ALPACA: Mining user opinions in mobile app reviews with linguistic patterns

Phong Minh Vu, Tam The Nguyen, Tung Thanh Nguyen
Auburn University
lenniel@auburn.com,tam@auburn.com,tung@auburn.edu

ABSTRACT

Mobile app development is increasingly competitive as millions of apps are available on major app markets. To improve overall customer satisfaction, mobile app developers need to know user opinions on specific features of their apps. In this paper, we introduce ALPACA, an automated tool for extracting relevant opinions, such as complaints, problem reports, feature requests, or product comparisons from users' app reviews. ALPACA allows users to define their topics of interests and employs pre-defined linguistic patterns to extract phrases in raw reviews matching such topics and patterns. Github page and video demo: https://github.com/phong1990/ALPACA

KEYWORDS

mobile app, review, response

1 INTRODUCTION

Millions of mobile apps are available on major app markets such as Apple AppStore and Google Play, making mobile app development increasingly competitive. Business models of most apps, especially free apps, are strongly dependent on users' adoption and engagement (e.g., displaying ads, purchasing in-app items, or subscriptions). Thus, it is important for app developers to know users' needs to make them satisfied and continue to use the apps.

An online app store typically provides a review system allowing mobile apps' users to write reviews about such apps. Prior studies suggest that such user reviews contain important user opinions, such as reporting a bug or requesting a new feature [4, 6, 21]. App developers can extract relevant users' opinions from their reviews to understand users' needs and improve the apps accordingly.

However, app reviews are typically numerous and noisy [4]. Reading those reviews manually to find relevant user opinions would be ineffective and time-consuming. Therefore, researchers have introduced several automated tools to analyzing and extracting user opinions from mobile app reviews. However, those tools do not support or would work ineffectively in some situations in practice.

Let us illustrate that via an example. 'Magic Tiles 3' is a piano tapping game for mobile phone developed by AMANOTES [1]. According to Google Play, this mobile app has between 50 to 100 million downloads and more than 500 thousands user ratings. The reviews are generally positive with an average rating of 4.5. Because this is a song-related app, the app's developers have told us that they want to know what songs the users want to add to the app' library. However, it would be impossible to read through thousands of reviews to look for user opinions on that topic.

Current automated tools for app review mining could not effectively extract such opinions. For example, AR-Miner [4] could predict if a review contain useful information, but could not determine if such useful information is about what songs the users want to add. SURF [6] could summarize user opinions in reviews into different intentions (e.g., feature requests or bug reports). However, it could only find opinions involving general topics, such as *"user interface"* or *"security"*. Meanwhile, the app developers are interested in a topic specific for this app (about "songs" for a piano tapping game), thus, SURF could not effectively identify such opinions. MARK [21] allows app developers to define a topic of interest, such as *"songs"* in this case, via relevant keywords. However, it can only export reviews containing keywords of that topic, so a user still need to read its report manually to identify reviews specially mention what songs to add to the app's library.

In this paper, we introduce ALPACA, an automated tool for extracting relevant opinions, such as complaints, problem reports, feature requests, or product comparisons from users' app reviews. ALPACA allows users to define their topics of interests and employs pre-defined lingusitic patterns to extract phrases in raw reviews matching such topics and patterns. In the next section, we will demostrate how ALPACA can effectively extract the users' opinions on app-specific topics from raw reviews.

2 TOOL USAGE

Figure 1 is the main working screen of ALPACA. It is implemented as an independent Java-based application. The detailed installation instructions and data for replication can be found at its webpage: https://github.com/phong1990/ALPACA

2.1 Extracting opinions about songs and music request of Magic Title 3

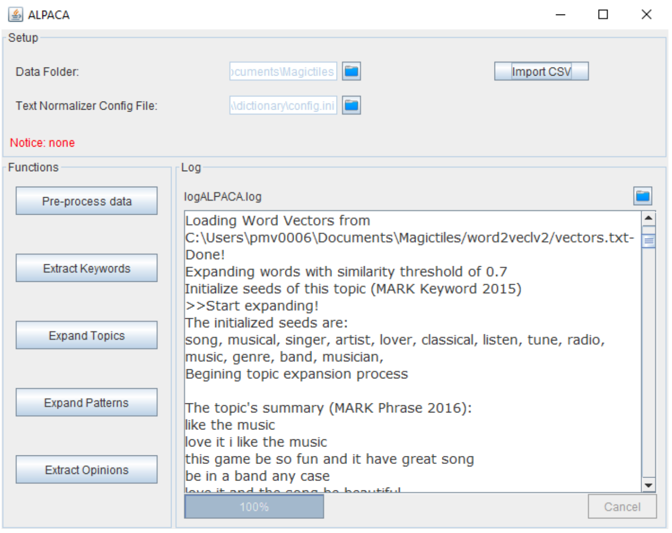
To extract opinions in app reviews from users of 'Magic Title 3' about their suggestions for what songs to add to the app music library, we operate ALPACA by the following steps.

Step 1. Setup a working data folder. To use ALPACA, we need to provide it the app reviews organized in a data folder. ALPACA provides a function to extract reviews from `csv` files and populate the user-specified data folder for it. Our replication package include a `csv` file for about twelve thousands reviews of Magic Tiles 3.

Step 2. Pre-process data. Before the data can be analyzed, ALPACA needs to pre-process it. In this stage, ALPACA will also need to produce a word2vec [14] model and a list of POS tag patterns [15]

ESEC/FSE '18, FSE
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3194972

Figure 1: **GUI of ALPACA**



```
Loading Word Vectors from
C:\Users\pmv0006\Documents\Magictiles/word2veclv2/vectors.txt-
Done!
Expanding words with similarity threshold of 0.7
Initialize seeds of this topic (MARK Keyword 2015)
>>Start expanding!
The initialized seeds are:
song, musical, singer, artist, lover, classical, listen, tune, radio,
music, genre, band, musician,
Begining topic expansion process

The topic's summary (MARK Phrase 2016):
like the music
love it i like the music
this game be so fun and it have great song
be in a band any case
love it and the song be beautiful
```

from the data. We also included a text corpus of 3 millions reviews in our package to help word2vec produces a more accurate model.

Step 3. Analyze keywords. Internally, ALPACA uses keywords and their rankings to find topics. Therefore, ALPACA provides a function to rank keywords in different ranking models. Advanced users can also add their own rankings to influence the final results.

Step 4. Expand topic "song and music". In MARK [21], we discovered that a topic can be represented by a collection of keywords that have the similar context semantics. However, developers or review analysts may not have the full vocabulary of a topic readily available to them. Therefore, ALPACA provides a function to further expand a given set of keywords into a bigger, more complete topic with descriptions. The keywords were found using the keyword similarity technique introduced in MARK and the descriptions of them are phrases extracted with the phrase-based miner[22]. The descriptions can be used to verify if our keywords are really representing the topic of interest.

In this example, we want to learn more about the topic of music and songs for Magic Tiles 3. Therefore, we put "music,song" as the initial keywords in the "Expand Topic" function of ALPACA. The results are two files: a keyword file containing the relevant keywords (`song, music, musical, single, artist, lover, classical, listen, tune, radio, genre, band, musician`) and a description file containing the phrases describe that topic.

Step 5. Expand request intention patterns. Prior works [6, 12] suggested that intentions often follow some linguistic patterns. For example, "`add [something] in [something]`", or "`should have [something]`" are linguistic patterns of request. Therefore, ALPACA uses intention linguistic patterns to match with phrases that have the intention of interest. For example, to find out if a review contains the intention or not, ALPACA would tag the first review in Table 1 as following:

`Can[MD] u[PRP] add[VB] ‘‘some hindi music’’[COMPNN] in[IN] it[PRP]. It[PRP] will[MD] be[VB] awesome[JJ].`

With these tags, ALPACA can match the request pattern "add `[something] in [something]`" with the phrase "add some hindi music in it". Note that we use COMPNN (a.k.a. noun combination) instead of a single POS tags for each word because it represents the semantic meaning of `[something]` in a linguistic pattern. Similarly, the fifth review was tagged as the following:

`I[PRP] loved[VBD] it[PRP] .But[CC] I[PRP] think[VB] they[PRP] should[MD] have[VB] ‘‘faster music’’[COMPNN] otherwise[CC] it[PRP] ’s[VB] 10[CD] out[IN] of[IN] 10[CD]`

In which, ALPACA found the phrase "they should have faster music" very similar to "`should have [something]`".

In practice, we do not use general notions like [something] or [someone] in the patterns, as they are not recognized by the POS tagger. Instead, we just use the POS tags. ("`should have [something]`" will become "`should have [COMPNN]`" and "`add [something] in [something]`" would become "`add COMPNN in COMPNN/PRP`")

With this pattern matching process, it is vital that we have a high coverage set of patterns for each intention. Before, linguistic patterns usually have to be manually derived from text [7], however, ALPACA can learn and improve the set of pattern directly from data with the "Expanding Intent Pattern" function. It looks through the POS sequences generated in the pre-processing step, and finds similar patterns to our initial pattern set. We discuss more about this in section 3.2.

For this example, we chose to expand the default request patterns provided by ALPACA. The result of this expansion process is a bigger and more data-driven patterns set for the request intention. This would greatly improve the coverage of opinion extraction and always be recommended. Beside the two default intention patterns provided by our tool, we encourage users to check our github place frequently for other intention patterns.

Step 6. Extract opinions. In this last step, we choose the keyword set obtained from step 4 and the request pattern set expanded in step 5. The final results are highlighted requests ALPACA found in reviews like in Table 1. We have sent this result to the developers at Magic Tiles 3, and they have confirmed that these requests were the same as what their human operators have collected manually from the reviews.

Table 1: Examples of highlighted reviews for requesting songs opinions in Magic Tiles

Date	Rating	Review text
9/17/2017	5	Can u add some hindi music in it. It will be awesome
10/22/2017	4	Pls add more of yiruma piano songs
11/25/2017	5	I love the game but the company should add some Christmas songs from the radio 97.5! (THIS WAS POSTED 11-25-17)
11/13/2017	5	You need to add "build our machine" to the music
11/25/2017	5	I loved it .But I think they should have faster music otherwise it's 10 out of 10
07/06/2017	3	You should add custom songs to play

2.2 Extracting complaints about songs and music of Magic Tiles 3

Similar to the previous example, we want to know what kind of complaint users have made about songs and music. For this example, we do not need to pre-process the data again, nor extract keywords, nor expanding topic again. Hence, there are only two steps:

Step 1. Expand complaint intention patterns. This time, we choose "Complaints" instead of "Requests" and expanded it.

Step 2. Extract opinions. In this step, we choose the keyword set obtained from step 4 of the previous example and the complaint patterns set expanded in step 1. The final results are highlighted opinions ALPACA found in reviews like in Table 2.

The results show a trend of complaining about how the songs and the input tiles are not really in sync to each other. The team at Magic Tiles 3 have confirmed that this is a known problem of the game, and they were really impressed by the feedback of ALPACA.

Table 2: Examples of highlighted reviews for requesting songs opinions in Magic Tiles

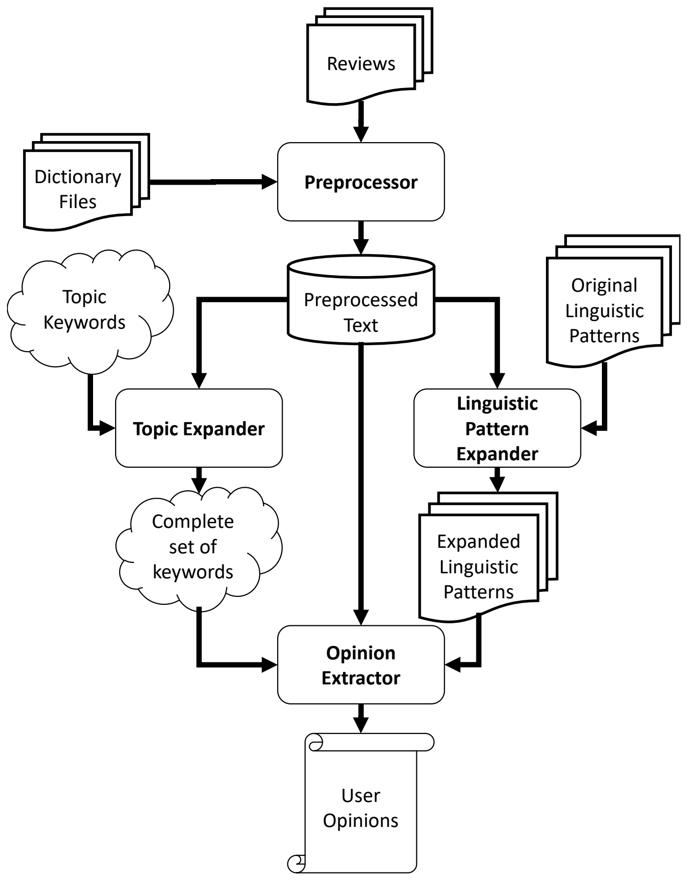
Date	Rating	Review text
11/07/2017	2	Song does not sync with my Samsung s8+. When fixed, will re-rate.
05/03/2017	1	The songs doesnt match the tiles, its so disturbing
10/01/2017	1	DO NOT INSTALL THIS GAME, IT IS A FAKE , YOU DON'T ACTUALLY PLAY THE SONG, THE SONG IS ALREADY PLAYING AND ALL YOU DOING IS TAPPING
10/26/2017	4	I think the music doesnt match the piano but love it still
10/21/2017	1	The game is a complete copy of the original piano tiles also most of the songs do not go with the tiles, that makes the tiles point less please aline the song with tiles and make to the song sound like played by piano.

3 SYSTEM DESIGN

3.1 Overview of ALPACA

Figure 2 illustrates the general architectural design of ALPACA. It has four major components: Pre-processor, Topic Expander, Opinion Extractor, and Pattern Expander. The Pre-processor module pre-processes raw reviews (e.g., extracts keywords, assigns part-of-speech (PoS) tags, or computes word vectors). Then, when the user provides a collection of keywords as the topic of interest and a collection of linguistic patterns as the intention of interest, the Opinion Extractor will search in the pre-processed reviews for ones having matches with the provided keywords and patterns and generates a report highlighting those results. Due to the variation in writing styles, there might be several similar patterns expressing the same intention (e.g., people can use different ways to write a request such as *"can you [do] [something]"* or *"you should [do] [something]"*). To address that problem, the Pattern Expander in ALPACA can expand an initial collection of pre-selected patterns with similar ones mined from actual reviews. Similarly, a topic can be described

Figure 2: **Overview of ALPACA**



in reviews by more keywords than user provided. Topic Expander can expand a small set of keywords into a complete set and help user validates it by providing descriptions of the expanded topic.

3.2 Linguistic Pattern Similarity (LPS) Measurement

A generic pattern can be something like *"can you [do] [something]"* that has a specific intention. However, even this generic pattern can varies into slightly different patterns with the same intention as well (e.g. *"you should [do] [something]"* or even more erratic as *"you have to [do] [something] about [something]"*). These kinds of variation could be limitless, despite of implying the same intention, which is a request. Further observation suggested that similar patterns often share a similar set of functional words (e.g. `to, have, should, can`, etc) and domain generic words (e.g. `add, rate, fix` in app review domain) in the same order (e.g. "have to" and "to have" describe two entirely different intentions) . Other words usually can be replaced in the placeholders to create new sentences and have little impact on how an intention is interpreted.

Therefore, by finding patterns of the same functional/domain generic words order, we can find more patterns of similar intentions.

ALPACA developed a Linguistic Pattern Similarity measurement using JACCARD similarity [17] of sub-sequences of those words from both patterns. Note that this measurement is based on the assumption that most patterns with similar functional words order have the same intention. There are cases that the more general functional words can combine with non-functional words to create patterns of several different intentions at once. Currently, to the best of our knowledge, there is no definite formula for the way English language is used and it is not yet possible to determine such cases. Therefore, we do not consider them in this ALPACA.
ϕ_i : Linguistic patterns i
ξ_i : set of functional subsequences belong to ϕ_i, in *Strict* version, this set include both functional and non-functional words

$$patternSim(\phi_1, \phi_2) = \frac{|\xi_1 \cap \xi_2|}{|\xi_1 \cup \xi_2|} \quad (1)$$

3.3 Pattern Matching Algorithm

To match the patterns with a text sequence, we first tag the text with corresponding POS tags before grouping the tags into compound tags (e.g. adjective + noun = compound noun). This helps greatly reduce the complexity of POS tags sequences and allow more matches. After this step, the tag sequence of a text would be greedily scanned against the intention patterns to single out the sequences with similarity crossed a threshold. We do not parse the grammar tree because patterns are usually already grammatically correct and traversing the tree is resource taxing. ALPACA offers several different thresholds for user to choose from (i.e. Exact Match - 1.0, Strictly Match - 0.8, Moderately Match - 0.7, Flexible Match - 0.6, and Loosely Match - 0.45). More detail of our algorithm and evaluations is discussed in our technical report [20].

3.4 Pattern Expanding Algorithm

Unlike pattern matching, expanding pattern is a more delicate process, as if a pattern is not grammatically sensible, it would match to the wrong sequence and return incorrect opinions. Therefore, before expanding the pattern from the data, we tasked the preprocessing stage with extracting all phrasal POS tag patterns of the data. This is done using Stanford Parser [5] by traversing the whole parse tree of each review to find the phrases. Despite of being resource heavily taxing, this step is only needed to be done one time for each dataset.

With the phrase patterns extracted from the dataset, we use our LPS measurement on each phrases against the set of seeding intention patterns. The chosen patterns are the ones passed a preset threshold. More detail of our algorithm and evaluations are discussed in our technical report [20].

4 RELATED WORK

In the past, there is a number of empirical and exploratory studies on the importance of app's reviews in app development process such as [2, 11, 13, 16, 18, 19]. These studies have greatly inspired the work of ALPACA.

There were several authors focusing on mining useful information from user's reviews, such Chandy et al. [3], Carreno et al. [9], Guzman et al[10]. While some other provided complete tool sets or prototypes such as Wiscom [8] or MARA[12], or AR-Miner [4].

Later on, MARK [23] proposed a keyword based approach to discovering topics and trends using their semantic meaning. However, MARK did not try to capture the intention of user on their reviews.

The most closely related work to ALPACA would be SURF [6] from Di Sorbo et al. This work helps developers find both intention and topic from reviews. However, the range of topics is limited within a number of predefined general topics and their linguistic patterns needed to be derived manually by human. It inspired our work to introduces a method for automatically expanding the linguistic patterns and match them by user defined topic. In other words, ALPACA provides an unbounded, flexible, and robust environment for opinions discovering on user reviews.

5 CONCLUSIONS

To improve overall customer satisfaction, app developers are interested in user opinions on specific features of their apps. This paper introduces ALPACA, an automated tool for extracting relevant opinions, such as complaints, problem reports, feature requests, or product comparisons from users' app reviews.

REFERENCES

[1] AMANOTES. 2017. Magic Tiles 3. (2017). https://play.google.com/store/apps/details?id=com.youmusic.magictiles
[2] G. Bavota, M. Linares-Vasquez, C.E. Bernal-Cardenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk. 2015. The Impact of API Change- and Fault-Proneness on the User Ratings of Android Apps. *SE* (2015).
[3] R. Chandy and H Gu. 2012. Identifying Spam in the iOS App Store *(WebQuality '12).*
[4] N. Chen, J. Lin, S. CH. Hoi, X. Xiao, and B. Zhang. 2014. AR-miner: mining informative reviews for developers from mobile app marketplace. In *ICSE.*
[5] M.-C. De Marneffe, B. MacCartney, C. D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *LREC.*
[6] A. Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A. Visaggio, G. Canfora, and H. C. Gall. 2016. What would users change in my app? summarizing app reviews for recommending software changes. In *FSE.*
[7] A. Di Sorbo, S. Panichella, C. A. Visaggio, M. Di Penta, G. Canfora, and H. Gall. 2016. DECA: development emails content analyzer. In *ICSE.*
[8] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh. 2013. Why People Hate Your App: Making Sense of User Feedback in a Mobile App Store *(KDD '13).*
[9] L.V. Galvis C. and K. Winbladh. 2013. Analysis of user comments: An approach for software requirements evolution. In *ICSE.*
[10] E. Guzman and W. Maalej. 2014. How do users like this feature? a fine grained sentiment analysis of app reviews. In *RE.*
[11] L. Hoon, R. Vasa, J. Schneider, and J. Grundy. 2013. *An Analysis of the Mobile App Review Landscape: Trends and Implications.* Technical Report.
[12] C. Iacob and R. Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *MSR.*
[13] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan. 2015. What do mobile app users complain about? *IEEE Software* (2015).
[14] Tomas M., Kai C., Greg C., and Jeffrey D. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR* (2013).
[15] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics* (1993).
[16] William Martin, Mark Harman, Yue Jia, Federica Sarro, and Yuanyuan Zhang. 2015. The App Sampling Problem for App Store Mining. (2015).
[17] Gary W Oehlert. 2000. *A first course in design and analysis of experiments.* Vol. 1. WH Freeman New York. http://users.stat.umn.edu/~gary/book/fcdae.pdf
[18] D. Pagano and W. Maalej. 2013. User feedback in the appstore: An empirical study. In *RE.*
[19] R. Vasa, L. Hoon, K. Mouzakis, and A. Noguchi. 2012. A preliminary analysis of mobile app user reviews. In *OzCHI.*
[20] P. M. Vu, T. T. Nguyen, and T. T. Nguyen. 2018. *Mining User Opinions in Mobile App Reviews with Linguistic Patterns.* Technical Report.
[21] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen. 2015. Mining user opinions in mobile app reviews: A keyword-based approach (t). In *ASE.*
[22] P. M. Vu, H. V. Pham, T. T. Nguyen, et al. 2016. Phrase-based extraction of user opinions in mobile app reviews. In *ASE.*
[23] P. M. Vu, H. V. Pham, T. T. Nguyen, and T. T. Nguyen. 2015. Tool support for analyzing mobile app reviews. In *ASE.*