2015 30th IEEE/ACM International Conference on Automated Software Engineering

Mining User Opinions in Mobile App Reviews: A Keyword-based Approach

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

***Abstract*—User reviews of mobile apps often contain complaints or suggestions which are valuable for app developers to improve user experience and satisfaction. However, due to the large volume and noisy-nature of those reviews, manually analyzing them for useful opinions is inherently challenging. To address this problem, we propose MARK, a keyword-based framework for semi-automated review analysis. MARK allows an analyst describing his interests in one or some mobile apps by a set of keywords. It then finds and lists the reviews most relevant to those keywords for further analysis. It can also draw the trends over time of those keywords and detect their sudden changes, which might indicate the occurrences of serious issues. To help analysts describe their interests more effectively, MARK can automatically extract keywords from raw reviews and rank them by their associations with negative reviews. In addition, based on a vector-based semantic representation of keywords, MARK can divide a large set of keywords into more cohesive subsets, or suggest keywords similar to the selected ones.**

***Keywords*—Opinion Mining, Review Analysis, Keyword**

I. Introduction

Mobile apps are the software applications developed specially for mobile devices such as smartphones and tablets. As the use of mobile devices explodes, developing mobile apps becomes a popular and profitable business in software development. However, it is also a highly competitive business, as millions and counting apps of different categories are made available on app markets. Since the revenue and profit of a mobile app is often proportional to the size of its userbase, improving user experience and satisfaction to retain existing users and attract new ones is of important to its developers. User opinions like complaints or suggestions would be valuable for that task.

As mobile app markets typically provide rating and reviewing mechanisms, reviews from users of apps purchased on those markets provide an important source of user opinions. However, analyzing those reviews manually for useful opinions would be inherently challenging. First, a popular app with millions of users often gets thousands of reviews each day and reading all of those reviews would be very time-consuming. In addition, user reviews of mobile apps are often noisy. They can have typos, acronyms, abbreviations, emoji icons, etc. Even worse, prior research reports that more than 60% of user reviews do not contain useful opinions [1].

In this paper, we introduce MARK (*Mining and Analyzing Reviews by Keywords*), a semi-automated framework for mining user opinions from user reviews of mobile apps. Considering this mining task as an information retrieval problem, MARK follows a keyword-based approach. That is, it allows a review analyst to specify his/her interests in one or some mobile apps by a set of keywords. Then, it uses those keywords to search for and visualize the most relevant reviews, expecting them to contain opinions usefully matched the analyst's specified interests. The key departure of MARK from a typical information retrieval system is that it employs several automated, customized techniques for extracting keywords from raw reviews, ranking those keywords based on review ratings and occurrence frequencies, grouping related keywords, searching for reviews that are relevant to a set of keywords, visualizing their occurrences over time, and reporting if such occurrences contain unusual patterns.

TABLE I. Negative keywords for Facebook Messenger

Rank	Keyword	Rank	Keyword	Rank	Keyword	Rank	Keyword
1	battery	6	expire	11	phone	16	say
2	message	7	drain	12	app	17	space
3	download	8	crash	13	keep	18	use
4	install	9	fix	14	facebook	19	freeze
5	session	10	log	15	reinstall	20	network
...	...	...	...	...	...	...	...

TABLE II. Clusters of negative keywords

Energy consumption	Unrecoverable error	Authentication
battery, drain, hog, consume	crash, freeze, hang, break	session, login, fail, connect

Let us illustrate MARK and those techniques via an example. Assume that a review analyst is interested in negative user opinions about Facebook Messenger, one of the most popular mobile apps with around 700 millions active users by July 2015 [2]. Initially, he has no idea about which aspects of the app that get negative opinions. Thus, MARK uses its customized keyword extraction technique discussed in details in Section II-A3 to extract all potential keywords from raw reviews of this app. Then, it ranks those keywords based on a negative scoring scheme discussed in details in Section II-A and presents the ranked list to the analyst. Table I illustrates the top ones among them.

We could see in the table that keywords are often related and indicate more general concerns. For example both keywords "crash" and "freeze" could be used to describe the app's status when an *"unrecoverable error"* occurs. Or, "battery" and "drain" often go together to describe the bad *"energy consumption"* of the app. Therefore, MARK can cluster, i.e. divide the listed keywords into smaller subsets, each for a

978-1-5090-0025-8/15 $31.00 © 2015 IEEE
DOI 10.1109/ASE.2015.85

TABLE III. EXPANDING KEYWORDS FOR "BATTERY" AND "DRAIN"

heat, hog, usage, consumption, consume, battery, drain, hogger, overheat, eater, eat, drainer, power, ram, cpu, storage, memory

TABLE IV. REVIEW SEARCH RESULT FOR KEYWORDS RELATED TO "ENERGY CONSUMPTION"

Battery drain. Latest version usually destroying my **battery** life, **consuming** almost a third of my phones **power consumption** and I haven't opened the app or gotten a single message all day!!! Please fix!
Batt **hogger**. Disabled notification sounds & vibrate. Still **drains** more **battery** than the screen itself. Can do without.
Excessive **battery usage**, **overheating**. Excessive **CPU** and **battery usage** is leading to quickly **drained battery** and **overheating** even when my tablet is sleeping. Uninstalling until this is fixed; it's killing my **battery**.

more general concern. Table II illustrates the resulted clusters produced for Facebook Messenger.

This clustering task is based on Word2Vec, a distributed, vector-based representation of words [3]. Word2Vecvec represents each word in the vocabulary as a high dimensional vector and learns those vectors from a large corpus of text such that words having similar or related syntactic roles or semantic meanings would have similar vectors. Based on Word2Vec, MARK can divide a set of keywords into smaller subsets of related ones by applying K-mean [4], a similarity-based clustering algorithm on the vectors representing those keywords. Details about Word2Vec and MARK's clustering technique will be discussed in Section III-B.

Instead of dividing a large set of keywords into smaller subsets, MARK can expand a small set of keywords into a bigger one, also based on the vector-based similarity of the keywords. This case often happens when the analyst has some ideas about the opinions he is looking for and wants to explore them in a broader context. For example, when analyzing the keywords in Table I, the analyst sees the keyword battery and is interested in the *"energy consumption"* aspect of this app. He assumes keywords "battery" and "drain" to be related to this topic, but expects users using other keywords to describe it. Therefore, he requests MARK to expand his initial keyword set {"battery", "drain"}. Table III shows the expanding results, with newly discovered keywords like "heat", "power", and "usage".

Once the analyst specifies a set of keywords (via clustering or expanding) that matches his interests, MARK can query its review database and returns ones that are most relevant to that keyword set. This querying task is based on the standard Vector Space Model [3]. That is, MARK applies the *tf.idf* [5] weighting scheme on the keywords and measures the relevance between the given keyword set to a review based on the cosine similarity of their *tf.idf* feature vectors. Table IV illustrates some reviews queried for the keyword set in Table III. As seen, those reviews contain several (negative) user opinions about the *"energy consumption"* aspect of this app.

MARK can also visualize and analyze the occurrences of a keyword set overtime for abnormal patterns. Prior studies [1], [6] suggest that when a new version of an app is released with some severe defects or issues making many users unsatisfied, there are often sudden changes in occurrences of related topics in user reviews. For example, in Feb 2015 a new version of Facebook Messenger is released with a critical error in

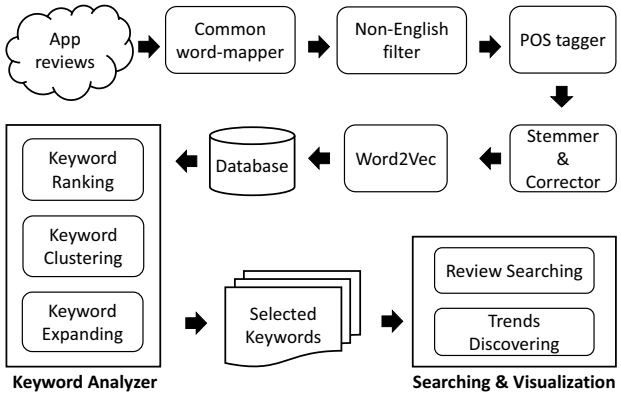

Fig. 1. System Architecture and Processing Pipeline of MARK

the syncing functionality [7] which generates extremely high CPU and network activities, and thus causes severe battery draining. This leads to an unusual occurrence of user reviews discussing *"energy consumption"* topic and related keywords. To detect such abnormalities, MARK considers the keywords' occurrence counts as a time series, computes its simple moving average (SMA) and the differences between actual counts and its SMA values. If a difference value is significant higher than the standard deviation of SMA values, it is considered as a sudden change in the corresponding occurrence count. Figure 6 illustrates the analysis result of MARK on this example. Section IV-B will discuss the technique in more details.

We have developed and deployed MARK as a web-based tool available online at http://useal.cs.usu.edu/mark. Figure 1 summarizes its system architecture and processing pipeline. In general, MARK works in three stages. In the pre-processing stage, app reviews are crawled from designated online app stores such as Google Play or Apple App Store and are tokenized as sequences of words. Then, the Common Word Mapper replaces frequently misspelled words and popular acronyms and abbreviations with their corrected forms. The Non-English Filter will analyze the reviews and discard ones not written in English. In the next step, word sequences of unfiltered reviews are analyzed by a part-of-speech (PoS) tagger. After this step, MARK keeps only nouns and verbs and transforms them to their base forms via stemming and spell-correction. After that, Word2Vec component processes those words to produce their semantic vectors. In the two remaining stages, MARK allows users select keywords of their interests by using Keyword Ranking, Keyword Clustering, and Keywords Expanding components. Once the keywords are selected, the user can view the related reviews provided by the Research Searching component or view the occurrence trends of those keywords via the Trends Discovering component.

The remaining of this paper is organized as the following. Section II describes MARK's keyword extraction and its customized regulation algorithms. Section III discusses keyword analysis techniques used in MARK. Section IV presents our trend abnormalities detection and visualization technique. Section V presents our empirical evaluations and case studies. Related works are discussed in Section VI and we conclude our study in Section VII.

TABLE V. TOP 10 MISSPELLED WORDS COLLECTED FROM 300,000 REVIEWS

Misspelled	Corrected	Count	Misspelled	Corrected	Count
usefull	useful	1104	appp	app	351
excelent	excellent	452	watsapp	whatsapp	348
helpfull	helpful	417	coool	cool	261
dosent	doesn't	374	exelent	excellent	241
frnds	friends	352	aplication	application	228

II. KEYWORD EXTRACTION

In this section, we describe how MARK pre-processes raw review data and extracts keywords from them. We first discuss the common issues of processing mobile app reviews including misspelled words and typos, acronyms, abbreviations, slangs, non-English text, etc. After that, we propose a keyword extraction procedure with several customized algorithms to address such problems.

A. Reviews Pre-processing

1) Misspelled words, acronyms, and abbreviations: Mobile app users are most likely to write their reviews on mobile devices. However, most of those devices have small sizes and no physical keyboards, making it difficult and time-consuming for typing. This situation leads to the frequent occurrences of typos (i.e. misspelled words), acronyms, and abbreviations in user reviews of those mobile apps.

We performed a preliminary analysis of 300,000 reviews collected from Google Play during January 2015. Checking against an English dictionary of more than 150,000 common words, we found nearly 5,500 words do not belong to that dictionary and occur at least 20 times in those reviews. Table V shows the top 10 among them. As illustrated in the table, most are misspelled words like "usefull" or "excelent". Some are abbreviations (e.g. "frnds" is an abbreviation of "friends") and some refer to special names (e.g. "watsapp" is a misspelled word for "WhatsApp" - a popular messaging app for mobile devices).

Common spell correctors like FAROO [8] or Peter Norvig's corrector [9] would not work well for such abbreviations and special names. Therefore, we decided to use a custom dictionary for the most frequent out-of-dictionary words we detected in the aforementioned preliminary analysis. To develop this dictionary, our team of four researchers analyzed each word and mapped it to the most plausibly corrected form. If a word is not so obvious, we will read all reviews containing that word to infer its corrected form from the context. For example, we consider intagram as a misspelled form of Instagram, a popular image-sharing service. Our dictionary also contains app names, acronyms, abbreviations, and technical terms that would not appear in common English dictionaries, like "reinstall", "hashtag", or "xml". This dictionary is made available on MARK's website.

2) Non-English reviews: Mobile apps are available globally, their users can come from different countries and cultures and thus, might write reviews in languages different from English. However, we only focus on analyzing English reviews (which are the most popular), non-English reviews would introduce noises to the analyses and thus, should be filtered. Prior work [6] filters non-English reviews based on the occurrences of non-ASCII characters. However, many languages can be written with ASCII characters, e.g. Spanish, French, or even Vietnamese. Therefore, we decide to develop a custom filtering algorithm for non-English reviews.

```
function Stem(a review r)                                  1
  for each sentence c ∈ r                                  2
    for each verb/noun w ∈ PoS(c)                          3
      if w is an irregular verb                            4
        return IrregularVerbTable[w]                       5
      else                                                 6
        apply stemming rules on w                          7
        return SpellingCorrect(w)                          8
```

Fig. 2. Customized Stemming Algorithm

We performed a preliminary analysis on non-English reviews for some insights on their characteristics. We randomly selected 400 reviews and manually labeled them as English (245 reviews) or non-English (155 reviews). We observed with no surprise that most words in non-English reviews do not appear in our English dictionary. However, sometimes a non-English review can contain English words (e.g. a technical term like "email") or words in foreign languages that accidentally have the same spellings with English words (e.g. "can" is a Vietnamese word that is also a meaningful English word). This is also true for English reviews, i.e. sometimes they also contain words not appear in English dictionaries, e.g. misspelled words that have not been corrected in previous pre-processing steps.

Designed based on such observations, our filtering algorithm considers a review to be written in English if the ratio of its single words (i.e. unigram) appearing in our English dictionary (which also includes special names, acronyms, abbreviations, and technical terms found in our previous analysis) exceeds a preset threshold. To improve the accuracy of this algorithm, we also compute the ratio of pairs of consecutive English words (bigram) in each review and compare it to another preset threshold. This rule is based on the assumption that English speakers would not make too many mistakes repeatedly (e.g. one can misspell a word but would be less likely to misspell two words in a row).

3) Word stemming and Part-of-Speech tagging: Information retrieval system commonly employ stemming, i.e. reducing words to its simplest form, to improve their searching capacity. However, common stemmers like Porter's Snowball [10] often over-stem words. For example, it would stem the words "conspiracy", "conspirator", "conspire", "conspired" and "conspiring" as "conspir". This is not desirable for our keyword-based analysis because we want the keywords to retain their original meanings. Lemmatization tools like Stanford Lemmatizer [11] do not over-stem words, but might be too slow for our review data.

In addition, we expect review analysts to be interested in user opinions related to features or issues of mobile apps and such features and issues would be described by verbs (e.g. "freeze" or "crash") and nouns (e.g. "battery" or "screen"). Adjectives and adverbs often describe additional information to the main nouns and verbs (e.g. "slow", "bad", or "great"), thus do not provide references to functions or features of mobile apps. Other parts of speech like pronouns or prepositions even contribute less to the general understanding of user opinions (they are often called stop words and removed by information

TABLE VI. STEMMING RULES

Case	Pre-condition	Condition	Action	Note/ Example
plural noun OR verb-present tense- 3rd person singular	length $<$ 4 OR not end with 's'		do nothing	Fail safe if PoS tagger makes mistake
	end with 'es', X-Y-Z: last 3 characters before 'es'	YZ is a SPECIAL PAIR	remove 's'	divestitures → divestiture
		Z = 'i'	remove 'es' and change 'i' to 'y' if Y is a vowel, just remove 'es' otherwise.	studies → study
		YZ = 'ss'	remove 'es'	masses → mass
		Z = 's'	remove 's'	bases → base
		Z = 'x', 'o', 'z'	remove 'es'	foxes → fox
		YZ = 'ch', 'sh'	remove 'es'	approaches → approach
		Z = 'h'	remove 's'	breathes → breathe
		everything else	remove 's'	combines → combine
	end with 's'		remove 's'	annoys → annoy
verb-past tense OR verb-past participle	length $<$ 5 OR not end with 'ed'		do nothing	Fail safe if PoS tagger makes mistake
	vowel count = 1		do nothing	
	X-Y-Z: last 3 characters before 'ed'	length = 5 && Y is a vowel	remove 'd'	fired → fire
		Z = 'i'	remove 'ed', change 'i' to 'y'	implied → imply
		vowel count = 2 && YZ = 'tt', 'nn', 'rr', 'dd', 'mm', 'ff', 'gg', 'pp', 'bb'	remove 'ed' and one last consonant	hogged → hog
		P(e|XYZ) * P(' '|YZe) $>$ P(' '|XYZ)	remove 'd'	3-gram model
	everything else		remove 'ed'	acted → act
verb-present participle (gerund)	length $<$ 6 OR not end with 'ing'		do nothing	Fail safe if PoS tagger makes mistake
	vowel count = 1		do nothing	
	X-Y-Z: last 3 characters before 'ing'	length = 6 && Y is a vowel	remove 'ing', append 'e'	firing → fire
		vowel count = 2 && YZ = 'tt', 'nn', 'rr', 'dd', 'mm', 'ff', 'gg', 'pp', 'bb'	removing 'ing' and one last consonent	hogging → hog
		P(e|XYZ) * P(' '|YZe) $>$ P(' '|XYZ)	remove 'ing', append 'e'	3-gram model
	everything else		remove 'ing'	acting → act

retrieval systems). Therefore, we need to identify the part-of-speech (PoS) tags of the words and keep only verbs and nouns as potentially useful keywords.

Due to such issues, we decide to design a customized stemmer to work exclusively for verbs and nouns. The main purpose of this stemmer is to reduce verbs and nouns in different tenses and forms back to their base forms, e.g. "frozen" to "freeze" or "notifications" to "notification". Figure 2 summarizes our stemming algorithm. The stemmer first breaks the given review into sentences and employs the Stanford PoS tagger [12] to find verbs and nouns in each sentence. If a verb is an irregular verb, its base form will be retrieved directly from a pre-defined table of irregular verbs [13]. Otherwise, stemming rules in Table VI will be applied directly on the word.

Those rules are designed based on English grammar and some rules of the Snowball stemmer. As seen in the table, we only stem plural nouns and verbs in the present tense having an "s" or "es" suffix, verbs in the past or past perfect tenses having an "ed" suffix, and verbs in the present continuous tense having an "ing" suffix. The stemming rules aim to remove those suffixes and convert the noun or verb back to their original forms before those suffixes are added.

Although less likely, our stemmer might still over-stem words. To address this problem, we train a tri-gram language model at character level on a dataset of one billion characters containing text crawled from Wikipedia [14], and use this model to check if a word is over-stemmed and fix it. For example, word "analyzing" is a verb in the present continuous tense. However, when our stemmer removes its suffix "ing", its new form "analyz" is over-stemmed because our tri-gram model suggests that "lyz" is less likely to appear as ending characters of an English word than "yze" and "e" is more likely to appear after "lyz". Thus, character "e" is added to the word, producing "analyze" as the final (stemmed) form of "analyzing".

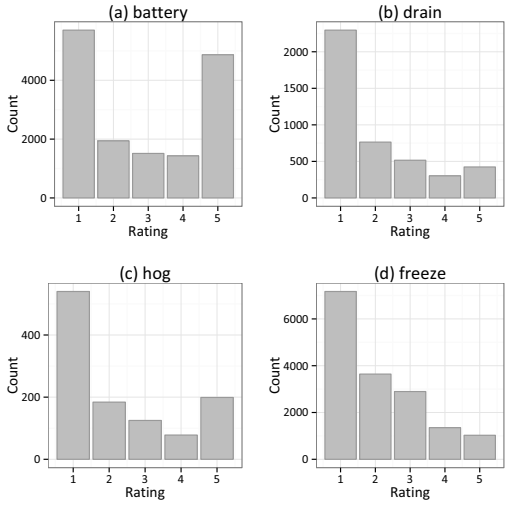

Fig. 3. Distribution of ratings for some keywords

We further fix the over-stemming problem by running the FAROO Spelling Corrector after applying our stemming rules in Table VI. We choose this spelling corrector because it is very time-efficient. We also train it with the aforementioned one-billion-character dataset. It should be noted that stemming and spell correcting are applied only for common words and not for words in the dictionary of popular misspelled words and special names discussed in Section II-A1 because the corrections for them have been included in that dictionary.

III. KEYWORD RECOMMENDATION

MARK can extract a vast number of keywords from a large amount of user reviews. Therefore, it has three recommendation techniques to help analysts navigate those keywords and describe their interests more accurately and effectively. The first is a ranking technique to identify keywords that often associate with negative reviews, which are likely to describe the issues or features of mobile apps that make users unhappy. Two other techniques are clustering and expanding, which could help an analyst divide a large set of keywords into smaller, more cohesive subsets or expand a small set of keywords into a larger, more comprehensive one. This section will describe those techniques in details.

A. Keyword Ranking

Information retrieval systems often rank their results to help their users focus on the most relevant ones. MARK also ranks the keywords it extracts from user reviews to help review analysts select the most important keywords. This ranking technique is designed based on the assumption that keywords frequently appearing with negative reviews are likely to describe the issues or features of the apps that cause bad user experience, i.e. making users unhappy or unsatisfied. Thus, those keywords would be of interested to app developers and review analysts because they can help to identify the bad aspects of an app and the user opinions about such aspects.

Table IV and Figure 3 provide some anecdotal evidence of that assumption. The reviews in the table show that users

use words like "battery", "drain", and "hog" when complaining about the excessive energy consumption of some mobile apps. Figure 3 shows the numbers of reviews containing those words grouped by their ratings in MARK most recent review dataset of more than two millions reviews. As seen, those words appears much more in negative (i.e. 1- or 2-star) reviews than in positive (i.e. 4- or 5-star) reviews. For example, "drain" appears in 3,060 negative reviews and 726 positive reviews, which is four times more negative than positive. "battery" is not as excessive, but the number of negative reviews is still higher than that of positive reviews (7,645 vs 6,297). The figure also shows the counts for word "freeze" which is often used to describe or complain about apps that suddenly stop working or responding. As seen "freeze" appears in negative reviews five times more than in positive reviews (10,814 vs 2,378).

Based on that assumption, MARK ranks the keywords by their association with negative reviews. It measures such association using a metric that we call *contrast score*:

$$\varphi = \frac{n}{p} \times (n - p)$$

In this formula, n and p are correspondingly the counts of negative (i.e. having ratings of 1 and 2 stars) and positive (i.e. having ratings of 4 and 5 stars) reviews containing the keyword of interest. For example, as shown in Figure 3, keyword "freeze" has $n = 10,814$ negative reviews and $p = 2,378$ positive reviews. Thus, its contrast score is $10,814/2,378 \times (10,814 - 2,378) = 38,362.87$. The contrast scores of "drain" and "battery" are $9,837.52$ and $1,636.57$, respectively.

In addition to the ratio of negative and positive reviews, the formula also combines their absolute difference because that will give more score to keywords occurring frequently in user reviews. Such a keyword is more likely to describe an issue that affect many users. This also help to downgrade keywords with low frequencies but extremely high contrast ratio, e.g. $n = 10$ and $p = 1$.

We also consider two other scoring techniques: the Pearson (linear) *correlation* of the ratings and their counts, and the *skewness* of the distribution of the ratings. The relevance of those scores can be inferred from Figure 3. For example, for keyword "freeze", higher ratings (e.g. 4 or 5 stars) have lower counts, leading to a negative linear correlation of -0.937. For the same reason, the distribution of the ratings is highly *left-skew*, with the skewness of 8.719.

We performed a preliminary analysis to compare those three scoring techniques. It is surprised that they produced nearly identical results. For example, the sets of top 1,000 keywords ranked by those three metrics share 96% to 99% of their keywords. Due to this preliminary result, we use our contrast score as the main ranking technique in the final version of MARK because it can be computed very efficiently and incrementally (e.g. we just need the counts for computation and we could store and update the counts when new data are crawled and pre-processed).

B. Keyword Clustering and Expanding

Because MARK performs review search and trend visualization based on keywords provided by the analysts, the more relevant of those keywords to the issues of interest, the more accurate and effective the search and visualization will be. In practice, users often use similar or related keywords to describe an issue or a feature. For example, we have seen in Table IV that "battery", "power", "drain", and "hog" are used together to describe the issue of excessive energy consumption. Therefore, grouping similar or related keywords would help analysts obtain more better analysis results.

```
function Cluster(keywordset S, number of clusters k)          1
  for i = 1 to k                                              2
    centroid[i] = a random vector ∈ S                         3
  repeat                                                      4
    for each keyword v ∈ S                                    5
      cluster[v] = i where i is the centroid closest to v     6
    for i = 1 to k                                            7
      centroid[i] = meanvector(∀v where cluster[v] = i)       8
  until cluster has no change                                 9
  return cluster                                              10
```

Fig. 4. Keyword Clustering Algorithm (k-mean)

```
function Expand(keywordset S, vocabulary V, threshold δ)      1
  repeat                                                      2
    for each keyword v ∈ V − S                                3
      c = meanvector(S)                                       4
      if distance(c, v) < δ                                   5
        S = S ∪ {v}                                           6
  until no more word added                                    7
  return S                                                    8
```

Fig. 5. Keyword Expanding Algorithm

MARK provides two different grouping functions for its keywords. It can divide (i.e. cluster) a large set of keywords into into smaller, more cohesive subsets. For example, an analyst can start by selecting the top 100 keywords from reviews of an app and then cluster them into 10 subsets, expecting each set to describe a specific issue of the app. In contrast, when the analyst starts with just a few keywords for a specific concern of her interests, MARK can expand that set into a larger, more comprehensive one by adding many more keywords that are similar or related to the selected ones.

Both of these grouping techniques require a similarity measure for keywords. Traditional approaches often use WordNet [15] to obtain similar words (i.e. synonyms). However, MARK utilizes Word2Vec, a distributed, data-driven, vector-based representation of words [3]. This technique represents each keyword as a high dimensional vector which it learns from our review data. Because its learning algorithm produces similar vectors for keywords having similar or related syntactic roles or semantic meanings, MARK measures the similarity of keywords based on the distances of their vectors.

By using vectors to represent keywords and vector distances to measure similarity between keywords, MARK can apply k-mean, a popular clustering algorithm, to divide a set of keywords into k subsets. Figure 4 illustrates this algorithm. Initially, it randomly selects k *centroids*, each is a vector and represents the center of a cluster. Then, it assigns each keyword v to the cluster whose centroid is closest to v (in term of vector distance). After that, each centroid is re-computed as the mean vector over all keywords assigned to the corresponding cluster. This process repeats until all clusters are stable, i.e. no word is assigned to a cluster different from its current assignment. Because the centroid is the mean vector of each cluster, it could

be considered to represent the "common meaning" of that cluster. Therefore, by assigning a keyword to the cluster with closest centroid, each keyword is considered to be assigned to a cluster most similar to it.

Figure 5 illustrates our keyword expanding algorithm. Although it performs the task opposite to keyword clustering, it reuses some ideas of the keyword clustering algorithm. That is, given a set of keywords S for expanding, our algorithm first computes the mean vector c over all keywords in S to represent that set. Then, it adds to S any keyword v that is close enough to c, i.e. the distance between c and v is smaller than a pre-defined threshold δ. After that, the mean vector c is re-computed and the process repeats until no new keywords are added. S is then returned with newly added keywords.

IV. Review Search and Trend Analysis

Once an analyst has selected a suitable (i.e. as concrete and comprehensive as possible) set of keywords to describe her interests, MARK can search for the actual user reviews that are most relevant to those keywords. This is the standard information retrieval task and thus, MARK employs the popular *tf.idf* weighting scheme and the Vector Space Model (VSM) for this task [5]. In addition to review search, MARK can visualize and analyze the occurrences of the selected keywords over time, helping the analyst to spot any unusual patterns or abnormalities in those occurrences. Let us discuss those two tasks in details.

A. Review Search

The main goal of the Review Search function is to provide the analyst the actual reviews that are the most relevant to his interests specified by the keywords she provides. To do that, we need a model to represent the reviews and measure the relevance between them and the provided keywords. Vector Space Model [5] is wide-used to represent textual documents in information retrieval systems. In Vector Space Model, a document (e.g. a review) or a query (i.e. a set of keywords) is represented by a high dimensional vector of which each element corresponds to a term or keyword in the vocabulary. Then, the relevance between a document and a query is often computed as the cosine between their vectors.

MARK employs Vector Space Model for its Review Search function. It represents each review by a vector and uses the *tf.idf* (term frequency - inverse document frequency [5]), a standard term weighting scheme to compute the element values for those vectors. The term frequency (tf) of a keyword in a review is the number of its occurrences in that review, while the document frequency (df) of a keyword is the number of reviews in the dataset containing it. We compute the tf.idf weight for a keyword as:

$$\text{tf.idf} = \frac{\text{tf}}{\log(\text{df} + 1)}$$

For example, if we just consider three reviews in Table IV, keyword "battery" appears in all of them and appears four times in the last review. Therefore, its term frequency for the last review is 4 and its document frequency (in the whole dataset) is 3. Thus, its tf.idf weight in vector for the last review is 4/(log(3+1)) = 2. Similarly, "drain" occurs in all three reviews but just once in the last review, thus, its tf.idf weight is 1/(log(3+1)) = 0.5.

MARK pre-computes the tf.idf weights and the vectors for all reviews when it extracts keywords from them (see Section II-A). When a set of keywords is given for review search, it computes the tf.idf vector for that set (term frequencies are set as 1 and document frequencies are taken from the review data). Then, it computes the cosine of the tf.idf vector of the query with the tf.idf vectors of all available reviews, ranks the reviews by those scores (the higher the more relevant), and presents the results back to the analyst.

B. Trend Analysis

Prior studies [1], [6] suggest that unusual patterns in user reviews often indicate severe, wide-spread issues or problems that make many users unsatisfied. Trend Analysis function in MARK allows review analysts to track user reviews on a topic over time and detect any abnormalities in their discussions. This can help app developers to spot issues related their apps early and address the occurred problems in a timely manner.

Trend Analysis is designed based on time series analysis techniques. Once an analyst describes her interest via a set of keywords and a specified time range, MARK computes the total occurrences of those keywords for any day in that range to produce a time series. Figure 6(a) shows such a time series for the set of keywords related to *"energy consumption"* specified in Table III in the time range from Jan 1, 2015 to Apr 24, 2015. We could see an unusual pike in the occurrences of those keywords around Feb 13, 2015. Minor fluctuations also appear in other days, e.g. around Feb 4, 2015 or Mar 4, 2015.

Such minor fluctuations could be considered as noise. To smooth out such noises and focus on major changes, MARK employs simple moving average (SMA), a simple but effective time series analysis technique. Given the time series $x_i, i = 1..n$ (x_i is the total occurrences of the keywords of interest in day i) and a lag parameter k, the k-day simple moving average of time series x_i is another time series a_i where

$$a_i = \sum_{j=i-k}^{i-1} x_j$$

In other words, the simple moving average of day i is the average of the last k days before day i.

Figure 6(a) also shows the 113-days simple moving average of the aforementioned time series for the *"energy consumption"* keywords. We could see that the simple moving average (the dashed line) is much smoother than the original time series (the solid line). That is because moving averaging acts like a low-pass filter. When averaging the original values via a sliding window, the short-term fluctuations in the original values cancel each other.

We could also see that sudden changes in the original time series could not be detected in the simple moving average series alone (they are actually smoothed out too). However, when comparing the original series and its simple moving average, we could see that a major change occurs when the original value is significant higher than the corresponding

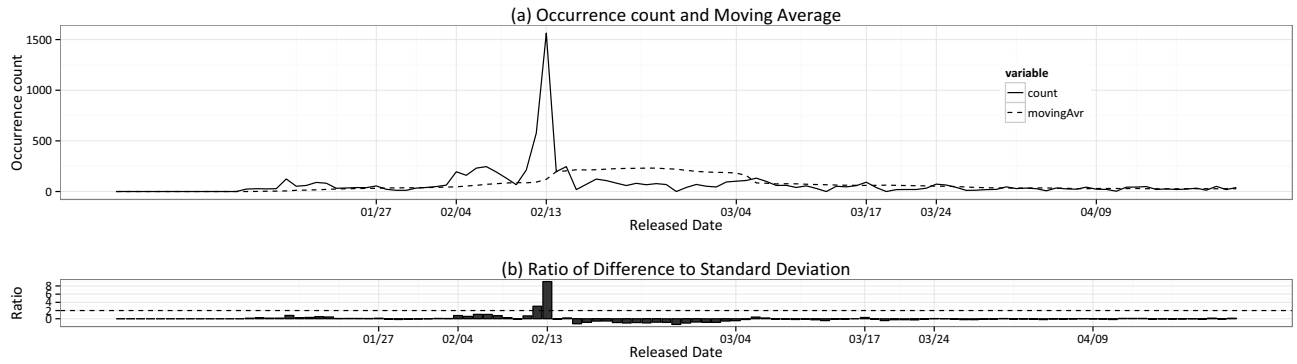

Fig. 6. Trend Analysis for "energy consumption" issue in Facebook Messenger. The keywords used are: *heat*, *hog*, *usage*, *consumption*, *consume*, *battery*, *drain*, *hogger*, *overheat*, *eater*, *eat*, *drainer*, *power*. Threshold for ratio is 2.

TABLE VII. STATISTICS OF COLLECTED DATA

Total number of mobile apps	95
Total number of crawled reviews	2,106,605
Average number of reviews per app	22,174
Max number of reviews (Clash Of Clans)	302,936
Min number of reviews (Amazon MP3)	1,001
Average words per review	11.1
Collection Time	Jan 1, 2015 to May 1, 2015

simple moving average value. Thus, to detect such changes, we define the *relative deviation score* as:

$$\tau_i = \frac{x_i - a_i}{\sigma_i}$$

In this formula, x_i, a_i, and σ_i are the corresponding the actual occurrence count, the moving average value, and the standard deviation of the moving average series at day i. a_i could be considered as the expected count for day i. So τ_i indicates how far the actual count differs from the expected count relative to the typical difference (i.e. the standard deviation). In statistics, a relative deviation of 2 (i.e. an actual difference of two standard deviations) is often considered as significant. Therefore, if $\tau_i > 2$, MARK will report day i to have a sudden, significant change in the trend for the corresponding keywords. Of course, it also allows users to use a different threshold.

Figure 6(b) plots the relative deviation score over time for the *"energy consumption"* keywords. As seen in the figure, it is lower than 2 most of the time, except two days Feb 12 and 13, 2015. Those two days are of course the time of a burst in occurrences of those keywords.

V. EMPIRICAL EVALUATION

In this section, we present and discuss our empirical evaluation of MARK on a large dataset of more than two millions reviews collected from Google Play. Our empirical evaluation focuses on the runtime parameters and the final accuracy of MARK in its keyword extraction and recommendation tasks. We also evaluate the correctness and potential usefulness of review search tasks.

A. Data Collection

Table VII summarizes the dataset we prepared for our evaluation. In total, we have crawled two millions of reviews from 95 mobile apps on Google Play from January 1, 2015 to May 1, 2015. Each crawled review contains a title, a long text description of the review content, the creation time, the reviewer ID, and the associated rating.

The apps chosen for our study are among the most popular apps in this market such as Facebook, Twitter, WhatsApp, Snapchat, Viber, Instagram, and Clash of Clans. The full list of apps can be accessed at MARK's website http://useal.cs.usu.edu/mark. On average, each app has about 20 thousands reviews. However, the actual number of crawled reviews varies between apps because some of them have more active users than others. For example, Clash of Clans is a very popular game with near 30 millions of daily active users [16]. Within five months of our crawling process, we have obtained more than 300 thousands reviews for that app.

Because we use an open-source crawler [17] and Google Play returns only 500 reviews for each request, we had to crawl the reviews of the chosen apps continuously overtime to obtain to the most recent ones. However, the crawled reviews might not be all available reviews because the Google Play APIs we use for crawling do not have any option for storing the current states or excluding the previously crawled results.

B. Keyword Extraction

1) Non-English reviews classifier: We use the dataset of 400 reviews we have manually labeled in a preliminary analysis (see Section II) to evaluate the accuracy of our English review classifier. This dataset contains 245 English reviews and 155 non-English ones. In our experiment, we varied the thresholds of English uni-gram and bi-gram ratios from 0 to 1 with the increment of 0.01 and used those thresholds to classify and compute the classification accuracy. Finally, the best accuracy of 86.5% was obtained with the uni-gram and bi-gram thresholds of 0.64 and 0.38, respectively. We consider this level of accuracy reasonable. However, in future work, we plan to investigate in more sophisticated classification techniques to achieve higher levels of accuracy.

Table VIII lists some examples of English (in the upper half) and non-English (in the lower half) reviews classified by our technique with the corresponding uni-gram and bi-gram ratios for some of our examples. As seen in the table, our

TABLE VIII. EXAMPLES OF UNI-GRAM AND BI-GRAM RATIOS ON DETECTING NON-ENGLISH REVIEWS

Word Count	Bi-gram Ratio	Uni-gram Ratio	Reviews	Note
12	0.58	0.67	it's so fun i'm gong to die.utut77brhrhuvrivriy7vht7hrtour7	Minor mistakes
59	0.58	0.68	es, incomparable.this perhas one, if not, the best radio stations i've tuned into. has variety and though i haven't yet figures it all out how to scroll through the stations, record and everything, i like it. si tubiera que recomendar musica de internet, definitivamente esta seria mi primera opcion. en tu idioma, canciones retro y variedad chingado. es inigualabe	Written in two languages
16	0.56	0.69	need support group! clans anonymous!.this game is completely addictive! i....neeed.....my.....nexxxttt.....doooossseeee l....offfff...thiiiiissssss....	Misspelled on purpose
15	0.38	0.67	.nce n i lov dis app.... n hlps tu clear de threats wch wre thre in ma phne.	Too many slangs
20	0.4	0.55	.maganda kasi disente pwede sa mga kids walang bastos d tulad ng ibang larong on-line..long live clash of clans!!!!!!!	Foreign language
5	0.4	0.6	buddha hong lien hoa.ommanipadmehum?!:-)...."	Foreign language

technique can recognize English reviews with words written in another language or even unrecognizable.

2) Customized Stemming: To evaluate our customized stemmer, we prepared a dataset by randomly selecting 1,000 words from our review data. This dataset includes both verbs and nouns which are tagged by the Stanford PoS Tagger. We manually stemmed those words to their respective base forms. Then we ran our stemmer on this dataset and computed the accuracy as the percentage of words whose base forms were derived by our stemmer match the base forms we have manually stemmed. We also run the widely used Stanford Lemmatizer tool for comparison purpose.

Overall, our stemmer was able to stem correctly 97.9% of the words while the Stanford Lemmatizer can only got 90.8% right. The main contributor for that improvement is likely the domain specific dictionary of misspelled words and special names (see in Section II-A1) included in our stemming and spelling correction procedure.

C. Keywords Recommendation

1) Keywords Clustering: To evaluate our Keyword Clustering technique, we selected the top 100 keywords ranked based on our contrast score. Then, we used the simple formula $k = \sqrt{n/2}$ to estimate the number of clusters (seven in this case). Table IX shows the results when MARK clustered our selected keywords into seven clusters.

We asked a team of eight Computer Science researchers to manually label those clusters and identify irrelevant words in each cluster. Then, we discussed to choose the clusters' labels and combined the results. In Table IX, the acceptance rate of each word is given next to it. For example, all eight researchers considered "crash" relevant for the issue of *"unrecoverable error"*, while only two considered "minute" relevant.

We measure the accuracy of each cluster as the average acceptance rate of its words. For example, cluster *"connection"* has its words well accepted, thus, has an accuracy of 87.5%. However, cluster *"battery & versioning"* is worse with an accuracy of 76%. The overall accuracy, as the average over all clusters, is 83%.

It is interesting that Facebook and Snapchat have their own clusters from top 100 most negative words. Our manual investigation of their recent releases suggests that they often have problems with such updates. Due to their popularity, i.e. having much higher number of active users than other apps, such problems lead to high amounts of negative reviews which could partially explain why many of their keywords appeared on the top negative list. This result suggests that their developers should be more careful in quality control of their future releases.

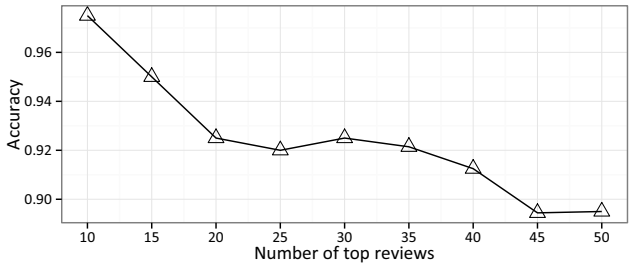

Fig. 7. Searching accuracy for top returned reviews for battery consumption concern. The keywords for this search are: *heat*, *hog*, *usage*, *consumption*, *consume*, *battery*, *drain*, *hogger*, *overheat*, *eater*, *eat*, *drainer*, *power*

2) Keywords Expanding: In this experiment, we asked MARK to expand keywords from review topics reported previously [6]. To simplify the experiment, we selected only one keyword of each topic for the initial set. Table X shows the expanding results. Similar to the previous experiment, we also asked the team of eight researchers to evaluate these results in the same manner, i.e. identifying the unrelated keywords.

The evaluation results in Table X indicate the overall accuracy of our technique is of 89.7%. Two topics *"ads"* or *"call"* accidently have low accuracy because of misspelled words like "cal" or "spoor". Nevertheless, the overall result suggests that our expanding technique can capture semantics of a keyword set and add relevant words to it.

We ran this experiment with the distance threshold δ of 0.25 and also ran with thresholds of 0.2 and lower. However, due to such low thresholds, some topics could not be expanded because MARK cannot find words whose similarity measurements could pass those thresholds.

D. Review Search

We used the same set of keywords about *"energy consumption"* from Section IV-B to evaluate our review search technique on Facebook Messenger. Our team of four authors read the top 50 returned reviews and manually validated if such a review is relevant to this topic or not.

Figure 7 shows the accuracy computed for top 10, 15, 20, 25, 30, 45, 50 reviews. As seen in the figure, the overall

TABLE IX. K-MEAN CLUSTERING RESULT FOR TOP 100 HIGHEST RANKED KEYWORDS. EACH WORD IS ACCOMPANIED BY ITS ACCEPTANCE RATE OF 8 RESEARCHERS (X/8). ACCURACY IS THE AVERAGE OF THE ACCEPTANCE RATES.

Main concerns	Battery & versioning	Connection	Unrecoverable error	Messaging	Snapchat	Authentication	Facebook
	drain (8/8) ruin (8/8) bug (8/8) break (6/8) version (5/8) disgust (6/8) lag (8/8) lollipop (6/8) downgrade (8/8) tab (3/8) pos (3/8) space (4/8)	data (7/8) connection (8/8) retry (8/8) permission (6/8) turn (3/8) network (8/8) wifi (8/8) server (8/8)	crash (8/8) open (6/8) freeze (8/8) close (8/8) stop (8/8) restart (8/8) shut (8/8) annoy (7/8) disappear (8/8) pop (5/8) minute (2/8) reason (1/8) hang (8/8)	message (8/8) send (8/8) notification (8/8) code (6/8) band (1/8) bar (2/8) receive (8/8) email (8/8) access (7/8) verification (7/8) verify (7/8)	update (8/8) dislike (5/8) story (7/8) load (8/8) video (8/8) screen (6/8) snap (8/8) upload (8/8) view (8/8) snapchat (8/8) mess (4/8) skip (5/8) bestfriend (6/8) discover (5/8)	fix (8/8) login (8/8) log (7/8) fail (7/8) uninstall (8/8) delete (7/8) expire (8/8) respond (2/8) session (8/8) say (6/8) error (8/8) download (7/8) reinstall (8/8) password (7/8) install (7/8) account (8/8) sign (7/8) refuse (8/8) happen (3/8) reset (7/8) waste (6/8) try (7/8) attempt (7/8)	feed (8/8) refresh (8/8) newsfeed (8/8) news (8/8) scroll (7/8) post (8/8) page (7/8) read (7/8) show (8/8) click (2/8) timeline (8/8) comment (8/8) middle (1/8)
Accuracy	**76.04%**	**87.50%**	**81.73%**	**79.55%**	**83.93%**	**86.41%**	**84.62%**
Avr. Accuracy	**83.11%**						

TABLE X. KEYWORD EXPANDING RESULTS FOR SEVERAL POPULAR TOPICS FROM WISCOM [6]. EACH WORD IS ACCOMPANIED BY ITS ACCEPTANCE RATE OF 8 RESEARCHERS (X/8). ACCURACY IS THE AVERAGE OF THE ACCEPTANCE RATES. CHOSEN THRESHOLD IS 0.75

Starter word	crash	compatibility	connection	pay	call	camera	ads
	reboot (8/8) close (8/8) shut (8/8) open (6/8) hang (8/8) restart (8/8) exit (7/8) freeze (8/8) stop (8/8) load (5/8) reopen (8/8) respond (8/8)	android (7/8)	wifi (8/8) network (8/8) 4g (8/8) 3g (8/8) connectivity (8/8) 2g (8/8) lte (8/8) signal (8/8) internet (8/8) connect (8/8)	buy (8/8) spend (8/8) purchase (7/8) money (8/8) earn (6/8)	voice (8/8) whatsapp (8/8) viber (8/8) cal (4/8) skype (8/8) tango (7/8) facility (3/8)	record (7/8) zoom (8/8) cam (8/8) flash (8/8)	commercia (6/8) advertisement (8/8) advert (8/8) spoor (0/8) advertise (8/8)
Accuracy	**93.75%**	**87.5%**	**93.18%**	**92.5%**	**82.14%**	**96.88%**	**75%**
Avr. Accuracy	**89.72%**						

accuracy is high in the range of 90-97%. The top 10 reviews are mostly relevant, and the accuracy is no surprise lower with more reviews. Nevertheless, these results suggest that our keyword-based review searching technique can reliably provides reviews relevant to the interests of app developers.

E. Threat of Validity

The top threat of validity of our study is the fact that our evaluation is performed on datasets manually labeled by the authors and is validated by human subjects. In addition, our subjects are students and professors, while the intended users of our tool should be professional review analysts and app developers. Finally, our datasets and experiments are small in scale (e.g. 95 downloaded apps vs millions available apps) and the apps chosen in our study might not be representative.

To address those threats of validity, we have publicly published our tool, provided our experiment data, and made our work open-source. We encourage fellow researchers to replicate and extend our experiments, send us feedbacks or feature requests, and contribute to our open-sourced tool.

VI. RELATED WORK

There are a number of empirical and exploratory studies on the importance of app's reviews in app development process.

In [18], Vasa et al. made an exploratory study about how users input their reviews on app stores and what could affect the way they write reviews. Later, Hoon et al. [19] analyzed nearly 8 millions reviews on Apple AppStore to discover several statistical characteristics to suggest developers constantly watching for the changes in user's expectations to adapt their apps. Again on Apple App Store, an emprical study about user's feedback was made by Pagano et al. [20]. Similarly, Khalid et al. suggest that there are at least 12 types of complaints about iOS apps [21]. They explored various aspects that influent user reviews such as time of release, topics and several properties including quality and constructiveness to understand their impacts on apps.

Other than reviews, price and rating of apps can also affect how people provide their feedbacks, as suggested in [22] by Iacob et al. Meanwhile, Bavota et al. [23] studied the relationship between API changes and their faulty level with app ratings. Recently, Martin et al. [24] reported the sampling bias researchers might encounter when mining information from app reviews.

Thus so far, there are very few works in mining useful information from user's reviews on app markets. One of the earliest work is from Chandy et al. [25] who proposed a classification model for spamming reviews on Apple AppStore using a simple latent model. On the other hand, Carreno et al. [26] extract changes or additional requirements for new versions automatically using information retrieval techniques. Our work is different from theirs in the main goals: We focus on extracting user opinions based on keywords to map the way developers express their concerns to user's way.

More to the mining techniques, in [27], Guzman et al. extracts features from app reviews in form of collocations and summarizes them with Latent Dirichlet Allocation (LDA) [28] and their sentiment. In their work, they score the sentiment of reviews by using SentiStrength [29], which is a lexical sentiment approach. The keywords extraction approach in our work is very close to the meaning of features extraction, but

with extra flexibility for developers and users to map their expressions. Instead of a collocation of two words, our keywords can come in a set and carries an entirely different dimension of information. We rank our keywords using ratings from users, which is a more domain specific approach for review analysis than Guzman's general lexical sentiment approach.

Another work, which is more closely related to our work, is Wiscom [6]. Their work on sentiment analysis of words using Linear Regression Model is comparable to our ranking scheme but with a different intention. We try to address the impact of keyword's concerns to users while they want to address the impact on sentiment of a keywords to discover inconsistent review. On "meso level" they use a LDA model to analyze topics of user reviews based on their distribution. Similarly, we group the keywords using K-mean clustering on vector-space representation of words, but our approach focus on exploiting different layers of semantic meaning for words inside the corpus, which give us a different perspective of user opinions. To the best of our knowledge, their work is also the first work to mention the use of timeseries on reviews for analysis, however, their approach was to use root cause analysis based on the observed busts in negative or positive comments, which does not address the problems that may lie inside normal stream of comments. Our approach using keywords can automatically discover changes in trends of given concerns regardless of the total reviews' number.

Interestingly, Iacob et al [30] designed a prototype (MARA) to retrieve app feature requests from comments using a set of linguistic rules. The rules were derived manually from actual text of reviews and the retrieved features are further analyzed using LDA to find common topics among them. This approach shares one similar aim with us: to extract features. However, their feature level stay at the phrase level while our features can be described by keywords, which may be more intuitive for both developers and users.

One of the most recent work in extracting information from reviews is AR-Miner [1]. Chen et al. propose a computational framework to extract and rank informative reviews at sentence level. They adopt the semi-supervised algorithm Expectation Maximization for Naive Bayes (EMNB) [31] to classify between informative and non-informative reviews. To rank the reviews, they use a ranking schema based on several meta-data of reviews and try to suggest the most informative ones. Our work is different from their work in both purpose and approach: Our purpose is to extract user opinion, while they want to find and rank most informative reviews, so the benefit for developers is different. Moreover, we focus on keywords level because their distributed representations can discover more detailed semantic meanings of user's reviews.

From an empirical study suggested that there are error-prone permissions reported in user reviews, Gomez et al. [32] developed a static error-proneness checker for app based on permissions. This work uses LDA to identify topics in reviews and reported permissions. As other works, it is different from our main purpose and approach of mining keyword's semantical meaning.

Finally, our framework is the first to provide a reliable way to search for most relevant reviews based on keywords approach, which is yet to be mentioned by any prior work.

VII. Conclusion

In this work, we proposed MARK as a semi-automated framework to collect and mine user opinions from app markets using a keyword-based approach. We developed and applied several automated, customized techniques for our main tasks, including: extracting keywords from raw reviews, ranking them, grouping them based on their semantic similarity, searching for most relevant reviews to a set of keywords, visualizing their occurrence over time and reporting any unusual patterns.

From our observations on the difficulties of processing raw reviews, we proposed an original technique to classify non-English ones and developed a customized stemmer to normalize their keywords. Our evaluations show that the classifier is able to correctly label 86.5% reviews in our test set. Our customized stemmer is also proved to be of higher accuracy for stemming app reviews data than the general purpose lematization tool from Stanford for our test set.

Exploiting the multi-degree semantical meaning property of distributed vector-space representation of words, our clustering and expanding techniques can discover highly related keywords that express common concerns. Our case studies show that these techniques is able to reach 83% accuracy for grouping and 89.7% accuracy for expanding tasks.

Using a set of keywords discovered from the above steps, MARK can help developers to search for most relevance reviews to that set. In our case study of searching for *battery consumption* concern in Facebook Messenger, we found that 90% of top 50 returned reviews satisfy the query.

In the tasks of discovering abnormalities of keywords occurrence in a time period, our analysis technique utilizes Simple Moving Average to alert developers when abnormal patterns appear. Our case study suggests that this technique is able to detect correctly a real problem in Facebook Message that annoyed many of its users. We conclude that it has the potential to reduce developer's effort in real life situation.

Finally, from the evaluations and case studies, we suggest that using MARK Framework would help developers to map their concerns/interests with user's via the common expression of keywords, thus save them time and effort for discovering and understanding user's opinions.

Acknowledgment

The authors would like to thank Dr. Young-Woo Kwon, Anand Ashok Bora, Sima Mehri Kotchaki, and Jiin Kim at Utah State University for their help in our evaluation. We also give our special thank to Dr. Tam Vu, Dr. Thang Dinh, and Ty Nguyen for their contributions to a prior study from which our work is inspired.

References

[1] 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*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 767–778. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568263

[2] "Number of monthly active facebook messenger users from april 2014 to june 2015." [Online]. Available: http://www.statista.com/statistics/417295/facebook-messenger-monthly-active-users/

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[4] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[5] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. Cambridge university press Cambridge, 2008, vol. 1. [Online]. Available: http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

[6] 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*, ser. KDD '13. New York, NY, USA: ACM, 2013, pp. 1276–1284. [Online]. Available: http://doi.acm.org/10.1145/2487575.2488202

[7] "Facebook messenger battery drain! - facebook's developer confirmed," Feb 12, 2015. [Online]. Available: http://androidforums.com/threads/facebook-messenger-battery-drain.902687/

[8] "Faroo spell corrector." [Online]. Available: http://blog.faroo.com/2012/06/07/improved-edit-distance-based-spelling-correction/

[9] "Peter norvic spell corrector." [Online]. Available: http://norvig.com/spell-correct.html

[10] M. Porter and R. Boulton, "Snowball stemmer," 2001.

[11] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014, pp. 55–60. [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-5010

[12] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 173–180. [Online]. Available: http://dx.doi.org/10.3115/1073445.1073478

[13] "Irregular english verbs." [Online]. Available: http://www.usingenglish.com/reference/irregular-verbs/

[14] M. Mahoney, "English wikipedia dump." [Online]. Available: http://mattmahoney.net/dc/textdata

[15] G. Miller and C. Fellbaum, "Wordnet: An electronic lexical database," 1998.

[16] "Clash of clans daily revenue at 5.15 millions usd- hacker," February 11, 2014. [Online]. Available: http://www.gamesindustry.biz/articles/2014-02-11-clash-of-clans-daily-revenue-at-5.15-million-hacker

[17] Akdeniz, "An opensource googleplay crawler." [Online]. Available: https://github.com/Akdeniz/google-play-crawler

[18] R. Vasa, L. Hoon, K. Mouzakis, and A. Noguchi, "A preliminary analysis of mobile app user reviews," in *Proceedings of the 24th Australian Computer-Human Interaction Conference*, ser. OzCHI '12. New York, NY, USA: ACM, 2012, pp. 241–244. [Online]. Available: http://doi.acm.org/10.1145/2414536.2414577

[19] L. Hoon, R. Vasa, J.-G. Schneider, and J. Grundy, "An analysis of the mobile app review landscape: Trends and implications," Tech. rep., Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia, Tech. Rep., 2013.

[20] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *Requirements Engineering Conference (RE), 2013 21st IEEE International*, July 2013, pp. 125–134.

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

[22] C. Iacob, V. Veerappa, and R. Harrison, "What are you complaining about?: a study of online reviews of mobile applications," in *Proceedings of the 27th International BCS Human Computer Interaction Conference*. British Computer Society, 2013, p. 29.

[23] G. Bavota, M. Linares-Vasquez, C. Bernal-Cardenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "The impact of api change- and fault-proneness on the user ratings of android apps," *Software Engineering, IEEE Transactions on*, vol. 41, no. 4, pp. 384–407, April 2015.

[24] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining."

[25] R. Chandy and H. Gu, "Identifying spam in the ios app store," in *Proceedings of the 2Nd Joint WICOW/AIRWeb Workshop on Web Quality*, ser. WebQuality '12. New York, NY, USA: ACM, 2012, pp. 56–59. [Online]. Available: http://doi.acm.org/10.1145/2184305.2184317

[26] 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*, May 2013, pp. 582–591.

[27] 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*. IEEE, 2014, pp. 153–162.

[28] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944937

[29] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[30] 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*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 41–44. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487094

[31] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[32] M. Gomez, R. Rouvoy, M. Monperrus, and L. Seinturier, "A recommender system of buggy app checkers for app store moderators," Ph.D. dissertation, Inria Lille, 2014.