Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 94 (2016) 168 – 175

Procedia Computer Science

The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016)

Mining Collective Opinions for Comparison of Mobile Apps

Haroon Malik[a]*, Elhadi M. Shakshuki[b]

[a]*Weisberg Division of Computer Science, Marshall University, WV, USA*
[b]*Jodery School of Computer Science, Acadia, University, NS, Canada*

Abstract

User review is a crucial component of open mobile app market such as Google Play Store. These markets allow users to submit feedback for downloaded apps in the form of a) start ratings and b) opinions in the form of text reviews. Users read these reviews in order to gain insight into the app before they buy or download it. The user opinion about the product also influence on the purchasing decisions of potential users; indeed play a key role in the generation of revenue for the developers. The mobile apps can contain large volumes of reviews and it is impossible for a user to skim through thousands of reviews to find the opinion of other users about the features he/she is interested in. Towards this end, we propose a methodology to automatically extract the features of an app from its corresponding reviews using machine learning technique. Moreover, our proposed methodology aid user to compare the features across multiple apps, using the sentiments, expressed in their associated reviews. The proposed methodology can be used to understand user's preference to a certain mobile app and can uncover the relational behind why users prefer an app over other.

Peer-review under responsibility of the Conference Program Chairs

Keywords: Opinion mining; Google play store; Sentiment analysis

1. Introduction

The proliferation of smartphones attracts more and more software developers to devote to building mobile applications ("apps"). These developers place their app on popular distribution channels from mobile device software

* Corresponding author. Tel.: +1-304-696-5655.
E-mail address: malikh@marshall.edu

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2016.08.026

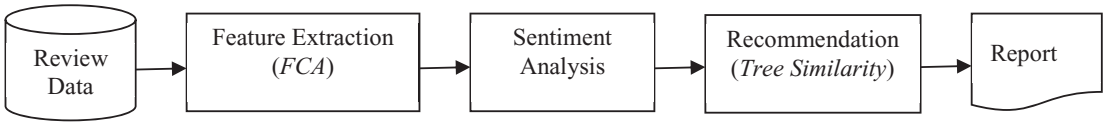

Figure 1: Overview of our proposed methodology

such as APP Store and Google Play Store. The distribution platforms allows users to search, buy and deploy software apps for mobile devices with a few clicks. These platforms also allow users to share their opinion about the app in text reviews, where they can, e.g., express their satisfaction with a specific app feature or request a new feature. Recent empirical studies[1,2,3] showed that app store reviews include information that is useful to analysts and app designers, such as user requirements, bug reports, feature requests, and documentation of user experiences with specific app features. This feedback can represent a "voice of the users" and be used to drive the development effort and improve forthcoming releases[4,5]. Moreover, the user opinion about the product also influence on the purchasing decisions of potential users; indeed play a key role in the generation of revenue for the developers.

However, there are several limitations which prevent analysts and development teams from using the information in the reviews. First, app stores include a large amount of reviews, which require a large effort to be analyzed. A recent study found that iOS users submit on average 22 reviews per day per app[6].Very popular apps such as Facebook get more than 4000 reviews per day. Second, the quality of the reviews varies widely, from helpful advice and innovative ideas to insulting comments. Third, a review typically contains a sentiment mix concerning the different app features, making it difficult to filter positive and negative feedback or retrieve the feedback for specific features. The usefulness of the star ratings in the reviews is limited for development teams, since a rating represents an average for the whole app and can combine both positive and negative evaluations of the single features.

Therefore it is difficult for both consumers and developers compare two or more mobile apps that offer similar function but with different properties. Towards this end we propose a systematic approach to mine opinions from crowd sourced reviews, i.e., App store reviews. In particular, the paper see to answer the following three research questions:

RQ. 1 What features can we extract from mobile app reviews?
RQ. 2 What are people's opinions about the products based on features extracted?
RQ. 3 How do we make recommendation based on the sentiment analysis of the extracted features?

2. Related Work

Our approach is mostly relevant to Hu and Liu[7] and Popescu and Etzioni[8]. In[7], they use Part-of-Speech (POS) tagging to collect nouns or noun phrases since features are nouns mostly. They produced POS tags for each word (whether the word is a noun or a verb). After that, association rule mining is applied to filter out the frequent feature item sets. The result of their research shows a good performance in analyzing electronic products like DVD player, MP3 player, digital camera and cellular phone. Obviously, our research is related but different from theirs in many ways. POS tagging and association rules mainly focused on noun features which may skip some words of their inputs that can imply features. For instance, there are some email mobile app that people prefer, 'multiple account support' ones rather than single account. In such condition, people may talk about their preference about "multiple account" when they refer to an app's feature. But "multiple account" is adjective in those sentences. Which means it would be filtered off when they try to sum up all the features. Our system based on the feature extraction does not have this problem. We did not remove words by part of speech. Instead, we comprehensively analyze input words from both frequency and relationship between different words. Moreover, they use comments on mobile apps from ecommerce Web sites as input. While we use data from Google Play Store that have a large number of short text with sparse words, which makes association rules not applicable. They demonstrated their algorithm with a small data set (500 records), while we tested our algorithm with more than 8,000 mobile reviews of thirty email apps. Our work is also different from the feature extraction method in[8], that they perform mining of consumer reviews and sentiment classification without comparing the pair of user-specified products based on the corresponding product features.

Table 1: Formal Context (features) across different mail apps

	Group Mail	Multiple Accounts	Signature	Quick Filters	Cloud Space	Message Search	Themes
Yahoo Mail	X	X	X		X	X	
K-9 Mail		X	X	X		X	X
Gmail	X	X	X	X	X	X	X
Blue Mail		X	X	X		X	
Microsoft Outlook	X	X	X	X		X	X
Cloud Magic			X		X	X	X

2.1. Feature Extraction Methods

When dealing with a large volume of text, we should scan through the text only once and generate a list of features or properties that can best represent the content of text. In doing so, we consider to extract the meaningful keywords and calculate their TFIDF[9] to represent text as feature vectors for the computational purpose. Feature extraction is an important step in text processing to transform input text into feature vectors. Guyon et al. [10] provided a comprehensive review on feature extraction from text data and relevant applications. Insightful discussions can also be found in [7, 13] and [13] Weka[11] has provided open source tools for feature extractions.

The task of feature extraction in this paper is to transform text data into a feature space that can best describe the interests of social network users who comment on the products or services. In brief, our feature extraction is to extract only product features[7, 12] that have appeared in the app reviews. In the feature extraction process we need to firstly search for the relevant text from reviews where the given products are mentioned, then we apply the feature extraction algorithms on the text to derive the features for those specific products.

In order to make products comparable to each other, the output product features need to be constructed as a tree structure which can be transformed from a concept lattice where some features are general and some features are specific. This requirement especially matches with the idea of discovering concept hierarchy by formal concept analysis (FCA) approaches[13].

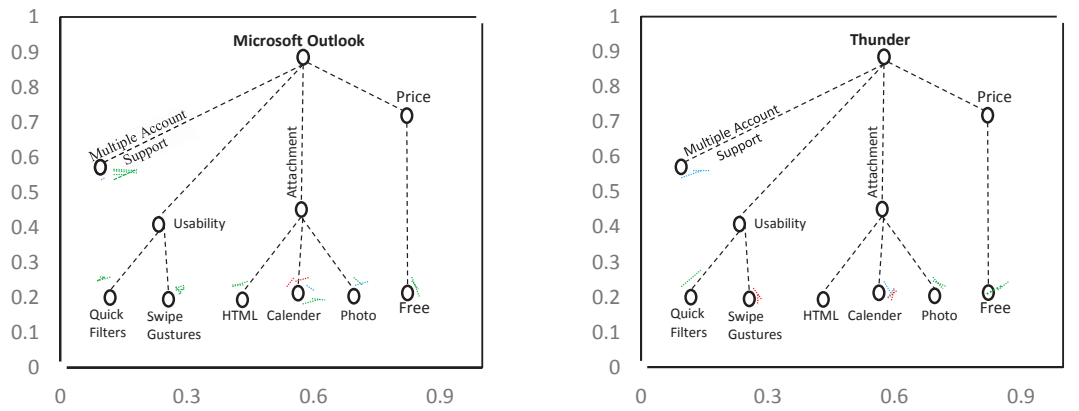

Figure 2: Sentimental analysis of hot features of two email apps (Green dots- positive; Red dots- negative; blue dots, neutral opinion)

2.2. Formal Concept Analysis

In Formal Concept Analysis, the elements of one type are called “formal objects”, the elements of the other type are called “formal attributes”. The adjective “formal” is used to emphasize that these are formal notions. “Formal

Table 2: **Algorithm Feature Extraction**

```
Input:
  TW: Pre-processes mobile app reviews
Output:
  T: Concept hierarchy of product features
Description:
  Begin
    Count tf of each word (wi) in Tw
    for (each word wi) (filter infrequent words appear in Tw)
        If tf>0.01 (set this by experiments, words' tf lower than 0.01 are not meaningful)
          add wi to word set W
for (wi1 in W) (the double loop analyses connections between any two words)
   for (wi2 in W)
      if (wi2 != wi1) {
        review collection c1, every element of it contains wi1 tweets collection c2, every element of it contains wi2
         if  (c1 \(\supset\) c2)
             wi2 is a sub-concept of wi1
             T add (wi1,wi2)
             For (any wjn that wi2 is a sub-concept of wjn)
               if  (wi1 \(\supset\) win)
                  wjn is a sub-concept of wi1
                  wi2 is a sub-concept of wjn
               else if (wjn \(\supset\) wi1)
                     wi1 is a sub-concept of wjn
                     wi2 is a sub-concept of wi1
        }
    return t
  end
```

objects" need not be "objects" in any kind of common sense meaning of "object". But the use of "object" and "attribute" is indicative because in many applications it may be useful to choose object-like items as formal objects and to choose their features or characteristics as formal attributes. In an information retrieval application, documents could be considered object-like and terms considered attribute-like. Other examples of sets of formal objects and formal attributes are tokens and types, values and data types, data-driven facts and theories, words and meanings and so on. The sets of formal objects and formal attributes together with their relation to each other form a "formal context", which can be represented by a cross table (see Table 1). The elements on the left side are formal objects; the elements at the top are formal attributes; and the relation between them is represented by the crosses. In this example, the formal objects are email app(s): Yahoo mail, K-9 mail, Gmail, Blue mail, Microsoft outlook and Cloud magic.

The attributes listed in the Table 1 describe the features, i.e., formal context of the objects; Allow group mail, support multiple accounts, facilitate constructing signature, allows message search in a mail box, provide cloud storage, and have multiple themes to enhance user experience. This is, of course, a toy example but it is sufficient to explain the basic features of FCA.

In our context, i.e., online reviews, the classical Formal Concept Analysis (FCA)[3] builds up a concept hierarchy by comparing the subset relationships amongst the associated terms of a concept. In FCA a concept can be associated with a single term or a set of terms. A term is a regarded as a meaningful word not appearing in the stop word list. When a term is used in describing a concept it is considered as an attribute of that concept. All the attributes that are associated with all concepts can be organized in a two dimensional matrix: one dimension (columns) is to list all attributes and the other (rows) is to list all the concepts. Then FCA algorithm will check the columns that corresponding to the matrix and form a lattice from that. It has been proven that there is a one-to-one mapping between each matrix and its corresponding lattice[13]. It can be seen that the critical step in FCA algorithm is to generate the attribute matrix for every concept by scanning the text only once.

3. Methodology

In this section we describe our proposed methodology. To motivate the idea behind constructing the methodology, we provide a motivating example. For email client, an app developer, after launching it via google paly store may be curious about how well the app is received by the customer and how is the app penetrating into the market. Similarly, customer shopping for an app like to know how green is an app?, i.e., is it energy notorious — since once would not

use an app if it drains the mobile phone battery quickly, or does it support polling mail from multiple email accounts? or it provides quick search of messages in mail box.

In order to answer such questions, we need to know the opinions of the people towards the features. App providers need to know what the most talked features among the customers are. We call them the '*Hot Features*'. The most talk features are the ones well received by customers as well as those features that customers complained a lot. Such a constructive feedback from users becomes extremely crucial for developers to fix bugs, implement new features, improve user experience agility as well polish the features of most interest to the users. Whereas, user need to know about the opinion of people towards the product feature. Such as opinion helps the user to locate and buy the best app when deciding among several app providing similar functionality.

A straight forward proxy to find the features claimed to be supported by a mobile app is to read its detailed description as submitted by its developer. However, similar apps can have one or set of desired features in common. For example, Microsoft outlook, Gmail, Yahoo and Blue mail, all support features, such as 'signatures', 'multiple-account support' and 'message search'. Reading the app descriptions do not guarantee how well the feature is implemented, if it is free from bug or it will provide functionality/behavior matching what is listed in the app's description[14,15] .

Like other mobile app markets, Google Play displays histograms of ratings and lists the comments/reviews by users, in addition to the app's descriptions submitted by its developer. Despite the app reviews being shorter in length (since most of them are typed and submitted from smart phones), but can range from hundreds to thousands for each app, depending upon its popularity. Therefore manually analyzing such a large volume of comments, especially quality of ratings for specific features and then comparing then against the reviews of other popular app can be hectic, time consuming and a painful process.

Towards this end to facilitate both app developers and customers to automatically compare the hot features among mobile apps, we propose our methodology. Given the set of mobile app reviews, the proposed methodology automatically extracts the hot features corresponding to the app and gauges the people opinions towards the mobile app features. The proposed system had three main steps, (1) feature extraction, (2) sentiment analysis and (3) recommendations. The input for our proposed methodology is set of reviews for one of more mobile apps. The output of the methodology is the counting of the people options toward the extracted features of the mobile apps and the set of recommendations. Below we detail the three major steps of our methodology.

3.1. *Feature Extraction*

Feature extraction step is the main contribution of the paper. The step extracts the 'hot features' of mobile app that people mostly talk/discuss and provide feedback. For the purpose all the comments/review of a particular apps are collected. The review do not suffice directly for applying our feature extraction algorithm listed in Table 2. This is due to the fact that large portion of reviews are submitted from mobile devices on which typing is not so easy. Therefore, we performed the following preprocessing steps:

- **Noun, verb, and adjective extraction.** We use the part of speech (POS) tagging functionality of the Natural Language Toolkit, NLTK4, for identifying and extracting the nouns, verbs, and adjectives in the reviews. We assume that these parts of speech are the most likely to describe features as opposed to others such as adverbs, numbers, or quantifiers. A manual inspection of 100 reviews confirmed this assumption.
- **Stopword removal.** We remove stopwords to eliminate terms that are very common in the English language (e.g., "and", "this", and "is"'). We use the standard list of stopwords provided by Lucene5 and expand it to include words that are common in user reviews, but are not used to describe features. The words we added to the stopword list are the name of the application itself, as well as the terms "app", "please", and "fix".
- **Lemmatization.** We use the Wordnet[16] lemmatizer from NLTK for grouping the different inflected forms of words with the same part of speech tag which are syntactically different but semantically equal. This step reduces the number of feature descriptors that need to be inspected later. With this process, for example, the terms describing the verbs "sees" and "saw" are grouped into the term "see".

- **Explicit Sentence.** Since people express opinions casually within app reviews, there may have either explicit or complete sentences[17], which we can easily know what they mean; or there may have implicit sentences that are incomplete sentences or just some phrases. For example, an implicit sentence in following is difficult for identifying its feature: "This game continues for a long time". In this case, it is difficult to tell whether this sentence is referring to the play time or battery life. Such sentences would have several different ways to express the same meaning which makes it even more difficult to find the patterns of features. Fortunately, we observed that those implicit sentences do not appear much in our data set (with less than 10% of the sentences). So we can focus on explicit statements in this paper and leave the process of implicit sentences to the future work.

We then apply our algorithm listed in Table 1. The algorithm can filter those words that are popular but not regarded as product features. It analyzes the processed reviews/comments of an app and finds out the hierarchy of the high term frequency-inverse document frequency (TFIDF) words. Suppose there are two random words in app reviews: w1, w2. The reviews set that contains all the appearance of word w1 is namely set c1. Similarly, the reviews set that contains all the appearance of word w2 is namely set c2. If set c1 is a superset of set c2, then more likely, w2 is a sub-concept of w1. A tree structure is used to express the hierarchy like w1, w2 instead of a lattice as it can be seen from Figure 2.

3.2. Sentiment Analysis

Sentiment analysis, which is also called as opinion mining, is an approach that's requires collection of people in real-time about a product, an event or a situation. Sentiment analysis is type of a reality check for events, people and products. People from different demographic areas have different views on certain issues. This gives a wider angle of thought to the initiator of the idea and also gives overall review on that subject or product. For example a new DSLR camera is launched with high specifications and improved technology and a website blog give all the positives of the product with all the new specifications in it[7]. But, when used by people around the world, the same product may get many negative reviews due to the DSLR's heavy nature. So these networking sites give a clear picture of the situation taking into account small details of the specific subject.

We used the approach, i.e., sentiment analysis as a part of our methodology to explore people's opinion about the hot mobile app features (features extracted using our algorithm presented in section 3.1). In general, the opinion of people can be classified into three category; *positive*, *negative* and *natural*. People use certain predictable words while giving comments or writing an app review to express their feelings. Here are two examples:

1. "*I hate the app. It keeps on crashing. Don't waste your time on it*"
2. "*This is awesome. Love it. Works with android the best*"

The first review express negative feelings towards and app. It uses a sentiments word 'hate' to express negative feeling. Whereas, the second review express positive feeling through an adjective "awesome" and a verb "love'. All these words are essential words that reflects the user's sentiment. However, there are many slang words that have no meanings such as '*Ummmm*', '*Urrr*' . '*phew*', '*oh man*' and '*huhh*'. We removed such words since they act as noise and do not contribute towards sentiment analysis. Then, we narrow down these input words again by using WordNet[16] to eliminate the words that are seldom used. We also delete the none-existing words. By tagging the existing words, a bitmap is established (listing all those existing words, tagging the existing words appeared in a review with value 1 the others with 0). Also tagging the orientation of each sentence is based on the sentimental orientation like positive, negative, or neutral. We show the result in the evaluation section. Besides, people's emotion can be divided into more types. WordNet[16] has divided some sentiment words into six types including disgust, anger, fear, sad, surprise and joy. Each of these six types shows different levels of emotions which may make the analysis more sophisticated. The taxonomy of product features provides an overview of hot features as well as the results of sentiment analysis of those features as shown in Figure 2

3.3. Recommendations

The last step of our methodology based on the qualitative and quantitative analysis on customers preferred features. The main motivation behind recommendation is that mobile app in similar category that have strong similarities. For

example, two email app(s), both support multiple accounts, signature and message search feature. If one customer likes one of these mobile app, probably he would like the other one as well. This step of our methodology employee Weighted-Tree Similarity Algorithm[18] on the extracted hot features and corresponding customer's sentiments. Due to space constraints, the recommendation algorithm is not listed in the paper. We collected six thousands reviews of two popular email apps, Microsoft outlook and Thunder. Using our methodology, we found that both of the email app are well received by the customers. Nevertheless, we use two thousands of reviews for each of the mobile app to plot the Figure 2.

Our system is implemented in Python. We crawled thirty reviews of thirty popular email clients. Our review dataset consisted of six thousands reviews stored in MySQL Database. First we preprocessed all the reviews. Then extracted the features for each of the app to construct a feature tree. For each of the feature corresponding review are analyzed to find out the opinion of the customers, i.e., applied sentiment analysis technique. Finally, we explored the similarity between the feature trees using tree-similarity comparison algorithm. Each app received a similarity score (between 0 and 1). For each app we recommended apps with the similarity threshold of 0.7 and above. Since the pool of our mobile email app is of managed size, i.e., thirty mobile app, we validated the results manually.

4. Conclusions and Future work

With the popularity of smartphones and mobile devices, mobile application (i.e., apps) market have been growing exponentially in terms of user and downloads. App developers spend considerable effort on collecting and exploiting user feedback to improve user satisfaction. On other hand, users use the comments/reviews to get insight into the experience, opinions and sentiments of other users about specific features and descriptions of experiences with these features. However, for many apps, the amount of reviews is too large to be process manually and their quality varies largely. Therefore, towards, this end, we proposed a methodology that automatically extracts the hot features of the mobile apps from the reviewers comment, mine the feelings of users towards those features and recommend them the mobile app with similar hot features.

The future research will improve on the scalability and effectiveness of our proposed methodology of mining social opinions on wide category of apps from different App stores.

References

1. L. V. Galvis Carreño and K. Winbladh. Analysis of user comments: an approach for software requirements evolution. In ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, pages 582–591. IEEE Press, May 2013.
2. M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: MSR for app stores. In Proc. of Working Conference on Mining Software Repositories - MSR '12, pages 108–111, June 2012.
3. D. Pagano and W. Maalej. User feedback in the appstore : an empirical study. In Proc. of the International Conference on Requirements Engineering - RE '13, pages 125–134, 2013.
4. W. Maalej and D. Pagano. On the Socialness of Software. In 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pages 864–871. IEEE, Dec. 2011.
5. N. Seyff, F. Graf, and N. Maiden. Using mobile re tools to give end-users their own voice. In Requirements Engineering Conference (RE), 2010 18th IEEE International, pages 37–46. IEEE, 2010.
6. D. Pagano and W. Maalej. User feedback in the appstore : an empirical study. In Proc. of the International Conference on Requirements Engineering - RE '13, pages 125–134, 2013.
7. Hu, M., Liu, B.: Mining opinion features in customer reviews, Proceedings of the National Conference on Artificial Intelligence. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; pp. 755–760 (2004)
8. Popescu, A.M., Etzioni O.: Extracting Product Features and Opinions from Re views. In Natural Language Processing and Text Mining, 2007, pp.9-28
9. Salton, G.,Buckley,C.: Term-weighting approaches in automatic text retrieval. Information processing & management, 24(5):513-523 (1988)
10. Guyon, I., et al (ed) : Feature Extraction: Foundations and Applications, Springer,(2006)
11. WEKA The University of Waikato, http://www.cs.waikato.ac.nz/ml/weka/
12. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment classification using machine learning techniques. Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002, pp.79–86.

13. Ganter, B., Wille,R.: Applied lattice theory: Formal concept analysis, In General Lattice Theory, G. Gr¨atzer editor, Birkha¨ user (1997)
14. Siqi Ma, Shaowei Wang, David Lo, Robert Huijie Deng, and Cong Sun. 2015. Active Semi-supervised Approach for Checking App Behavior against Its Description. In *Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference - Volume 02* (COMPSAC '15), Vol. 2. IEEE Computer Society, Washington, DC, USA, 179-184.
15. Alessandra Gorla, Ilaria Tavecchia, Florian Gross, and Andreas Zeller. 2014. Checking app behavior against app descriptions. In *Proceedings of the 36th International Conference on Software*
16. WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press, http: //wordnet.princeton.edu
17. Hu, M., Liu, B.: Mining and Summarizing Customer Reviews. In: Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining, pages 168–177. ACM Press New York, NY, USA (2004)
18. Bhavsar, Virendrakumar C., Boley, Harold., Yang, Lu. A Weighted-Tree Similarity Algorithm for Multi-Agent Systems. In: Proceedings of E-Business Environments Computational Intelligence, volume 0, issue 4, pages 584-609, d004.