Feature extraction from app reviews in google play store by considering infrequent feature and app description

IOP Conf. Series: Journal of Physics: Conf. Series **1230** (2019) 012007 doi:10.1088/1742-6596/1230/1/012007

Feature extraction from app reviews in google play store by considering infrequent feature and app description

Q L Sutino, D O Siahaan

Informatics Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Email: qonitaluthfia@gmail.com

Abstract. Google Play Store is one of the platforms used for distributing various kinds of mobile app from the developer to the users. Through this platform, users are allowed to give their comments about the app. These user reviews could be used to extract potential app's requirements. They are important information for developer to further develop the app. There have been some previous researches about extracting mobile app features which are frequently mentioned in user reviews. There are less researches that focus on extracting infrequent features. Nevertheless, extracting infrequent features is also important. It is because there is a possibility that important needs contained in the review which are not extracted as frequent features. One of the challenges in infrequent feature extraction was the irrelevant features contained in extracted features. To overcome the problem, this study aims to extract app frequent feature in reviews by finding collocation and infrequent feature in reviews based on dependency as extraction rules. Afterward, it compares the similarity of all the extracted features from review with features from app description. The implied technique of similarity measure includes similarity of 1) single-term by matching each term of feature, 2) synonym referring to WordNet synsets, and 3) sentence based on calculation of lexical semantic vector and cosine similarity. The implementation result is evaluated using precision and recall calculations. The result shows that features extracted by proposed method are more relevant than previous method.

1. Introduction

Nowadays, the development of mobile app is growing rapidly. At the same time, the maintenance of the app is required to conform users need. To maintain the app, developer need to discover which feature of app that need to be improved or added. Google Play Store is one of platform where the user is able to comment about an app. Moreover, in this platform, the developer can describe the app and find out the user need based on the user review. However, there might be a lot of reviews given by a lot of users. It means, there is more effort for developer to check the user reviews one by one. To overcome this problem, the app reviews need to be extracted so that the developer can get the app feature efficiently.

Previously, there have been studies about feature extraction from app reviews, like MARK [1] and CrossMiner [2], but those studies focus on extracting feature as keyword utilized for searching in reviews. Meanwhile, SURF [3], PUMA [4], CLAP [5], AR-Miner [6], and [7] have been proposed to extract more brief information from app review and group it into some topics. Some researches that are particularly in the field of extracting app features from review are MARA [8] that focuses on extracting app feature which is only described as feature request based on user review text and metadata and [9] that focuses on extracting features which are classified to bug reports, feature

requests, user experiences, and ratings. Other researches that also concentrates on extracting app feature are [10] and [11]. These studies consider sentiment aspect in review to extract the app features.

Different with those researches, this study focuses on extracting mobile app feature from review by considering infrequent feature. Infrequent feature is obtained from existing review that has not result any app features from the extracting process. It implies that the feature extracting process is twice. This infrequent feature extraction method is referenced from [12]. But, the result shows that some of extracted features are irrelevant. Besides, [13] extracts mobile app feature which is frequent by considering the similarity between extracted feature from review with extracted feature from app description. The result indicates that the method delivers more relevant features. Therefore, this research aims to extract more relevant features by combining both study methods. We propose extraction of mobile app feature from review considering infrequent ones and the similarity between extracted features from review and from description. This extraction method is depicted in Figure 1 and explained in the following sections.

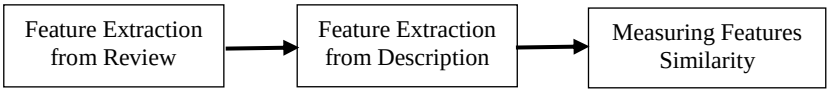

Figure 1. Feature Extraction Method.

2. Feature Extraction from Review

The method for extracting feature from review is composed of three processes listed as follows:

2.1. Frequent Feature Extraction

This process aims to generate app feature from user review by finding collocation. In English, collocation can be defined as a group of words that often exist together and the meaning of it might be different if the words are being apart. However, before searching the collocation, the review text is being processed first.

The pre-processing is made up of these four techniques. First is tag Part-of-Speech (POS). The POS used here is only the POS that might indicate a feature (i.e., noun, verb, and adjective). Second is eliminate stopwords. Stopwords are terms that have no significant meaning like '*the*', '*an*', etc. Third is correct spelling. Because, the characteristic of review text tends to be free and informal, checking the spelling in review text is necessary. Fourth, there is lemmatization using wordnet. This technique is implemented to lessen the variant of words in review by grouping semantically similar words.

After obtaining the outcome of pre-processing, then the searching of collocation is performed. In this research, the number of words used as the collocation is two and the distance between each word is up to three words. The found collocation is delivered according to likelihood ratio test [14].

2.2. Mapping Feature

Every feature yielded from above process is mapped to its review. The review that does not have any extracted feature mapped is being processed again to make sure whether there is any significant feature missed to be extracted. The process is called infrequent feature extraction, so that the result of this extraction is infrequent feature.

2.3. Infrequent Feature Extraction

The method executed in this re-extracting step is different from method used in the frequent feature extraction. For delivering infrequent feature, linguistic pattern enables uncommon feature identified, because it is based on the structure of language [10]. In this study, infrequent feature is extracted using extraction rules which are generated from dependency analysis of linguistic patterns denoting a feature [12]. The extraction rules are listed in Table 1.

Table 1. The Extraction Rules.

No.	Rules
1	*nsubj(t,h)*, where *t* has dependency as
	a. *dobj(t, n)* , the feature: '*t n*'
	b. *aux(t, n1)* and *neg(t, n2)*, the feature: '*t*'
2	*nsubjpass(t,h)*, where *t* has dependency as *aux(t, n1)* or *neg(t, n2)*, the feature: '*h*'
3	No subject dependency
	a. *dobj(t,h)*, where *t* has dependency as *amod(t, n1)* or *advmod(t, n2)*, the feature: '*t h*'
	b. *aux(t,n1)*, where *t* has dependency as *neg(t, n2)* or *advmod(t, n2)*, the feature: '*t*'
4	'No' + noun or noun phrase ~ *neg(t, h)*, where *h* = 'no' and there is
	a. *mark(m, n)*, the feature: '*m*'
	b. no mark dependency, the feature: '*t*'

After delivering features based on extraction rules above, it inspects whether the feature *f* has dependency as *conj(f, c)*. If there is conjunction dependency of *f*, the *c* is extracted as feature too. Additionally, to normalize the infrequent features, there are two processes of pruning the extracted features. These pruning methods are compactness and redundancy. Compactness method is intended to remove 2-words-feature which is irrelevant according to the interval between these 2 words. If the interval is more than 3 words, the features is considered irrelevant. Besides, the purpose of implementing redundancy pruning is to eliminate 1-word-feature by counting the p-support of the feature. P-support is a value representing the number of the sentence that contains feature *f* as noun or noun phrase. Nevertheless, this sentence must not contain any superset of feature *f*. For example, there are feature *f* '*sound*' and '*sound setting*', as superset of '*sound*'. The sentence is counted as p-support if the sentence contains '*sound*' and does not contain '*sound setting*'. The feature is not pruned if the p-support is more than or equal 3.

3. Feature Extraction from Description

This part depicts the method utilized for extracting app feature from description. Based on method implemented in [13] for extracting features in description text, SAFE (Simple Approach for Feature Extraction) method includes two main steps which are pre-processing and implementation of SAFE patterns. Each of the steps is described as follows.

3.1. Pre-processing Step

Before performing feature extraction, the description text is being processed in pre-processing step. This step includes some text processing techniques. First, tokenization used for transforming sentence become some chunks (words). Second, explanation removal. In description text, the existence of brackets often indicates additional information which is not necessary. Because of that, the brackets and the information contained in brackets needs to be removed. Third, noise filtering. URL, email address, and quotation are considered as noises so that the sentence that has this kind of noise is eliminated. Besides, if a sentence has bullet points or symbols (e.g., '*' or '#'), this bullet or symbol is removed, but the sentence is remained. Fourth, subordinate removal. It is stated that the subordinate clauses in description mostly do not denote any app features. Fifth, stopwords removal. Same with method applied in extracting frequent feature, the stopwords are removed, but for description extraction, the app name is also treated as stopwords. Lastly, POS Tag. Compared to the POS which is used in aforesaid review extraction technique, the POS tagged for this description extraction process is not limited to just noun, verb, or adjective.

3.2. Implementation of SAFE Patterns

The SAFE patterns are divided into two parts, there are sentence and POS patterns. The output of pre-processing is being processed by analysing the sentence pattern in the text so that it produces raw features. Afterwards, the other sentences which do not suit the sentence patterns and the yielded raw features are being analysed based on the patterns of POS. The detail of both patterns is drawn here.

3.2.1. Sentence Patterns. These patterns are based on enumeration, conjunction, and feature identifier in a sentence that often indicate the feature. These patterns are summed up into 5 cases. These cases are listed as follow.

- Case 1: If there is a conjunction, there might be feature identifiers in the left and right side of the conjunction
- Case 2: If there is a conjunction after a comma, there might be feature identifiers in the left of the comma and in the right of the conjunction
- Case 3: If there are two conjunctions, there might be four feature identifiers: 1) two in the left and right side of the first conjunction, 2) two in the left and right side of the second conjunction
- Case 4: If there are two conjunctions and a comma before the second conjunction, the feature identifiers might be: 1) in the left and right side of the first conjunction, 2) in the right side of the second conjunction, 3) in the left side of comma
- Case 5: If there are two conjunctions and a comma before every conjunction, the feature identifiers might be: 1) in the left of the comma, 2) in the right side of each conjunction

3.2.2. POS Patterns. The features existed in a description text can be found by detecting the pattern of POS that often emerge within the text and is indicated as feature. Based on POS Tag in Penn Treebank, these 18 POS Patterns are Noun-Noun, Verb-Noun, Adjective-Noun, Noun-Conjunction-Noun, Adjective-Noun-Noun, Noun-Noun-Noun, Verb-Pronoun-Noun, Verb-Noun-Noun, Verb-Adjective-Noun, Adjective-Adjective-Noun, Noun-Preposition-Noun, Verb-Determiner-Noun, Verb-Noun-Preposition-Noun, Adjective-Noun-Noun-Noun, Adjective-Conjunction-Adjective, Verb-Preposition-Adjective-Noun, Verb-Pronoun-Adjective-Noun, and Noun-Conjunction-Noun-Noun.

After generating features in accordance with the POS patterns, the output is being checked in case there is a feature with redundancy like '*sounds sounds*'.

4. Measuring Features Similarity

This section describes the method used for finding out that the features extracted from app review is conformed the features extracted from app description. This method is composed of similarity matching techniques. The techniques are divided into three steps: single-term-based similarity, synonym-based similarity, and sentence-based similarity.

4.1. Single-term-based Similarity

For this technique, the measurement of similarity between feature from review and feature from description is based on every term composes both features. In other words, the number of terms contained in each of features needs to be equal and the terms within a feature is similar with the terms within another feature. For instance, '*post status*' as a feature from review and *'status post*' as a feature from description are considered similar so that '*post status*' is extracted as final feature.

4.2. Synonym-based Similarity

If feature f_1 from review and feature f_2 from description consist of the same number of terms (f_1: (t_{11}, t_{12}) and f_2: (t_{21}, t_{22})), the similarity between each feature term will be assessed by the synonym similarity. As the example, f_1 is '*take picture*' and f_2 is '*get photo*', then f_1 is similar with f_2. It means, $f_2$is generated as final feature.

4.3. Sentence-based Similarity

This similarity measure technique prevails in determining similarity between features that have different number of terms. Corresponding to method proposed in [15], the similarity between extracted feature from review and description is calculated. This method uses semantic vector to measure the semantic similarity between sentences. The steps of this method are conveyed below.

- Join two sets of sentences (T_1 and T_2) become one-word set (T) without derive morphologically redundant word
- Arrange lexical semantic vector ($\check{s}_i$) is a vector which is delivered from the joint word set where *i* is the number of this vector entry which equals the number of words in T. This vector consists of semantic similarity between words in 1) T and T_1 2) T and T_2.
- Calculate the similarity between words in T and each of original sentence set which the words do not exist in both T and the original set. This similarity calculation is based on two functions which are path length between words ($f_1(l)$) and depth of the subsumer ($f_2(h)$). The formula is presented in equation (1) and (2) as follows:

$$s(w_1, w_2) = f_1(l) \cdot f_2(h) \tag{1}$$

$$s(w_1, w_2) = e^{-\alpha l} \cdot [(e^{\beta h} - e^{-\beta h}) \cdot (e^{\beta h} + e^{-\beta h})^{-1}] \tag{2}$$

  where the knowledge-based parameters $\alpha \in [0, 1]$ and $\beta \in (0, 1]$. The highest value is considered the most similar. Otherwise, the similarity is setted to 1 if the word appears in both sets.
- Before putting the similarity value in vector, the value is multiplied by probability of word in each of T and original set in Brown Corpus. To get this value, the formula is shown in equation (3) where n is the frequency of word w and N is the number of words in corpus.

$$I(w) = 1 - [\frac{log(n+1)}{log(N+1)}], \; I \in [0, 1] \tag{3}$$

- After all the values within vector completed, the vectors of T-T_1 and T-T_2 are being calculated according to cosine similarity formula that can be seen in equation (4).

$$S_s = \frac{s_1 \cdot s_2}{\|s_1\| \cdot \|s_2\|} \tag{4}$$

- The sentences are similar if the result of cosine similarity calculation ranges from 0,7 to 1.

In the end of the matching features process, there are the final features extracted from review which are relevant to the app description.

5. Experiment and Evaluation

5.1. Dataset

The methods explained above are implemented to the dataset that has been crawled from Google Play Store. This dataset consists of the app description and review. The total of app which the description and the review had been taken is 36 apps. It means, the number of descriptions is also 36. Besides, the total amount of reviews employed here is 1372.

5.2. Experiment

The experiment of this research is designed as illustrated in Figure 1. Each of steps have been described in Section 1, 2, and 3. Moreover, the program which the methods are applied is built in

Python using NLTK, except the method in extracting infrequent feature from reviews. The method for infrequent feature extraction which is using dependency analysis is built in Java using Stanford Dependency.

5.3. Evaluation

To validate the relevancy of extracted features, we calculate precision and recall of the results. The precision value represents extracted features that suit the evaluated features set. On the other hand, the recall value indicates elements of evaluated features set which are successfully extracted.

Every feature including in a review is assessed by an annotator. The average of precision and recall are delivered based on 1) frequent feature from using collocation finding method, 2) infrequent feature from implementing extraction rules, and 3) both frequent and infrequent features from using proposed extraction method that applies similarity measurement between extracted review features and description features. The detail of this evaluation is presented in Table 2.

Table 2. Evaluation Result.

App Category	Number of Apps	Number of Reviews	Precision			Recall			Number of Extracted Features		
			CM	ER	PM	CM	ER	PM	CM	ER	PM
Art & Design	4	129	0.22	0.23	0.25	0.21	0.18	0.22	1	9	5
Business	4	159	0.36	0.36	0.35	0.35	0.33	0.34	3	11	2
Comics	5	195	0.38	0.36	0.39	0.37	0.31	0.37	3	20	6
Education	3	120	0.47	0.41	0.47	0.47	0.40	0.46	0	4	2
Entertainment	4	151	0.32	0.33	0.35	0.32	0.31	0.34	0	12	6
Finance	3	117	0.36	0.36	0.36	0.36	0.34	0.35	0	6	2
Health & Fitness	5	181	0.27	0.25	0.26	0.26	0.23	0.25	3	9	3
Lifestyle	2	80	0.30	0.20	0.26	0.29	0.18	0.25	1	4	2
Personalization	3	120	0.25	0.23	0.23	0.24	0.22	0.22	2	4	2
Photography	3	120	0.26	0.26	0.28	0.25	0.24	0.26	1	10	7
Total			**0.32**	**0.30**	**0.32**	**0.31**	**0.28**	**0.31**	**14**	**89**	**37**

CM = Collocation Method.
ER = Extraction Rules.
PM = Proposed Method.

Based on the evaluation results, the precision and recall average of PM feature (i.e., 0.32 and 0.31) are higher than precision and recall average of ER feature (i.e., 0.30 and 0.28). It shows that the result of proposed method tends to have features which are more relevant than extraction rules method has. The precision and recall average of the frequent extraction method are relatively equal to the proposed method. It means that the proposed method would produce features that are as relevant as features produced by collocation method. Besides, the infrequent feature extraction method that implements extraction rules generates more features than collocation finding method. This characteristic of result is similar with [12] result. However, in this study, the extracted feature implementing extraction rules does not result more than 50% of relevant results.

6. Conclusion

To get more relevant features from the app reviews crawled from Google Play Store, this proposed method extracts not only the frequent features, but also the infrequent ones from reviews. By finding collocation in pre-processed reviews, it delivers frequent features. These features are mapped to the origin reviews and the unmapped reviews are employed as the input for infrequent features extraction. The infrequent is obtained by analysing the unmapped reviews that fit the extraction rules. The rules are based on dependency between words which often denote a feature. Moreover, all the extracted

features from review are compared according to the similarity with features in description. There are single-term, synonym, and sentence measure of features similarity where the sentence similarity is determined by lexical semantic vector and cosine similarity calculation. According to the evaluation results, the proposed method produces more relevant features than features extracted by implementing infrequent method only. It means, the similarity measure between extracted review features and features within description can filter the irrelevant features.

References

[1] Vu PM, Nguyen TT, Pham HV, Nguyen TT. 2015. Mining User Opinions in Mobile App Reviews : A Keyword-based Approach
[2] Man Y, Gao C, Lyu MR, Jiang J. 2016. Experience Report : Understanding Cross-Platform App Issues From User Reviews
[3] Sorbo A Di, Panichella S, Alexandru C V, Visaggio CA, Canfora G. 2017. SURF : Summarizer of User Reviews Feedback. *2017 IEEE/ACM 39th IEEE Int Conf Softw Eng Companion*. pp. 55–8
[4] Vu PM, Pham HV, Nguyen TT, Nguyen TT. 2016. Phrase-Based Extraction of User Opinions in Mobile App Reviews. *2016 31st IEEE/ACM Int Conf Autom Softw Eng*. pp. 726–31
[5] Villarroel L, Bavota G, Russo B, Oliveto R, Penta M Di. 2016. Release Planning of Mobile Apps Based on User Reviews. *2016 IEEE/ACM 38th Int Conf Softw Eng*
[6] Chen N, Lin J, Hoi SCH, Chen N, Lin J, Hoi SCH, et al. 2014. AR-Miner : Mining Informative Reviews for Developers from Mobile App Marketplace. *36th Int Conf Softw Eng*
[7] Panichella S, Di Sorbo A, Guzman E, Visaggio CA, Canfora G, Gall H. 2015. How Can I Improve My App ? Classifying User Reviews for Software Maintenance and Evolution. *IEEE Int Conf Softw Maint Evol*
[8] Iacob C, Harrison R. 2013. Retrieving and Analyzing Mobile Apps Feature Requests from Online Reviews. *2013 10th Work Conf Min Softw Repos*. pp. 41–4
[9] Maalej W, Nabil H. 2015. Bug Report , Feature Request , or Simply Praise ? On Automatically Classifying App Reviews. *2015 IEEE 23rd Int Requir Eng Conf*
[10] Guzman E. 2014. How Do Users Like This Feature ? A Fine Grained Sentiment Analysis of App Reviews. pp. 153–62
[11] Malik H, Shakshuki EM. 2016. Mining Collective Opinions for Comparison of Mobile Apps. *The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016)*. **94**. pp. 168–75
[12] Putri DGP, Siahaan DO. 2016. Software Feature Extraction using Infrequent Feature Extraction.
[13] Johann T, Stanik C, B AMA, Maalej W. 2017. SAFE : A S imple A pproach for F eature E xtraction from App Descriptions and App Reviews. pp. 21–30
[14] Manning CD, Manning CD, Schütze H. 1999. Foundations of statistical natural language processing. (Cambridge: MIT press)
[15] Li Y, McLean D, Bandar ZA, O'Shea JD, Crockett K. 2006. Sentence Similarity based on Semantic Nets and Corpus Statistics. *IEEE Trans Knowl Data Eng*. **18**. pp. 1138–50