AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace

Yifan Zhao
North Carolina State University
yzhao48@ncsu.edu

ABSTRACT

App developers spend considerable effort on collecting and exploiting user feedback to improve user satisfaction, but suffer from the absence of effective review analytics tools. To facilitate mobile app developers discover the most "informative" reviews from a large and rapidly increasing pool of user reviews, a novel computational framework for App Review Mining, "AR-Miner", is presented, which performs comprehensive analytics from raw reviews by first extracting informative reviews by filtering noisy and irrelevant ones, then grouping the informative reviews automatically using topic modeling, further priortizing the informative reviews by an effective ranking scheme, and finally presenting the groups of most "informative" reviews via an intuitive visualization approach.

KEYWORDS

User feedback, mobile application, user reviews, data mining

1 INTRODUCTION

There are two outstanding obstacles for app developers to obtain valuable information from App Store or Google Play. First of all, the proportion of "informative" review is relatively low. Second, for some popular apps, the volume of user reviews is simply too large to do manual check on all of them.

In this poster, I will present the the work of Chen, et al[1]. Their main contributions are as follows:

- Formulated a new problem that aims to discover the most "informative" information from raw user reviews in app markets;
- Presented AR-Miner as a novel analytic framework to tackle this new problem, which includes a new and promising ranking scheme to prioritize the "informative" reviews;
- Evaluated AR-Miner based on user reviews of four Android apps, in which empirical results show that AR-Miner is effective.

2 METHODOLOGY AND FRAMEWORK

Consider an individual app, in a time interval $\mathcal{T}$, it receives a list of user reviews $\mathcal{R}*$ with an attribute set $\mathcal{A} = \{A_1, A_2, ..., A_k\}$, and $r_i = \{r_i.A_1, r_i.A_2, ..., r_i.A_k\}$ is the i-th review instance in $\mathcal{R}*$. In this work, they chose $\mathcal{A} = \{Text, Rating, Timestamp\}$ since they are the common attributes supported in all mainstream app marketplaces. In this problem, each r_i in $\mathcal{R}*$ is either "informative" or "non-informative".

Foundations of Software Science, Fall 2018, Computer Science
© 2018 North Carolina State University
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$9999.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

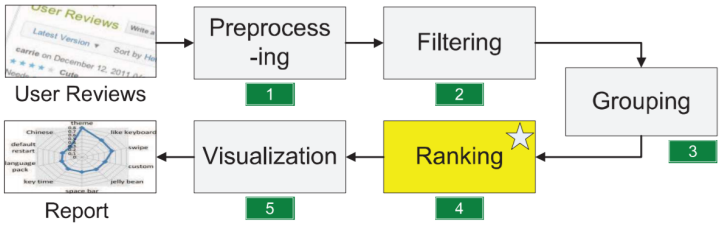

Figure 1: Visulization of Top-10 ranked results achieved by AR-Miner (SwiftKey)

The framework consists of five steps: preprocessing, filtering, grouping, ranking (most important) and visualization as shown in Figure 1.

2.1 Preprocessing

The first step of AR-Miner preprocesses the collected raw data by converting the raw user reviews into sentence-level review instances.The *Text* attribute of a raw review often consists of more than one sentence. In this work, the authors split *Text* into several sentences via a standard sentence splitter provided by LingPipe[2]. For each sentence, a review instance r_i with *Rating* and *Timestamp* equal to the values of the corresponding raw review is generated.

2.2 Filtering

To filter "non-informative" review instances, Expectation Maximization for Naive Bayes (EMNB) is adopted in this work. Because in this problem, they could get a mass of unlabeled data almost freely, but labeling training data is time consuming and labor intensive. And EMNB can use a small amount of labeled data along with plenty of unlabeled data to train a fairly good classifier.

2.3 Grouping

This step is to partition the remaining review instances ($\mathcal{R}$) in several groups such that the *Text* of review instances in a group is more semantically similar to each other than the *Text* of reviews instances in other groups.

The authors adopted *topic modeling* to group the review instances, which could assign multiple topics to each review instance. For example, the review "Just add emojis and more themes." is modeled as a distribution over two topics (50% "emoji", 50% "theme").

2.4 Ranking

The step of ranking is aimed to determine the relative importance of groups.To measure the importance of different groups, the set

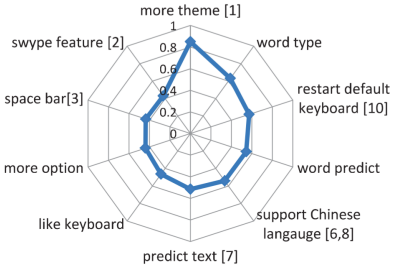

Figure 2: Visulization of Top-10 ranked results achieved by AR-Miner (SwiftKey)

of functions $f^G = \{f^G_{Volume}, f^G_{TimeSeries}, f^G_{AvgRating}\}$ is used in this work.

Volume: The volume of group g is computed by

$$f^G_{Volume}(g) = \sum_{i=1}^{n} p_{r_i g}$$

where $p_{r_i g}$ is the probability of review instance r_i being in group g.

Time Series Pattern: Set $v(\mathcal{T}_k)$ as the total number of review instances posted during a given time window $\mathcal{T}_k$. Then for a group g, the volume of review instances posted during $\mathcal{T}_k$ is $v(g,k) = \sum_{r_j \in \mathcal{R}_{\mathcal{T}_k}} p_{r_j g}$ Then

$$f^G_{TimeSeries}(g) = \sum_{k=1}^{K} \frac{p(g,k)}{p(g)} \times l(k)$$

where $p(g,k) = v(g,k)/v(\mathcal{T}_k)$, $p(g) = \sum_{k=1}^{K} p(g,k)$ and $l(k)$ is a monotonically increasing function of k, since later $\mathcal{T}_k$ should have more weight.

Average Rating:

$$f^G_{AvgRating}(g) = \frac{f^G_{Volume}(g)}{\sum_{i=1}^{n} p_{r_i g} \times r_i.R}$$

where $r_i.R$ is the rating of given review instance and smaller $r_i.R$, i.e., larger $f^G_{AvgRating}(g)$ indicates more importance.

2.5 Visualization

The last step of AR-Miner is to visualize the results generated by the ranking model. Figure 2 shows a possible visulization of top-10 ranked results.

3 EMPIRICAL EVALUATION

3.1 Dataset

Four Android apps was selected from Google Play, i.e., SwiftKey (keyboard), Facebook (social), Temple Run 2 (game), and Tap Fish (game). These apps are selected because they cover different app domains and range from large datasets (Facebook and Temple Run 2) to relatively small datasets (SwiftKey and Tap Fish).

3.2 Performance Metrics

The first set of metrics includes Precision, Recall and F1 score. In addition, the well-known Normalized Discounted Cumulative Gain

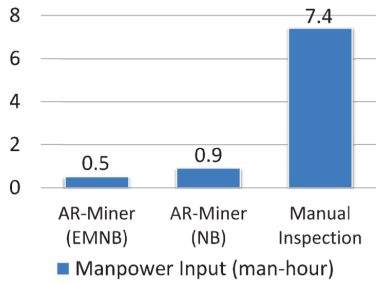

Figure 3: Evaluation Results on Facebook, Manpower Comparison with Manual Inspection

(NDCG) is also adopted as a measure for evaluating the quality of top-k ranking results:

$$NDCG@k = \frac{DCG@k}{IDCG@k}$$

where NDCG@k∈[0,1], and the higher value implies greater agreement between the predicted rank order and the ideal rank order.

3.3 Comparison with Manual Inspection

In this case study, the authors applied three schemes: (i) AR-Miner with EMNB filter; (ii) AR-Miner with NB filter; and (iii) purely manual inspection, respectively, to the test set of the Facebook dataset. Figure 3 has shown the approximate manpower input of the first author for finding the most "informative" information by these three schemes. First, AR-Miner (EMNB filter, 0.5 man-hours) is much more efficient than purely manual inspection (7.4 man-hours). The reason is that AR-Miner only requires humans to label some training data. Second, AR-Miner (NB) needs more human efforts than AR-Miner (EMNB) since building an NB filter whose performance is comparable to an EMNB filter requires manually labeling more training data.

4 LIMITATIONS

Despite the encouraging results, this work has two potential threats to validity. First, the authors are not professional app developers, and thus the defined category rules of informativeness might not be always true for real app developers. The second threat relates to the generality of the framework. It is unclear that if this framework could attain similar good results when being applied to other kinds of Android apps and apps on other platforms.

5 CONCLUSION

The paper presented AR-Miner, a novel framework for mobile app review mining to facilitate app developers extract the most "informative" information from raw user reviews in app marketplace with minimal manual effort.

REFERENCES

[1] Ning Chen, Jialiu Lin, Steven Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-Miner: mining informative reviews for developers from mobile app marketplace. In *2014 the International Conference on Software Engineering*. ACM, 767–778.
[2] LingPipe. http://alias-i.com/lingpipe/.