Online App Review Analysis for Identifying Emerging Issues

Cuiyun Gao, Jichuan Zeng, Michael R. Lyu, and Irwin King
Shenzhen Research Institute of The Chinese University of Hong Kong, China
The Chinese University of Hong Kong, China
{cygao,jczeng,lyu,king}@cse.cuhk.edu.hk

ABSTRACT

Detecting emerging issues (*e.g.*, new bugs) timely and precisely is crucial for developers to update their apps. App reviews provide an opportunity to proactively collect user complaints and promptly improve apps' user experience, in terms of bug fixing and feature refinement. However, the tremendous quantities of reviews and noise words (*e.g.*, misspelled words) increase the difficulties in accurately identifying newly-appearing app issues. In this paper, we propose a novel and automated framework IDEA, which aims to IDentify Emerging App issues effectively based on online review analysis. We evaluate IDEA on six popular apps from Google Play and Apple's App Store, employing the official app changelogs as our ground truth. Experiment results demonstrate the effectiveness of IDEA in identifying emerging app issues. Feedback from engineers and product managers shows that 88.9% of them think that the identified issues can facilitate app development in practice. Moreover, we have successfully applied IDEA to several products of Tencent, which serve hundreds of millions of users.

CCS CONCEPTS

• **Software and its engineering** → *Dynamic analysis*; • **Information systems** → *Web and social media search*;

KEYWORDS

App reviews, online analysis, emerging issues

ACM Reference format:
Cuiyun Gao, Jichuan Zeng, Michael R. Lyu, and Irwin King. 2018. Online App Review Analysis for Identifying Emerging Issues. In *Proceedings of ICSE '18: 40th International Conference on Software Engineering , Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18),* 11 pages.
https://doi.org/10.1145/3180155.3180218

1 INTRODUCTION

App developers are eager to know what is going on with their apps after published [38]. Timely and precisely identifying the emerging issues of apps is of great help for app developers to update their apps, such as fixing bugs, refining existing features, and adding new functions.

User reviews are direct feedback from the users who have experienced the apps, and reflect the instant user experience [33]. The emerging issues detected from user reviews, such as the existing bugs (*e.g.*, crashes) and unfavorable app features (*e.g.*, too many ads) [13], can provide informative evidence for app developers in maintaining their apps and scheduling the app updates. For example, Facebook Messenger received massive one-star ratings (the lowest rating) in August, 2014, accounting for nearly 94% of all its reviews on Apple's App Store[1], and suffered a large loss of users [8], since the version contained severe privacy issues (*e.g.*, accessing the photos and contact numbers in users' phones). However, such issues had already been flushed out with complaints from over 12,600 user reviews on App Store one month ago. The situation could be effectively alleviated if the emerging issues were timely detected from user reviews. Therefore, user reviews provide an effective and efficient way to identify the emerging issues of apps, which would be a significant help to the developers.

The characteristics of user reviews make accurate issue detection very challenging. First, app reviews are generated everyday in large volume. Manual analysis is prohibitively time-consuming for apps with large numbers of reviews (*e.g.*, Facebook receives more than 10,000 reviews in Google Play every day [2]). Second, app reviews contain numerous noise words, such as misspelled words, repetitive words, and non-English words. Also, they are often shorter in length, since most of them are written by users via mobile terminals. Third, only 30% of the reviews provide informative user opinions for app updates [6]. Furthermore, detailed and newly-appearing app issues are hard to be predefined, because they are diverse for different apps and versions. Previous research mainly focuses on reducing the manual power in extracting software aspects or user preferences, such as establishing dictionaries for preprocessing reviews [42], filtering out non-informative reviews [6], or classifying reviews to predefined topics [40]. However, effectively detecting emerging issues from user reviews has rarely been studied.

We propose a novel and automated framework IDEA for detecting emerging issues/topics[2] based on online review analysis. IDEA takes reviews of different versions as input. To track the topic variations over versions, a novel method AOLDA (Adaptively Online Latent Dirichlet Allocation) is employed for generating version-sensitive topic distributions. The emerging topics are then identified based on the typical anomaly detection method. To make the topics comprehensible, IDEA labels each topic with the most relevant phrases and sentences based on an effective ranking scheme considering both semantic relevance and user sentiment. The prioritized topic labels are the app issues identified. Finally, IDEA visualizes the variations of app issues along with versions, and highlights the emerging ones for better understanding.

[1]The App Store in this paper refers to Apple's App Store.
[2]The topics and issues are semantically equal in this paper.

ICSE '18, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5638-1/18/05…$15.00
https://doi.org/10.1145/3180155.3180218

To verify the effectiveness of IDEA, we consider the official app changelogs as ground truth, since they encompass the primary changes of the releases and represent the issues concerned by developers. Our experiments are conducted on six popular apps, with two of them from App Store and the others from Google Play. We compare IDEA with the method based on OLDA (Online Latent Dirichlet Allocation) [1], one classical method for emerging issue detection. Results indicate that the average precision, recall, and F-score of IDEA on the subject apps are 60.4%, 60.3%, and 58.5% respectively, which increases the F-score of the OLDA-based method by 72.0%. We also conduct a user survey in Tencent, indicating that 88.9% of respondents think that the identified issues of IDEA can facilitate app development in practice. Moreover, we apply IDEA to four Tencent[3] products which serve hundreds of millions of users worldwide, and confirm the effectiveness and efficiency of IDEA in industrial practice.

The contributions of our paper are elaborated as below.

- We propose a framework called IDEA to automatically identify emerging issues from app reviews effectively. Also, IDEA is an online analysis tool and can process new app reviews in a timely fashion.
- We propose a novel method called AOLDA for online review analysis, which adaptively combines the topics of previous versions to generate topic distributions of current versions.
- We visualize the variations of the captured (emerging) app issues along with versions, with the emerging ones highlighted. We publish the code and review data on website[4].
- We verify the effectiveness of IDEA based on the app reviews of six popular apps which are from different categories and platforms. The survey and application in Tencent also validate the performance of our framework in practice.

The remainder of the paper is organized as follows. Section 2 describes the motivation and the background of our work. Section 3 outlines the overall picture and details each step involved in the framework. Section 4 illustrates experiment results. Section 5 presents the practical usage of our framework. Section 6 discusses possible limitations, with related work introduced in Section 7. Section 8 concludes the paper.

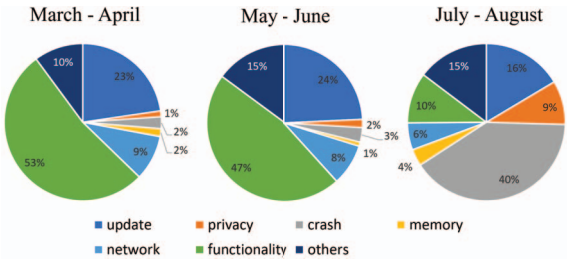
2 BACKGROUND AND MOTIVATION

2.1 Emerging App Issues

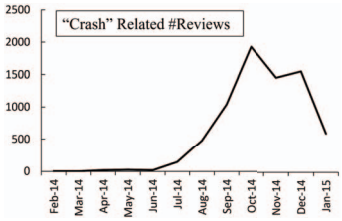
For an app issue to be considered an emerging issue, it must be (heavily) discussed in this time slice but not previously [16]. Figure 1 (a) presents the issue distributions of Facebook Messenger in three periods (March-April, May-June, and July-August), based on the manually labelled 100 review samples from each period. Generally, the issue distributions are nearly consistent along with periods, *e.g.*, from March-April to May-June in Figure 1 (a). However, emerging issues can influence the issue distribution of one period, creating significant differences with those of previous periods in terms of proportion. For example, the proportion of the crash issue presents a huge increase during the July-August period. We further investigate the number of reviews containing the keyword "crash" along with their timing, and present the results in Figure 1 (b). The volume of the crash issue shows a sudden increase around July-August, which signifies that the issue tends to be an emerging issue during that period.

Definition 2.1 (Emerging Issues in User Reviews). An issue in a time slice is called an emerging issue if it rarely appears in previous slice but is mentioned by a significant proportion of user reviews in current slice.

In Definition 2.1, the "time slice", the degree of "rarely", and the "significant proportion" can be defined according to different situations. For example, the "time slice" in this paper corresponds to the app version. Based on the detected emerging issues, developers can locate the buggy features of their apps efficiently, update the apps accordingly, and ultimately improve the user experience.



(a) Issue distribution in Facebook Messenger.



(b) The number of reviews containing the keyword "crash".

Figure 1: Illustration of emerging issues.

2.2 Online Review Analysis

Online review analysis (ORA) is an automated way to acquire and process user reviews in real time as reviews are arrived continuously. As shown in Figure 2, ORA takes the reviews of slice t (*current review slice*) as input, and outputs analysis results, such as tracking user preference and detecting emerging issues. In this way, the urgent user concerns incarnated by app reviews can be captured by ORA in a timely manner and fed back to developers for instant bug fixing or feature improvement. Thus, ORA is a crucial component in the closed cycle of app development.

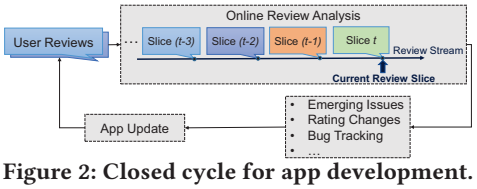


Figure 2: Closed cycle for app development.

[3]The company has many popular products, such as WeChat, QQ, and Honor of Kings, and serves billions of users worldwide.
[4]https://github.com/ReMine-Lab/IDEA

Currently, most of the app issues mined from user reviews are manually settled or defined [24, 40, 42], such as privacy and GUI, which are usually general categories. Although such definition facilitates the process of task assignment to individuals, it is unfavorable for detecting newly-presented and more detailed issues (*e.g.*, notification center). Thus, for detecting emerging issues, ORA is a practical way due to its timeliness and no need for predefined issues, which has rarely been studied previously.

2.3 App Changelogs

App changelogs describe the noticeable modifications of the latest versions for attracting users to install and experience new releases. Similar to user reviews. changelogs also correspond to specific versions. Generally, developers write into the changelogs with information related to whether the apps are adding or removing features, and whether the apps have made improvements with certain devices or to specific bugs. Figure 3 illustrates a sample changelog of NOAA Radar Pro, a weather alerts & forecast app in App Store.

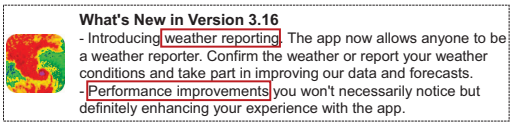


Figure 3: Changelog of NOAA Radar Pro. The rectangles highlight two key terms which represent the major changes of Version 3.16.

As Figure 3 indicates, the new version introduces new functionality (*i.e.*, weather reporting) and refines performance issues. The delivered changes exhibit the issues that are concerned by developers. Although the changelogs may not cover all the modifications to the releases, they represent a lower bound and the prominent part of the changes. Hence, changelog is a reasonable ground truth for verifying whether the extracted emerging issues are helpful for developers.

3 METHODOLOGY

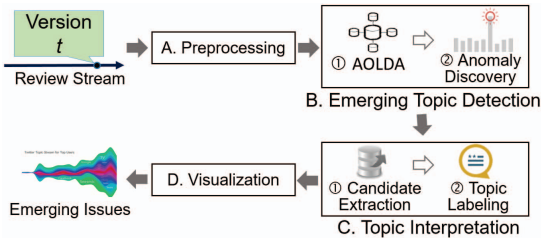


Figure 4: Framework of IDEA.

In this section, we first outline the overall framework of IDEA in Figure 4 and then elaborate on the four components involved in the framework. Each time, in the first stage (Part A in Figure 4), IDEA preprocesses a version of raw reviews from the review stream for reducing noisy words and non-informative words, and extracts phrases for subsequent analysis (Section 3.1). In the second stage (Part B in Figure 4), the proposed algorithm AOLDA captures the topic distributions of each version by considering the topics in previous versions, based on which emerging topics are identified using anomaly discovery (Section 3.2). Then, to interpret the topics (Part C in Figure 4), IDEA employs the meaningful phrases and sentences as candidates to label each topic according to their semantic relevance and user sentiment (Section 3.3). The topic labels are the identified app issues. Finally (Part D in Figure 4), IDEA visualizes the app issues along with the different versions, and highlight the emerging ones for better understanding (Section 3.4).

3.1 Preprocessing

Since app reviews are generally submitted via mobile terminals and written using limited keyboards, they contain massive noisy words, such as casual words, repetitive words, misspelled words, and non-informative words (*e.g.*, the words simply describing users' feelings). In the following, we introduce our rule-based methods for formatting words, the phrase extraction process, and our filtering method for reducing non-informative words.

3.1.1 Word Formatting. We first convert all the words in the review collection into lowercase, and then stem each word into its original form. We employ the preprocessing method in [26] for lemmatization. We then replace all digits with "<digit>". Since new terms and casual words would continuously increase in user reviews, we do not employ the dictionaries provided by [42] for avoiding over correction. We adopt the rule-based methods based on [42, 26] to rectify repetitive words, misspelled words, and non-English words.

3.1.2 Phrase Extraction. Since phrases (mainly referring to two consecutive words in our paper) are employed in Part C of IDEA for interpreting topics, they should be extracted in the preprocessing step and trained along with all the other words in Part B. In this way, we can capture the semantics of each phrase, based on which we can label the topics with the most relevant phrases. Since the topic labels in phrases should be meaningful and comprehensible, we use a typical phrase extraction method based on PMI (Pointwise Mutual Information) [35], which is effective in identifying meaningful phrases based on co-occurrence frequencies:

$$PMI(w_i, w_j) = \log \frac{p(w_i w_j)}{p(w_i)p(w_j)}, \quad (1)$$

where $p(w_i w_j)$ indicates the co-occurrence probability of the phrase $w_i w_j$ and $p(w_i)$ (or $p(w_j)$) represents the probability of the word w_i (or w_j) in the whole review collection. Higher PMI values exhibit that the combination of the two words is more likely to be a meaningful phrase. We extract the meaningful phrases by experimentally set a threshold for PMI. The phrases with PMIs larger than the threshold are extracted.

3.1.3 Filtering. The filtering step aims to reduce the non-informative words, such as emotional words (*e.g.*, "bad" and "nice"), abbreviations (*e.g.*, "asap"), and useless words (*e.g.*, someone). Non-informative words are summarized by two researchers from 1,000 reviews, which are also referred to as *predefined stop words*. The box below lists 18 of the total 78 non-informative words due to space limitations. The *predefined stop words* are filtered out together with

the stop words provided by NLTK [34]. We do not employ the supervised method in [6] for filtering, since in this work labeling massive non-informative reviews requires a great deal of manual effort. Finally, all the remaining words and extracted phrases (where the words in each phrase are connected with "_") are fed into the next step for emerging topic detection.

Predefined Stop Words: *cool, fine, hello, alright, poor, plz, pls, thank, old, new, asap, someone, love, like, bit, annoying, beautiful, dear.*

3.2 Emerging Topic Detection

In this section, we aim to detect the emerging topics of current versions by considering the topics in previous versions. We first introduce the proposed method AOLDA for adaptively online topic modeling, from which we capture the topic evolutions along with versions. We then present how we discover the emerging topics (*e.g.*, anomaly topics).

3.2.1 AOLDA - Adaptively Online Latent Dirichlet Allocation. Online Latent Dirichlet Allocation (OLDA) [1] is a classic method for tracking the topic variations of text streams, which models the topics of texts in one time slice based on the topics of the last slice. However, app reviews are typically short and contain massive noise words. Such review features can influence the topic distributions in consecutive versions with OLDA, and thereby decrease the performance of emerging topic detection. To reduce the influence of noise words and more accurately capture the topic evolution along with versions, we propose an adaptively online topic modeling method, AOLDA. The proposed AOLDA improves OLDA by adaptively combining the topic distributions in previous versions. The details are described below.

The preprocessed reviews are divided by version, denoted as $R = \{R^1, R^2, ..., R^t, ...\}$ (where t indicates the t-th version), and input into AOLDA one by one. In AOLDA, each review is treated as one document. The prior distributions over document-topic and topic-word distributions are defined initially, represented as α and β respectively. β determines the topic distributions of the terms in the input. The number of the topics is specified as K. For the k-th topic, ϕ_k^t is the probability distribution vector over all the input terms. We introduce the parameter - window size w, which defines the number of previous versions to be considered for analyzing the topic distributions of the current version. The overview of the model AOLDA is depicted in Figure 5.

Different from OLDA, as Figure 5 shown, we *adaptively* integrate the topic distributions of the previous w versions, denoted as $\{\phi^{t-1}, ..., \phi^{t-i}, ..., \phi^{t-w}\}$, for generating the prior β^t of the t-th version. The *adaptive* integration refers to summing up the topic distributions of different versions with different weights $\gamma^{t,i}$:

$$\beta_k^t = \sum_{i=1}^{w} \gamma_k^{t,i} \phi_k^{t-i}, \tag{2}$$

where i denotes the i-th previous version ($1 \le i \le w$). The weight $\gamma_k^{t,i}$ is determined by the similarity of the k-th topic between the $(t-i)$-th version and the $(t-1)$-th version, which is calculated by the softmax function [39]:

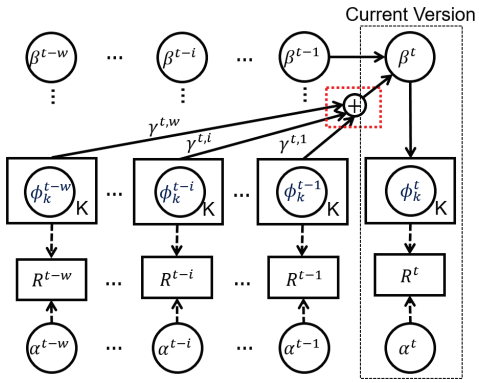


Figure 5: Overview of AOLDA. The red rectangle with dashed dots highlights the adaptive integration of the topics of the w previous versions for generating the prior β in the t-th version. R^t is the review corpus in the t-th version. The dotted lines indicate that we simplify the original LDA [4] steps for clearness.

$$\gamma_k^{t,i} = \frac{\exp(\phi_k^{t-i} \cdot \beta_k^{t-1})}{\sum_{j=1}^{w} \exp(\phi_k^{t-j} \cdot \beta_k^{t-1})}, \tag{3}$$

where the dot product $(\phi_k^{t-i} \cdot \beta_k^{t-1})$ computes the similarity between the topic distribution ϕ_k^{t-i} and the prior of the $(t-1)$-th version β_k^{t-1}. Such adaptive integration can endow the topics of the previous versions with different contributions to the topic distributions of the current version.

3.2.2 Anomaly Discovery. Based on the captured topic evolution by AOLDA, we identify the anomaly topics which present obvious differences with those of the previous versions. The identified anomaly topics are regarded as emerging topics. To obtain the difference of the k-th topics between two consecutive versions, *e.g.*, ϕ_k^t and ϕ_k^{t-1}, we employ the classic Jensen-Shannon (JS) divergence [19]. JS divergence measures the similarity between the two probability distributions:

$$D_{JS}(\phi_k^t || \phi_k^{t-1}) = \frac{1}{2} D_{KL}(\phi_k^t || M) + \frac{1}{2} D_{KL}(\phi_k^{t-1} || M), \tag{4}$$

where $M = \frac{1}{2}(\phi_k^t + \phi_k^{t-1})$. The Kullback-Leibler (KL) divergence D_{KL} is utilized to measure the discrimination from one probability distribution P to another Q, computed by:

$$D_{KL}(P||Q) = \sum_i P(i) \log(P(i)/Q(i)), \tag{5}$$

where $P(i)$ is the i-th item in P. Higher JS divergence indicates that the two topic distributions have a larger difference.

Based on the computed divergences D_{JS} between the topics of consecutive versions, we capture anomaly topics by leveraging a typical outlier detection method [37]. The method assumes that the divergences follow a Gaussian distribution with the mean and variance at μ and σ^2 respectively. The anomaly topics are then detected by setting a threshold δ. For the t-th version, the threshold δ^t is dynamically defined according to the following steps.

1. We compute D_{JS} of the previous w versions for each topic, which generates a $w \times K$ matrix (where K is the number of topics).
2. We compute the mean μ and variance σ^2 of all the values in the computed D_{JS} matrix.
3. We set the threshold δ^t as $\delta^t = \mu + 1.25\sigma$, where the coefficient 1.25[5] is experimentally set for accepting 10% of topics as anomaly topics, as shown in Figure 6.

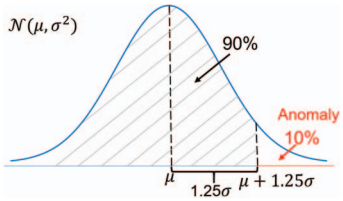


Figure 6: Gaussian distribution for anomaly discovery. The shaded area means the integral of the Gaussian distribution, which equals 90%. The topics with divergence larger than δ^t are considered as emerging topics.

For the t-th version, the topics with divergences higher than the defined threshold δ^t are regarded as emerging topics.

3.3 Topic Interpretation

The topics based on AOLDA are represented as the probability distributions over all the input terms. One snapshot of the top five terms to each topic is illustrated in Table 1. By only observing a few words, it would be non-trivial for developers to capture the meaning of the topics. In this section, we aim to interpret the topics automatically. To interpret each topic, we can utilize words, phrases, sentences, or entire reviews. However, single words may be ambiguous in semantics and cannot display the complete meanings of the topic. For example, we list the top five relevant words for each of the four topics of YouTube, as shown in Table 1, although both the words "video" and "work" are most relevant to Topics 2 and 4, these two topics may deliver different meanings, *e.g.*, Topic 2 is related to the video descriptions and Topic 4 is about loading videos. Moreover, one review may complain about several issues. For example, one Instagram user complains about the videos and stories in one review: *Videos don't post. Videos don't load. Stories disappear all the time.* Therefore, topic labels in words or reviews may not be helpful in accurately capturing the semantics of the topics. To render the topics comprehensible, we employ the most relevant phrases and sentences to label each topic in this section.

3.3.1 Candidate Extraction. We obtain candidate phrases and sentences for labeling topics.

Phrase Candidate: The candidates of the phrase labels are generated based on the extracted phrases in Section 3.1. **Three rules** are employed to identify more meaningful phrases: 1) Length limit: The length of each word in the phrase should be no less than three; 2) Stop word limit: The phrase should not contain words that are in the stop word list of NLTK [34]; and 3) Part-Of-Speech limit: The phrase should include at least one noun or verb, and no adverbs (*e.g.*, "here") or determiners (*e.g.*, "the").

[5]The coefficient can be adjusted according to the percentage of anomaly topics to be discovered. We use 1.25 here for accepting 10% of the total topics as anomalies.

Table 1: Top five terms for each topic of YouTube.

Topic	Topic 1	Topic 2	Topic 3	Topic 4
Term	comment	link	back	load
	say	video	also	video
	reply	open	button	even
	try	work	change	work
	error	description	go back	take

Sentence Candidate: We employ the reviews before the filtering step in Section 3.1, starting by chunking them into sentences based on NLTK's punkt tokenizer [36]. Then we retrieve sentences with more than four words, during which the noisy sentences (such as *so far so bad* and *great one*) are filtered out. The remaining sentences are regarded as our sentence candidates.

3.3.2 Topic Labeling. The topic labeling method is a ranking method, which considers two aspects: the semantic similarity between the candidates and the topics, and also the user sentiment of the candidates.

Semantic Score: Good topic labels should cover the latent meaning of the topic [30]. The semantic score measures the semantic similarity between the candidate and the topic. Moreover, the labels of different topics should be discriminative and cover different aspects of input reviews, instead of delivering overlapping information. Hence, the semantic score of one candidate involves the semantic similarity to the target topic and also the semantic similarities to all the other topics. A good topic label should be similar to the target topic and also different from the other topics in semantics.

We employ the method in [30] to measure the semantic similarity between one phrase candidate a and the target topic ϕ_k^t, defined as:

$$sim(a, \phi_k^t) = -D_{KL}(a||\phi_k^t) \approx \sum_w p(w|\phi_k^t) \log \frac{p(a, w|C)}{p(a|C)p(w|C)}, \quad (6)$$

where $p(w|\phi_k^t)$ is the probability of term w in the topic distribution ϕ_k^t. $p(w|C)$ and $p(a|C)$ denote the percentages of the terms w and a in the whole review collection C, respectively. The $p(a, w|C)$ indicates the co-occurrence probability of the two terms a and w in the collection C. For the sentence candidates s, we utilize Equation (7) to calculate the similarity.

$$sim(s, \phi_k^t) = -D_{KL}(s||\phi_k^t) \approx \sum_w p(w|\phi_k^t) \log \frac{p(w|s)/len(s)}{p(w|\phi_k^t)}, \quad (7)$$

where $p(w|s)$ can be calculated by the term frequency of w in the sentence s. The semantic score is then defined by combining $sim(l, \phi_k^t)$ with the similarity scores to other topics $\sum_{j \neq k} sim(l, \phi_j^t)$, which means the label l should be semantic close to the topic distribution ϕ_k^t and discriminate from other topic distributions.

$$Score_{sem}(l, \phi_k^t) = sim(l, \phi_k^t) - \frac{\mu}{K-1} \sum_{j \neq k} sim(l, \phi_j^t), \quad (8)$$

where l can be a phrase candidate a or sentence candidate s, and K is the number of topics. The parameter μ is utilized to adjust the penalty for the semantic similarities to other topics. Larger μ signifies that the candidates that are more different from other topics.

Sentiment Score: The topic labels should reflect users' concerns. Generally, the reviews with low ratings tend to express poor user experience and app issues [6], and the reviews with longer lengths are more likely to provide valuable information to developers. Therefore, we compute the sentiment score $Score_{sen}$ of one candidate l by combining the user ratings and review lengths:

$$Score_{sen}(l) = \exp(\frac{r_l}{\log(h_l)}), \tag{9}$$

where l can be a phrase candidate or sentence candidate. r and h denote the average user rating and the average word length of the reviews containing l, respectively.

Overall Score: We prioritize the candidates for each topic based on their semantic scores and sentiment scores. The overall score $Score(l, \phi_k^t)$ is defined as:

$$Score(l, \phi_k^t) = Score_{sem}(l, \phi_k^t) + \lambda\, Score_{sen}(l), \tag{10}$$

where the weight λ is to balance the two aspects. In this manner, all the topics including the detected emerging topics are labeled with the prioritized candidates. The topic labels are the identified app issues. For each topic, there is a trade off between the number of prioritized labels and the cost of user comprehension (*e.g.*, too many labels usually spend users more time in understanding the meaning of the topic). According to the survey [43], three labels are the moderate choice for users to comprehend the topics. Therefore, for one topic, we choose the ***three*** most relevant phrases and sentences respectively as labels for each topic.

3.4 Visualization

In this part, we visualize the the evolution of app issues (*i.e.*, topic labels) along with versions for better understanding. We employ an *issue river* to display issue variations. Figure 7 presents one example of YouTube for iOS. All the app issues constitute one river and each branch of the river indicates one topic. By moving the mouse over one topic, one can track detailed issue changes along with versions, where the emerging issues are highlighted as shown in Figure 7. The app issues with wider branches are of greater concern to users, where the *width* of the k-th branch in the t-th version is defined as:

$$width_k^t = \sum_a \log Count(a) \times Score_{sen}(a), \tag{11}$$

where $Count(a)$ is the count of the phrase label a in the review collection of the t-th version, and $Score_{sen}(a)$ denotes the sentiment score of the label a.

4 EXPERIMENTATION

We evaluate the performance of IDEA in identifying emerging app issues based on case studies. In this section, we explain how we select the subject apps for experiments, the performance metrics, and finally the comparison results of different methods. We focus on answering the following three research questions.

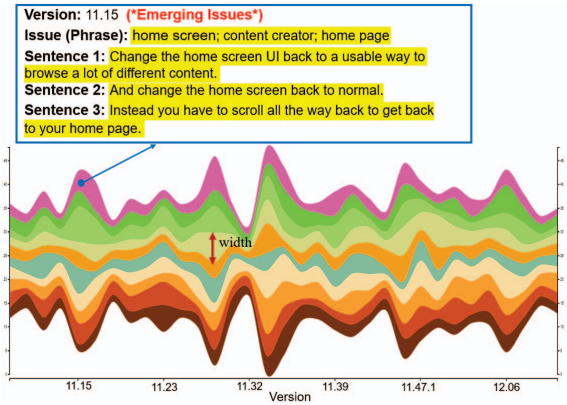


Figure 7: Issue River of YouTube for iOS. The number of topics K is set as 10, corresponding to 10 branches of the river. The horizontal axis represents the app versions, and the branches with larger widths indicate that the corresponding issues are more cared about by users at those versions.

RQ1: What is the performance of IDEA in identifying emerging app issues?

RQ2: Can IDEA achieve better performance compared with other methods?

RQ3: How do different parameter settings impact the performance of IDEA?

4.1 Dataset

We select the subject apps based on the following four criteria: The apps are i) popular apps in the app markets - indicating that the developers would update their apps regularly and the user reviews can be collected from several consecutive versions; ii) apps from different categories and platforms - to ensure the generalization of the proposed framework; iii) apps with enough user reviews - which necessitates an automated analysis; and iv) apps with detailed changelogs for most versions - to facilitate our validation process.

To obtain apps that satisfy the first three criteria, we randomly inspect the apps ranked in the top 100 on either App Store or Google Play according to App Annie [2], an app analytics platform. Only the apps with more than 2,000 US reviews [6] are inspected further, since significant effort is required for manual analysis. To filter out the apps that do not meet the fourth criterion, we check the historical changelogs of these apps. We eliminate apps with more than five successive sketchy changelogs, *i.e.*, the changelogs provide no details related to what functionality had been changed or how the user experience was being affected. One example of sketchy changelogs is "Multiple bug fixes and improvements across the entire app", where the bugs and improvement are not concrete enough for verifying prioritized app issues. Finally, we select six subject apps, with the details illustrated in Table 2.

In Table 2, we list the subject apps with the app name, category, platform, the number of reviews crawled, and the number of versions in the review collection. Overall, we obtain 164,026 reviews (from August 2016 to April 2017) for the six apps, from 89 versions in total. The apps are distributed in different categories, with two

Table 2: Subject apps.

App Name	Category	Platform	#Reviews	#Versions
NOAA Radar	Weather	App Store	8,363	16
YouTube	Multimedia	App Store	37,718	33
Viber	Communication	Google Play	17,126	8
Clean Master	Tools	Google Play	44,327	7
Ebay	Shopping	Google Play	35,483	9
Swiftkey	Productivity	Google Play	21,009	16

of them from App Store and the others from Google Play. With multiple categories and platforms, the generalization of IDEA can be ensured.

4.2 Performance Metrics

The app changelogs, *i.e.*, our ground truth, are collected from App Annie. Since the prioritized issues of IDEA are in phrases and sentences, we manually extract key terms from these changelogs for verification. One example is illustrated in Table 6, with the key terms highlighted. For each key term in changelogs, we validate whether the term is covered by the prioritized issues. Since the *word2vec* model [31] can accurately capture the semantic meanings of input terms based on their vector representations, we obtain the cosine similarities between each key term and the phrase-level issues based on the model. The key term is considered covered if its similarity to one of the issues is larger than 0.6 [18]. For sentence-level issues, we split the sentences into terms (including phrases and words) and verify whether the key term in changelogs can be covered in a similar way. We employ such semi-automatic evaluation method to facilitate parameter adjustment and comparison with other methods.

We employ three performance metrics[6] for verifying the effectiveness of IDEA. The first metric is for measuring the accuracy in detecting emerging issues, defined as $Precision_E$. The second is to evaluate whether our prioritized app issues (including both emerging and non-emerging issues) reflect the changes mentioned in the changelogs, defined as $Recall_L$. We introduce the third metric F_{hybrid} to measure the balance between $Precision_E$ and $Recall_L$. Higher values of F_{hybrid} indicate that changelogs are more precisely covered by detected emerging issues and more changelogs are reflected in the prioritized issues.

$$Precision_E = \frac{I(E \cap G)}{I(E)}, \quad Recall_L = \frac{I(L \cap G)}{I(G)}, \\ F_{hybrid} = 2 \times \frac{Precision_E \times Recall_L}{Precision_E + Recal_L}. \tag{12}$$

where E, G, and L are three sets, containing the detected emerging issues, the key terms in the changelogs, and all app issues (including both emerging and non-emerging issues), respectively. $I(\cdot)$ denotes the number of the issues in $\cdot$. During evaluation, we experimentally set the parameters as $w = 3$, $K = 10$, $\lambda = 0.5$, $PMI = 5$, and $\mu = 0.75$. We also initialize α and β with 0.1 and 0.01 respectively.

[6]We do not involve $Recall_E$ for validation since changelogs possibly include partial emerging issues. Also, $Precision_L$ cannot be considered because changelogs may cover items other than user-concerned issues. Here, $Precision_E$ and $Recall_L$ measure the precision of the emerging issues and coverage rate of changelogs by all the extracted issues respectively, which are consistent with the standards and convincing for this task.

Table 3: Topic-word distributions based on AOLDA.

	v11.07	v11.10	v11.11
Topic 1	link	open	video
	open	video	watch
	video	work fine	go
	work	go	want
	description	**click**	change
Topic 2	make	<digit>	back
	want	thing	make
	button	get	would
	back	interface	button
	use	want	people

4.3 Result of RQ1: Case Study

In this part, we evaluate the performance of IDEA by employing a case study on YouTube. We first present the results of the version-sensitive topic distributions based on AOLDA, then exhibit the prioritized labels to interpret the topics, and finally illustrate the performance of the proposed framework on YouTube.

4.3.1 Result of AOLDA. Table 3 depicts the example topic-word distributions based on AOLDA, where the top five words are listed for each topic. According to the table, the general meanings of the topics are consistent along with versions. For example, Topic 1 is related to the video for all the three versions, and Topic 2 is constantly related to the user interface. However, for one topic, the specific meanings may be distinguished in the three versions. Take Topic 1 as an example. The topic may discuss the video description/link for version 11.07, while it talks about "click"-related things in version 11.10. It would be very laborious for developers to comprehend topics based on the top words. Therefore, we conduct automatic topic interpretation in the next step.

4.3.2 Result of Topic Interpretation. Table 4 illustrates the prioritized phrases for labeling topics, where only one of the three labels are listed for saving space. The highlighted labels in Table 4 are the emerging app issues detected by the anomaly discovery method in Section 3.2.2. Topic 1 of version 11.07 is interpreted as "description box", which is consistent with the meaning of that topic in Table 3 intuitively. Table 5 illustrates the ranked sentence for labeling each topic. Although phrase labels can be quickly understood, we discover that sentence labels can detail the information conveyed by phrases and interpret the topics more comprehensively. For example, the sentence label of Topic 1 for version 11.07 (*i.e.*, "...click a link in the description...") provides more details than the corresponding phrase label ("description box") in Table 4. With both issues in phrases and sentences, developers can efficiently spot and locate specific app issues. To help developers gain better understanding, we visualize the identified issues along with versions in Figure 7. By moving the mouse over Topic 10 of version 11.15, we can observe both phrase-level and sentence-level issues, among which the emerging ones are highlighted. Developers can readily track the changes of each topic and discover urgent issues in a timely manner.

4.3.3 Performance Evaluation. We collect the ground truth of YouTube based on the method in Section 4.2. Table 6 displays part of the changelogs. We manually inspect whether the identified app issues of one version can be reflected in the changelogs of the next version. According to Table 6, version 11.10 improves the user

Table 4: Topic labels in phrases for YouTube. The highlighted ones indicate detected emerging issues. The value after each label is the overall score of the label.

	v11.07	v11.10	v11.11
Topic 1	description box: 2.03	comment section: 1.48	notification center: 1.33
Topic 2	user interface: 1.25	split screen: 1.23	split screen: 0.94
Topic 3	playback error: 1.44	battery drain: 0.99	performance improvement: 1.41
Topic 4	certain spot: 1.81	cpu usage: 0.85	camera roll: 1.22
Topic 5	profile picture: 2.19	main page: 1.11	home screen: 1.18
Topic 6	say playback error: 1.54	long period: 0.92	force quit: 1.26
Topic 7	copyright issue: 1.11	bring back: 1.14	nothing happen: 1.53
Topic 8	take forever: 1.88	ten minute: 1.12	pure torture: 1.02
Topic 9	sound quality: 1.55	major issue: 1.45	buffer forever: 1.03
Topic 10	home button: 1.15	full screen: 1.07	home page: 1.29

Table 5: Topic labels in sentences for YouTube. The highlighted ones are the detected emerging issues.

	v11.07	v11.10
Topic 1	I mean it work but why do you take off where you would click a link in the description and it doesn't even let me go through the video: -0.05	It say error every time I try to reply back to a comment: 0.52
Topic 2	But right now the lack of multitasking have actually make it a better experience to use YouTube in safari: -0.79	Add split view and slide over but no picture in picture: -1.36
Topic 3	Please fix this app fix this bug and that playback error: -0.80	Dear YouTube please release a fix for overheat issue on older iPhone and the battery drain just too ridiculous: -0.45

Table 6: Changelog of YouTube

Version	Date	Changelog
11.10	22-Mar-16	(1) Added slide over and split view support (2) Moved home tabs into navigation bar for iPad in landscape mode (3) Fixed bug that prevented URLs in video descriptions from opening
11.11	29-Mar-16	(1) Fixed bug where accessibility VoiceOver looped over the same elements (2) Fixed issue where the video couldn't be exited after completing (3) Bug fixes and stability improvements

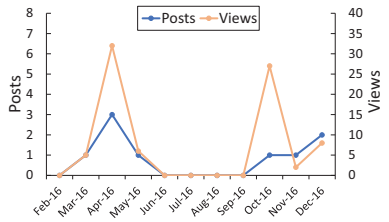


Figure 8: Count of posts and views related to the battery issue in YouTube iOS forum.

interface by adding the functionality of multitasking (*i.e.*, sliding over and splitting view [32]) and fixes the bug in video descriptions. Referring to Table 4 and Table 5, we discover that the two issues are detected by IDEA in Topic 1 and Topic 2 of the previous version 11.07. Then to statistically measure the performance of our framework, we employ the proposed three metrics in Section 4.2. Based on the collected 33 versions for YouTube, IDEA achieves $Precision_E$, $Recall_L$, and F_{hybrid} at 0.628, 0.666, 0.636 in sentence-level issues and 0.592, 0.472, and 0.523 in phrase-level issues, respectively.

Discussion of the performance: Since the changelogs may not cover all the changes in releases, the metric $Precision_E$ represents a lower bound of the performance. For example, the highlighted emerging issues, such as "split screen" and "battery drain" for version 11.10 in Table 4, are not clearly embodied by the changelog of version 11.11 (shown in Table 6). We then inspect the reason why the detected issues "fail" to be noticed by developers. We discover that "split screen" is one new added feature of version 11.10 and it is reasonable for a hot discussion about the drawbacks of this feature in the user reviews, which explains why "split view" is identified as one emerging issue. Then for the issue "battery drain", we dig into the official user forum of YouTube for iOS [41], and observe the number of posts and views of the issue by searching the phrase (illustrated in Figure 8). We find that there exists a sudden increase in the counts of posts and views around May 2016, which also demonstrates that the battery issue was an emerging issue for the version. Therefore, we summarize that changelogs may not completely cover all emerging issues, and our performance metric computes a lower bound of the performance of IDEA. The comparison with other methods can validate our proposed framework more sufficiently.

Table 7: Comparison result of different methods on six subject apps. The value under each app name indicates the average number of reviews across the versions.

App Name (#avg. reviews)	Method	Phrase			Sentence		
		$Precision_E$	$Recall_L$	F_{hybrid}	$Precision_E$	$Recall_L$	F_{hybrid}
NOAA Radar (523)	OLDA	0.468	0.528	0.473	**0.482**	0.622	0.534
	IDEA-R	**0.606**	0.461	0.520	0.478	0.570	0.503
	IDEA-S	0.250	**0.530**	0.340	0.417	0.547	0.473
	IDEA$^+$	0.571	0.497	**0.531**	0.476	**0.639**	**0.546**
Youtube (1,143)	OLDA	0.441	0.462	0.451	0.578	0.664	0.597
	IDEA-R	0.506	0.429	0.456	0.550	0.659	0.586
	IDEA-S	0.548	0.466	0.502	0.456	0.656	0.522
	IDEA$^+$	**0.592**	**0.472**	**0.523**	**0.628**	**0.666**	**0.636**
Viber (2,141)	OLDA	0.157	0.305	0.166	0.313	0.550	0.375
	IDEA-R	0.542	0.326	0.407	**0.625**	0.571	0.597
	IDEA-S	0.500	**0.342**	0.406	0.500	0.518	0.509
	IDEA$^+$	**0.625**	0.340	**0.440**	**0.625**	**0.651**	**0.638**
Clean Master (6,332)	OLDA	0.300	0.269	0.160	0.200	0.421	0.129
	IDEA-R	0.500	0.216	0.301	**0.750**	0.377	0.502
	IDEA-S	0.067	0.289	0.366	0.500	0.398	0.443
	IDEA$^+$	**0.667**	**0.318**	**0.431**	0.667	**0.434**	**0.526**
Ebay (3,943)	OLDA	0.167	0.238	0.196	0.500	0.488	0.494
	IDEA-R	**0.229**	0.243	0.220	**0.646**	0.496	0.561
	IDEA-S	0.125	**0.285**	0.132	0.354	0.476	0.406
	IDEA$^+$	**0.229**	0.251	**0.227**	**0.646**	**0.527**	**0.580**
SwiftKey (1,313)	OLDA	0.100	0.567	0.148	0.367	0.617	0.458
	IDEA-R	0.333	0.611	0.376	0.417	**0.733**	0.515
	IDEA-S	0.333	0.622	0.372	0.500	0.711	**0.587**
	IDEA$^+$	**0.517**	**0.653**	**0.523**	**0.583**	0.700	**0.587**

4.4 Result of RQ2: Comparison Results with Different Methods

For validating the performance of AOLDA in IDEA, we choose the typical method for online topic modeling - OLDA [1]. For evaluating the proposed topic labeling method in Section 3.3.2, we also compare with the method only considering the sentiment score for labeling (denoted as IDEA-R), and the method only considering the semantic score for labeling (denoted as IDEA-S). For clarity, our proposed framework is represented as IDEA$^+$. Table 7 illustrates the comparison results on the six subject apps. We discuss the performance of IDEA from three aspects in the following subsections.

4.4.1 Issues in Phrases v.s. Issues in Sentences. According to the results of IDEA$^+$ in Table 7, issues in sentences can attain better performance than those in phrases by 30.7%, 52.5%, and 43.2% in $Precision_E$, $Recall_L$, and F_{hybrid} on average respectively. This may be attributed to the fact that sentences can deliver more detailed and complete information than phrases (explained in Section 4.3.2), and

thereby cover more key terms in changelogs. Focusing on the metric F_{hybrid}, employing sentence-level issues improves the properties of using phrase-level issues by 2.7%∼1.56 times. For $Precision_E$, the issues in sentences increase those in phrases by -16.7%∼1.8 times. The negative increase only occurs to the app NOAA Radar, which may be because the small datasets of the app (512 reviews per version) introduce instability for our framework [23]. For the metric $Recall_L$, IDEA$^+$ shows an increase range of 7.1%∼1.1 times. Overall, sentence-level issues can better represent app issues, and we employ such issue representations for comparing with different methods in the following.

4.4.2 AOLDA v.s. OLDA. On average, IDEA$^+$ achieves 0.604, 0.603, and 0.585 for $Precision_E$, $Recall_L$, and F_{hybrid} respectively, while the OLDA-based method only obtains 0.407, 0.560, and 0.431 for the three metrics. Considering the metric F_{hybrid}, AOLDA enhances the performance of OLDA by 2.1%∼3.08 times, where OLDA presents the poorest performance (0.129) on the app with the largest quantity of reviews (*e.g.*, Clean Master with 6,332 reviews per version). For the metrics $Precision_E$ and $Recall_L$, our framework can improve the performance by -1.1%∼2.33 times and 0.3%∼18.4% respectively. Although IDEA$^+$ exhibits a slightly lower $Precision_E$ than the OLDA-based method for the app NOAA Radar, it shows better performance in both F_{hybrid} and $Recall_L$, which indicates that our framework can well balance the precision and recall in issue detection.

4.4.3 IDEA v.s. Different Topic Labeling Methods. We discover that IDEA$^+$ can achieve better performance than IDEA-R and present the increase rates at 7.1%, 7.3%, and 7.7% on average for the three metrics respectively. For F_{hybrid}, our framework improves IDEA-R by 3.4%∼14.0%. When compared with IDEA-S, our framework increases by 34.9%, 10.4%, and 20.7% on average in $Precision_E$, $Recall_L$, and F_{hybrid}, respectively. Therefore, both the user sentiment and semantic similarity should be considered for topic labeling.

4.5 RQ3: Effect of Different Parameter Settings

In this part, we demonstrate the impact of different parameter settings on the performance of our framework. We focus on analyzing two important parameters, including the window size w and the number of topics K. We also explain how we choose the parameters in our experiments.

4.5.1 Window Size. Figure 9 illustrates the results of different window sizes on two apps, including YouTube and Ebay. For both apps, the values of F_{hybrid} present an inverted "U" shape for both phrase-level and sentence-level issues. We attribute this to the reason that the topic distributions of the current version are strongly dependent on those of the previous versions. When the window size is set relatively small, the detected issues of current versions may be more divergent and unstable. However, larger window sizes may weaken the distinction of app issues among versions, which is unfavorable for detecting the emerging issues. Since w = 3 can achieve the best performance on our datasets (indicated in Figure 9), we set the window size as three in our experiments.

4.5.2 The Number of Topics. Generally, the topic number should be defined according to the size of the review collection [3]. In IDEA, a larger topic number can bring more prioritized app issues,

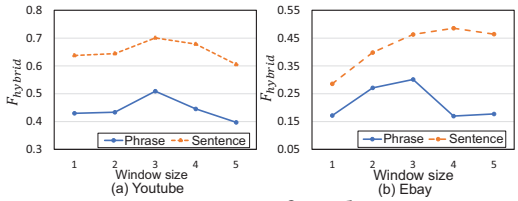


Figure 9: Impact of window size.

which can cover more changelogs (*i.e.*, increasing $Recall_L$). However, more app issues may be a double-edged sword, since the metric $Precision_E$ can be decreased. Figure 10 shows the results of different topic numbers on two apps, including NOAA Radar and Ebay. For Ebay (on average 3,943 reviews per version), the values of F_{hybrid} display an ascending tendency in both phrase-level and sentence-level issues. But for NOAA Radar (on average 523 reviews per version), a larger topic number will reduce the performance when using phrase-level issues. To better balance the precision and recall, we set the topic number as 10 during experiments.

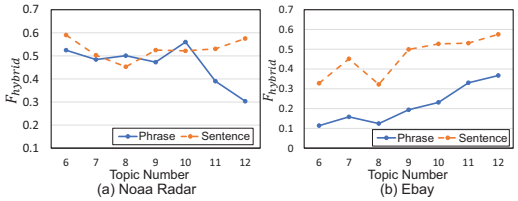


Figure 10: Impact of topic number.

5 IDEA IN PRACTICE

In this section, we explore the performance of IDEA in practice. First, we introduce a user survey conducted in Tencent. Then we describe the successful application of IDEA in Tencent's products.

5.1 User Survey

To further demonstrate the significance and effectiveness of our work, we conduct a user study among 45 staff in Tencent, with 29 developers (64.4%), five data analysts (11.1%), four product managers (8.9%), three maintenance engineers (6.7%), one test engineer (2.2%), and three from other positions (6.7%). The user study is conducted through an online questionnaire, which consists of six questions: one question on participants' background, four questions for experimental assessment, and one question for understanding their attitude towards such automatic analysis.

5.1.1 Changelogs as Ground Truth. We interview the participants about their opinions of using changelogs as ground truth, since changelogs may only include partial changes of the releases. The survey results indicate that 31 (68.9%) of the interviewees agree that changelogs can reflect modified issues of the new releases, and 10 (22.2%) of them indicate a strong approval. Moreover, 88.9% of participants think that changelogs embody the user concerns of the previous releases, with 11.1% echoing strong agreement. Since our framework aims to prioritize app issues based on user reviews, using changelogs as ground truth is reasonable.

5.1.2 Effectiveness of Our Framework. During the survey, we validate our framework in terms of three aspects: the presentation

style of IDEA's results, the performance achieved by our framework, and the significance of such automatic analysis. The survey results indicate that 75.6% of participants think the visualization with an *issue river* is comprehensible for them, while the phrase-level issues (only with the approval rate at 11.1%) are considered more difficult to understand than sentence-level issues (with an approval rate of 37.8%). For inquiring about their opinions of the performance of IDEA, we present the example results of WizNote [44] with $Precision_E$ and F_{hybrid} at 50%~60%. According to the survey, 88.9% of the interviewees think that the performance is acceptable in practical usage, and 31.1% strongly approve of such performance. In addition, all the participants think such automatic analysis of detecting emerging issues is significant for app development, with 73.3% of them strongly agreeing with this sentiment. These results provide strong evidence of the effectiveness of our framework.

5.2 Successful Story in Industrial Practice

Team X of Tencent aims to provide developers with abnormal events report and operation statistics of 20+ apps of Tencent. Traditional review analysis in X requires lots of manpower. With the increasing quantities of app reviews and the onslaught of spam in user reviews, X has been seeking a means of automatic analysis. We have successfully applied IDEA into X to maintain four apps with review quantities at 500~5,000 per day. The four apps serve hundreds of millions of users worldwide, and their quality is very important for the company. IDEA obtains user reviews by the hour or day based on the review collection API provided by X. The collected reviews are grouped by versions and processed in real time. The detected emerging issues are fed back to developers for further analysis.

In July of 2017, App Y encountered a serious problem when the content search service was not available for a period of time, and received a sudden increase in the amount of user feedback. With IDEA, the team X quickly identified the issue and reported it to the development team. The team also confirmed this issue.

Moreover, IDEA can efficiently analyze large numbers of reviews. We deploy IDEA on a PC with Intel(R) Xeon E5-2620v2 CPU (2.10 GHz, 6 cores) and 16GB RAM. For 36,000 product reviews per version, IDEA achieves a high throughput (nearly 160 reviews per second), and only consumes 1.02GB of memory on average. Overall, IDEA is proved to be effective and efficient in quickly pinpointing urgent app issues for developers in the industrial practice.

6 THREATS TO VALIDITY

First, we only select six subject apps for validating our framework and the apps represent a tiny portion of all apps on app markets. Since we utilize user reviews for detecting emerging issues, our methods can be easily applied to other apps, even those with other languages. Also, we alleviate this threat by choosing the apps from different categories and platforms. Second, the number of user reviews can impact the performance of IDEA. However, since small datasets can be easily analyzed manually, our framework aims for automatic analysis of large review datasets. We also mitigate this threat by selecting apps with different quantities of user reviews (on average 523~6,332 reviews per version). Third, the topic number should be manually defined, which can influence the performance of our framework. Such a threat stems from the original topic modeling method [4], which is still a great challenge in academia [45]. In this paper, we alleviate this threat by testing on different topic numbers (introduced in Section 4.5.2). In practice, we can employ heuristic approaches [45] to determine the optimal topic number.

7 RELATED WORK

7.1 App Review Mining

Some previous work [12, 20] focuses on identifying users' major concerns or preferences from app reviews [28]. Different from these work, where the reviews are manually analyzed, there exists some research which extracts app issues automatically. For example, [17, 42] design frameworks for automatic retrieval of mobile app feature requests from reviews. Mcilroy *et al.* [29] contribute to automatically assigning multiple labels to each review. Although the work [40, 25, 14] classify app reviews into different categories for recommending software updates, they mainly analyze static reviews and pay little attention to tracking issue changes. In [21, 9, 15, 27], the authors analyze variations in app ratings, prices, or review sizes along with time, but the issues are neither identified automatically nor studied online. Similarly, online review analysis is not the focus of Gao *et al.*'s work [10, 11]. There also exists some work [13] focusing on analyzing the parts of apps that are loved by users. Different from previous efforts, our work aims to detect the emerging issues automatically and dynamically. We employ changelogs for verifying effectiveness of our framework. Moreover, we present app issues in an interactive and comprehensible manner.

7.2 Emerging Topic Detection

There are research efforts focused on detecting emerging topics in social media, such as Twitter [5] and Microblog [7]. Online topic models [1, 22] are the typical methods for discovering burst topics. We are the first to apply online topic modeling methods into app reviews, and we improve on previous work by proposing a novel AOLDA. AOLDA can adaptively combine the topics in previous app versions and greatly enhances the performance of OLDA [1].

8 CONCLUSION

Timely and effectively detecting app issues is crucial for app developers. We propose IDEA, a novel framework for automatically identifying emerging issues from user reviews. Our framework can be easily applied to text-based online detection tasks and report emerging issues timely. Industrial practice also validates the effectiveness of IDEA. In the future, we will refine IDEA to be capable of defining the topic number automatically, and make IDEA a distributed algorithm for supporting ultra-large-scale datasets.

ACKNOWLEDGMENTS

We greatly thank Weiwen Qiu and Jun Ouyang of Tencent for assisting our industry study. This work was supported by the Key Project of National Natural Science Foundation of China (No. 61332010 and No. 61472338), the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14234416 and No. 14208815 of the General Research Fund), and Microsoft Research Asia via 2018 MSRA Collaborative Research Award.

REFERENCES

[1] Loulwah AlSumait, Daniel Barbará, and Carlotta Domeniconi. "On-line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking". In: *Proceedings of the 8th IEEE International Conference on Data Mining, ICDM 2008, December 15-19, 2008, Pisa, Italy*. 2008, pp. 3–12.

[2] *App Annie*. https://www.appannie.com/en/.

[3] R. Arun et al. "On Finding the Natural Number of Topics with Latent Dirichlet Allocation: Some Observations". In: *Advances in Knowledge Discovery and Data Mining, 14th Pacific-Asia Conference, PAKDD 2010, Hyderabad, India, June 21-24, 2010. Proceedings. Part I*. 2010, pp. 391–402.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.

[5] Mario Cataldi, Luigi Di Caro, and Claudio Schifanella. "Emerging topic detection on twitter based on temporal and social terms evaluation". In: *Proceedings of the Tenth International Workshop on Multimedia Data Mining (MDMKDD)*. ACM. 2010, p. 4.

[6] Ning Chen et al. "AR-miner: mining informative reviews for developers from mobile app marketplace". In: *36th International Conference on Software Engineering, ICSE 2014, Hyderabad, India - May 31 - June 07, 2014*. 2014, pp. 767–778.

[7] Yan Chen et al. "Emerging topic detection for organizations from microblogs". In: *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2013, Dublin, Ireland - July 28 - August 01, 2013*. 2013, pp. 43–52.

[8] *Facebook Messenger is getting slammed by tons of negative reviews*. http://www.businessinsider.com/facebook-messenger-app-store-reviews-are-humiliating-2014-8.

[9] Bin Fu et al. "Why people hate your app: making sense of user feedback in a mobile app store". In: *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. 2013, pp. 1276–1284.

[10] Cuiyun Gao et al. "AR-Tracker: Track the Dynamics of Mobile Apps via User Review Mining". In: *2015 IEEE Symposium on Service-Oriented System Engineering, SOSE 2015, San Francisco Bay, CA, USA, March 30 - April 3, 2015*. 2015, pp. 284–290.

[11] Cuiyun Gao et al. "PAID: Prioritizing app issues for developers by tracking user reviews over versions". In: *26th IEEE International Symposium on Software Reliability Engineering, ISSRE 2015, Gaithersbury, MD, USA, November 2-5, 2015*. 2015, pp. 35–45.

[12] Judith Gebauer, Ya Tang, and Chaiwat Baimai. "User requirements of mobile technology: results from a content analysis of user reviews". In: *Inf. Syst. E-Business Management* 6.4 (2008), pp. 361–384.

[13] Xiaodong Gu and Sunghun Kim. ""What Parts of Your Apps are Loved by Users?" (T)". In: *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*. 2015, pp. 760–770.

[14] Emitza Guzman and Walid Maalej. "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews". In: *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*. 2014, pp. 153–162.

[15] Leonard Hoon et al. "An analysis of the mobile app review landscape: trends and implications". In: *Faculty of Information and Communication Technologies, Swinburne University of Technology, Tech. Rep* (2013).

[16] Jiajia Huang et al. "A probabilistic method for emerging topic tracking in Microblog stream". In: *World Wide Web* 20.2 (2017), pp. 325–350.

[17] Claudia Iacob and Rachel Harrison. "Retrieving and analyzing mobile apps feature requests from online reviews". In: *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR 2013, San Francisco, CA, USA, May 18-19, 2013*. 2013, pp. 41–44.

[18] Aminul Islam and Diana Zaiu Inkpen. "Semantic text similarity using corpus-based word similarity and string similarity". In: *TKDD* 2.2 (2008), 10:1–10:25.

[19] *Jensen Shannon divergence*. https://en.wikipedia.org/wiki/Jensen-Shannon_divergence.

[20] H Khalid et al. "What do mobile app users complain about? A study on free iOS apps. 2014". In: *IEEE Software* 10 (2015).

[21] Jieun Kim et al. "Trends and relationships of smartphone application services: Analysis of apple app store using text mining-based network analysis". In: *Proceedings of the 4th ISPIM Innovation Symposium*. 2012.

[22] Jey Han Lau, Nigel Collier, and Timothy Baldwin. "On-line Trend Analysis with Topic Models: \#twitter Trends Detection Topic Model Online". In: *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*. 2012, pp. 1519–1534.

[23] *LDA on small datasets*. https://stats.stackexchange.com/questions/78926/at-what-point-does-lda-latent-dirichlet-allocation-not-make-sense-to-use.

[24] Qingwei Lin et al. "iDice: problem identification for emerging issues". In: *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*. 2016, pp. 214–224.

[25] Walid Maalej and Hadeer Nabil. "Bug report, feature request, or simply praise? On automatically classifying app reviews". In: *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*. 2015, pp. 116–125.

[26] Yichuan Man et al. "Experience Report: Understanding Cross-Platform App Issues from User Reviews". In: *27th IEEE International Symposium on Software Reliability Engineering, ISSRE 2016, Ottawa, ON, Canada, October 23-27, 2016*. 2016, pp. 138–149.

[27] William Martin, Federica Sarro, and Mark Harman. "Causal impact analysis for app releases in google play". In: *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*. 2016, pp. 435–446.

[28] William Martin et al. "A Survey of App Store Analysis for Software Engineering". In: *IEEE Trans. Software Eng.* 43.9 (2017), pp. 817–847.

[29] Stuart McIlroy et al. "Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews". In: *Empirical Software Engineering* 21.3 (2016), pp. 1067–1106.

[30] Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. "Automatic labeling of multinomial topic models". In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*. 2007, pp. 490–499.

[31] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 2013, pp. 3111–3119.

[32] *Multi-tasking in iOS*. https://developer.apple.com/ios/human-interface-guidelines/features/multitasking/.

[33] Thanh-Son Nguyen, Hady Wirawan Lauw, and Panayiotis Tsaparas. "Review Synthesis for Micro-Review Summarization". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*. 2015, pp. 169–178.

[34] *NLTK*. http://www.nltk.org.

[35] *PMI*. https://en.wikipedia.org/wiki/Pointwise_mutual_information.

[36] *Punkt tokenizer*. http://www.nltk.org/modules/nltk/tokenize/punkt.html.

[37] Peter J. Rousseeuw and Mia Hubert. "Robust statistics for outlier detection". In: *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery* 1.1 (2011), pp. 73–79.

[38] Federica Sarro et al. "Feature lifecycles as they spread, migrate, remain, and die in App Stores". In: *23rd IEEE International Requirements Engineering Conference, RE 2015, Ottawa, ON, Canada, August 24-28, 2015*. 2015, pp. 76–85.

[39] *Softmax function*. https://en.wikipedia.org/wiki/Softmax_function.

[40] Andrea Di Sorbo et al. "What would users change in my app? summarizing app reviews for recommending software changes". In: *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, Seattle, WA, USA, November 13-18, 2016*. 2016, pp. 499–510.

[41] *User forum of Youtube iOS*. https://productforums.google.com/forum/#!forum/youtube.

[42] Phong Minh Vu et al. "Mining User Opinions in Mobile App Reviews: A Keyword-Based Approach (T)". In: *30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015*. 2015, pp. 749–759.

[43] Xiaojun Wan and Tianming Wang. "Automatic Labeling of Topic Models Using Text Summaries". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. 2016.

[44] *WizNote*. https://www.wiz.cn/.

[45] Weizhong Zhao et al. "A heuristic approach to determine an appropriate number of topics in topic modeling". In: *BMC Bioinformatics* 16.13 (2015), S8.