Phrase-Based Extraction of User Opinions in Mobile App Reviews

Phong Minh Vu, Hung Viet Pham, Tam The Nguyen, Tung Thanh Nguyen
Computer Science Department
Utah State University
{phong.vu,hung.pham,tam.nguyen}@aggiemail.usu.edu
tung.nguyen@usu.edu

ABSTRACT

Mobile app reviews often contain useful user opinions like bug reports or suggestions. However, looking for those opinions manually in thousands of reviews is ineffective and time-consuming. In this paper, we propose PUMA, an automated, phrase-based approach to extract user opinions in app reviews. Our approach includes a technique to extract phrases in reviews using part-of-speech (PoS) templates; a technique to cluster phrases having similar meanings (each cluster is considered as a major user opinion); and a technique to monitor phrase clusters with negative sentiments for their outbreaks over time. We used PUMA to study two popular apps and found that it can reveal severe problems of those apps reported in their user reviews.

CCS Concepts

•**Software and its engineering** → *Software maintenance tools;*

Keywords

Opinion Mining; Review Analysis; Phrase Extraction

1. INTRODUCTION

Millions of mobile apps are available on app markets, making app development a highly competitive business. To attract, retain, and improve users' satisfaction, app developers are interested in user opinions about their apps. Fortunately, app markets allow users to rate and write reviews about the apps they use. Those user reviews often contain useful information such as reports of bugs, suggestions of new features or improvements, or simply praises or frustrations [2].

However, a popular app often receives thousands of reviews each day and most of them do not contain information useful for app developers [2]. Due to this large volume and noisy nature of such reviews, reading and analyzing them manually for relevant opinions is very ineffective and time-consuming. Therefore, researchers have proposed and developed many automated and semi-automated techniques to analyze app reviews [5, 8, 17].

Most current approaches represent user opinions using single words [2, 17]. For example, MARK detects and ranks keywords frequently appearing in reviews with negative ratings and allows users search for reviews with a specified set of keywords. AR-Miner clusters reviews into different topics using Latent Dirichlet Allocation (LDA) and Aspect and Sentiment Unification Model (ASUM). However, single words could only briefly and loosely describe semantics and sentiments of reviews. For example, if MARK reveals that keyword "connect" appears frequently in negative reviews, we could guess that users are complaining about some "connection" issues. A phrase like "cannot connect to Facebook" will inherently provide more useful information.

Therefore, in this paper, we propose PUMA, a novel approach to extract user opinions using phrases. PUMA considers a *phrase* as a sequence of consecutive words corresponding to a grammatically correct phrase, clause, or sentence in English. The noun phrase "Facebook Messenger", the verb phrase "kill my phone battery", or the sentence "the game app gets the wrong profile" are examples of phrases we aim to extract in app reviews.

We could extract phrases directly by parsing the reviews and collect word sequences correspondingly to phrase, clause, or sentence structures in the resulted parse trees. However, app reviews are often noisy, e.g. containing many typos and grammar errors. Thus direct parsing would be inefficient and non-scalable to large amount of reviews. To avoid this problem, PUMA uses Part-of-Speech (PoS) tags of words rather than parse trees to extract phrases in user reviews.

The PoS tag of a word in a given sentence indicates its functional type, e.g. whether it is used as a *noun* (NN), a *verb* (VB), a *modal verb* (MD), a *determiner* (DT), etc. Given a word sequence, PoS tagging is the task to infer the PoS tag for each of its words, which is clearly much easier than parsing it. For example, PoS tagging the sentence "my phone does not connect to Facebook" results in the PoS tag sequence "PRP$ NN MD NOT VB TO NN". Note that we use the POS tags definition from Stanford POStagger [4].

Let us explain how PUMA uses PoS tags to extract phrases via an example. First, we could observe that many valid noun phrases contain two consecutive nouns, such as "network connection" or "battery consumption". That means, they all have the tag sequence "NN NN". Thus, if a word sequence has the tag sequence "NN NN", like "account profile" or "SMS message", it is likely a valid noun phrase.

Our phrase extraction technique has two phrases. First,

ASE'16, September 3–7, 2016, Singapore, Singapore
© 2016 ACM. 978-1-4503-3845-5/16/09...$15.00
http://dx.doi.org/10.1145/2970276.2970365

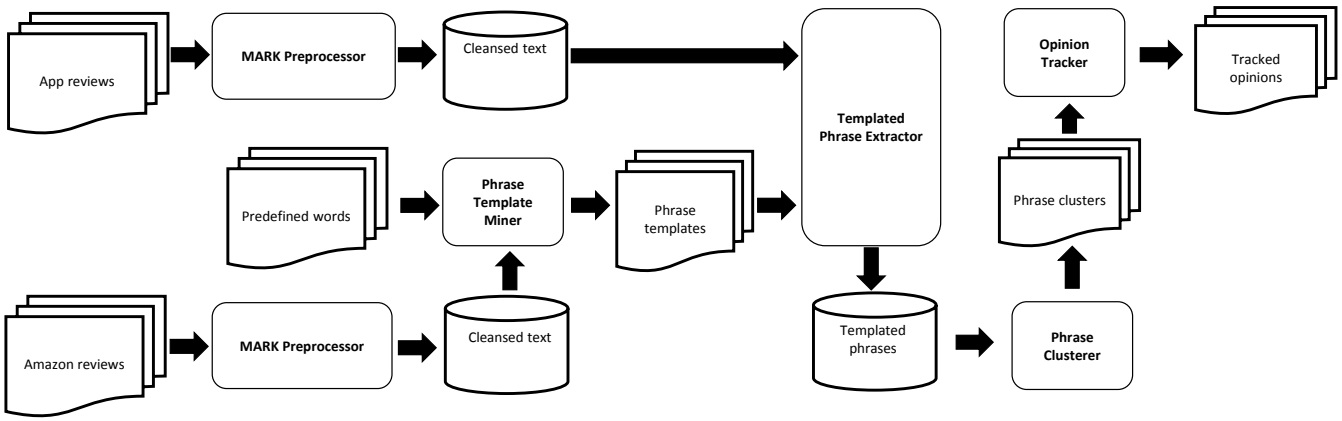

Figure 1: Flow of our Phrase-based approach to extract user opinions

Table 1: Example of phrases extracted from a review

Actual Phrases	Phrase Templates	Templated Phrases
can not connect to facebook	MD NOT VB TO NN	can not connect TO facebook
kill my phone battery in game	VB PRP$ NN NN IN NN	kill PRP$ phone battery IN game
game freeze	NN VBP	game freeze
facebook messenger and internet connection	NN NN and NN NN	facebook messenger AND internet connection
game app get the wrong profile	NN NN VBP DT JJ NN	game app get the wrong profile

we mine the *phrase templates*, which are tag sequences frequently appearing in valid phrases in a corpus of high quality product reviews from Amazon.com, like "NN NN" mentioned above. Then, we extract in mobile app reviews the *templated phrases*, i.e. word sequences whose their tag sequences match the mined *phrase templates*.

The same opinion can be expressed by different phrases having similar meanings. For example, "Internet connection" has similar meaning to "network access", or "connect to Internet". Thus, it would be more convenient to analysts of app reviews to group similar phrases. To do that in PUMA, we have proposed a new similarity measure for phrases based on the similarity of single words appearing in those phrases, which in turn is computed based on vector-based representation of those words (word embedding). Then we implemented a soft clustering algorithm to group phrases based such similarity measurements between them.

After extracting and clustering phrases, PUMA can rank the resulted clusters of similar phrases based on their associations with the ratings of reviews. Thus, it can discover phrases frequently appearing in negative reviews, which likely indicate some complaints or issues that make users unhappy. PUMA can also monitor the occurrences of resulted clusters of similar phrases over time. Thus, it can help app developers to identify issues or complaints that are currently affect a lot of users.

In short, our contributions in this work include:
1. A new phrase extraction approach using *phrase templates*
2. A new similarity measurement to estimate phrase similarity using vector representation of words.
3. New techniques to cluster, rank, and track groups of similar phrases overtime.

We organized the remaining of this paper the following. Section 2 describes our approach in details. Section 3 presents preliminary evaluation results of our approach. Section 4 discusses related work and section 5 concludes this paper.

2. APPROACH

In this section, we present our approach PUMA in details. Figure 1 illustrates its main components and data flow. As seen in the figure, PUMA mines *phrase template* from a corpus of high quality product reviews from Amazon.com. After mining, it fetches the raw app reviews from app markets and uses MARK tool [18] to pre-process them. After that, it extracts phrases in the pre-processed reviews using phrase templates that we have mined. Then, it clusters the extracted phrases into groups with similar meanings and consider each group to express a major opinion. Finally, PUMA rank and monitor those groups over time. We now discuss the details of each component.

2.1 Preprocessing

PUMA uses MARK [17] to pre-process both the raw app reviews and the product reviews from Amazon.com. MARK pre-processor has a customized word stemmer to convert English words into its base form without being over-stemmed. It can also correct errors, typos, slang words, acronyms and abbreviations typically appear in reviews of mobile apps.

2.2 Mining Phrase Templates

In our approach, a phrase template is a sequence of PoS tags corresponding to a large number of valid (i.e. grammatically correct) phrases in English, such as **"NN NN"** (noun phrases containing two nouns). To mine those phrase templates, we collected a corpus of 200,000 high quality reviews of products in Amazon.com [10]. We mine phrase template from this dataset instead of mining directly from reviews of mobile apps because we expect the product reviews in Amazon.com to be better in grammar and spelling (i.e. less noisy than mobile app reviews), while having the expressions and writing styles of user reviews.

After pre-processing the collected reviews, PUMA parsed them using Stanford Parser [4]. For each resulted parse tree,

it located the sub-tree for a phrases, a clause, or a sentence and counted its corresponding tag sequence. After counting, we only kept sequences that have at least 20 counts and have length of at most 8.

We observe in the review data that some functional words (e.g. connectors, intensifiers, negators) have the same POS tag but lead to very different meanings. For example, both "and" and "but" are tagged as Coordinating Conjunction (CC). Therefore, we obtained a list of functional words from several English websites[1][2][3] so we can keep those functional words instead of replacing them with their PoS tags. For example, **"bread and butter"** will correspond to the tag sequence **"NN and NN"**, not **"NN CC NN"**.

We also investigated the mined *phrase templates* manually and removed ones that obviously do not make sense, such as ending with **"but"** or **"and"**. Finally, we have more than 90,000 phrase templates. Although parsing Amazon reviews, counting tag sequences of valid phrases, and validating the resulted sequences manually are time-consuming, we only need to do it once.

2.3 Extracting Templated Phrases

Figure 2 illustrates the algorithm in PUMA to extract phrases using the mined phrase templates. It first separates a review into sentences. For each sentence, it uses a PoS tagger to produce the corresponding tag sequence. Then, it collects all word sequence in that sentence whose tag sequence matches a mined phrase template. If a word sequence is collected, all its sub-sequences are disregarded. In addition, sequences containing only stop words are also filtered because they would not express any useful opinion.

PUMA also performs some reductions to increase the generality of the extracted phrases. First, some modal verbs like "do", "can", "will", etc. would not contribute significantly to the meanings of phrases containing them. Thus, PUMA replaces them with their PoS tag "MD". As MARK, PUMA only keeps verbs, nouns, and additionally adjectives commonly used to describe user sentiments, e.g. intrusive, unreliable. All other words, e.g. possessive adjectives and pronouns like "my", "yours", "our", etc. are replaced by their PoS tags. Table 1 provides some examples of the phrase templates, the actual phrases in reviews and the final templated phrases after extraction and reductions.

2.4 Clustering Phrases

Users' opinions can change over time after each bug fix or update. Therefore, monitoring the major problems/opinions and discover new ones is an important task for app developers. The simplest way to monitor these opinions is to track the occurrences overtime of the phrases expressing such opinions. However, an opinion could be described in many ways, or in other words, many phrases could describe the same opinion. Because there are many phrases having the same meanings, thus, trying to list similar phrases manually for tracking would be time-consuming. It is better to extract phrases from reviews and cluster them into more coherent topics for tracking.

[1]http://www.myenglishteacher.eu/question/list-of-sentence-connectors-in-english/

[2]http://learnenglish.britishcouncil.org/en/english-grammar/adjectives/intensifiers

[3]http://dictionary.cambridge.org/us/grammar/british-grammar/negation

```
function ExtractTemplatedPhrase                                              1
  Input: T is a list of phrase templates                                     2
         R is a review                                                       3
         W is a list of predefined words                                     4
         L is the maximal length of extracted phrases                        5
  Output: P is a list of templated phrases                                   6
                                                                             7
  P = ∅                                                                      8
  S = extractSentence(R)                                                     9
  for each sentence s ∈ S                                                    10
    α = {sequence x ∈ s:length(x) ≤ L}                                       11
    β = {sequence x ∈ α:∃t ∈ T such that x matches with t}                   12
    Remove subsequences of larger sequences in β                             13
    Remove sequences containing only stop words in β                         14
    Add β to P                                                               15
  return P                                                                   16
```

Figure 2: Extraction algorithm for *Templated Phrase*

2.4.1 Phrase Similarity

Our clustering algorithm requires a reliable method to compute the similarity or distance between the templated phrases. However, we could not find a well-established technique in the literature for that task. Lin *et al.* [14] propose to use Point-wise Mutual Information to find the feature vector of each phrase based on their co-occurrence with other words in the same context. Mikolov *et al.* [15] proposed word2vec tool with a similar intention of finding context vector for words or phrases. However, both approaches suffer from the fact that our templated phrases can be too long (e.g. can contain up to 8 words) and the average length of a review can be very short (our dataset of 1.8 million reviews has an average of 11.24 words per review). Sometimes, the phrase can be as long as a whole sentence, or even a review. Thus, the methods of finding context based on co-occurrence of words is not applicable for our phrases.

To address this limitation, we propose our own technique to compute similarity between two phrases using vector representation of words from word2vec. Our technique considers two phrases as two bags of word of size m and n. First, it greedily finds the pairs of words δ_i in our phrases that has highest vector similarity (measured by cosine similarity). Similarity of two phrase would then be equal to the average similarity of these pairs times with the modified Jaccard similarity of the two bags of word. This Jaccard similarity considers all paired words to be in the intersection set instead of only identical words. This modification favors phrases of similar lengths. The following formula shows how we calculate such similarity:

$$phraseSim(\rho_a, \rho_b) = \frac{\sum_{i=1}^{k} \delta_i}{n + m - k}$$

k: number of word pairs δ

ρ_a, ρ_b: two phrases that have the number of word n and m, respectively.

δ: similarity of a pair of words.

2.4.2 Soft Clustering

The Soft Clustering algorithm uses the *templated phrase*'s vectors as shown in algorithm 3.

In each iteration, the algorithm does the following steps:

(1) Starting: choose a starting phrase with highest negative sentimental score in the pool of available phrases S. The negative sentimental score is the same *contrast score* we used in MARK [17]. The negative sentimental score can also be

```
function SimpleSoftCluster                                              1
  Input: S is a list of templated phrases                               2
         k is the number of clusters                                    3
         ρ is the releasing percentage                                  4
         μ is threshold for distance similarity                         5
  Output: P is a list of clusters                                       6
                                                                        7
  Sort S by negative sentimental score in DESC order                    8
  C = ∅                                                                 9
  δ = 0                                                                 10
  while(!isEmpty(S))                                                    11
    H := PopFirst(S)                                                    12
    β.add(H)                                                            13
    δ := centroid(β)                                                    14
    ω = 0                                                               15
    for each λ in S                                                     16
      c := CosineSimilarity(δ,λ)                                        17
      if(c >= ω)                                                        18
        ω = c                                                           19
        λ' = λ                                                          20
    if(omega >= μ)                                                      21
      β.add(λ')                                                         22
      δ := centroid(β)                                                  23
    if β.size() == k                                                    24
       break;                                                           25
    Release top ρ% Phrases closest to centroid in β from S              26
    add β to C                                                          27
  return P                                                              28
```

Figure 3: Simple Soft Clustering Algorithm for *Templated Phrase*

modified to positive sentimental score simply by changing negative ratings for positive ratings.

(2) Expanding: Choose a phrase L with the closest Cosine Similarity to the centroid of chosen phrases, given that the distance is no more than a threshold mu. Stop expanding when there is no phrase left to chose in the pool, or a K number of phrase in cluster is reached.

(3) Releasing: Release top δ percents of the chosen phrases closest to centroid from phrase pool for the next iteration. This step helps allocating phrases with different meanings in different context to all clusters they belongs, hence the name *Soft Clustering*.

Each of the final clusters represents a topic that users are talking about. We rank them by the sum of negative sentimental score of each phrase within them by month. This way, analysts can find out which problems have changed their ranking over time. Moreover, they can narrow down the list of reviews they need to read.

Our case studies in section 3.2 show that our tracking approach can catch real problems for the studied apps.

3. EVALUATION

In this section, we evaluate our phrase-based approach on the ability to track real-life problems. The first subsection discusses about our review data used for this evaluation. In the second subsection, we show 2 case studies resulted from our clustering and tracking of opinions.

3.1 Data

The dataset we prepared for our evaluation is showed on Table 2. We have crawled nearly three millions of reviews from 120 mobile apps on Google Play from January 1, 2015 to September 1, 2015. Each crawled review contains the creation time, the reviewer ID, a title, a long text description of the review content, and the associated rating. The crawling process is the same as in MARK [17]. We need all 120 apps for just two case studies is because the more apps we use, the more general templated phrases we can extract.

Table 2: Statistics of collected data

Total number of mobile apps	120
Total number of crawled reviews	2,944,335
Total number of reviews can be extracted for phrases	1,809,231
Min number of reviews (Amazon MP3)	1,001
Average number of reviews per app	15,076.925
Average words per review	11.24
Collection Time	Jan 1 to Sep 30, 2015

Table 3: Example of reviews from our case-studies

Candy Crush Saga	facebook issue. there have been several times now my king games wo not connect to my facebook after they have been connected. i have tried everything right down to uninstalling everything and installing it over again. this bug fix needs to be fixed in every king game not just this one.
Facebook Messenger	poorest app of all time. not many knows but this app constantly runs in background whether you log in or not. draining battery and hampering performance. this kind of app disappoint android os as a whole.

This data was passed to MARK for preprocessing, and then we extracted *templated phrases* on it. However, due to the messy nature of app reviews, even our MARK custom spelling correction could not correct all words. This left some strange misspelled words inside the cleansed reviews. Stanford POS tagger sometime label those words as foreign words or some other POS tags did not appeared in the Amazon review text dataset. Therefore, we were only able to extract *templated phrases* with a threshold of at least 3 occurrence on 1.8 millions reviews. This can be improved with a better spelling corrector and a more extensive set of *phrase templates* in the future. For the purpose of evaluating our approach, we used this 1.8 millions reviews dataset through out this evaluations.

3.2 Case Studies For Problem Tracking

By applying the opinion tracking technique in section 2.4, we were able to discover many interesting clusters for various mobile apps. Further investigations on 2 problems discovered by our technique for 2 different categories of app showed us that our technique can discover real major user opinions/problems. The following sub-sections shows our finding for *Candy Crush Saga*, *Facebook Messenger*:

3.2.1 Facebook Connection in Candy Crush Saga

We detected a cluster of phrases related to the complaint "cannot connect to Facebook" that has an abnormal increase in frequency on February, 2015 (Figure 4. Some example phrases are shown in Table 4). Investigation on this complaint showed that starting from February 1, 2015, all games from King (developers of Candy Crush Saga) cannot connect to the login system using Facebook [4] [5]. Since this happened to all games, not just this game, it could be a back-end systems problem. Even though we have never got an official announcement from King regarding this problem. The number of complaints was going down on the next month, which was likely because developers had fixed it.

3.2.2 Battery consumption in Facebook Messenger

Similar to our finding in prior work MARK [17], we also found a cluster dedicated to "battery draining" problem on

[4]http://www.isitdownrightnow.com/king.com.html
[5]http://www.product-reviews.net/2015/02/01/candy-crush-problems-as-app-cant-connect-to-facebook/

Table 4: Examples of phrases extracted from reviews of Candy Crush Saga belong to the cluster of facebook connection problem

connection to facebook
access to facebook
my facebook account
let me connect
no facebook connect
...

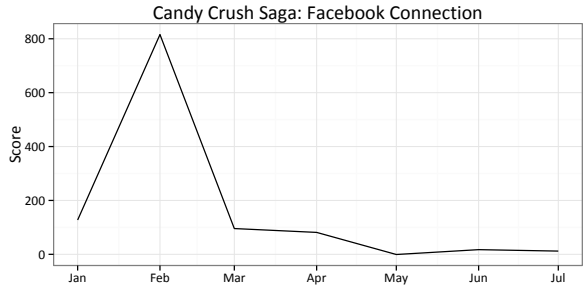

Figure 4: Candy Crush Saga: outbreak of the cluster for "Facebook Connection"

Table 5: Examples of phrases extracted from reviews of Candy Crush Saga belong to the cluster of facebook connection problem

started using most of my battery
it drains my battery
running in the background and drain
waste my battery
using 90 of my battery
...

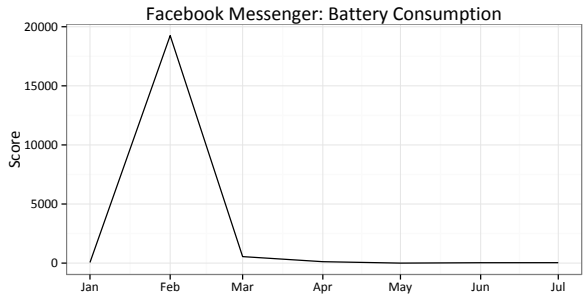

Figure 5: Facebook Messenger: outbreak of the cluster for "Battery Consumption"

Facebook Messenger on February (Figure 5 and example phrases in Table 5). On Feb 12, users of Facebook Messenger noticed a significant loss in their battery power and started complaining about it. On the same day, Facebook developers confirmed it was a syncing problem with their app and released a new update to fix it [6].

This is an interesting finding since MARK [18] was a semi-automated tool, which means we had to use human intervention to find this problem. However, the soft clustering approach can automatically discover it.

3.3 Threat To Validity

There are two threats to validity of our evaluation process. The first is, we only applied our process on 1.8 millions reviews of 120 apps. Therefore, the apps used in this research might not be representative. To address this threat, we made our data publicly available when requested to encourage researchers to replicate and extend our experiments.

The second is, our approach cannot extract opinions corresponding to textual expressions that were not extracted from Amazon reviews dataset.

4. PRIOR AND RELATED WORKS

In the recent years, researchers had answered quite a few interesting questions about app reviews. As of 2014, Hamad Khalid [11] [13] found that there are at least 12 types of user complaints present in mobile app reviews by manually inspecting 6,390 reviews from iOS apps. Guzman and Maalej [7] used Topic modeling techniques to group extracted features from reviews and found users' sentiments for them. Another study was conducted by Khalid *et al.* [12] with reviews is to explore the correlation between them and the devices that the app is running on, then use this information to help Quality Assurance prioritize their effort. Wano and Lio found out that review styles are different for each category in their research [19].

More into this trend, researchers found that app reviews could contain many valuable information. A number of recent researches have focused on extracting such information such as the work of Iacob *et al.* [8] that extract feature requests from reviews. In the same year (2013), Carreno *et al.* [6] also proposed a way to extract changed or new requirements for next versions of the app from user feedbacks. Following the trend, Fu *et al.* [5] proposed Wiscom, a tool that can discover inconsistencies in reviews, identify why an app is being liked or disliked, and provide some users' major concerns and preferences of different types of apps.

In 2014, Iacob *et al.* [9] introduced the next version of MARA to extract relevant information such as feature requests and bug reports. AR-Miner [2] from Chen *et al.* was the first framework to extract informative user reviews then group and rank them.

While some researchers study mobile app reviews in general, some others prefer to target specific needs and concerns that are expressed in the reviews. For instant, Chen *et al.* [3] made a study about the unreliable maturity content ratings for mobile apps on Android market and Apple store. They discovered that these ratings can be not accurate and inconsistent with the published policies. Security and Privacy issues were extracted by Cen *et al.* [1] using a two dimensional labeling system. Palomba *et al.* [16] developed CRYSTAL to find traceability link between code changes and app reviews to analyze how reviews affect development process.

However, mining user opinion using a phrase-based approach was not mentioned in prior works.The most similar work to our research is our prior work with MARK framework [17]. In this work, we use keyword-based approach as a mean to discover user opinions. However, using keywords alone cannot express as many sentiment levels as the phrases because of their abstract nature. Without phrase-based approach, currently there is no method to show a complete

[6]http://androidforums.com/threads/facebook-messenger-battery-drain.902687/

user opinion using only keyword.

5. CONCLUSION AND FUTURE WORK

We proposed a novel phrase-based approach for user opinions discovering in mobile app reviews. Our approach includes: (1) a phrase extraction process to extract *templated phrases* from reviews (2) an user opinions tracking techniques using phrases. Moreover, for phrase extraction, we have semi-manually developed a dataset of 90125 *phrase templates* for extracting user reviews.

We also conducted a preliminary evaluation of our approach on 2 case studies of real-life problems, showing that it can automatically detect major user opinions and support developers in understanding them.

For future development, we plan to conduct more experiments to: compare the effectiveness of our approach with the parsing approach on a reasonable-sized dataset; compare the explanation power of single word approach to our phrase-based approach; discover newly introduced problems on several popular apps and contact developers to verify if our approach really helps them discovering problems. Moreover, we intent to develop a technique that can automatically highlight the important opinions on actual reviews to support developers on the reading task using our approach.

6. REFERENCES

[1] L. Cen, L. Si, N. Li, and H. Jin. User comment analysis for android apps and cspi detection with comment expansion. In *Proceeding of the 1 st International Workshop on Privacy-Preserving IR: When Information Retrieval Meets Privacy and Security (PIR 2014)*, page 25.

[2] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang. Ar-miner: Mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 767–778, New York, NY, USA, 2014. ACM.

[3] Y. Chen, H. Xu, Y. Zhou, and S. Zhu. Is this app safe for children?: A comparison study of maturity ratings on android and ios applications. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 201–212, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[4] M.-C. De Marneffe, B. MacCartney, C. D. Manning, et al. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454, 2006.

[5] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 1276–1284, New York, NY, USA, 2013. ACM.

[6] L. Galvis Carreno and K. Winbladh. Analysis of user comments: An approach for software requirements evolution. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 582–591, May 2013.

[7] E. Guzman and W. Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 153–162. IEEE, 2014.

[8] C. Iacob and R. Harrison. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 41–44, Piscataway, NJ, USA, 2013. IEEE Press.

[9] C. Iacob, R. Harrison, and S. Faily. Online reviews as first class artifacts in mobile app development. In *Mobile Computing, Applications, and Services*, pages 47–53. Springer, 2014.

[10] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM, 2008.

[11] H. Khalid. On identifying user complaints of ios apps. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1474–1476, Piscataway, NJ, USA, 2013. IEEE Press.

[12] H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan. Prioritizing the devices to test your app on: A case study of android game apps. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pages 610–620, New York, NY, USA, 2014. ACM.

[13] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan. What do mobile app users complain about? a study on free ios apps. 2014.

[14] D. Lin and X. Wu. Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1030–1038. Association for Computational Linguistics, 2009.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[16] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia. User reviews matter! tracking crowdsourced reviews to support evolution of successful apps. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 291–300. IEEE, 2015.

[17] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen. Mining user opinions in mobile app reviews: A keyword-based approach (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 749–759. IEEE, 2015.

[18] P. M. Vu, H. V. Pham, T. T. Nguyen, and T. T. Nguyen. Tool support for analyzing mobile app reviews. In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 789–794, Nov 2015.

[19] M. Wano and J. Iio. Relationship between reviews at app store and the categories for software. In *Network-Based Information Systems (NBiS), 2014 17th International Conference on*, pages 580–583, Sept 2014.