to extend the analysis results. The focus of evaluation in this study is to quantify:

**RQ1:** *The correlation between feature extraction and sentiment analysis in terms of acquiring relevant requirements.*

**RQ2:** *The correlation between feature extraction and sentiment analysis in informing system analysts in the decision-making of new requirements.*

**RQ3:** *The correlation between feature extraction and sentiment analysis pertaining to verbatim reuse of extracted features as new requirements.*

To compare the data, a questionnaire was used. Systems analysts were asked to evaluate whether or not the system’s software features are relevant to the requirement documents, whether the extraction results prompt to new features addition, and whether the features are to be use verbatim as new requirements. We employed ten system analysts who previously composed the requirements document. The questionnaire was prepared based on the requirements report. This questionnaire consists of extracted software feature with positive and negative sentiments. The sentiments with high similarity values were

also included in the comparison. The questionnaire displays the results of extraction—ten positive phrases, ten negative phrases, and ten phrases with the best similarity values. We generate a questionnaire according to the software feature that the application has (15 questionnaires for the earthquake app and ten questionnaires for the e-marketplace app). We ask three system analysts to evaluate each software feature based on research questions (RQ1–RQ3). The consensus was reached based on a majority assessment results. The data were then be further analyzed to see the sentiment correlation (polarity and subjectivity), the relevance, the decision to change the requirements, and the list of new features to be added.

**Table 11.** An example of a requirements report.

Initial Requirement: Notification When Earthquake Happened									
No	Software Feature	Count	Similarity	Polarity			Subjectivity		
				Min	Max	Mean	Min	Max	Mean
<b>High Similarity</b>									
1.	immediate notification quake	1	0.83	0.00	0.00	0.00	0.00	0.00	0.00
2.	tsunami alert	3	0.81	−0.35	0.47	0.04	0.00	0.67	0.42
3.	earthquake trigger tsunami	2	0.80	0.00	0.00	0.00	0.90	0.90	0.90
4.	set earthquake magnitude alert	2	0.79	0.00	0.00	0.00	1.00	1.00	1.00
5.	see earthquake happened nearby	1	0.77	0.00	0.00	0.00	0.00	0.00	0.00
<b>Positive Sentiment</b>									
1.	tsunami warning added	1	0.74	1.00	1.00	1.00	1.00	1.00	1.00
2.	seismic report	1	0.72	1.00	1.00	1.00	1.00	1.00	1.00
3.	geographical earthquake	2	0.69	0.80	0.80	0.80	0.75	0.75	0.75
4.	monitoring seismic activity worldwide	2	0.60	0.78	0.78	0.78	1.00	1.00	1.00
5.	providing magnitude	1	0.57	1.00	1.00	1.00	1.00	1.00	1.00
<b>Negative Sentiment</b>									
1.	fake earthquake alarm	1	0.76	−0.50	−0.50	−0.50	1.00	1.00	1.00
2.	magnitude earthquake	2	0.74	−0.80	0.00	−0.40	0.00	0.90	0.45
3.	earthquake alarm bit annoying	1	0.68	−0.80	−0.80	−0.80	0.90	0.90	0.90
4.	felt huge earthquake application show nothing	1	0.63	−1.00	−1.00	−1.00	1.00	1.00	1.00
5.	slow detect earthquake	2	0.62	−0.30	−0.30	−0.30	0.40	0.40	0.40

### 3. Results

#### 3.1. Software Feature Extraction Methods

We compared the approaches used to decide which one was the best for the software feature extraction. The approaches were POS chunking, Textblob, and Spacy (see Table 12). We applied the POS tag pattern; the noun phrases function provided by the Textblob and SpaCy libraries; and syntactic dependency attributes, such as dobj and pobj in SpaCy. We compared the results from these with the results from manual tagging. We asked two experts to tag software features from sample user reviews. The comparison results show that the software feature produced by POS chunking has the best F-measure value compared to other approaches, which indicates the effectiveness of the POS tag pattern (See Figure 2). As such, we used POS chunking as the software feature extraction method in this study.

**Table 12.** Comparisons of software feature extraction methods.

	Precision	Recall	F-Measure
TextBlob nounphrases	0.295	0.375	0.330
POS-chunking	0.303	0.785	0.437
Spacy dobj-pobj	0.133	0.321	0.188
Spacy pobj	0.125	0.178	0.147
Spacy dobj	0.133	0.142	0.137
Guzman and Maalej [26]	0.601	0.506	0.549
SAFE [17]	0.239	0.709	0.358

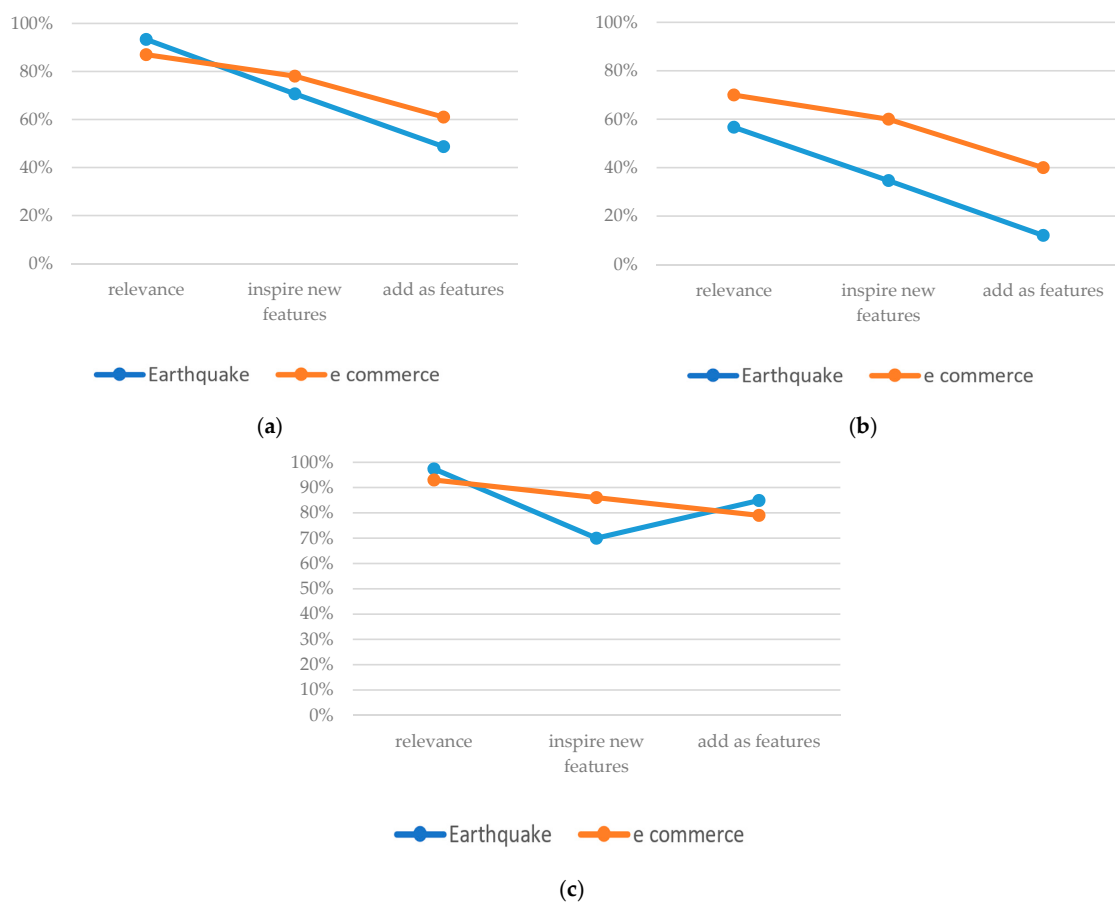
We also attempted to corroborate the comparisons by using other approaches. However, it should be noted that the data used for comparisons are those reported by the study, so this data cannot be used as an apple-to-apple comparison. We used these comparisons to check whether our approach has delivered similar results compared to other approaches. The results show that the POS chunking is suitable for software feature extraction. Another important point to note is that we encountered the same problem as the previous study [17,26], namely the low value of precision. The user review data has a non-standard language format and does not always refer to software features.

### 3.2. Questionnaire Results

The software features results were separated based on their polarity (positive sentiment and negative sentiment). We added features with high similarity value as the benchmarks as per standard of requirement reuse. The questionnaire required the respondents to assess whether the feature extraction results are relevant, prompt the addition of new requirements, and verbatim reuse of extracted features as new requirements. We assume that the extracted features have some degree of relevancy to the initial requirements and to new features addition. Figure 4 shows the questionnaire results.

From the highest relevance results, the features with high similarity have an average of 95%, followed by positive sentiment at 90%. Negative sentiment scores are lower at 63%. Whether or not the results prompt system analysts to update the existing requirements list, the high similarity and the positive sentiment score an average value of 78% and 74%, respectively. Meanwhile, the negative sentiment scores only 47%. In terms of adding new requirements, the high similarity has the highest value at 82%, whereas the positive and negative sentiment values score significantly lower at 55% and 22%, respectively.

The results of the questionnaire can be seen in Table 13, which maps the results of feature extraction with positive, negative, and high similarity sentiment categories along with the actual values of polarity, subjectivity, and similarity. We compared the value of feature extraction that was rated ‘yes’ or ‘no’ by respondents to show their assessment on the relevance value, the influence in the decision-making of adding new features, and the verbatim addition of the outcomes as new features. In Figure 5, we compared the polarity, subjectivity, and similarity values of the relevant features, prompt new requirements, and verbatim reuse of new requirements. Based on Table 13 and Figure 5, an analysis of the feature extraction sentiment analysis was carried out. The ‘yes’ response tends to have a higher polarity value for features extracted compared to the ‘no’ responses in all three categories. However, this does not apply in the high similarity category. High similarity features tend to have a polarity value of 0.06–0.187 (having a positive sentiment value but close to a neutral value). Additionally, there was a slight shift in the polarity value towards a more neutral value for the positive sentiment, negative sentiment, and high similarity when viewed regarding the relevance, the prompting of new requirements, and the addition of features as new requirements. For example, in the positive sentiment group, the polarity value for the relevance was 0.731, and this was lower at 0.715 in the prompting new requirement category.



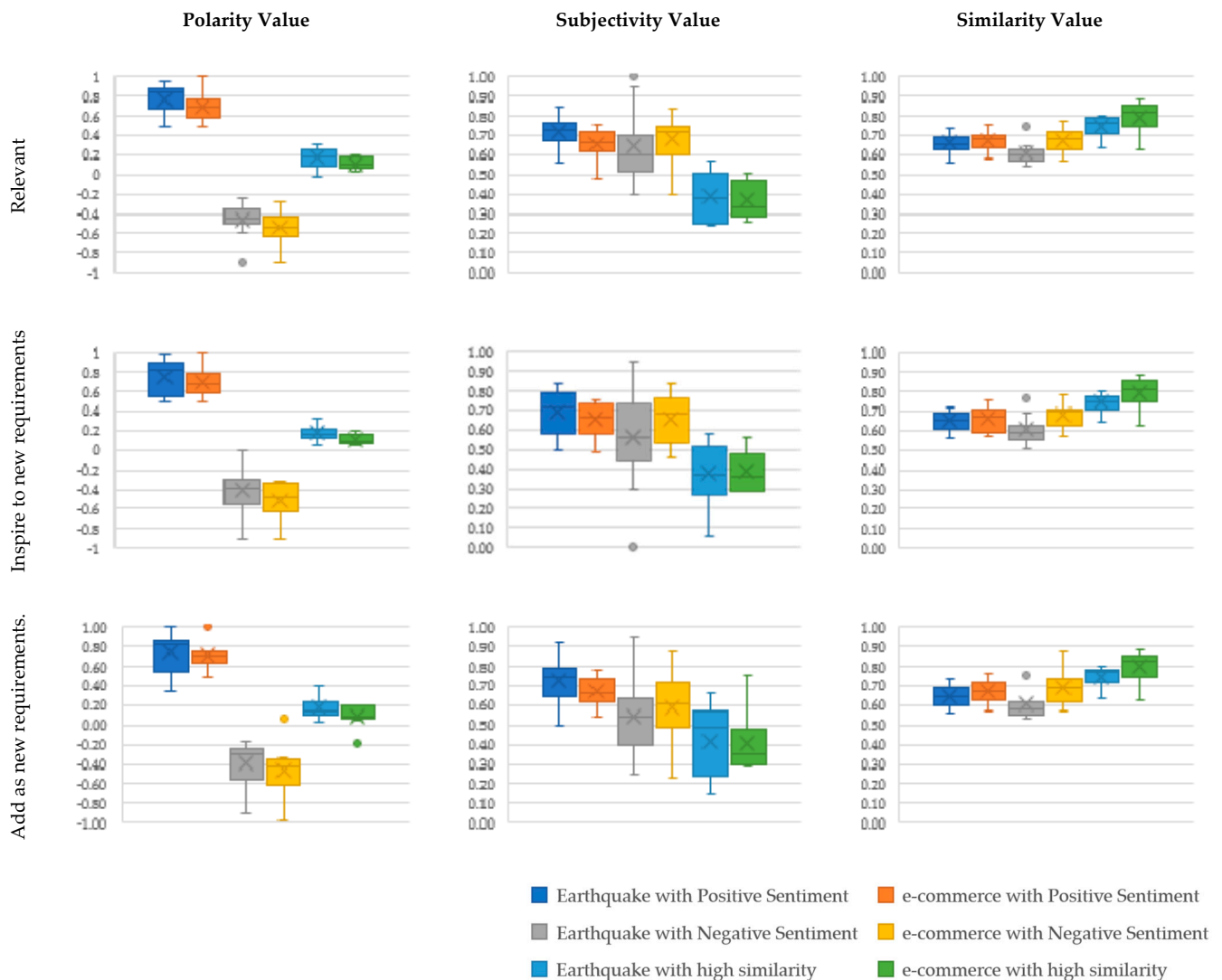
**Figure 4.** Comparison of the percentage of extracted features whether the results are relevant, prompt new requirements, and determine the addition of requirements verbatim from the feature extraction by using earthquake and e-commerce case studies (a) positive sentiment; (b) negative sentiment; (c) high similarity.

**Table 13.** Results of the questionnaire.

			Positive Sentiment	Negative Sentiment	High Similarity
Polarity	Relevance	Yes	0.731	−0.500	0.141
		No	0.672	−0.547	0.060
	Inspire new features	Yes	0.715	−0.488	0.132
		No	0.680	−0.590	0.187
	Add as a feature	Yes	0.721	−0.433	0.129
		No	0.717	−0.575	0.181
Subjectivity	Relevance	Yes	0.685	0.658	0.381
		No	0.599	0.659	0.455
	Inspire new features	Yes	0.677	0.637	0.399
		No	0.638	0.687	0.305
	Add as a feature	Yes	0.698	0.566	0.412
		No	0.651	0.712	0.354
Similarity	Relevance	Yes	0.664	0.638	0.769
		No	0.623	0.630	0.802
	Inspire new features	Yes	0.657	0.640	0.766
		No	0.666	0.635	0.793
	Add as a feature	Yes	0.658	0.647	0.772
		No	0.668	0.633	0.782

There was no clear pattern found in terms of relationship between relevance, prompting new requirements, and adding features verbatim as new requirements in the subjectivity value. However, the high similarity group's value had a higher objectivity value than the positive and negative sentiment groups.

Meanwhile, the comparison of similarity values does not have a significant difference. For example, the relevance value of high similarity was 0.769, while the similarity value for the positive and negative sentiment group was 0.664 and 0.638, respectively. This can be used as a basis to determine a similarity threshold value for the requirement reuse from positive and negative sentiments.



**Figure 5.** The relationship between polarity, subjectivity, and similarity of extracted features between case studies can be compared using this figure: For RQ1, the correlation between feature extraction and sentiment analysis in terms of acquiring relevant requirements; for RQ2, the correlation between feature extraction and sentiment analysis in informing system analysts in the decision-making of new requirements; and for RQ3, the correlation between feature extraction and sentiment analysis pertaining to verbatim reuse of extracted features as new requirements.

## 4. Discussion

### 4.1. Findings Related to the Research Question

The following are the answers to the research questions.

#### 4.1.1. RQ1: What Is the Correlation between Feature Extraction and Sentiment Analysis in Terms of Acquiring Relevant Requirements?

The features with positive sentiments, negative sentiments, and high similarity have a high relevance level based on the questionnaire results; this means that all extracted features are relevant to the initial requirements. This is because all extracted features have been filtered by applying a specific similarity threshold. The average similarity value for all extracted features is above 0.5. The average subjectivity value between features with positive and negative sentiment is at least above 0.5. Meanwhile, the subjectivity value of features with high similarity is consistently below 0.5. This means that the high similarity feature has better objectivity than the features with positive and negative sentiments.

#### 4.1.2. RQ2: What Is the Correlation between Feature Extraction and Sentiment Analysis in Informing System Analysts in the Decision-Making of New Requirements?

This research question means whether the extracted features can inspire new ideas for app features. As expected, the value is lower in comparison with the relevance value. Features with positive sentiment and high similarity experienced an expected decline, but negative sentiment features experienced a drastic decline. The average value of similarity did not change significantly compared to RQ1. Meanwhile, the average subjectivity value has a broader range than RQ1, which indicates a higher value of subjectivity, especially for the positive and negative sentiment category.

#### 4.1.3. RQ3: What Is the Correlation between Feature Extraction and Sentiment Analysis Pertaining to Verbatim Reuse of Extracted Features as New Requirements?

This question seeks whether or not the systems analyst would be willing to reuse the extracted features to revise the requirement document. For positive and negative sentiment categories, the percentage results are lower compared to RQ2. Most the features with negative sentiment score considerably lower, below 50%. Likewise, the features with positive sentiment also score lower, although not as significant as that of the negative sentiment. Meanwhile, the high similarity feature has a consistent value compared to the RQ2. The subjectivity and similarity have the same pattern as RQ2.

### 4.2. Main Finding

The main finding from this research show that features with a high similarity outperformed the positive and negative sentiments in terms of acquiring relevant features. The high similarity category has a consistent value of above 70% for relevancy; the positive sentiment group can only offset this result. The value is lower in the prompting new requirements category and in the category of using the features verbatim as new requirement. Meanwhile, the value for the negative sentiment category is deficient, which means that although it is possible to obtain relevant features based on negative sentiment, it is unlikely to produce significant results. However, it should be noted that the results of feature extraction can have negative or positive sentiment values; this applies to the relevant feature category, prompting a similar new feature category, and adding verbatim as a new feature category. To obtain the best results, filtration with a certain similarity threshold is recommended, then, categorizing them under high similarity, positive sentiment, and negative sentiment. Thus, feature extraction for the requirement reuse that considers polarity and subjectivity values can be shown to the analysis system. However, since each project is unique, the primary determinant of feature reuse should involve human intervention, hence comparing the results with the system analysts' assessments.

### 4.3. Related Studies

Based on the previous studies, user reviews can provide excellent feedback. User reviews may contain appreciations, bug reports, and new feature requests, which could be useful for a software and requirements engineering development [27]. User reviews in software development can be used in many ways. Jiang et al. [24] used online reviews to conduct requirements elicitation. Guzman et al. [26] used user reviews to analyze an

application feature systematically. Bakar et al. [18] extracted software user reviews to reuse software features. Keertipati et al. [25] extracted application reviews to prioritize feature improvements. In this study, user reviews are used to assist systems analysts in extending the scope of the existing software requirements.

Previous research analyzes in more depth the opinions or sentiments of user reviews. Jiang et al. [24] adopted a mining technique to extract opinions about software features. Keertipati et al. [25] found how negative sentiment in software features helped developers prioritize features that needed improvement. Meanwhile, in this study, we provide subjectivity and polarity scores for each extracted feature. The system analysts can choose which features to be used for requirements reuse to expand existing software requirements. Table 14 shows a comparison of this study with related research.

**Table 14.** Comparison with related research.

	Jiang, et al. [24]	Guzman et al. [26]	Bakar, et al. [18]	Keertipati et al. [25]	Proposed Method
Objective	Use online reviews to requirements elicitation	Use user reviews to help app developers analyze user opinions about features.	Software features reuse from user reviews.	App reviews to prioritize feature enhancements using sentiment analysis.	Extending the scope of existing software requirements using sentiment analysis.
Input	User reviews	User reviews	User reviews, Legacy requirements, and Product descriptions	User reviews	User reviews and a draft of software requirements The Requirements report contains a list of software features along with the sentiment value and similarity attributes.
Output	Evolutionary Requirements Document	High-level features with sentiment score	software feature	List of features that need to be improved.	
Feature Extraction Techniques	Expanded version of Syntactic relation-based propagation approach (SRPA+)	Natural Language Processing (NLP) collocation finding, and Latent Dirichlet Allocation (LDA) topic modelling.	Feature Extraction for Reuse of Natural Language requirements (FENL)	Feature identification using POS Tag and N-gram	Software feature extraction using POS-Tag and Chunking
Sentiment Analysis Techniques	Using existing sentiment words rule to assign polarity	Using SentiStrength	-	Using LIWC dictionary to analyze emotion	Using sentiment feature in Text Blob

#### 4.4. Thread to Validity

The results of our study have limitations and should be considered within the study context. The extracted software features have low precision, recall, and F-measure values, which has been a problem in many studies. For this reason, we used selected features for the respondents' data to determine relevance, inspiring new features, and adding them verbatim as new features.

We only measured a few samples from user reviews, which precludes us from filtering the extracted software features based on their importance level. Another aspect to note is the manual tagging done by the system analysts, which will vary when carried out in other studies and by different people. This is because the definition of software feature will differ from one to another depending on the research objectives. In our study, this issue may impact the software feature's extraction results, which are evaluated by the experts to measure the relevancy. To solve this issue, in this study, we selected the ten best extraction results with positive and negative sentiments, and with the best similarity compared to the existing requirements.

To some extent, this study's results still rely on human involvement as the main determining factor in software feature selection. The domain may also become a limitation as different domains will have different results. Finally, this study uses only two case

studies with different domains. The results are similar but it cannot be guaranteed that the results will be the same in other domains.

## 5. Conclusions

In this research, we performed software feature extraction from user reviews to extend the existing software requirements by using sentiment analysis. The extraction results were grouped based on the polarity, subjectivity, and similarity value. We evaluated the correlation between software feature extraction results with regards to the polarity, subjectivity, and similarity. This was done to enhance the requirements elicitation process.

Both extracted features with positive and negative sentiments can be used as a requirement reuse, primarily to expand the existing requirements. However, to obtain the best results, we recommend that all extracted features are filtered based on the similarity value by using a certain threshold. This is crucial in filtering out features that are not relevant to the existing requirements. From this study, it can be concluded that features that have positive or negative sentiments can be used to acquire the relevant requirements, to inform the system analysts in determining new requirements, and reuse the extraction results verbatim as new requirements. However, the positive sentiment group yields a better performance than the negative sentiment group. In fact, the positive sentiment group has a value close to the features with high similarity. The primary source for requirement reuse is the feature extraction with a high similarity value. However, this does not mean we should overlook the feature extraction based on the positive and negative sentiments because it still provides the relevant information to help the determination of requirement reuse.

Further research should be undertaken to investigate the specific roles of the sentiment analysis in the feature extraction and analysis. This includes the semantic meaning of a feature, such as whether a feature has negative sentiment when the feature is not working, incomplete, or has other issues.

**Author Contributions:** I.K.R.: Conceptualization, methodology, writing—original draft, writing—review and editing, supervision. V.A.: Software, investigation, data curation, writing—original draft. B.Z.: Methodology, writing—review and editing, supervision. A.J.: Writing—review and editing. S.S.M.F.: Writing—review and editing. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by Universitas Airlangga under Penelitian Unggulan Fakultas Grand number 346/UN3/2020.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The codes and dataset are available at <https://github.com/indrakharisma/SoftFeatExt-SentAnt> (accessed on 3 March 2021).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Zowghi, D.; Coulin, C. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In *Engineering and Managing Software Requirements*; Springer: Berlin/Heidelberg, Germany, 2006.
2. Al-Alshaikh, H.A.; Mirza, A.A.; Alsalamah, H.A. Extended Rationale-Based Model for Tacit Knowledge Elicitation in Requirements Elicitation Context. *IEEE Access* **2020**, *8*, 60801–60810. [[CrossRef](#)]
3. Canedo, E.D.; Mendes, B.C. Software Requirements Classification Using Machine Learning Algorithms. *Entropy* **2020**, *22*, 1057. [[CrossRef](#)] [[PubMed](#)]
4. Pohl, K.; Rupp, C. *Requirements Engineering Fundamentals*, 2nd ed.; Rocky Nook Inc.: San Rafael, CA, USA, 2015; ISBN 9781937538774.
5. Dewi, M.R.; Raharjana, I.K.; Siahaan, D.; Fatichah, C. Software Requirement-Related Information Extraction from Online News using Domain Specificity for Requirements Elicitation. In Proceedings of the 10th International Conference on Software and Computer Applications (ICSCA 2021), Kuala Lumpur, Malaysia, 23–26 February 2021.
6. Carrizo, D.; Dieste, O.; Juristo, N. Systematizing requirements elicitation technique selection. *Inf. Softw. Technol.* **2014**, *56*, 644–669. [[CrossRef](#)]



7. Bourque, P.; Fairley, R.E.; Society, I.C. *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0)*, 3rd ed.; IEEE Computer Society Press: Washington, DC, USA, 2014; ISBN 9780769551661.
8. Suali, A.J.; Fauzi, S.S.M.; Nasir, M.H.N.M.; Sobri, W.A.W.M.; Raharjana, I.K. Software quality measurement in software engineering project: A systematic literature review. *J. Theor. Appl. Inf. Technol.* **2019**, *97*, 918–929.
9. Liu, L.; Zhou, Q.; Liu, J.; Cao, Z. Requirements cybernetics: Elicitation based on user behavioral data. *J. Syst. Softw.* **2017**, *124*, 187–194. [[CrossRef](#)]
10. Raharjana, I.K.; Ibadillah, I.; Hariyanti, E. Incident and Service Request Management for Academic Information System based on COBIT. In Proceedings of the 2018 5th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI 2018), Malang, Indonesia, 16–18 October 2018.
11. Ferrari, A.; Spoletini, P.; Gnesi, S. Ambiguity and tacit knowledge in requirements elicitation interviews. *Requir. Eng.* **2016**, *21*, 333–355. [[CrossRef](#)]
12. Sari, D.A.P.; Putri, A.Y.; Hanggareni, M.; Anjani, A.; Siswondo, M.L.O.; Raharjana, I.K. Crowdsourcing as a tool to elicit software requirements. *AIP Conf. Proc.* **2021**, *2329*, 050001. [[CrossRef](#)]
13. Raharjana, I.K.; Harris, F.; Justitia, A. Tool for Generating Behavior-Driven Development Test-Cases. *J. Inf. Syst. Eng. Bus. Intell.* **2020**, *6*, 27–36. [[CrossRef](#)]
14. Khusidman, V.; Bridgeland, D.M. A Classification Framework for Software Reuse. *J. Object Technol.* **2006**, *5*, 43. [[CrossRef](#)]
15. Acher, M.; Cleve, A.; Perrouin, G.; Heymans, P.; Vanbeneden, C.; Collet, P.; Lahire, P. On extracting feature models from product descriptions. In Proceedings of the Sixth International Workshop on Computing Education Research, ICER 2010, Aarhus, Denmark, 9–10 August 2010.
16. Davril, J.-M.; Delfosse, E.; Hariri, N.; Acher, M.; Cleland-Huang, J.; Heymans, P. Feature Model Extraction from Large Collections of Informal Product Descriptions. In Proceedings of the Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering; Association for Computing Machinery: New York, NY, USA, 2013; pp. 290–300.
17. Johann, T.; Stanik, C.; Maalej, W. SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews. In Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 4–8 September 2017; pp. 21–30.
18. Bakar, N.H.; Kasirun, Z.M.; Salleh, N.; Jalab, H.A. Extracting features from online software reviews to aid requirements reuse. *Appl. Soft Comput.* **2016**, *49*, 1297–1315. [[CrossRef](#)]
19. Putri, D.G.P.; Siahaan, D.O. Software feature extraction using infrequent feature extraction. In Proceedings of the 2016 6th International Annual Engineering Seminar (InAES), Yogyakarta, Indonesia, 1–3 August 2016.
20. Puspaningrum, A.; Siahaan, D.; Fatichah, C. Mobile App Review Labeling Using LDA Similarity and Term Frequency-Inverse Cluster Frequency (TF-ICF). In Proceedings of the 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), Bali, Indonesia, 24–26 July 2018.
21. Guzman, E.; Alkadhi, R.; Seyff, N. A Needle in a Haystack: What Do Twitter Users Say about Software? In Proceedings of the 2016 IEEE 24th International Requirements Engineering Conference (RE), Beijing, China, 12–16 September 2016; pp. 96–105.
22. Nayebi, M.; Cho, H.; Ruhe, G. App store mining is not enough for app improvement. *Empir. Softw. Eng.* **2018**, *23*, 2764–2794. [[CrossRef](#)]
23. Raharjana, I.K.; Siahaan, D.; Fatichah, C. User Story Extraction from Online News for Software Requirements Elicitation: A Conceptual Model. In Proceedings of the 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE), Chonburi, Thailand, 10–12 July 2019.
24. Jiang, W.; Ruan, H.; Zhang, L.; Lew, P.; Jiang, J. For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews. *Lect. Notes Comput. Sci.* **2014**, *8444 LNAI*, 584–595. [[CrossRef](#)]
25. 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 Intelligent User Interfaces, Sonoma, CA, USA, 7–10 March 2016.
26. Guzman, E.; Maalej, W. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In Proceedings of the 2014 IEEE 22nd International Requirements Engineering Conference (RE), Karlskrona, Sweden, 25–29 August 2014; pp. 153–162.
27. Pagano, D.; Maalej, W. User feedback in the appstore: An empirical study. In Proceedings of the 21st IEEE International Requirements Engineering Conference (RE), Rio de Janeiro, Brazil, 15–19 July 2013.
28. Raharjana, I.K.; Siahaan, D.; Fatichah, C. User Stories and Natural Language Processing: A Systematic Literature Review. Under review, **2021**.
29. Prastyo, P.H.; Sumi, A.S.; Dian, A.W.; Permanasari, A.E. Tweets Responding to the Indonesian Government’s Handling of COVID-19: Sentiment Analysis Using SVM with Normalized Poly Kernel. *J. Inf. Syst. Eng. Bus. Intell.* **2020**, *6*, 112–122. [[CrossRef](#)]
30. Rintyarna, B.S.; Sarno, R.; Fatichah, C. Semantic Features for Optimizing Supervised Approach of Sentiment Analysis on Product Reviews. *Computers* **2019**, *8*, 55. [[CrossRef](#)]
31. Zhang, L.; Hua, K.; Wang, H.; Qian, G. Sentiment Analysis on Reviews of Mobile Users. *Procedia Comput. Sci.* **2014**, *34*, 458–465. [[CrossRef](#)]
32. Panichella, S.; Di Sorbo, A.; Guzman, E.; Visaggio, C.A.; Canfora, G.; Gall, H.C. How can I improve my app? Classifying user reviews for software maintenance and evolution. In Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, Germany, 29 September–1 October 2015; pp. 281–290.

33. Chen, N.; Lin, J.; Hoi, S.C.H.; Chen, N.; Lin, J.; Hoi, S.C.H.; Xiao, X.; Zhang, B. AR-Miner: Mining informative reviews for developers from mobile app marketplace. In Proceedings of the ICSE 2014 the 36th International Conference on Software Engineering, Hyderabad, India, 31 May 31–7 June 2014.
34. Bird, S. NLTK: The Natural Language Toolkit. In Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions, Sydney, Australia, 18 July 2006; pp. 69–72.
35. Hasan, A.; Moin, S.; Karim, A.; Shamshirband, S. Machine Learning-Based Sentiment Analysis for Twitter Accounts. *Math. Comput. Appl.* **2018**, *23*, 11. [[CrossRef](#)]
36. Taylor, A.; Marcus, M.; Santorini, B. *The Penn Treebank: An Overview*; Word Sense Disambiguation: San Francisco, CA, USA, 2003.
37. Montani, I.; Honnibal, M.; Honnibal, M.; Landeghem, S.; Van Boyd, A.; Peters, H.; Samsonov, M.; Geovedi, J.; Regan, J.; Orosz, G.; et al. Explosion/spaCy: v3.0.3: Bug Fixes for Sentence Segmentation and Configure Filling 2021. Available online: <https://newreleases.io/project/github/explosion/spaCy/release/v3.0.3> (accessed on 1 March 2021).
38. Srinivasa-Desikan, B. *Natural Language Processing and Computational Linguistics: A Practical Guide to Text Analysis with Python, Gensim, Spacy, and Keras*; Packt Publishing Ltd.: Birmingham, UK, 2018.