AR-Tracker: Track the Dynamics of Mobile Apps via User Review Mining

Cuiyun GAO*†, Hui XU*†, Junjie HU†, and Yangfan ZHOU*‡
*Shenzhen Research Institute, The Chinese University of Hong Kong, China
†Dept. of Computer Science and Engineering, The Chinese University of Hong Kong, China
‡School of Computer Science, Fudan University, Shanghai, China
{cygao, hxu, jjhu, yfzhou}@cse.cuhk.edu.hk

***Abstract*—User-generated reviews on mobile applications (apps) are a valuable source of data for developers to improve the quality of their apps. But the reviews are usually massive in size and span over multiple topics, thus leading to great challenges for developers to efficiently identify the key reviews of interest. In recent studies, automatic user review mining has been recognized as a key solution to address this challenge. The existing methods, however, require extensive human efforts to manually label the training data. Besides, they only analyze the static characteristics over the whole set of collected reviews, while ignoring the dynamic information embedded in the reviews of different time periods. In this paper, we propose 'AR-Tracker', a new framework to mine user reviews without the need of human labeling and track the dynamics from the top-ranked reviews. Through extensive experiments on the reviews of four popular mobile apps collected over 7 months, we show that AR-Tracker can still achieve comparable accuracy with the state-of-the-art methods, e.g., AR-Miner. Additionally, a case study on Facebook reviews further validates the effectiveness of 'AR-Tracker' in tracking the dynamics.**

***Keywords*-user reviews; dynamics; mobile apps**

I. Introduction

The portability of the Dalvik virtual machine [1] promotes the proliferation of mobile apps. In general software development process, developers constantly update their apps according to the user feedbacks, e.g., reviews and ratings [2]. In the area of mobile applications, constant updates make a great impact on the app's growth trajectory, being hot or abortive. For instance, with refinements based on numerous reviews, the game app "Flappy Bird" shot to the top of the App Store with zero marketing spent, a spot estimated to cost over $80,000 through customer acquisition [3]. Handling massive user reviews manually is prohibitively time-consuming for developers, especially for popular apps. How to extract valuable information from user feedback is a critical problem yet to be well addressed [4].

Different from other online reviews (such as shopping reviews and hotel reviews), user reviews of mobile apps are (a) generally shorter in length as a majority of them are posted via mobile devices; (b) often specific to a particular app version and vary over time [5]. Existing work focuses more on topic-extracting from user reviews on mobile apps [6, 7]. The dynamics of the topics are however largely ignored [5]. We notice that apps are typically rapidly evolving software: New versions of an app come out with its constant modification by the developer [8]. User feedback often reflects only the problems in the current version of the app. Hence, we can track the dynamics of user feedback to help developers determine the bugs they need to fix or features to add [9].

To this end, we propose a novel tool, namely 'AR-Tracker' (App Review Tracker), which can extract and visualize the main themes from the raw user review data in a dynamic version. In order to find the most appropriate modeling method for our framework, we compare various topic modeling algorithms. Regarding how to rank the topics mined from user reviews, existing approaches consider only the features, such as ratings and dates that can explicitly reflect the user feedback, while largely overlooking the implicit relations between topics. In contrast, the ranking scheme we propose takes both the topic similarities [10] and other user-generated information into account. We compare our algorithm with the state-of-the-art algorithm in [4]. Then we analyze the dynamics of the main user feedback through mining user reviews. Finally, we visualize the results in an interpretable way.

The paper makes the following contributions.

- We propose a new topic-ranking and review-ranking scheme to prioritize the user reviews, which can help developers capture the most up-to-date issues with proper topic modeling.
- We trace the changes of main user reviews on mobile apps over time, which can help developers understand the user demands and shed light to the new version design.
- We visualize the results in a user-friendly way, so that developers can observe the trends of hot issues clearly.
- We evaluate our framework by comparing it with the state-of-the-art work based on a large-scale experiment involving more than 500k user reviews.

The rest of the paper is organized as follows: we discuss related work in Section II; Section III gives the problem setting; in Section IV, we introduce the AR-Tracker framework; and Section V illustrates the empirical results; finally the conclusion is described in Section VI.

978-1-4799-8356-8/15 $31.00 © 2015 IEEE
DOI 10.1109/SOSE.2015.13

II. RELATED WORK

Two lines of work inspire the design of AR-Tracker, namely short-text mining and mining user reviews on mobile apps.

A. Short-text Mining

Recently, short-text analysis and classification has been paid more and more attention [11]. Generally, topic extraction is related to semantic analysis, which has already been reflected in many classic works. Latent Dirichlet Allocation (LDA) [12] is a standard topic modeling method. In this paper, we implement and discuss different kinds of topic modeling methods for our app reviews, so as to seek out the most appropriate one.

B. Mining User Reviews on Mobile Apps

With the swift growth of mobile application markets, many researches have been conducted in mining app features. Claudia and Rachel [13] proposed a prototype MARA (Mobile App Review Analyzer) to automatically retrieve request features of online reviews. The features extracted in their work mainly focused on game apps and simply included some keywords, which might be not sufficient for practical review analysis.

Also, there are many other aspects of applications. A case study of Android Game Apps was conducted in [14]. It extracted the devices mentioned in most user reviews to help developers prioritize their limited Quality Assurance (QA) efforts. In [8], a framework named as 'Apposcopy' was proposed to determine whether a given app matched a malware signature based on semantic characteristics. Features of different types of apps were mined in [5]. Although [5] provided topic analysis for different segments of time series, the number of user reviews was analyzed in it, instead of topics or main reviews. It merely focused on the topic distributions when the number of reviews was abnormal. In [4], Ning et al. produced a framework named as 'AR-Miner' for mining informative user reviews. A filtering method base on supervised learning was proposed for extracting informative reviews before topic analysis. However, their work did not consider the essential attribute of user reviews, that is, the pertinence to a time period or a specific version of mobile app. Furthermore, 'AR-Miner' needs manual labeling before filtering process, which might be not applicable for massive reviews.

III. PROBLEM SETTING

Considering an individual app $\mathcal{A}$ from a certain app store, it involves a list of n review instances with review texts $R = \{r_1, r_2, \cdots, r_n\}$, user ratings $A = \{a_1, a_2, \cdots, a_n\}$, post time $T = \{t_1, t_2, \cdots, t_n\}$ and corresponding versions $V = \{v_1, v_2, \cdots, v_n\}$. Therefore, for each review instance r_i, we have its attributes: post date t_i, rating a_i, and version v_i. The general structure of data is designed as $\mathcal{A} : \{R, A, T, V\}$.

Table I: Examples of User Reviews

ID	Rating	Date	Version	Review Text
1	5.0	11/08/14	21.0.0.23.12	Never had an issue with it
2	1.0	9/08/14	20.0.0.25.15	Hate that I have to download.
3	2.0	07/14/14	12.0.0.15.14	Can't download videos.
...	...	...	...	...
n	a_n	t_n	v_n	r_n

Notes: Each row means a review instance, including a review text, user rating, post date and the corresponding app version.

Table I shows the notations of all the variables and a review sample with n review instances respectively.

To track the dynamics of the main themes, we divide reviews into several time sequences $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_z\}$, where z indicates the number of time periods. The detail will be discussed in Section V.

IV. AR-TRACKER

In this section, we give an overview of the proposed framework - 'AR-Tracker' initially. And then we introduce each procedure of AR-Tracker in detail.

A. Overview of AR-Tracker

The general framework of AR-Tracker is illustrated in Fig 1, which comprises 5 main steps. To begin with, we need to collect and preprocess massive user reviews and other basic information of the app from the Internet. The second step is to extract topics from messy reviews using various topic modeling methods (Section IV-B). Then, we implement our proposed ranking scheme to present the topics in order of importance (Section IV-C). In the next step, on the basis of the ranked topics, we prioritize the user review instances, described in Section IV-D. We compare the results of different modeling methods with the real user feedback presented on the official website, in order to find the most appropriate one for mining app reviews (Section V-C). Finally, by using the method with a more effective result in Section V, we visualize the dynamics of main themes generated from user reviews (Section V-D).

B. Topic Extraction

User experiences possess intangible values. Developers can be reminded of the bugs in their apps or the features they need to refine through mining user-generated contents.

Topic modeling methods are statistical methods that can analyze the words of the original texts to discover the main topics [15]. They do not require any prior annotation or labeling of the documents - the topics emerge from the analysis of the texts. To our best knowledge, little research has been conducted systematically on determining which kind of method is better for app review mining. In the paper, we compare different topic modeling methods on massive user review instances. The methods include Latent Semantic Indexing (LSI) [16], Latent Dirichlet Allocation (LDA) [12], Random Projection (RP) [17], Non-Negative

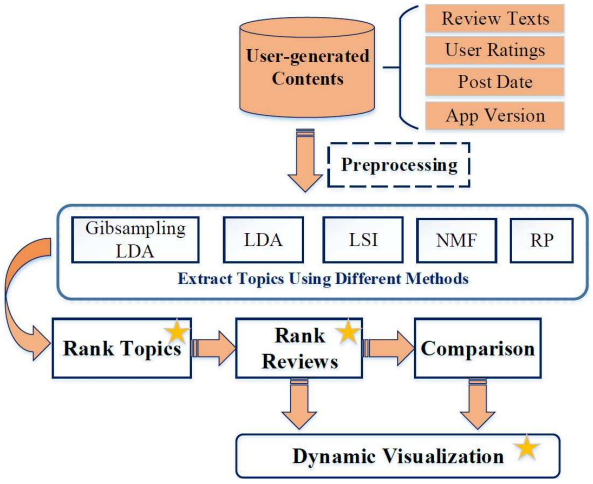

Figure 1: Framework of AR-Tracker

Matrix Factorization (NMF) [18], and Gibbs Sampling of LDA [19].

C. Topic Ranking

In order to discover the major topics, we need to prioritize the topics extracted from user reviews. Existing methods merely consider the user-generated contents, such as ratings, dates, and duplicates, etc., and ignore the relations between topics. Generally, the topic with the larger similarity with other topics would likely to be the main concern of users.

Given a list of review texts $R = \{r_1, r_2, \cdots, r_n\}$, we can simply obtain the corresponding vocabulary $D = \{\omega_1, \omega_2, ..., \omega_d\}$ (d is the magnitude of the vocabulary, ω means one specific token). Topics $\beta = \{\beta_1, \beta_2, ..., \beta_k\}$ (k is the number of topics) represent the topics extracted through topic modeling. A review text r can also be expressed as a probability distribution over the topics β, as shown in Table II.

Table II: Review-Topic Matrix

	β_1	β_2	$\cdots$	β_k
r_1	p_{11}	p_{12}	$\cdots$	p_{1k}
r_2	p_{21}	p_{22}	$\cdots$	p_{2k}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
r_n	p_{n1}	p_{n2}	$\cdots$	p_{nk}

The topic-ranking schema involves two factors - one is the similarities with other topics, and the other is the user ratings. The score S of the i th topic can be represented as $\mathbf{S}(\beta_i) = \{S(d_i), S(a_i)\}$, where $S(d_i)$ and $S(a_i)$ indicate the scores stemmed from the above two influence factors. Next we will introduce the detail of the computation.

1) Topic Similarity: Commonly, ranking and clustering are regarded as orthogonal techniques [20]. Intuitively, if one topic has a larger similarity with other topics, this topic tends to have more significance than the other and can be ranked higher. As shown in Fig. 2, Topic C seems closer to the other topics than Topic A or Topic B does, which implies that the words in Topic C are more related to those in B and C. Here, we introduce Hellinger distance, a statistical method to quantify the similarity between two probability distributions.

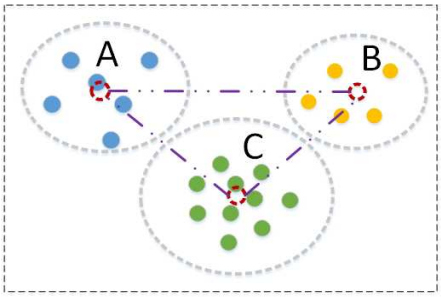

Figure 2: Explanation of Topic Similarity. **Note:** The clusters A, B and C indicate three topics. And the dots inside each cluster represent the words belonging to that topic.

For any two topics β_i and β_j, their discrete probability distributions among review instances are $\beta_i = (p_{i1}, \cdots, p_{in})$ and $\beta_j = (p_{j1}, ..., p_{jn})$ respectively, where n means the number of review instances. The Hellinger distance between these two topics is defined as:

$$H(\beta_i, \beta_j) = \frac{1}{\sqrt{2}} \sqrt{\sum_{u=1}^{n} (p_{iu}^{\frac{1}{2}} - p_{ju}{}^{\frac{1}{2}})^2}. \quad (1)$$

The score of topic similarity for the topic β_Q, $Q \in [1, k]$, is the inverse distance to all the other topics, that is:

$$S(d_Q) = \frac{1}{\sum_{i=1}^{k} H(\beta_Q, \beta_i)}, \quad (2)$$

where k is the number of the topics.

2) User Rating: Generally, user reviews with lower ratings imply the users are highly frustrated with some features of the app. And reviews with medium or higher ratings indicate the corresponding features are not very critical.

In the paper, we define $\mathcal{I}_1$ as the set of reviews with rating 1.0 or 2.0, $\mathcal{I}_2$ as the set with rating 3.0, and $\mathcal{I}_3$ as the set with rating 4.0 or 5.0. Then the score of user ratings for the topic β_Q can be described as:

$$S(a_Q) = w_1 \sum_{i \in \mathcal{I}_1} P_{iQ} + w_2 \sum_{i \in \mathcal{I}_2} P_{iQ} + w_3 \sum_{i \in \mathcal{I}_3} P_{iQ}, \quad (3)$$

where w_1, w_2 and w_3 are the weights corresponding to each index set $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}_3$ respectively and $w_1 + w_2 + w_3 = 1$, $0 \leq w_3 \leq w_2 \leq w_1 \leq 1$.

3) Overall Score: In terms of topic similarities and user ratings, the overall score of the topic β_Q is defined as:

$$\mathbf{S}(\beta_Q) = (S(a_Q) + \gamma_1) \cdot (S(d_Q) + \gamma_2), \tag{4}$$

where γ_1 and γ_2 are the weights that can be modified according to individual requirements and $0 \leq \gamma_1, \gamma_2 \leq 1$.

D. Review Ranking

Our goal is to provide developers more interpretable and direct results, so only a few undefined topics are not satisfying. We need to prioritize the review instances among massive raw user reviews and show the representative ones to developers.

Given a list of review texts $R = \{r_1, r_2, ..., r_n\}$ and their rating information, we get the topic probability distributions of each review text in Table II and the sorted topics in Section IV-C. The review-ranking scheme involves 3 elements - topic importance, user rating and review instance similarity. We adopt the probability distributions among different reviews to measure the similarity. The grade G of the i th review instance can be described as $\mathbf{G}(r_i) = \{G(m_i), G(a_i), G(d_i)\}$, where $G(m_i)$, $G(a_i)$ and $G(d_i)$ mean the grades stemmed from the above three factors respectively. The detail is described as below.

1) Topic Importance: For a review text, a larger probability distribution of the more important topic means that instance also tends to be more prioritized. Here, we group reviews according to the topic probability distributions, which means a review text belongs to the topic with the largest proportion. Thus, we have k groups of reviews, each with n_j (where $j \in [1, k]$, k is the number of topics) review texts. The score of topic importance for the review text r_Q can be defined as:

$$G(m_Q) = \frac{\sum_{j=1}^{k} \lambda_j \cdot p_{Qj}}{k}, (0 \leq \lambda_j \leq 1), \tag{5}$$

where k represents the number of extracted topics. λ_j means the weight to the j th topic and $\lambda_j = \frac{n_j}{n}$, where n is the total number of review instances. p_{Qj} is the probability distribution of j th topic to the review text r_Q.

2) User Rating: Analogous to the impact of user ratings on sorting topics it also influences the order of review instances. Here, we adopt the similar grouping method as the one described in the last section. That is, $\mathcal{I}_1$ means the review group with rating 1.0 or 2.0, $\mathcal{I}_2$ with rating 3.0, and $\mathcal{I}_3$ with rating 4.0 or 5.0. The score of user ratings $G(a_Q)$ for the reveiw r_Q is described as:

$$G(a_Q) = w_i, i \in [1, 3], \tag{6}$$

where w_i is the wight to the i th review group. Generally, $w_1 + w_2 + w_3 = 1$, $0 \leq w_3 \leq w_2 \leq w_1 \leq 1$. A larger w_i means the review group with a certain rating has more priority.

3) Review Similarity: If a review text has more resembling texts, the theme reflected by this review would tend to be significant. Thus, this review instance can be ranked higher. A review text can be regarded as a probability distribution over different topics, as shown in Table II. Here, we also adopt the Hellinger distance to measure the similarity between reviews. The score of review similarity $G(d_Q)$ for the review r_Q is defined as:

$$G(d_Q) = \frac{1}{\sum_{i=1}^{n} H(r_Q, r_i)}, \tag{7}$$

where n is the number of user reviews in R. $H(r_Q, r_i)$ means the Hellinger distance between r_Q and all the reviews in R.

4) Overall Score: With respect to the above three factors, the overall score $G(r_Q)$ of the review r_Q is denoted as:

$$\mathbf{G}_Q = (\alpha_1 \cdot G(m_Q) + \alpha_2 \cdot G(a_Q)) \cdot G(d_Q), \tag{8}$$

where $\alpha_i (i \in [1, 2])$ means the weight to each factor and $(0 \leq \alpha_1, \alpha_2 \leq 1)$. Developers can adjust them according to their own demands.

V. EMPIRICAL STUDY

We implement on a large sample of online reviews from mobile apps. The two main objects of our experiments: (i) to find a more appropriate topic modeling method for our framework; (ii) track the dynamics of the themes reflected by the representative reviews.

A. Dataset

We collected meta-information and user reviews of mobile apps by (a) building a web crawler using a web-automation and testing tool named Selenium based on Python; (b) utilizing the app-crawling API.

Table III: Experimental Dataset

	Facebook	Facebook Messenger	TempleRun2	Instagram
11-01	12,678	14,515	2,531	11,836
10-15	31,787	29,715	7,312	31,797
10-01	26,690	23,966	6,259	26,035
09-15	23,538	30,143	7,195	27,315
09-01	29,096	39,793	6,901	28,144
08-15	31,303	40,510	7,933	5,728
08-01	21,270	26,863	4,676	-

Here we choose 4 main apps - Facebook (social app), Facebook Messenger (communication app), Templerun2 (action game app), and Instagram (social app) - for dynamic analysis. The reasons why we choose these apps are that they belong to several categories and some of them are possessed by the same developer. So we can expect to detect some problems of apps belong to one company. Most importantly, they are prevalent in the worldwide, making sure the numbers of reviews are considerable and sufficient for tracking the dynamics.

Also, we utilize the data of SwiftKey (6,282 reviews in total) in [4] for comparison and evaluation. Table III illustrates numbers of reviews within different periods or time sequences $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_7\}$. As can be observed from the Table III, all the data are collected in the recent 4 months. In the first column of the table, '11-01' means the time period from '11-01' to the latest date of the reviews; '10-15' indicates the period from '10-15' to '11-01'.

B. Performance Metrics

In this section, we introduce the performance metrics used in our experiments. To compare with the state-of-the-art method in [4], we adopt the same metrics such as Hit-rate (recall) and NDCG@k.

$$Recall(Hit-rate) = \frac{TP}{TP+FN}, \tag{9}$$

where TP, FN indicate the numbers of true positives (hits) and false negatives (misses), respectively.

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{10}$$

Also, since our framework does not filter the non-informative user reviews, we use the Info-rate as one index for analyzing the proportion of informative reviews in the whole result.

$$Info-rate = \frac{Number\,of\,informative\,reviews}{Number\,of\,returned\,results}, \tag{11}$$

where $Info-rate \in [0,1]$, and the higher value implies more informative reviews are included in the results.

C. Comparison with AR-Miner

In the experiment, we choose w_1, w_2, w_3 equal to 0.85, 0.1, and 0.05, respectively. The whole parameters are illustrated in Table IV. Due to the limitation of writing space, we just describe the top 5 results of AR-Tracker(Gibbs Sampling LDA), shown in Fig 3. We can see that the fifth review text is non-informative and 3 of the top 5 reviews hit the groundtruth.

Table IV: Experimental Parameters

Parameter	k	α_1, α_2	γ_1, γ_2	w_1	w_2	w_3
Value	10	1	0	0.85	0.1	0.05

The groundtruth given in [4] is derived from SwiftKey Feedback Forum. To compare with AR-Miner, we use the same dataset (6,282 reviews totally) and performance metrics. The results are illustrated in Fig 4.

As can be seen from Fig 4, Gibbs Sampling LDA can achieve the identical hit-rate as AR-Miner(LDA) higher than AR-Miner(ASUM). Also, LDA without EMNB for filtering process would not prioritize user reviews satisfyingly. In terms of Info-rates and Hit-rates (only applicable for methods without filtering), LSI, LDA, RP, and NMF all represents

Rank	Review	Key Phrase
1	It went off for a repair and came back perfect until I installed swiftkey 3 again.	Update
2	Everything was great, after updating to jellybean stock rom on tmobile I can no longer keep set as my default keyboard, have to set it every time I turn my phone on, getting annoying, please fix.	Jellybean[10]
3	But between it shoving words after commas without me noticing, overly aggressive spell check and if I let it type out a sentence by just hitting space bar a log it types out my cover letter for a job I applied for over a year ago.	Auto-space after punctuation[3]
4	Doesn't predict swears, custom words, or abbreviations w/ symbols (ie. "w/").	AutoText-word substitution[5], Custom words[4]
5	This has happened multiple times which is annoying because I paid for this.	

Figure 3: Top 5 Review Texts of Gibsampling LDA

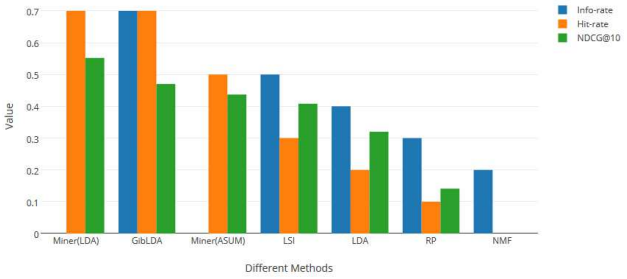

Figure 4: Comparison of Different Methods

lower than Gibbs Sampling LDA, which means these methods are more vulnerable to noises (non-informative reviews). Although the NDCG@10 of Gibbs Sampling LDA is a little lower than that of AR-Miner(LDA), Gibbs Sampling LDA is higher than AR-Miner(ASUM). And most importantly, Gibbs Sampling LDA spends less manual labor and are more particularly applicable for tremendous numbers of raw reviews. In AR-Miner, the total number of reviews amounts to just 6,282 (3,282 for unlabeled set), far less than the actual number (thousands of reviews per day) for popular apps. As [4] stated, AR-Miner took 0.5 man-hours (for 3,000 labeled data) for EMNB filtering and 7.4 hours for purely manual inspection, while filtering is non-necessary in our framework. Therefore, we suppose our framework possesses more practical implications. And we will use AR-Tracker(Gibbs Sampling LDA) for the dynamic analysis of user reviews in the next section.

D. Dynamic Analysis

We analyze the reviews of Facebook over time. From the top 10 review texts of different time sequences, we extract 10 key themes - Crash, Newsfeed, Picture, Post, Notification, Privacy, Space, Video, Messenger, and Navigation. We summarize the ranks and frequencies of occurrence of these themes in specific time period, illustrated in Fig 5. In the cell, the number outside the bracket indicates the rank with

the occurrence frequency inside. The only one figure in the cell represents the corresponding rank.

	08-01	08-15	09-01	09-15	10-01	10-15	11-01
Crash	5(2)	7	8(2)	4	2(3)	4(2)	1(3)
Notification	9	0	0	0	0	0	2
Privacy	6(2)	2(3)	1	0	4	0	0
Space	4	0	0	1(3)	2	5	0
Navigation	8	0	0	2	0	1	4
Newsfeed	8	0	0	3	1(2)	9	3
Picture	0	6	0	5(2)	0	3	4
Post	1	0	0		4	2	8
Video	2	6(2)	0	9(2)	1	10	10
Messenger	1(5)	1(8)	2(5)	1	8(2)	7(2)	0

Figure 5: Summary of Top 10 Reviews in Different Periods

To clearly describe the changes of each theme, we divide the themes into two groups: General issues (with orange ground in Fig 5) and Content issues (with gray ground). And we define $Importance - rate$ as below to illustrate the significance level of each theme for content issues.

$$Importance - rate = \frac{1 - rank}{N} * (\lambda + frequency), \quad (12)$$

where $rank$ and $frequency$ represent the rank and occurrence frequency of the theme in specific period, respectively. λ is for regularization and here we set $\lambda = 0.1$. The $Importance - rate$ of each theme over time is illustrated in Fig 6.

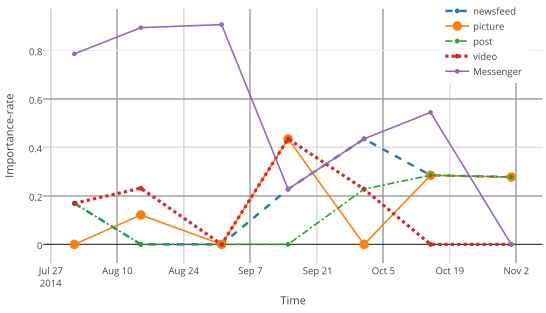

Figure 6: Dynamics of Topical Issues

As Fig 6 described, the trend of 'Messenger' is the sharpest and most interesting. This is because users were forced to download Messenger app to check Facebook messages when the app first launched, which aroused much discontent and strong response, also reported by [21]. From the aforementioned analysis, we can conclude that tracking the dynamics of the reviews can really help developers fix the main bugs or determine some new features.

Further, we summarize the Info-rates of these apps during time periods, described in Fig 7. It shows that Game app TempleRun2 has the lowest Info-rates but not less than 0.5 and all the other Info-rates are higher than 0.8, which indicates the validity of our framework on avoiding noises.

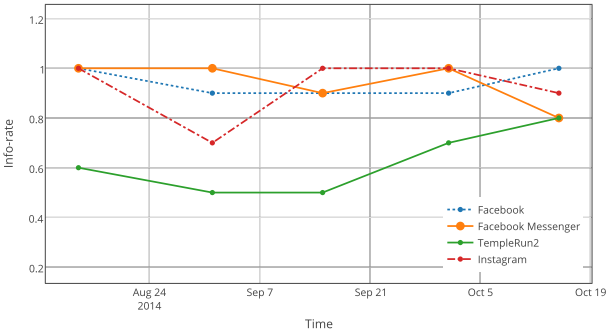

Figure 7: Info-rates of Different Apps Over Time

VI. Conclusion

User-generated contents are a favorable and crucial repository for mobile app developers. Since the user reviews are normally massive and messy for popular apps, manual labor is rather time-consuming and inapplicable. In the paper, we propose a novel review-ranking scheme without manual labeling for filtering. Also, we produce a new index for measuring the proportion of informative reviews in the top reviews. Furthermore, we visualize the dynamics of the main themes represented by these top reviews and evaluate the results with the factual reports or app change history.

There are two major advantages of our framework: (i) it does not need any manual labor and can achieve similar effect as the state-of-the-art method - AR-Miner. (ii) It can track the dynamics of main themes reflected by the top reviews, which would facilitate developers to determine the next commit.

Acknowledgment

This work was supported by the National Basic Research Program of China under 973 Project No. 2011CB302603, the National Natural Science Foundation of China under Project No. 61100077, and the Shenzhen Basic Research Program under Project No. JCYJ20120619152636275. Yangfan Zhou is the corresponding author.

References

[1] X. Lu, H. Wang, J. Wang, J. Xu, and D. Li, "Internet-based virtual computing environment: beyond the data center as a computer," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 309–322, 2013.

[2] "The importance of reviews and ratings." http://app-promo.com/aso-tip-5-the-importance-reviews-ratings/.

[3] "How in-app review mechanics pushed Flappy Bird to the top of the charts." http://venturebeat.com/2014/02/11/how-in-app-review-mechanics-pushed-flappy-bird-to-the--of-the-charts/.

[4] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: mining informative reviews for developers from mobile app marketplace." in *ICSE*, 2014, pp. 767–778.

[5] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1276–1284.

[6] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 113–120.

[7] C. Wang, D. Blei, and D. Heckerman, "Continuous time dynamic topic models," *arXiv preprint arXiv:1206.3298*, 2012.

[8] Y. Feng, S. Anand, I. Dillig, and A. Aiken, "Apposcopy: Semantics-based detection of android malware through static analysis," in *SIGSOFT FSE*, 2014.

[9] S. Havre, E. Hetzler, P. Whitney, and L. Nowell, "Themeriver: Visualizing thematic changes in large document collections," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 8, no. 1, pp. 9–20, 2002.

[10] A. Bougouin, F. Boudin, B. Daille *et al.*, "Topicrank: Graph-based topic ranking for keyphrase extraction," in *International Joint Conference on Natural Language Processing (IJCNLP)*, 2013, pp. 543–551.

[11] H. Yu, C. Ho, Y. Juan, and C. Lin, "Libshorttext: a library for short-text classification and analysis," Technical Report. http://www. csie. ntu. edu. tw/˜ cjlin/ papers/libshorttext. pdf, Tech. Rep., 2013.

[12] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[13] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*. IEEE, 2013, pp. 41–44.

[14] H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan, "Prioritizing the devices to test your app on: A case study of android game apps," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering. ACM*, 2014.

[15] D. M. Blei, "Probabilistic topic models," *Communications of the ACM*, vol. 55, no. 4, pp. 77–84, 2012.

[16] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391–407, 1990.

[17] P. Kanerva, J. Kristofersson, and A. Holst, "Random indexing of text samples for latent semantic analysis," in *Proceedings of the 22nd annual conference of the cognitive science society*, vol. 1036. Citeseer, 2000.

[18] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation*, vol. 19, no. 10, pp. 2756–2779, 2007.

[19] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen, "Cqarank: jointly model topics and expertise in community question answering," in *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2013, pp. 99–108.

[20] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu, "Rankclus: integrating clustering with ranking for heterogeneous information network analysis," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. ACM, 2009, pp. 565–576.

[21] "Facebook Messenger users gripe and grumble in online reviews." http://www.cnet.com/news/facebook-users-share-messenger-displeasure-in-online-pool/.