Emerging App Issue Identification via Online Joint Sentiment-Topic Tracing

Cuiyun Gao, Jichuan Zeng, Zhiyuan Wen, David Lo, Xin Xia,
Irwin King, *Fellow, IEEE*, and Michael R. Lyu, *Fellow, IEEE*

Abstract—Millions of mobile apps are available in app stores, such as Apple's App Store and Google Play. For a mobile app, it would be increasingly challenging to stand out from the enormous competitors and become prevalent among users. Good user experience and well-designed functionalities are the keys to a successful app. To achieve this, popular apps usually schedule their updates frequently. If we can capture the critical app issues faced by users in a timely and accurate manner, developers can make timely updates, and good user experience can be ensured. There exist prior studies on analyzing reviews for detecting emerging app issues. These studies are usually based on topic modeling or clustering techniques. However, the short-length characteristics and sentiment of user reviews have not been considered. In this paper, we propose a novel emerging issue detection approach named MERIT to take into consideration the two aforementioned characteristics. Specifically, we propose an Adaptive Online Biterm Sentiment-Topic (AOBST) model for jointly modeling topics and corresponding sentiments that takes into consideration app versions. Based on the AOBST model, we infer the topics negatively reflected in user reviews for one app version, and automatically interpret the meaning of the topics with most relevant phrases and sentences. Experiments on popular apps from Google Play and Apple's App Store demonstrate the effectiveness of MERIT in identifying emerging app issues, improving the state-of-the-art method by 22.3 percent in terms of F1-score. In terms of efficiency, MERIT can return results within acceptable time.

Index Terms—User reviews, online topic modeling, emerging issues, review sentiment, word embedding

1 Introduction

Mobile apps keep gaining popularity over the last few years. According to Statista [1], the global mobile internet user penetration in 2016 has exceeded half the world's population. During the third quarter of 2018, Android users were able to choose from 2.1 million apps, while Apple's App Store[1] provided almost 2 million apps. While users have a large number of products to choose from, the apps are facing immensely fierce competition to survive.

The popular mobile app stores, such as Google Play and App Store, use the star-rating mechanism to gather users' ratings and feedback. The feedback and ratings can impact an app's ranking on these stores, and further influence its discovery and trial. A survey in 2015 [2] reported that only 15~50 percent of the users would consider downloading a low-rated app, while for the high-rated apps, the ratio reached 96 percent. Thus, ensuring good user experience and keeping users engaged can help maintain high download numbers and increase benefits to app developers.

Recent studies [3], [4], [5] showed that frequently-updated apps could benefit in terms of increase in ranking. This is the case since the popular app stores factor in the freshness of an app in the ranking process. Additionally, app updates can also improve user experience. Specifically, McIlroy *et al.* [3] found that the rationale behind updates is often related to bug-fixing (63 percent of the time), new features (35 percent), and feature improvement (30 percent). However, not every update can definitely lead to positive user experience and high ranking [6]. For example, the updated Android and iOS versions of Skype released in June 2017 received a flood of complaints as the new design removed the key functionality and features available in the older version, such as the visibility of online friends [7]. As a result, its user rating on the App Store plunged from 4.5 to 1.5 stars shortly after the update [8]. Such situations are not unusual, c.f. [9], [10], and can cause customer churn and losses to app developers. The losses could be limited if the issues were recognized timely. In this work, we aim at accurately detecting *emerging* app issues by analyzing user feedback.

IDEA [11] is one of the most recent works that can be directly applied to detect emerging issues/topics[2] from user feedback. IDEA takes user reviews distributed in

[1] Apple's App Store is indicated as App Store for simplicity throughout the paper.

[2] The topics and issues are semantically equal in this paper.

- *Cuiyun Gao, Jichuan Zeng, Irwin King, and Michael R. Lyu are with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. E-mail: gaocuiyun@hit.edu.cn, {jczeng, king, lyu}@cse.cuhk.edu.hk.*
- *Zhiyuan Wen is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong, China. E-mail: cszwen@comp.polyu.edu.hk.*
- *David Lo is with the School of Information Systems, Singapore Management University, Singapore 188065. E-mail: davidlo@smu.edu.sg.*
- *Xin Xia is with the Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia. E-mail: xin.xia@monash.edu.*

Manuscript received 14 September 2020; revised 15 March 2021; accepted 16 April 2021. Date of publication 28 April 2021; date of current version 15 August 2022.
(Corresponding author: Jichuan Zeng.)
Recommended for acceptance by L. Tan.
Digital Object Identifier no. 10.1109/TSE.2021.3076179

consecutive app versions as input, and outputs emerging app issues in the level of phrases and sentences. A modified online topic modeling approach is utilized to infer topics of the text corpus in consecutive time periods. Finally, IDEA employs a *topic labeling* approach to automatically prioritize the phrases/sentences that are semantically representative of the topics. The prioritized phrases/sentences are regarded as descriptions of emerging issues. Although the approach achieves reasonable performance, it has several limitations in accurately detecting emerging app issues as it does not consider the following characteristics of user feedback and exists inefficiency during topic labeling:

1) *Short-Length Nature of User Feedback:* User feedbacks are usually short in length, providing limited context. According to Genc-Nayebi and Abran [12], the average length of app reviews is 71 characters. Besides, since the proposed online topic modeling approach in IDEA is built upon LDA (Latent Dirichlet Allocation) [13] and LDA is not considered to work well on short texts [14], [15], IDEA may fail to accurately capture the topics of user reviews.

2) *Sentiment of Topics:* Emerging issues are generally the issues that negatively impact user experience, such as bugs, or new features requested by users. Reviews corresponding to these issues are usually accompanied by poor ratings. However, current topic-modeling-based approaches do not explicitly distinguish topics based on their sentiment, which may identify the positive ones as emerging and generate false positives.

3) *Ineffectiveness of the Topic Labeling Approaches:* The previous topic labeling approaches represent topics [11], [16] with representative candidate phrases/sentences based on their similarities with the current topics in terms of topic distributions. Since topic-word distributions may not capture well the semantic relations between words [17], [18], [19], [20], these topic labeling approaches may choose improper phrases/sentences for interpreting the topics. For example, the words "*click*" and "*popup*" are semantically related to the word "*ad*", but they may not have high probabilities in the topic distribution of the "ad" issue. As a result, the phrases/sentences containing "*click*" or "*popup*"may not be recommended as the topic label. As the topic interpretations directly represent app issues, false emerging issues would be alerted.

In this paper, we propose an i**M**proved **E**me**R**ging **I**ssue de**T**ection approach, named **MERIT**, to mitigate these limitations and more accurately detect emerging app issues. Different from the topic modeling approach in IDEA [11], where a topic is a probability distribution over single words, MERIT considers topics over a mixture of *biterms*. Here, a *biterm* is an unordered word-pair co-occurring in a short context. The biterm-based model has been shown to be effective in alleviating the data sparseness problem of short texts and significantly enhance the topic learning [14]. To tackle the second limitation, MERIT distinctly considers sentiment-related prior during topic modeling, and thereby can well distinguish positive and negative topics. The negative topics are adopted for emerging app issue detection. For the third problem, MERIT employs word embedding [21] which has been shown to be effective in converting words into their distributed representations [22], [23], during the topic labeling process.

To evaluate the effectiveness of MERIT, we perform experiments on the same six real-world apps as IDEA [11]. Our results show that MERIT can more accurately identify emerging app issues than the baselines, with improvements in precision, recall, and F1-score of 21.0, 20.9, and 22.3 percent respectively. We discover that MERIT can capture more coherent topics (i.e., the top words belonging to one topic are more semantically consistent) from user reviews, focus on the negative topics, and better prioritize phrases/sentences for interpreting topics. We also demonstrate that MERIT can output results with reasonable time cost despite its more complex design than IDEA.

The main contributions of our work are as follows:

- We propose a novel online topic modeling approach for detecting emerging app issues. The proposed approach can not only generate more coherent topics but also well distinguish positive and negative topics during analysis of user reviews.
- We design a novel topic labeling approach based on word embedding techniques to well prioritize phrases/sentences for interpreting the meaning of each topic.
- We develop MERIT,[3] a new tool that can detect emerging app issues from online reviews.
- We evaluate the effectiveness and efficiency of MERIT on real-world mobile apps.

Paper Structure. Section 2 describes the background knowledge and motivation of our work. Section 3 presents the methodology we propose for accurate emerging app issue detection. Section 4 introduces the experimental setup. Section 5 describes the evaluation results, followed by Section 6 that discusses the limitation of our approach. Section 7 presents related studies. We conclude and mention future work in Section 8.

2 Preliminaries

In this section, we present the background knowledge for facilitating readers' understanding, including emerging issue detection, topic modeling, and word embeddings.

2.1 Emerging Issue Detection

In mainstream topic detection studies [24], [25], [26], [27], an event/issue is considered emerging if it is (heavily) discussed in current time slice but not previously. The application scenario of these studies is generally targeted at social media platforms, e.g., Twitter and Sina Weibo. However, user discussion on social media and app stores has significant differences. One difference is that the app reviews usually associate with specific app versions, while typical social media contents do not concern with the version concept. Another big difference is that emerging event detection for social media is simply dependent on volume of user posted content, regardless of the sentiment associated with it; while for app reviews, user sentiment is one indicator to emerging issues [28], [29]. Thus, simply applying standard emerging issue detection methods from the social media field is not

[3] Available at https://github.com/armor-ai/MERIT.

optimal for our scenario. In this paper, we define an emerging app issue as follows.

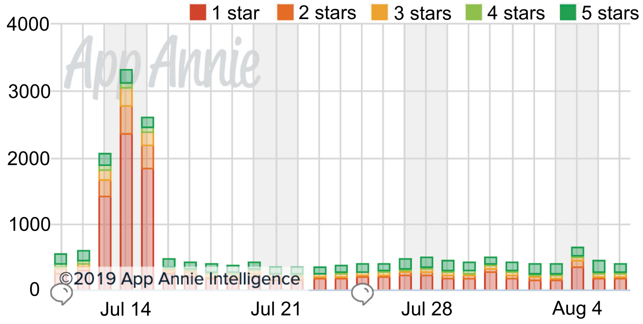


Fig. 1. Number of review changes along with time for the Facebook app on Google Play. Different color bars represent the rating distributions with reference shown on the top right. (Statistics from App Annie [31]).

Definition (Emerging App Issue). An issue reported in user review(s) at a particular time slice is defined as an emerging issue if its distributions presented marginal fluctuations in previous time slices in previous time slices, corresponds to a *significant increase* in terms of *the percentageof reviews* reporting it, and is *negatively reported by users* in the current time slice.

For example, Fig. 1 illustrates the changes of the number of reviews and user rating distributions over time for Facebook. As can be seen, the number of reviews received on July 14, 2019 is significantly larger than the number of reviews received on July 12, 2019; this is true especially for the one-star reviews (represented by the red bar), which means that an emerging issue may exist for the recent update. By checking the detailed user reviews, we find that it was related to a huge redesign of the app in July [30]. With continuous monitoring and accurate identification of emerging issues, such problem can be detected in a timely manner. Developers can then be alerted of the need to perform further maintenance activities to ensure good user experience. We also discover that both number of reviews reporting the issue and user rating can indicate the emergence of an issue. Involving review ratings in the analysis can help our tool avoid false positives, i.e., topics that are mentioned in many reviews but do not correspond to important problems that need to be rectified urgently [27].

2.2 Topic Modeling

Topic models [13], [32] have been proven useful for discovering latent structures in a collection of documents [33], [34]. The models capture the co-occurrence of words in the collection under a probabilistic framework by assuming that each topic can be represented by a set of word clusters. In this way, the models can uncover latent semantic structures (i.e., topics) in the documents. Due to the unsupervised nature, the models such as Probabilistic Latent Semantic Analysis (PLSA) [32] and Latent Dirichlet Allocation (LDA) [13] have been widely applied in mining software repositories [11], [29], [35] where labeled datasets are not available in practice. The outputs of the topic models are two matrices: (1) Document-topic matrix, denoted as $\Theta \in \mathbb{R}^{D \times K}$, where D is the number of documents and K is the number of topics. The ith row of the matrix, i.e., $\theta_i \in \mathbb{R}^K$, is a topic distribution vector for the ith document; (2) Topic-word matrix, denoted as $\Phi \in \mathbb{R}^{K \times V}$, where V is the total number of unique words (i.e., vocabulary). The ith row of the matrix, that is, $\phi_i \in \mathbb{R}^V$, is a word distribution vector for the ith topic. Table 1 shows an example of the output of topic models, with top five words and corresponding probabilities presented for each topic.

TABLE 1
Example for Illustrating the Output of Topic Models

Topic 1		Topic 2		Topic 3	
term	weight	term	weight	term	weight
video	0.014	update	0.021	link	0.021
play	0.011	better	0.006	notification	0.015
stop	0.009	slow	0.006	sign	0.013
audio	0.008	stupid	0.005	browser	0.012
music	0.007	stick	0.004	open	0.011

The top-five words of each topic are listed, with corresponding probabilities. The meaning of each topic can be manually deduced from the top-ranked words.

Latent Dirichlet Allocation (LDA) [13] assumes that each document consists of a mixture of topics, and each word in documents belongs to one topic.

Biterm Topic Model (BTM) [14] is designed specifically for modeling topics of short texts. BTM extends a document into a set of biterms (i.e., two terms) which includes all combinations of any two distinct words appearing in one document. In this way, BTM can enrich the short texts by explicitly modeling the word co-occurrence patterns. In BTM, instead of assuming that each word belongs to one topic, it assumes that each biterm relates to one topic. BTM has demonstrated better performance than LDA in modeling short texts [36], [37].

Joint Sentiment/Topic Model (JST) [38] is also a variant of LDA, but involves the sentiment of each topic. JST assumes that each word should be associated with one sentiment polarity, such as positive, neutral, and negative. The output of JST is also two matrices but with three dimensions: (1) Document-sentiment-topic matrix, denoted as $\Theta \in \mathbb{R}^{D \times 3 \times K}$ where 3 is the number of sentiment polarities (positive, negative, and neutral); (2) Sentiment-topic-word matrix, i.e., $\Phi \in \mathbb{R}^{3 \times K \times V}$.

Fig. 2 illustrates the differences among the above topic models in terms of their input and output. However, documents mostly come as ephemeral streams in most scenarios,

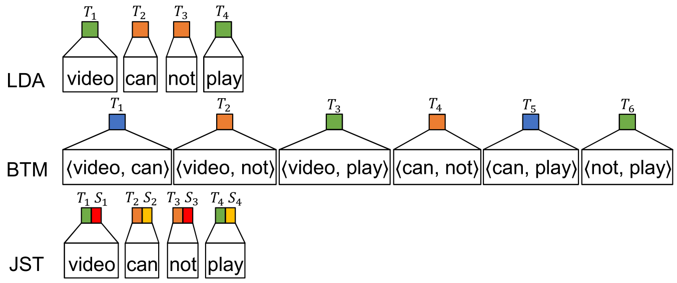


Fig. 2. Illustration of three typical topic modeling approaches, including LDA, BTM, and JST, based on an example of user review ("*video can not play*"). The symbols T_i and S_i denote the inferred topic and sentiment of the ith token in the review, respectively. Different colors indicate different topics or sentiments conveyed by the corresponding words.

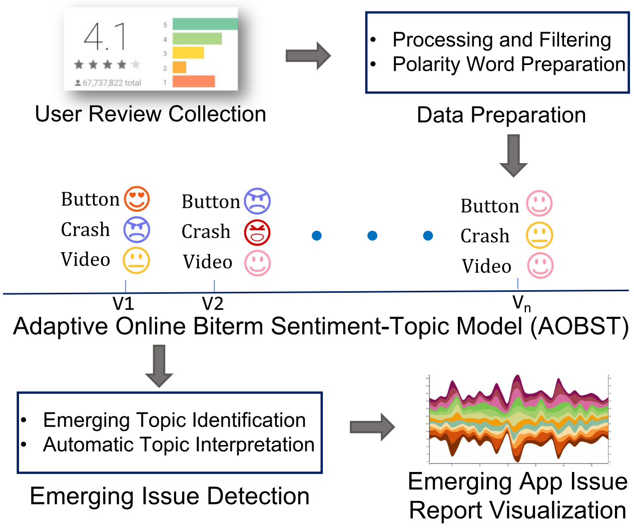

4.1
67,737,822 total
User Review Collection
Processing and Filtering
Polarity Word Preparation
Data Preparation
Button
Crash
Video
Button
Crash
Video
Button
Crash
Video
V1
V2
Vn
Adaptive Online Biterm Sentiment-Topic Model (AOBST)
Emerging Topic Identification
Automatic Topic Interpretation
Emerging Issue Detection
Emerging App Issue Report Visualization

Fig. 3. Overview of the proposed framework - MERIT.

such as scientific articles and Twitter messages, and thus the topics and subordinate attributes (e.g., word distributions) in the documents are time-evolving [33]. To capture such topic variations, online topic models, including Online LDA (OLDA) [38], have been proposed. The output of online topic models is topic distributions along with consecutive time slices, and Table 1 can be regarded as the topic distributions of the documents in one time slice.

2.3 Word Embeddings

Word embedding (also known as distributed representation [22], [39]) is one of the most popular techniques that represent document vocabulary by training on a large text corpus. They map each word to a low-dimension real-valued vector and are capable of capturing the context of a word based on the semantic similarity relations with other words. The words that exhibit the same semantics have similar vector representations. For example, suppose the word *"photo"* is represented as [0.53, -0.21, 0.02] and the word *"image"* is represented as [0.49, -0.35, 0.01]. From their vectors, we can estimate their distance and identify their semantic relation. Word embedding is usually implemented through training a machine learning model such as CBOW and Skip-Gram [39] on large datasets. Phrases, sentences, and documents can also be embedded as vectors based on word embedding techniques. For example, a simple way of sentence embedding is to compute the average word embeddings in the sentence [40].

3 Overview of MERIT

Fig. 3 illustrates the detailed steps of the proposed framework - MERIT, mainly including five steps: user review collection, data preparation (Section 3.1), training and use of an adaptive online biterm sentiment-topic model (Section 3.2), emerging issue detection (Section 3.3), and emerging app issue report visualization (Section 3.4).

3.1 Data Preparation

Since user reviews are usually written on mobile phones with limited keyboards on mobile screens, they often contain a large number of noisy words, such as misspelled words and abbreviations (e.g., *"asap"*). In the following, we elaborate on the preprocessing method and also the method to prepare *polarity-carrying words* (i.e., the words that carry sentiment polarities, e.g., positive or negative) for the subsequent modeling process.

3.1.1 *Preprocessing and Filtering*

We adopt the preprocessing method described in [11]. For completeness sake, we briefly describe the steps here. We first convert all the words into their lowercase and then lemmatize them into their root forms following the lemmatization method described in [41]. We also adopt the rule-based methods in [41], [42] to rectify repetitive words (e.g., *"very very good"* to *"very good"*), misspelled words, and remove non-English words. Then, we extract phrases (mainly referring to two consecutive words following our previous work [11]) for the topic interpretation procedure in Section 3.3. We use PMI (Point-wise Mutual Information) [43], a measure of word association in information theory and statistics, to identify meaningful phrases based on co-occurrence frequencies.

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

where $p(w_i w_j)$ and $p(w_i)$ (or $p(w_j)$) indicate the co-occurrence probability of the phrase $w_i w_j$ and the probability of the word w_i (or w_j) in the review collection. A higher PMI value illustrates that the two words appear together more frequently, and are more likely to be a meaningful phrase. The phrases with PMIs larger than a manually-defined threshold[4] are extracted. Finally, we reduce the non-informative words using the predefined list of to-be-filtered words proposed by Gao *et al.* [11], including abbreviations (e.g., *"ur"*) and stop words (e.g., *"is"*).

3.1.2 *Polarity Word Preparation*

To infer the sentiment affiliated with each topic, we first create a list containing words and their corresponding polarities. We build the word list leveraging opinion lexicons published by Hu and Liu [44], [45], which include 2,006 positive words and 4,783 negative words identified from customer reviews. However, since the published lexicons are from product reviews, there may exist discrepancies with the app review scenario. To mitigate the discrepancy, we adopt the collected reviews and extract 15,704 opinion words, including verbs, adverbs, or adjectives based on part-of-speech tagging [46]. Due to the huge effort in manually labeling polarities of all the extracted opinion words, we randomly select 500/15,704 words based on their frequencies for manual labeling. The selected words are a statistically significant proportion of the whole opinion words, providing us with a confidence level of 95 percent and a confidence interval of 5 percent. The labeling process is conducted by the first author and two Computer Science Ph.D. students. Each word needs to be labeled by two of the annotators, and the label options can be "1 (positive)", "0

[4] The threshold is experimentally set.

TABLE 2
Examples of Labeled Word Polarities

Word	Sentiment	Word	Sentiment
comfortable	1	unnecessary	-1
buggy	-1	learn	0
weird	-1	unclear	-1
beneficial	1	exclude	-1
consistent	1	blame	-1
inform	0	unlock	0

Positive, neutral, and negative sentiments are indicated as "1", "0", and "-1", respectively.

(neutral)", or "-1 (negative)". The labeling achieves 0.79 agreement rate[5] and full agreement after discussion. Table 2 lists some examples of the labeled word polarities. We combine the manually-labeled 500 opinion lexicons from the collected app reviews with the published ones [44] as our *word polarity list*.[6] By integrating the opinion words from app reviews, we can mitigate the polarity discrepancy caused by solely using the polarity words from product reviews.

All the non-filtered words, phrases extracted following Equ. (1) (where the words in each phrase are concatenated with "_"), and the word polarity list are fed into the topic modeling process.

3.2 Adaptive Online Biterm Sentiment-Topic Model (AOBST)

Inspired by existing topic modeling techniques [14], [38], we propose a novel unsupervised model named AOBST (Adaptive Online Biterm Sentiment-Topic Model) for jointly modeling the topics and sentiment of app reviews. We will first illustrate the proposed biterm sentiment-topic model for building connections between topics and sentiment, and then elaborate on its online adaption.

3.2.1 Biterm Sentiment-Topic Model

To address the first two limitations described in Section 1, including the short-length and sentiment characteristics of app reviews, we propose a Biterm Sentiment-Topic (BST) model. The BST model is built upon BTM and JST, since BTM has shown better performance than LDA in modeling short texts and JST can jointly model topics and sentiment. We introduce the details of the proposed BST below.

BST assumes that each app review is a set of biterms B, and each biterm $b = (w_i, w_j)$ belongs to one sentiment polarity s and one topic z. The modeling process can be described as below:

- Construct a sentiment distribution $\pi \sim Dir(\gamma)$.
- For each sentiment polarity s:
  - Construct a topic distribution for sentiment s, $\theta_s \sim Dir(\alpha)$.
  - For each topic z:
    - Construct a word distribution for sentiment s and topic z, $\phi_{s,z} \sim Dir(\beta)$.
- For each biterm b in the biterm set B:
  - Choose a sentiment polarity $s_b \sim Multi(\pi)$.
  - Choose a topic assignment $z_b \sim Multi(\theta_{s_b})$.
  - For each word w_i in the biterm
    - Choose a word w_i based on the distribution over words, i.e., $w_i \sim Multi(\phi_{s_b,z_b})$, where z_b and s_b denotes the topic and sentiment polarity, respectively.

The hyperparameters γ, α, and β in BST can be treated as the prior counts of the sampled sentiment polarity s, the sampled topic z associated with sentiment polarity s, and the sampled words for topic z and sentiment polarity s, respectively. $Dir(\cdot)$ and $Multi(\cdot)$ represent Dirichlet distribution and multinomial distribution parameterized by $\cdot$, respectively. The probability of a biterm $b = (w_i, w_j)$ can be calculated as

$$\begin{aligned} P(b) &= \sum_{s,z} P(z|s)P(w_i|s,z)P(w_j|s,z) \\ &= \sum_{s,z} \theta_s \phi_{i|s,z} \phi_{j|s,z}. \end{aligned} \tag{2}$$

The parameter matrices, i.e., $\{\Theta \in \mathbb{R}^{3\times K}, \Phi \in \mathbb{R}^{3\times K\times V}\}$, of BST can be inferred through Gibbs sampling [48] efficiently, given all the biterms B. The parameter matrix Φ is the sentiment-topic-word matrix, with an example shown in Fig. 5a. The first dimension of Φ is the sentiment polarity (i.e., $s \in \{1, 2, 3\}$ for representing each of the three sentiment — 1 = negative, 2 = neutral, 3 = positive). We can regard the second and third dimensions of Φ as a topic-word matrix ($\mathbb{R}^{K\times V}$), with each row indicating the probability distribution over words for the topic. By inspecting the topic examples extracted from Φ, shown in Fig. 5b, we can discover that the topics exhibit different sentiment polarities from the sentiment perspective.

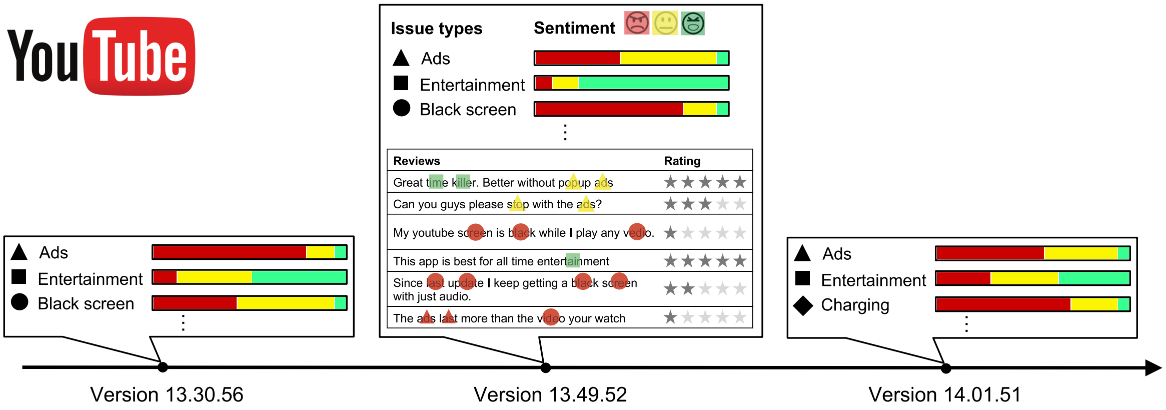
3.2.2 Adaptive Online Joint Sentiment-Topic Tracing

In the previous section, we have introduced BST for inferring sentiment-aware topics from an app review collection. In this section, we will describe an online adaption of BST to trace topic variations of review collections from consecutive app versions.

We first divide collected app reviews according to app versions, denoted as $R = \{R_1, R_2, \dots, R_t, \dots\}$, where R_t indicates all the reviews pertaining to the tth app version. In order to capture the topic evolution along with versions, we apply an adaptively online topic modeling mechanism [11] to BST, i.e., adaptive online biterm sentiment-topic model (AOBST). AOBST adaptively connects the sentiment-topic word distributions in previous app versions with the prior for the word distribution β of current app version. Specifically, we denote the sentiment-topic word distributions in previous ω version as $\{\phi^{t-1}, \dots, \phi^{t-i}, \dots, \phi^{t-\omega}\}$, where ω is the version window size determining the number of previous versions to be considered for analyzing the sentiment-topic word distributions of the current version. The *connection strength* η between the sentiment-topic word distribution ϕ^{t-i} in the previous ith version and the prior β^t

[5] The agreement rate is computed based on Cohen's kappa [47].

[6] For the coincident words in the 500 opinion lexicons and 6,789 published polarity words, we choose their polarities as the labels in app review scenario. In total, we obtain 7,215 opinion words and their polarities. Full list of the word polarities can be found in our replication package.



Reviews	Rating
Great time killer. Better without popup ads	★★★★★
Can you guys please stop with the ads?	★★★☆☆
My youtube screen is black while I play any vedio.	★☆☆☆☆
This app is best for all time entertainment	★★★★★
Since last update I keep getting a black screen with just audio.	★★☆☆☆
The ads last more than the video your watch	★☆☆☆☆

Fig. 4. Illustration of the output of the proposed adaptive online biterm sentiment-topic model (AOBST). The horizontal axis represents examples of released consecutive versions for YouTube on App Store. For each version, the AOBST model generates its topic distributions, indicated by different shapes, and the corresponding sentiment distributions (displayed with color bars beside the topics). Different colors represent different sentiment levels, from angry to just-so-so to happy.

of the current tth version is defined as their similarity, which is calculated in the following:

$$\eta_{s,z}^{t,i} = \frac{\exp(\phi_{s,z}^{t-i} \cdot \beta_{s,z}^{t-1})}{\sum_{j=1}^{\omega} \exp(\phi_{s,z}^{t-j} \cdot \beta_{s,z}^{t-1})}, \tag{3}$$

where i denotes the ith previous version ($1 \leq i \leq \omega$), and s and z indicate the current sentiment and topic respectively. The dot product $\phi_{s,z}^{t-i} \cdot \beta_{s,z}^{t-1}$ computes the similarity between the word distribution of the previous ith version $\phi_{s,z}^{t-i}$ and the prior of $(t-1)$th version $\beta_{s,z}^{t-1}$. Such adaptive connection can endow the sentiments and topics of the previous versions with different contributions to the sentiment-topic inference of the current version [11]. The prior β^t is calculated as

$$\beta_{s,z}^{t} = \sum_{i=1}^{\omega} \eta_{s,z}^{t,i} \phi_{s,z}^{t-i}. \tag{4}$$

Based on AOBST, we can trace the variations of topics for different sentiment polarities along with app versions, as shown in Fig. 4. We describe the approaches to detect the emerging topics and automatically interpret the topic meanings with phrases and sentences in the next section. Since we aim at detecting app issues, which are generally expressed in an unfavorable manner by users, we focus on the negative topics during emerging issue detection.

3.3 Emerging Issue Detection

In this section, we describe how we determine the emerging app issues based on the evolution of the topics belonging to negative sentiment along with app versions.

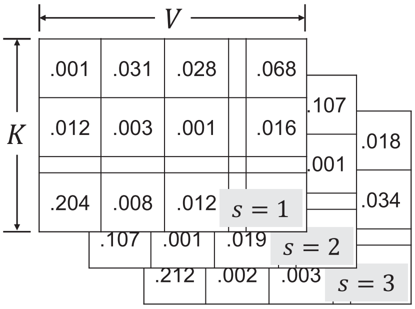
3.3.1 *Emerging Topic Identification*

Following the previous study in anomaly detection [49], anomalies are defined as data points that deviate significantly from the majorities within a group. In this work, we define the emerging topics as those present obvious differences with the counterparts in the previous versions. The identified topics are regarded as emerging topics. We focus on the topics inferred as negative during emerging topic detection.

We compute the difference of the zth negative topics between two consecutive versions, e.g., ϕ_z^t and ϕ_z^{t-1} and adopting the classic Jesen-Shannon (JS) divergence [50]. Higher JS value indicates that the two topic distributions exhibit a larger difference. In this way, we generate a $\omega \times K$ divergence matrix D_{JS} (where ω and K are the number of window size and topics respectively) for the versions in a window. We then use the typical outlier detection method [51] to detect the anomalies

$$\frac{\{D_{JS}\}_z^t - \overline{D_{JS}}}{\sigma} > \delta, \tag{5}$$

where $\overline{D_{JS}}$ and σ denote the mean and standard deviation of all the values in the computed D_{JS} matrix. The threshold



(a) Example of the 3-D matrix Φ.

Positive sentiment (s=1)				Neutral sentiment (s=2)				Negative sentiment (s=3)			
Topic 1		Topic 2		Topic 1		Topic 2		Topic 1		Topic 2	
w	$P(w\|s,z)$	w	$P(w\|s,z)$	w	$P(w\|s,z)$	w	$P(w\|s,z)$	w	$P(w\|s,z)$	w	$P(w\|s,z)$
great	.084	use	.059	see	.035	time	.078	bug	.069	ad	.073
best	.045	easy	.034	weather	.030	<digit>	.052	see	.065	pay	.035
fun	.036	accurate	.032	also	.023	come	.032	write	.034	get	.031
like	.034	detail	.019	get	.021	location	.031	map	.029	never	.030
favorite	.028	keep	.018	map	.018	real	.026	detail	.017	option	.024

(b) Example of the output of BST, with top-five words listed for each topic.

Fig. 5. Illustration of the learned sentiment-topic-word matrix Φ from BST.

δ determines how far the current JS divergence differs from the expected divergence value as compared to the typical difference (i.e., the standard deviation). We set $\delta = 1.25$ for accepting 10 percent of the total topics as anomaly topics following our previous work [11].

3.3.2 Automatic Topic Interpretation

By directly observing the top few words per topic as shown in Table 1, developers may find it difficult to capture the concrete meaning of each topic. In this section, we aim at automatically interpreting each topic. We choose phrases and sentences for the interpretation, since the meanings of single words may be ambiguous and entire reviews with more than one sentence can express totally different aspects. The phrases are prioritized from the candidates extracted during the preprocessing step in Section 3.1.1. To solve the third limitation described in Section 1, i.e., ineffectiveness of the topic labeling approach in [11], we combine word embeddings with topic distributions as the semantic representations of words. We denote the proposed New Topic Labeling approach as NTL. The details are described as follows.

(1) Interpreting Topics with Phrases.

The similarity *Score* between each phrase candidate a and topic ϕ_z^t is calculated in two levels: *topic level* and *embedding level*.

Topic Level. The topic distributions over words obtained from AOBST indicate the topical relevance of each word in the vocabulary to the topic. If one phrase candidate and topical words are closer to each other in the topic space, the candidate is more representative of that topic. We employ the method in [52] to measure the topical similarity between the phrase candidate a and target topic ϕ_z^t, defined as

$$Sim_{topic}(a, \phi_z^t) = -D_{KL}(a||\phi_z^t), \tag{6}$$

where D_{KL} denotes the Kullback-Leibler (KL) divergence [50] which is utilized to measure the distance between two probabilistic vectors.

Embedding Level. In the embedding space, if the phrase candidates and topical words are closer to each other, the candidates are more semantically representative. For this, we propose a semantic match score based on the attention mechanism [53]

$$Sim_{embed}(a, \phi_z^t) = \sum_w \frac{\exp(e_{a,w})}{\sum_w \exp(e_{a,w})} \phi_{z,w}^t, \tag{7}$$

where e_{a,w_i} and $\exp(\cdot)$ indicates the cosine similarity score between two embeddings and its exponential format. The fractional term represents the similarity match score between the phrase candidate and topical words in the embedding space. A phrase candidate with a higher match score with the top topical words will be ranked higher.

(2) Interpreting Topics With Sentences. For a sentence candidate s, its topic-level and embedding-level similarity scores are computed as below.

Topic Level. A sentence candidate is more representative of one topic if it comprises more words presenting higher topic relevancy to that topic. The similarity between a sentence candidate s and topic ϕ_z^t is computed as

$$\begin{aligned} Sim_{topic}(s, \phi_z^t) &= -D_{KL}(s||\phi_z^t) \\ &\approx \sum_w -D_{KL}(w||\phi_z^t)p(w|s), \end{aligned} \tag{8}$$

where $p(w|s)$ denotes the term frequency of w in the sentence s.

Embedding Level. Similarly, we calculate the embedding-level similarity of one sentence to the topic based on its constituent words, defined as

$$Sim_{embed}(s, \phi_z^t) = \sum_w Sim_{embed}(w, \phi_z^t). \tag{9}$$

The overall similarity score of each candidate l (indicating a phrase a or sentence s) is determined based on the combination of both topic-level and embedding-level scores

$$Score(l, \phi_z^t) = Sim(l, \phi_z^t) - \frac{\mu}{K-1} \sum_{j \neq z} Sim(l, \phi_j^t), \tag{10}$$

and

$$Sim(l, \phi_z^t) = m * Sim_{topic}(l, \phi_z^t) + (1-m) * Sim_{embed}(l, \phi_z^t), \tag{11}$$

where $m \in (0, 1)$ is a real-valued weight for balancing the two levels of similarity scores, l can be a phrase candidate a or sentence candidate s, and μ is a penalty factor to adjust the similarities to other topics.

3.4 Emerging App Issue Report Visualization

For facilitating developers to efficiently understand the identified emerging app issues, we visualize the evolution of app issues along with versions based on *issue river* [11]. Fig. 6 (Left) shows an example for Swiftkey for Android. The whole river represents all the app issues, and different branches indicate different topics. The *width* of each branch k presents the user-concern degree of the issue for the corresponding version t, defined as

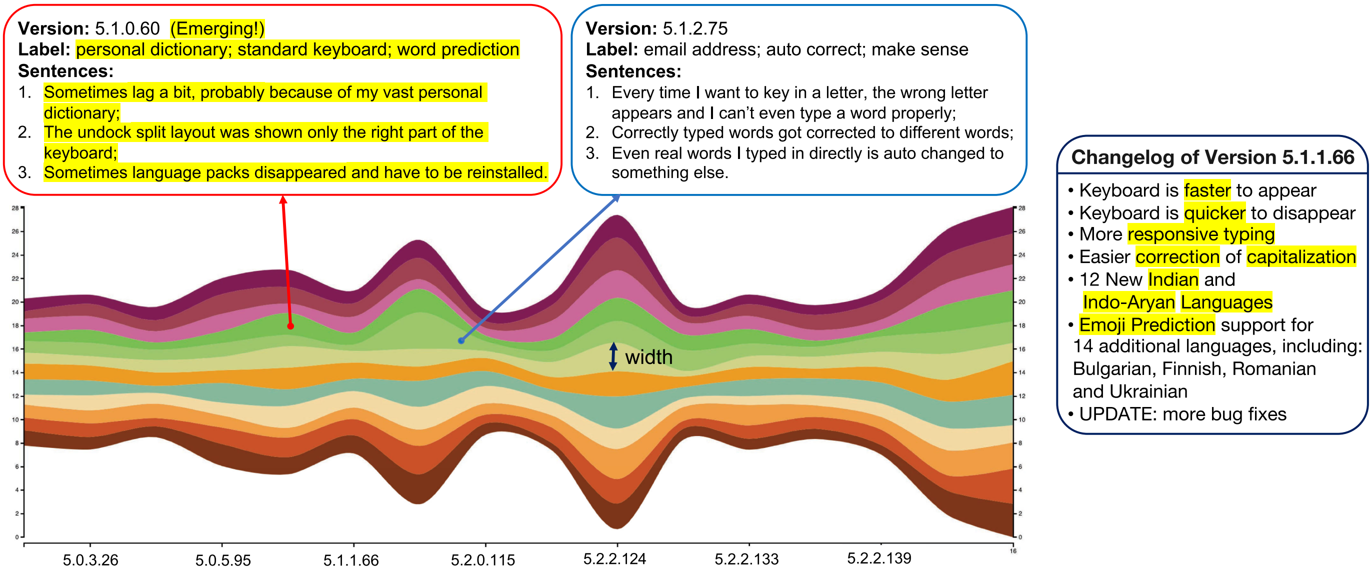
$$width_k^t = \sum_a \log Count(a) * \phi_k^t, \tag{12}$$

where $Count(a)$ means the count of the phrase label a in the review collection of the tth version. So, wider branches are of more concern to users. By moving the mouse over one topic (i.e., branch), developers can track detailed issues along with versions, where the emerging ones are highlighted with yellow background, as shown on the top left box in Fig. 6. We also show an example of changelog on the right of Fig. 6. We can discover that the identified emerging issue *lag during word prediction* was fixed by the next immediate version, described as *“More responsive typing”* (the third item) in the corresponding changelog.

4 Experimental Setup

4.1 Dataset

We employ the same dataset by Gao *et al.* [11] for evaluation. Details of the dataset are shown in Table 3. The dataset includes 164,026 reviews (from August 2016 to April 2017)



Version: 5.1.0.60 (Emerging!)
Label: personal dictionary; standard keyboard; word prediction
Sentences:

1. Sometimes lag a bit, probably because of my vast personal dictionary;
2. The undock split layout was shown only the right part of the keyboard;
3. Sometimes language packs disappeared and have to be reinstalled.

Version: 5.1.2.75
Label: email address; auto correct; make sense
Sentences:

1. Every time I want to key in a letter, the wrong letter appears and I can't even type a word properly;
2. Correctly typed words got corrected to different words;
3. Even real words I typed in directly is auto changed to something else.

Changelog of Version 5.1.1.66

- Keyboard is faster to appear
- Keyboard is quicker to disappear
- More responsive typing
- Easier correction of capitalization
- 12 New Indian and Indo-Aryan Languages
- Emoji Prediction support for 14 additional languages, including: Bulgarian, Finnish, Romanian and Ukrainian
- UPDATE: more bug fixes

Fig. 6. Issue river of the SwiftKey app (left) and changelog of the version 5.1.1.66 (right). For the issue river, the whole topic flow is visualized as a river, and the number of topics K is set as 12, corresponding to 12 branches of the river. The horizontal axis presents consecutive app versions, and the branches with larger widths illustrate that the corresponding issues relatively concern users more for those versions. For the changelog, we highlight keywords in yellow background.

for six apps, from 89 versions in total. The apps are distributed in different categories, with two of them from the App Store and the others from Google Play.

The word embeddings employed for topic labeling are trained on 4,663,316 app reviews released by Man *et al.* [42] by using the Gensim tool [54]. The dimension of the word embeddings is set as 200, with other parameters set following our prior study [11]. We openly release the trained word embeddings in our replication package.

4.2 Evaluation Methods

We use the keywords in changelogs as ground truth (one example shown on the right of Fig. 6) and employ the three metrics as used by Gao *et al.* [11] for verifying the effectiveness of MERIT. We define an app issue to be successfully identified by MERIT and its baselines if its corresponding description in the changelog of the immediate version present a high similarity[7] with the identified issues (either at phrase level or sentence level). The evaluation method is based on the hypothesis that emerging issues need to be quickly solved in an updated version and thus are typically reflected in the changelog of the immediate version. Here we use three performance metrics as used by Gao *et al.* [11] for verifying the effectiveness of MERIT. 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 $Recall_L$. The last metric F_{hybrid} is for measuring 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.

$$Precision_E = \frac{I(E \cap G)}{I(E)}, \quad Recall_L = \frac{I(L \cap G)}{I(G)},$$
$$F_{hybrid} = 2 \times \frac{Precision_E \times Recall_L}{Precision_E + Recall_L}. \tag{13}$$

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 $\cdot$. We experimentally set the parameters as $\omega = 3$, $K = 13$, $PMI = 5$, $\mu = 1.0$, and $m = 0.5$. We also initialize α and β with 0.1 and 0.01, respectively.

4.3 Baseline Approaches

We compare the effectiveness of our proposed framework with a popular emerging event detection approaches on social networks, OLDA [56] and the state-of-the-art emerging app issue identification approach, IDEA [11].

On-line Latent Dirichlet Allocation (OLDA) is an online version of Latent Dirichlet Allocation (LDA) [13] that manually captures the topic patterns and identifies topics of text streams and their changes over time. It generates an evolutionary word distribution matrix for each topic. In this way,

TABLE 3
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

[7] The similarity is measured as the cosine score between the two vector representations in the word embedding space, and it is high if the cosine score is larger than 0.6 [55].

TABLE 4
Comparison Results With Baseline Approaches

App Name (#avg. reviews)	Method	Phrase			Sentence		
		$Precision_E$	$Recall_L$	F_{hybrid}	$Precision_E$	$Recall_L$	F_{hybrid}
YouTube (1,143)	OLDA	0.441	0.462	0.451	0.578	0.664	0.597
	IDEA	0.592	0.472	0.523	0.628	0.666	0.636
	MERIT	**0.625**	**0.551**	**0.586**	**0.667**	**0.760**	**0.710**
Clean Master (6,332)	OLDA	0.300	0.269	0.160	0.200	0.421	0.129
	IDEA	**0.667**	0.318	0.431	0.667	0.434	0.526
	MERIT	**0.667**	**0.468**	**0.550**	**0.833**	**0.848**	**0.841**
Viber (2,141)	OLDA	0.157	0.305	0.166	0.313	0.550	0.375
	IDEA	0.625	0.340	0.440	0.625	0.651	0.638
	MERIT	**0.667**	**0.706**	**0.686**	**0.833**	**0.809**	**0.821**
Ebay (3,943)	OLDA	0.167	0.238	0.196	0.500	0.488	0.494
	IDEA	0.229	0.251	0.227	0.646	0.527	0.580
	MERIT	**0.889**	**0.508**	**0.646**	**1.000**	**0.749**	**0.857**
SwiftKey (1,313)	OLDA	0.100	0.567	0.148	0.367	0.617	0.458
	IDEA	0.517	**0.653**	0.523	0.583	0.700	0.587
	MERIT	**0.800**	0.633	**0.707**	**0.800**	**0.867**	**0.832**
NOAA Radar (523)	OLDA	0.468	0.528	0.473	0.482	0.622	0.534
	IDEA	0.571	0.497	0.531	0.476	0.639	0.546
	MERIT	**0.750**	**0.654**	**0.699**	**0.750**	**0.840**	**0.793**

*The value under each app name indicates the average number of reviews across the versions, and **bold** figures highlight better results.*

it incrementally builds an up-to-date model when new documents appear. The emerging topics in the current app version are determined by comparison with the topic distributions in the previous version.

IDEA is a state-of-the-art emerging app issue identification approach proposed recently. It improves OLDA by considering the topic distributions in previous versions within a version window during emerging topic detection. The improved method is named as Adaptive OLDA (AOLDA). It also includes an automatic topic interpretation method for labeling each topic with the most representative phrases and sentences.

5 Experimental Results

In this section, we describes results of the evaluation of MERIT through experiments and compare it with the state-of-the-art tool, IDEA [11], and another competing approach, OLDA [56], to assess its capability in identifying emerging app issues for developers. Our experiments are aimed to answer the following research questions:

RQ1: What is the performance of MERIT in detecting emerging app issues?

RQ2: What is the impact of different extensions on the performance of MERIT? The extensions include adopting BTM for topic modeling instead of LDA, considering sentiment for each topic, and the new topic labeling approach.

5.1 RQ1: What is the Performance of MERIT in Detecting Emerging App Issues?

This research question relates to the capability of MERIT in identifying accurate and complete emerging app issues in comparison with IDEA [11] and OLDA [56]. Having too many false positives would end up being counterproductive, whereas having too many false negatives would mean that the proposed framework is not able to alert emerging issues in many cases where those are important. Table 4 displays the comparison results.

As seen in Table 4, the proposed MERIT approach outperforms the baseline approaches on all the metrics. We discuss the performance of MERIT from two aspects as below.

Result 1: Interpreting Topics With Phrases versus Sentences. As mentioned in Section 3.3.2, there are two ways to represent an app issue: by phrases and by sentences. For example, the "Label" and "Sentences" in the top boxes of Fig. 6 are the phrase and sentence representations respectively. As shown in Table 4, considering all the three methods, issues in sentences present better performance than those in phrases with 9.5, 29.1, and 15.6 percent increase in $Precision_E$, $Recall_L$, and F_{hybrid} on average respectively. This result may be attributed to the fact that sentences can convey more details than phrases and thereby cover more key terms mentioned in changelogs, which is also in line with findings of our previous study [11]. Specifically, the sentences identified by MERIT can enhance the performance of phrases by 8.1, 22.6, and 16.3 percent wrt. three metrics, respectively. We then use Wilcoxon signed-rank test [57] for statistical significance test, and Cliff's Delta (or d) to measure the effect size [58]. The significance test result ($p - value < 0.05$) and large effect size ($d = 2.76$) on the difference in the mean of the F_{hybrid} scores of phrase-level issues and sentence-level issues confirm the better performance of sentence representations over phrase representations.

Result 2: MERIT versus Baselines. Comparing MERIT with baseline approaches, we find that MERIT can outperform both baselines in all the three metrics with respect to

TABLE 5
Ablation Experiments With Different Extensions Turned On/Off

App Name (#avg. reviews)	Method	Phrase			Sentence		
		$Precision_E$	$Recall_L$	F_{hybrid}	$Precision_E$	$Recall_L$	F_{hybrid}
YouTube (1,143)	IDEA	0.592	0.472	0.523	0.628	0.666	0.636
	+BTM	0.525	0.576	0.416	0.592	0.843	0.696
	+Sentiment	0.483	**0.592**	0.532	0.550	**0.868**	0.673
	+NTL	0.544	0.477	0.523	0.582	0.773	0.612
	MERIT	**0.625**	0.551	**0.586**	**0.667**	0.760	**0.710**
Clean Master (6,332)	IDEA	0.667	0.318	0.431	0.667	0.434	0.526
	+BTM	0.444	0.420	0.432	0.778	0.761	0.769
	+Sentiment	0.417	0.335	0.371	0.625	0.748	0.681
	+NTL	**0.833**	0.299	0.440	0.747	0.500	0.599
	MERIT	0.667	**0.468**	**0.550**	**0.833**	**0.848**	**0.841**
Viber (2,141)	IDEA	0.625	0.340	0.440	0.625	0.651	0.638
	+BTM	0.625	0.692	0.657	0.750	**0.809**	0.778
	+Sentiment	**0.778**	0.395	0.524	0.778	0.566	0.655
	+NTL	0.667	0.366	0.473	0.664	0.778	0.716
	MERIT	0.667	**0.706**	**0.686**	**0.833**	**0.809**	**0.821**
Ebay (3,943)	IDEA	0.229	0.251	0.227	0.646	0.527	0.580
	+BTM	0.667	0.402	0.502	0.833	0.640	0.780
	+Sentiment	0.361	0.310	0.333	0.833	0.516	0.637
	+NTL	0.542	0.418	0.472	0.676	0.667	0.671
	MERIT	**0.889**	**0.508**	**0.646**	**1.000**	**0.749**	**0.857**
SwiftKey (1,313)	IDEA	0.517	0.653	0.523	0.583	0.700	0.587
	+BTM	0.500	**0.767**	0.605	0.750	**0.900**	0.818
	+Sentiment	0.500	0.667	0.571	0.750	0.867	0.704
	+NTL	0.500	0.733	0.595	0.500	0.767	0.605
	MERIT	**0.800**	0.633	**0.707**	**0.800**	0.867	**0.832**
NOAA Radar (523)	IDEA	0.571	0.497	0.531	0.476	0.639	0.546
	+BTM	0.611	0.575	0.592	0.611	0.773	0.683
	+Sentiment	**0.796**	0.612	0.692	0.667	0.829	0.739
	+NTL	0.667	0.566	0.612	0.619	0.710	0.662
	MERIT	0.750	**0.654**	**0.699**	**0.750**	**0.840**	**0.793**

The value under each app name indicates the average number of reviews across the versions, and ***bold*** *figures highlight better results. The methods "+BTM", "+Sentiment", "+NTL" respectively represent the extensions upon IDEA that we propose in this work, including using BTM for topic modeling instead of LDA, combining topics with sentiment, and enhancing the topic labeling step with word embeddings.*

sentence-level issues. For phrase-level issues, although MERIT shows a slightly lower $Recall_L$ than IDEA for the SwiftKey app, it exhibits better performance in both $Precision_E$ and F_{hybrid}. On average, MERIT can achieve precision, recall, and f-score of 81.4, 81.2, and 80.9 percent respectively, and outperform OLDA by 37.8 percent and IDEA by 22.3 percent for F_{hybrid}, which indicates that MERIT can better balance the precision and recall in emerging issue detection. Besides, the significant statistical test results ($p-value < 0.01$) and large effect sizes ($d > 2$) on the F_{hybrid} scores for both phrase and sentence -level issues of MERIT and IDEA/OLDA confirm the superiority of MERIT over IDEA/OLDA.

5.2 RQ2: What is the Impact of Different Extensions on the Performance of MERIT?

MERIT extends IDEA by (1) adopting BTM for topic modeling instead of LDA, (2) jointly modeling sentiment and topics, and (3) employ the proposed word-embedding-based topic labeling (NTL) approach. We perform ablation experiments by considering each of the 3 extensions one-at-a-time, which we refer to as "+BTM", "+Sentiment", and "+NTL" respectively. Table 5 shows the results of comparing each of these 3 approaches with the baselines.

Unsurprisingly, the combination of all extensions gives the greatest improvements in terms of F_{hybrid}, and all the components are beneficial on their own. Similar to the answer to RQ1, we also observe that sentence-level issues generally present better performance than the phrase-level issues.

Specifically, with respect to each extension considered independently using BTM instead of LDA for topic modeling can enhance the average performance by 8.8 and 16.9 percent for the phrase-level and sentence-level F_{hybrid} scores respectively. In terms of $Precision_E$ and $Recall_L$, with BTM involved, the performance increases by 11.5 and 19.0 percent, respectively. When jointly modeling topics with the sentiment, the F_{hybrid} scores are increased by 5.8 and 19.4 percent in terms of phrase and sentence representations, respectively. On average, both precision and recall show an increasing trend, +9.6 and +13.4 percent, respectively. The results indicate that by the considerations of sentiments, overall results including both precision and recall have been improved. But for some apps, such as YouTube and Clean Master, although the recall is increased (+20.2 and +31.4 percent respectively), the precision is slightly dropped (-7.8 and -4.2 percent respectively). This may be because with the sentiment involved, the topics predicted as negative sentiment tend to be identified as emerging issues,

TABLE 6
Comparison on the Topics Generated by LDA and BTM for the YouTube iOS App

Method	Topic 1	Topic 2	Topic 3
	Battery drainage	*Play button*	*Video Recommendation*
LDA	<digit>	video	video
	quality	problem	watch
	battery	also	see
	io	make	channel
	iphone	button	recommend
	use	there_be	find
	6s	go	anymore
	video	great_app	i_want
BTM	<digit>	video	video
	battery	play	home
	video	button	watch
	drain	screen	screen
	use	auto	page
	cause	arrow	thumbnail
	6s	watch	see
	watch	pause	recommend

*The topics are related to "*battery drainage*", "*play button*", and "*video recommendation*" respectively, each with top eight terms presented. Fonts with wavy underlines highlight the terms that are not semantically related to the issue topic.*

TABLE 7
Examples of Ranked Phrases With the Original Topic Labeling Approach (denoted as MERIT+TL) and the New Word-Embedding-Based Approach (denoted as MERIT+NTL)

	Topic 1	Topic 2	Topic 3
	Third party utility for split view	*Subscription box*	*Comment section*
Top Terms	picture	watch	comment
	support	video	video
	feature	say	change
	add	subscription	better
	video	show	description
	slide_over	even	comment_section
	ipad	mark	back
	split_screen	see	move
MERIT +TL	playback error galore	playback error	comment section
	split screen	sub box	character limit
	browser base	low quality	push notification
MERIT +NTL	split view	subscription fee	channel name
	split screen	sub box	comment section
	third party	subscription box	main page
Ground Truth	Added slide over and **split view** support.	Easier access to your full **subscription list**.	Click on timestamp links in **comments** advances the **video** to correct position.

Fonts with wavy underlines highlight the phrase labels that are not semantically related to the corresponding topic. We also present the ground truth corresponding to each topic in boldface.

which is helpful for enhancing the recall. But the negative topics might not always be emerging, such as some constantly recurring topics (e.g., the "*screen*" topic for YouTube and the "*battery*" topic for Clean Master), so the precision is slightly weakened. Besides, involving word embeddings during topic interpretation gives us a 7.3 percent increase for phrase-level issues and a 5.9 percent increase for sentence-level issues with respect to F_{hybrid}. We also observe that although the YouTube app (with 1,143 reviews per version) shows a slightly decrease (-2.4 percent) on the F_{hybrid} score, all the other apps, especially NOAA Radar which has only 523 reviews per version, enjoy an increase. Thus, the experiment results demonstrate that the novel topic labeling method can work well even for apps with few reviews. Moreover, the gain from different extensions is not fully cumulative since the information delivered by these components overlaps. For instance, both the topic modeling and topic labeling steps help capture the semantics of the words in app reviews to generate accurate emerging issues.

6 Discussions

In this section, we discuss the advantages of MERIT, its limitations, time cost, impacts of different parameters, and the threats.

6.1 Why Does Our Model Work?

We have identified three advantages of MERIT that may explain its effectiveness in detecting emerging app issues.

Observation 1: MERIT Can Better Model the Topics of Short Texts. In this work, we propose to use the biterm topic model (BTM) for short text mining instead of LDA. Since LDA learns review-level word co-occurrence patterns to reveal topics, it suffers from the severe data sparsity in short review texts. Instead, BTM learns the topics from word co-occurrence patterns directly and thus alleviate the data sparsity problem. Table 6 shows the top eight terms of three example topics obtained from LDA and BTM. We discover that BTM can generate more semantically-coherent terms for each topic. For example, the terms "*also*", "*there_be*", and "*great_app*" terms are not related to Topic 2 "*play button*". The semantic inconsistency of the top terms in one emerging issue would confuse developers or influence the performance of subsequent automatic topic interpretation step. By using BTM instead of LDA for review modeling, the semantics of top terms belonging to one topic can be more coherent.

Observation 2: MERIT Can Focus on Negative Topics. As shown in Fig. 5, the topics extracted from reviews are usually mingled with various polarities. Even for the same topics, users may express totally different opinions. Motivated by the intuition that developers are more concerned about the negative app aspects [28], [29], MERIT focuses on the topics inferred as negative instead of incorporating topics in all sentiment polarities. Thus, MERIT can expose the topics likely corresponding to app issues.

Observation 3: MERIT Can Interpret Topics With More Representative and Coherent Labels. For accurate topic labeling, we combine word embeddings to prioritize semantically-representative phrase/sentence candidates. Table 7 shows the ranked phrases with and without word embeddings involved, respectively. We can discover that the proposed word-embedding-enhanced topic labeling (NTL) approach can better interpret the topic meanings in terms of coherence and semantic accuracy. For example, the original topic labeling approach selects "*playback error galore*" and "*playback error*" as the most representative phrases of Topic 1 and Topic 2 respectively, which are intuitively different from the general meaning of the topics (i.e., split view and subscription box respectively) and can cause confusion to developers in understanding the detected emerging issues. Instead, the top three phrases of these two topics obtained by MERIT are semantically coherent, all about split view or subscription.

6.2 Why Does Our Model Fail?

We have also summarized two main scenarios that may lead to inaccurate emerging issue prediction.

Observation 1: MERIT May Miss the Emerging App Issues Only Mentioned in Few User Reviews. For the emerging issues only expressed in few (e.g., three or four) reviews, they are difficult to be exposed through topic modeling approaches [26], [27]. For example, one major modification claimed by the version 5.9.3 of the Clean Master app, i.e., "*Added Cloud Recycle Bin - Recover misdeleted photos from the cloud up to 30 days after deleting*", MERIT misses capturing any emerging issue related to "*recycle bin*". After inspecting the collected corpus, we find that only three reviews received in the previous version are describing the recycle bin, which possibly leading to the omission. We discover similar failure scenarios for other apps. For instance, for the NOAA Radar iOS app, it made a major change about its widget in its version 2.0 that fixes an issue, i.e., "*Tap on Today tab, scroll to the bottom and tap Edit*" as written in the changelog. MERIT fails to identify the issue since it is only discussed by three pieces of reviews in the corpus of the previous version.

Observation 2: Official Changelogs May Not Cover All the App Issues of the Previous Version That are Fixed in the Current Version. Although app markets such as the App Store encourages app developers to write what is actually happening to the apps in the changelog [59], app developers tend to write sketchy and vague bullet points for the changes, such as "*Bug fixes*" and "*We're always trying to improve your experience*". Although we already filter such changelogs out during validation, the release notes may not cover all the major changes made to current versions and could lead to false negatives.

First, the app issues may not be fixed instantly in the next updated version. For example, MERIT detects an emerging issue associated with video orientation modes, described as "*portrait mode*" and "*full screen mode*", for version 11.39 of the YouTube iOS app. One user complained that "*Sometimes when I'm on full screen mode, I click the minimise screen button and it doesn't work. I try to flip my phone and it doesn't minimise.*", and gave a two-star rating. The issue also aroused heated discussion on the YouTube online forum [60]. We discover that the issue was fixed in a later version 12.05 instead of the immediate next version, as indicated in the changelog "*Fixed delay when pressing the full screen and minimize buttons in the player*". Such postponement of issue fixing is reasonable since not all bugs in apps would be addressed right away [61], [62]. Second, changelogs may describe modifications in general terms. For example, MERIT alerts an emerging issue related to "*Samsung keyboard*" and "*force close*" for version 5.0.4.93 of the SwiftKey app. Although the issue is greatly relevant to the corresponding changelog "*Fixed issues causing repeated crashes on some devices when loading the keyboard*", the evaluation of MERIT regards "*device*" and "*Samsung*" as mismatched. Finally, changelogs may not cover all the major app changes. For instance, we find that the voice dictation issue identified by MERIT for version 5.1.0.60 of the SwiftKey app is a change (i.e., forcing install additional app for voice dictation) made in the app version but not described in the changelog. For example, one user commented that "*After update, force install additional app 'google voice search' for voice dictation, which was not previously required. .... I wish I knew so I would not update the app.*".

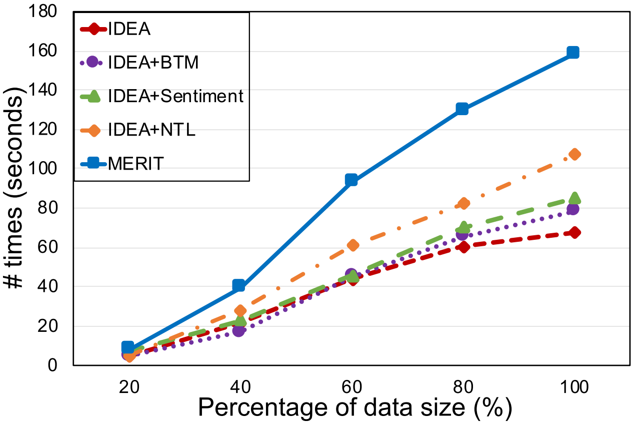


Fig. 7. Efficiency of MERIT and the comparison models on different data sizes of 5,000 reviews.

6.3 Efficiency of MERIT

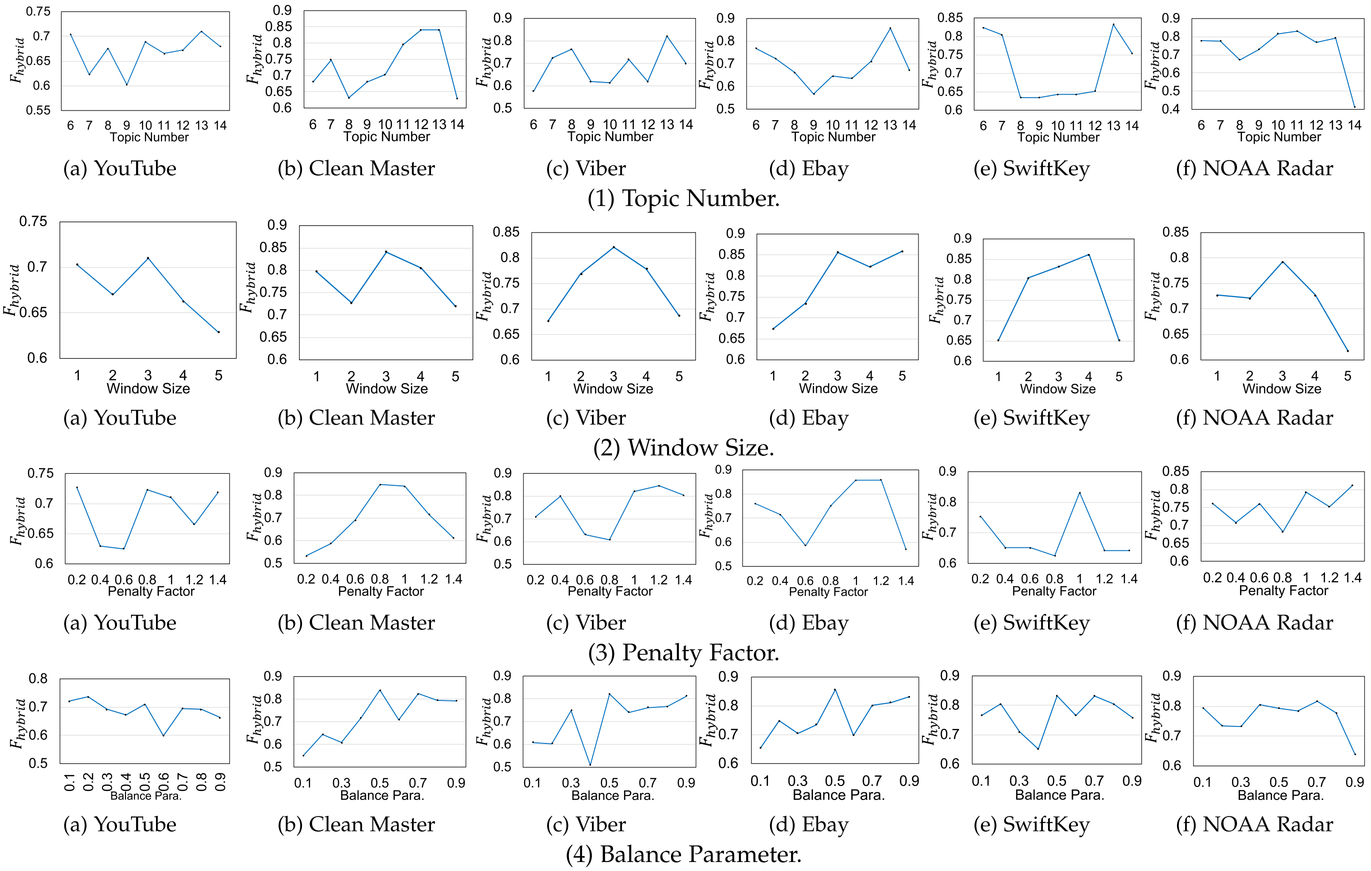
We evaluate whether MERIT can output emerging app issues within reasonable time, by comparing the execution time of MERIT on the subject apps with IDEA, and also with different extensions of IDEA (i.e., "+BTM", "+Sentiment", and "+NTL"). In this experiment, we randomly select subsets of the 5,000 reviews from the YouTube dataset (of different sizes) and run all the models. We run our experiments on a PC with Intel(R) Xeon E5-2620v2 CPU (2.10 GHz, 6 cores) and 16GB RAM. Fig. 7 displays the comparison results of time consumed on different dataset sizes. As can be seen from Fig. 7, all the models spend more time as the amount of data increases. We also find that the "IDEA+BTM", "IDEA+Sentiment" and "IDEA+NTL" models cost 16.0, 25.8, and 58.4 percent more time than the IDEA model when handling 5,000 reviews, respectively. Undoubtedly, MERIT incurs the highest time cost among all the models due to its higher complexity, which can cost 1.3 times more time than IDEA when processing the 5,000 reviews. In spite of the higher time cost, MERIT can deal with 1,000 reviews within eight seconds and 5,000 reviews within three minutes, which we believe to be still acceptable. Therefore, our experiments demonstrate that MERIT can detect emerging app issues more accurately while preserving reasonable time costs.

6.4 Parameter Analysis

We also quantitatively compare the performance of MERIT in different parameter settings. We analyze three parameters, that is, the number of topics K, the window size ω, the penalty factor μ (in Equ. (10)), and balance parameter m (in Equ. (11)). We vary the values of these four parameters and evaluate their impact on the performance of MERIT. The results are shown in Fig. 8.

6.4.1 The Number of Topics

As can be seen in Fig. 8 (1), the F_{hybrid} score curves created by varying topic numbers are not consistent among the



(a) YouTube (b) Clean Master (c) Viber (d) Ebay (e) SwiftKey (f) NOAA Radar

(1) Topic Number.

(a) YouTube (b) Clean Master (c) Viber (d) Ebay (e) SwiftKey (f) NOAA Radar

(2) Window Size.

(a) YouTube (b) Clean Master (c) Viber (d) Ebay (e) SwiftKey (f) NOAA Radar

(3) Penalty Factor.

(a) YouTube (b) Clean Master (c) Viber (d) Ebay (e) SwiftKey (f) NOAA Radar

(4) Balance Parameter.

Fig. 8. Impact of different parameters on the F_{hybrid}.

apps. For some apps such as YouTube, Clean Master, and Ebay apps, larger topic numbers can achieve better performance. However, for the SwiftKey and NOAA Radar apps, smaller topic numbers are preferred. This may be because the YouTube, Clean Master, and Ebay apps have relatively larger review volumes than the SwiftKey and NOAA Rader apps in the collected dataset, so more topics may exist. To better balance the precision and recall, we set the topic number as 13 during experiments.

6.4.2 Window Size

According to Fig. 8 (2), the performance varies along with different window sizes. On the whole, the trends are analogous to an inverted "U" shape, such as the Viber, SwiftKey, and NOAA Radar apps. Such a phenomenon is reasonable since the topic distributions of the current version strongly rely on those of the previous versions within the window size. Smaller window sizes render the topic distributions of current versions more sensitive and unstable. Although larger window sizes can weaken the sensitiveness, they may also lack the sensitivity to emerging issues. We set window size ω as three since the setting can bring relatively better performance on the studied apps (indicated in Fig. 8).

6.4.3 Penalty Factor

As shown in Fig. 8 (3), an approximately inverted "U" shape can also be observed in most apps, such as the Clean Master, Ebay, and SwiftKey apps. Smaller penalty values may lead the ranked sentences to not distinguish the two topics well, and thereby prioritize similar sentence labels for the topics. In this way, the issues in sentences would not be able to cover all the emerging issues. However, larger penalty values may cause the label prioritization to put more weights on the distinguishability instead of the semantic similarity between the labels and topics, so the sentence labels may not well represent the meanings of current topics. We choose the penalty factor $\mu = 1.0$ since the value presents almost the best performance on all the studied apps.

6.4.4 Balance Parameter

The results under different balance parameters are illustrated in Fig. 8 (4). We can observe that generally higher balance parameters can lead to better performance for the studied apps, such as the Clean Master, Viber, and Ebay apps. However, such patterns are not applicable to the other apps. Since $m = 0.5$ can achieve a good performance on our datasets, we set the balance parameter as 0.5 in our experiments.

6.5 Manual Inspection

In this paper, to automate the parameter tuning and result verification processes, we adopt a common semantic measurement metric [63], i.e., cosine similarity. However, the automatic evaluation results might not exactly reflect practical performance. Manual evaluation is therefore needed to comprehensively evaluate the consistency between detected emerging issues and the ground truth. Since validating all

TABLE 8
The Results of Manual Evaluation on the YouTube Dataset

Method	$Precision_E$	$Recall_L$	F_{hybrid}
IDEA	0.576 (-0.052)	0.673 (+0.007)	0.621 (-0.015)
MERIT	0.618 (-0.049)	0.801 (+0.041)	0.698 (-0.012)

The values inside brackets indicate the fluctuation ranges comparing to the scores computed by cosine similarity. Here, the comparison is based on the sentence-level issue representations.

the emerging issues would consume huge human effort, we chose the YouTube app, which contains the most versions among the studied apps (accounting for 37.1 percent of all the app versions and containing 70 official issues in total), for manual verification. The first two authors independently examined the sentence-level issues detected by MERIT with the descriptions in the changelogs of the immediate versions. The initial kappa agreement score between the two raters is 0.725 (substantial agreement). Finally, the two raters discussed the sentences with discrepant labels one-by-one to reach a consensus. The whole labeling process costs us around two weeks. To facilitate future research, we release our labeled dataset publicly.[8]

The results are shown in Table 8. As can be seen, comparing manual inspection results with the scores computed by cosine similarity, the differences range from -5.2 to +4.1 percent. Also, in terms of the F_{hybrid} scores, there is only a small disparity (around -1 percent). Thus, using cosine similarity could be regarded as a reliable way to alleviate the labor burden and time consumed in both parameter tuning and verification. The effectiveness of MERIT is consequently confirmed.

6.6 Survey on Industry Practitioners

To further demonstrate the effectiveness and practicability of our proposed approach, we conducted a user study with 44 full-time employees from three large IT companies in China including Tencent, Alibaba, and Netease. We contacted the cooperative partners from the engineering department in each of the companies and asked their help to distribute our online questionnaire. Each participant would receive a small compensation as reward. In the end, we obtained answers from 28, 11, and 5 employees from Tencent, Alibaba, and Netease, respectively. We provide a summary of the survey results as below.[9]

The 44 participants include 19 developers (43.2 percent), 9 data analysts (20.5 percent), 6 test engineer (13.6 percent), 2 product managers (4.5 percent), and 10 from other positions (22.7 percent).[10] Around 80 percent of the participants have more than one year of software engineering experience. The online questionnaire consists of five questions: two questions on participants' background and three questions for understanding their attitude towards the practicability of MERIT. During the survey, we validated the practicability of MERIT in terms of three aspects: acceptability of the provided emerging issues, preference of higher accuracy but with more time consumption, and willingness of adopting such a tool into the development pipeline.

6.6.1 Acceptability of the Provided Emerging Issues

Prior studies [64], [65] define "acceptability" of an application as a measure of users' overall experience with an application, including perceived ease of use, usefulness of its functionalities, and quality of user experience. In our study, "acceptability" mainly corresponds to the usefulness of the results provided by MERIT, and thus is narrower than the aforementioned definition of the term. Specifically, we survey the participants about their opinions on the presented descriptions of the emerging issues, by providing one official changelog example of YouTube and the corresponding detected emerging issues. Following the definition of Likert scale for attitude measurement [66], the aspect is rated on a 1-5 Likert scale (5 for strong agreement, 4 for agreement, 3 for undecided, 2 for disagreement, and 1 for strong disagreement). The survey results indicate that all the interviewees (100 percent) agree that the provided issues are acceptable, among which 14 (31.8 percent) of them are strongly in favor of the usefulness of the issue descriptions.

6.6.2 Preference of Higher Accuracy But With More Time Cost

To investigate on developer's opinions regarding the performance of MERIT, especially whether the higher accuracy is worth the additional time cost, we present the performance of two model examples and ask participants to choose the preferred one. The two models are: Model A can process 200 reviews in one second and obtains 60 percent accuracy; while model B achieves 80 percent accuracy but can only process 125 reviews per second. According to the survey, 28 (63.6 percent) of the interviewees prefer model B which achieves a higher accuracy but with a lower processing speed, and six interviewees (13.6 percent) consider both models to be acceptable. Only 10 survey respondents (22.7 percent) chose model A over model B. The results indicate that industry practitioners possibly prefer MERIT than the baselines regarding the performance.

6.6.3 Willingness of Applying MERIT into Industry

We collect participants' opinions of whether they are willing to employ or recommend developers to employ our tool MERIT. This aspect is also rated on a 1-5 Likert scale (5 for strong agreement, 4 for agreement, 3 for undecided, 2 for disagreement, and 1 for strong disagreement). According to the survey, around 86.4 percent of the interviewees express that they strongly agree or agree to use MERIT in their development pipelines, and 36.4 percent convey a strong agreement. The results demonstrate the potential benefit of MERIT to developers.

6.7 Comparison With TopicSketch

In this section, we compare MERIT with TopicSketch [67], one of the latest emerging event detection methods on Twitter data stream [68], to further demonstrate the effectiveness of the proposed approach. Specifically, the "sketch" topic

[8] https://github.com/armor-ai/MERIT/tree/master/dataset

[9] The online questionnaire and collected feedback can be found at https://wj.qq.com/s2/5848368/94afand https://github.com/armor-ai/MERIT/tree/master/dataset, respectively.

[10] 2/44 participants are from two different positions, respectively.

TABLE 9
Comparison Results With TopicSketch

App Name (#avg. reviews)	Method	$Precision_E$ (Phrase)
YouTube (1,143)	TopicSketch	0.132
	MERIT	**0.625**
Clean Master (6,332)	TopicSketch	0.101
	MERIT	**0.667**
Viber (2,141)	TopicSketch	0.157
	MERIT	**0.667**
Ebay (3,943)	TopicSketch	0.203
	MERIT	**0.889**
SwiftKey (1,313)	TopicSketch	0.091
	MERIT	**0.800**
NOAA Radar (523)	TopicSketch	0.189
	MERIT	**0.750**

The value under each app name indicates the average number of reviews across the versions, and ***bold*** *figures highlight better results.*

provides a "snapshot" of the content in current Twitter stream and updates along with timestamp. TopicSketch detects emerging events by detecting acceleration in three quantities: the whole Twitter stream, every word and every pair of words. The sketch-based topic modeling approach triggers emerging topic inference when an acceleration on these stream quantities is detected. We map consecutive app versions to sequential timestamps for fitting TopicSketch into our app review scenario.

The comparison results are illustrated in Table 9. Since TopicSketch only outputs phrase-level topic labels for emerging events and the computation of $Recall_L$ requires both emerging and non-emerging events, we only consider $Precision_E$ for phrase-level issues during comparison. As can be seen, MERIT outperforms TopicSketch on all the subject apps, showing an increase at 58.7 percent in terms of the $Precision_E$ on average. The results further indicate the effectiveness of MERIT in emerging app issue detection.

6.8 Industrial Practice

Team X in one IT Company Y aims to provide developers with abnormal events report and operation statistics of tens of apps of the company. With increasing quantities of the app reviews, it is necessary for Team X to automate the manual anomaly analysis process. We have successfully deployed MERIT to help Team X maintain six apps that receive 1,000-5,000 reviews daily. The six apps cover four categories, including social, tool, music, and communication. MERIT detects emerging app issues in real time and feeds the issues back to the developers. Due to the confidentiality rules and regulations of the company, the details of the detected issues are not allowed to be published. Nevertheless, we received the following encouraging comments from the developers:

"The tool is great and helps us reduce lots of manpower. The visualization way is impressive and intuitive."

"The model deployment is convenient. I also like the performance of the emerging app issue detection."

"I think we could establish long-term collaboration on daily monitoring of app reviews. The tool brings lots of convenience to us."

These comments indicate that MERIT can be practically useful and helpful to the developers. The positive feedbacks indicate that MERIT can indeed be appreciated by app developers in assisting their emerging app issue detection for precise and timely response to their end users.

6.9 Threat to Validity

First, our model evaluation is based on the six subject apps in [11], which may not guarantee the generalization of the findings. We pick the dataset used in [11] to allow for fair comparison. Second, app versions with few user reviews can impact the performance of MERIT. Since small datasets can be easily analyzed manually, MERIT is targeted for automatic analysis of large review datasets. Also, MERIT"s good performance on different quantities of user reviews (on average 523~6,332 reviews per version) show that MERIT would well adapt to different review sizes. Third, the 500 opinion words manually labeled during polarity word preparation procedure may not be the optimal opinion words for inferring the sentiment associated with each topic. To mitigate the threat, we randomly selected the words weighted by their frequencies, so the words with higher frequencies are more likely to be selected and labeled. Also, we ensure the sample corresponds to a statistically significant proportion of the whole opinion lexicons. In practice, app developers can choose a different opinion word set for emerging issue detection. How to select an optimal set of opinion words for better sentiment inference can be future work. Fourth, the changelogs and cosine similarity measurement that we adopt for evaluation may not accurately reflect the practical performance. We mitigate this threat by manually validating the results on a sample of reviews and demonstrating that the results reported using cosine similarity are consistent with those obtained via manual inspection.

Another threat is that during the manual inspection, we choose only the YouTube app for analysis, and the YouTube app may not be representative of all the studied apps. To mitigate this threat, we ensure that the selected app versions account for a significant proportion (37.1 percent) of all the studied versions. Moreover, in this study, we only combine the sentiment characteristic of app reviews into MERIT while other factors such as device types that may be helpful for emerging issue detection are not involved. Future studies should broaden the set of features used to characterize app reviews in our study and investigate the impact of different characteristics on the performance of emerging issue detection. In addition, there may be alternative approaches to combine topics and sentiment for emerging issue detection, e.g., implementing a two-step pipeline where extracting negative review sentences or paragraphs is the first step, and modeling topics of the negative texts is the second step. We leave implementation and evaluation of these alternatives to future work. Finally, MERIT shares the same limitation with the adopted topic modeling approach [14], i.e., the number of topics should be determined initially. This limitation is brought by the unsupervised nature of the approach. There are studies [69], [70], [71] on automatically identifying the optimal topic number, but they are not easy to be adapted to online topic modeling approaches, which is the core of our proposed framework. How to efficiently discover the optimum topic numbers for online topic models can be regarded as a challenging and interesting work for future research.

7 Related Work

We discuss two threads of studies that inspire our work: App review analysis and emerging topic detection.

7.1 App Review Analysis

Since app reviews serve as an essential channel between users and developers, and provide rich information about app usage, the number of studies on user review analysis is on the rise [72]. Recent research has leveraged Natural Language Processing and Machine Learning techniques to extract useful information from online app products to help developers realize, test, optimize, maintain, and release apps (see e.g., [73], [74], [75], [76]). The major goal of these studies is to alleviate the burden of summarizing useful knowledge from a relatively huge quantity of unstructured texts. Here, we focus on the research that exploiting app reviews to facilitate the process of app maintenance and release.

A number of studies [77], [78], [79], [80] categorize user reviews based on their sentiment (e.g., either praise or complaint) and general topics (e.g., bug report or feature request). Di Sorbo *et al.* [81] presented an approach called SURF to further classify reviews into fine-grained topics (e.g., GUI and security). Based on the categorized reviews, Gu and Kim [82] applied aspect opinion mining and sentiment analysis to find the most popular features of an app. Although they can present the rating changes of one app feature over time, the ratings are tracked based on feature words instead of topics. Moreover, their work does not establish the relation of features with star-ratings [83]. Besides, Islam and Zibran [84] and Calefato *et al.* [85] design sentiment analysis tools specific to software development.

Topic modeling is widely used in different domains, and interesting results have been inferred [86], [87]. Consequently, some researchers rely on topic modeling technique [13], [14] to analyze user reviews. Iacob and Harrison [88] and Guzman and Maalej [28] applied LDA to extract app features. Chen *et al.* [29] adopted LDA to capture the topic distribution of each user review, based on which they prioritized useful user reviews to developers. Fu *et al.* [89] analyzed the changes in the review number associated with each topic over time. Noei *et al.* [83] used LDA to determine the key topics of user reviews for different app categories. Gao *et al.* [16], [87] resorted to topic modeling methods for prioritizing app issues. None of the papers mentioned above have considered the sentiment changes of topics along with time or exploited the changes to detect emerging app issues.

Our previous work [11] is the most recent study focusing on tracking app issues along with release versions. Specifically, the IDEA proposed in [11] analyzed issue changes along with app versions using online topic modeling during which the emerging app issues are identified. Another of our recent work [90] also aimed at detecting emerging app issues but mainly during the beta testing periods. Although the IDEA model performs well on the studied apps, the proposed model still meets several limitations as discussed in Section 1.

Nayebi *et al.* investigate app updating frequency and its impact [91], [92]. They find that users prefer to install apps that were updated more recently and less frequently. Thus, frequent updates are not always considered positively in practice. Updating frequencies should also be carefully determined. Determining the sweet spot for app update frequency is an important and an interesting research topic. As this is beyond the scope of the current paper, we will leave this for future work.

7.2 Emerging Topic Detection

An event,[11] in the context of social media, is an occurrence of interest in the real world which initiates a discussion on the event-associated *topic* on social media platforms, either soon after the occurrence or, sometimes, in anticipation of it. The emerging event detection approaches can be based on term interestingness [93], incremental clustering [94], or topic modeling [95], etc. For example, Li *et al.* [96] identified emerging events from Twitter by first selecting top bursty words and then conduct word clustering, which is a term-interestingness-based approach. A comprehensive survey on emerging topic detection approaches can be found in Hasan *et al.*'s work [68]. Different from texts on social media, each review is specific to one app version, and generally shorter in length [12], [97], which renders app review mining a more challenging task [16], [73], [98], specialized to software engineering context. However, the existing studies [68] in machine learning field either do not involve automatic labeling of topics or do not consider the short length nature of input texts, so directly applying them into our app review scenario will not be optimal.

Thus far in the app review analysis literature, term interestingness [41], [90], [99] and topic modeling methods [11] have been widely used.

The term-interestingness-based methods rely on tracking the terms likely to be related to being an event, and are usually followed with clustering methods. Various approaches are proposed to determine the interestingness score[12] of each term. For example, Minh Vu *et al.* [41], [99] first grouped the keywords using clustering algorithms and then determine the emergent clusters based on the occurrence frequencies of the keywords in each cluster.

The topic-modeling-based methods associate each document with a probability distribution over various latent topics and track the topic distributions over time. For example, our previous work [11] proposed an online topic modeling approach to infer the topic distributions of user reviews along with app releases. The emerging topics are identified based on their differences with the corresponding topics in previous time slices.

[11] An *event* in social media corresponds to an *app issue* in the context of app reviews.

[12] The *interestingness score* refers to the possibility of a term to be related to an emerging event.

The term-interestingness-based methods can be regarded as a down-top model (i.e., from word to topic), while the topic-modeling-based methods are top-down (i.e., from topic to word). Since the abruptness of one topic does not indicate that all the words belonging to the topic show bursty trends, term-interestingness-based models may generate true negatives due to missing bursty words. Thus, our proposed model is based on the topic modeling approach.

8 Conclusion and Future Work

To ensure good user experience and maintain high-quality apps, identifying emerging issues in a timely and accurate manner is critical. In this paper, we propose a novel topic-modeling-based framework named MERIT for detecting emerging issues by analyzing online app reviews. MERIT improves the state-of-the-art method by better modeling of short review texts, jointly modeling topics and sentiment, and using word embeddings to better interpret topics. Extensive experiments verify the effectiveness and efficiency of our proposed framework, MERIT. In the future, we will conduct evaluations using a larger dataset and deploy the model with our industry partners.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62002084, in part by the Research Grants Council of the Hong Kong Special Administrative Region, China under Grant CUHK 14210717, and in part by the National Research Foundation, Singapore under its Industry Alignment Fund Pre-positioning (IAF-PP) Funding Initiative. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] Number of apps available in leading app stores, 2018. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[2] The mobile marketer's guide to app store ratings & reviews, [Online]. Available: https://www.apptentive.com/blog/2015/05/05/app-store-ratings-reviews-guide/

[3] S. McIlroy, N. Ali, and A. E. Hassan, "Fresh apps: An empirical study of frequently-updated mobile apps in the google play store," *Empirical Softw. Eng.*, vol. 21, no. 3, pp. 1346–1370, 2016.

[4] S. L. Lim and P. J. Bentley, "Investigating app store ranking algorithms using a simulation of mobile app ecosystems," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 2672–2679.

[5] App store optimization: 8 tips for higher rankings, [Online]. Available: https://searchenginewatch.com/sew/how-to/2214857/app-store-optimization-8-tips-for-higher-rankings.

[6] W. Martin, F. Sarro, and M. Harman, "Causal impact analysis for app releases in google play," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 435–446.

[7] People are really hating the new Microsoft Skype redesign, [Online]. Available: https://www.uctoday.com/collaboration/team-collaboration/people-really-hating-new-microsoft-skype-redesign/

[8] Microsoft admits users think the new Skype is 'exceptionally bad', [Online]. Available: http://www.digitaljournal.com/tech-and-science/technology/microsoft-admits-users-think-the-new-skype-is-exceptionally-bad/article/497609

[9] Facebook Messenger is getting slammed by tons of negative reviews," [Online]. Available: https://www.businessinsider.com/facebook-messenger-app-store-reviews-are-humiliating-2014–8

[10] "Pokémon Go is suddenly getting a lot of bad reviews," [Online]. Available: https://www.vocativ.com/347862/pokemon-go-is-suddenly-getting-a-lot-of-bad-reviews/index.html.

[11] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng.*, 2018, pp. 48–58.

[12] N. Genc-Nayebi and A. Abran, "A systematic literature review: Opinion mining studies from mobile app store user reviews," *J. Syst. Softw.*, vol. 125, pp. 207–219, 2017.

[13] D. M. Blei, A. Y. Ng, and M. Jordan, "Latent dirichlet allocation," in *Proc. Neural Inf. Process. Syst.*, 2001, pp. 601–608.

[14] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 1445–1456.

[15] C. Li, H. Wang, Z. Zhang, A. Sun, and Z. Ma, "Topic modeling for short texts with auxiliary word embeddings," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 165–174.

[16] C. Gao, B. Wang, P. He, J. Zhu, Y. Zhou, and M. R. Lyu, "PAID: Prioritizing app issues for developers by tracking user reviews over versions," in *Proc. IEEE 26th Int. Symp. Softw. Rel. Eng.*, 2015, pp. 35–45.

[17] R. Das, M. Zaheer, and C. Dyer, "Gaussian LDA for topic models with word embeddings," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process. Asian Federation Natural Lang. Process.*, 2015, pp. 795–804.

[18] D. Jiang, L. Shi, R. Lian, and H. Wu, "Latent topic embedding," in *Proc. 26th Int. Conf. Comput. Linguistics, Proc. Conf. Tech. Papers*, 2016, pp. 2689–2698.

[19] W. Hu and J. Tsujii, "A latent concept topic model for robust topic inference using word embeddings," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 380–386.

[20] S. Li, T. Chua, J. Zhu, and C. Miao, "Generative topic embedding: A continuous representation of documents," in *Proc.54th Annu Meeting Assoc. Comput. Linguistics*, 2016, *arXiv:1606.02979*.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[22] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1532–1543.

[23] X. Gu, H. Zhang, and S. Kim, "Deep code search," in *Proc. 40th Int. Conf. Softw. Eng.*, 2018, pp. 933–944.

[24] Y. Chen, H. Amiri, Z. Li, and T. Chua, "Emerging topic detection for organizations from microblogs," in *36th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2013, pp. 43–52.

[25] Q. Diao, J. Jiang, F. Zhu, and E. Lim, "Finding bursty topics from microblogs," in *Proc. 50th Annu Meeting Assoc. Comput. Linguistics*, 2012, pp. 536–544.

[26] X. Yan, J. Guo, Y. Lan, J. Xu, and X. Cheng, "A probabilistic model for bursty topic discovery in microblogs," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 353–359.

[27] J. Huang, M. Peng, H. Wang, J. Cao, W. Gao, and X. Zhang, "A probabilistic method for emerging topic tracking in microblog stream," *World Wide Web*, vol. 20, no. 2, pp. 325–350, 2017.

[28] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. 22nd Int. Conf. Requirements Eng.*, 2014, pp. 153–162.

[29] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "Ar-miner: Mining informative reviews for developers from mobile app marketplace," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 767–778.

[30] A huge Facebook redesign is coming but it's far more than a new website," [Online]. Available: https://www.techradar.com/news/facebook-redesign-2019

[31] App Annie, [Online]. Available: https://www.appannie.com/

[32] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. 22nd Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 1999, pp. 50–57.

[33] A. Ahmed and E. P. Xing, "Timeline: A dynamic hierarchical dirichlet process model for recovering birth/death and evolution of topics in text stream," in *Proc. 26th Conf. Uncertainty Artif. Intell.*, 2010, pp. 20–29.

[34] A. Srivastava and C. Sutton, "Autoencoding variational inference for topic models," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[35] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proc. IEEE 22nd Int. Requirements Eng. Conf.*, 2014, pp. 153–162.

[36] V. K. R. Sridhar, "Unsupervised topic modeling for short texts using distributed representations of words," in *Proc. 1st Workshop Vector Space Model. Natural Lang. Process.*, 2015, pp. 192–200.

[37] T. Chen, S. W. Thomas, and A. E. Hassan, "A survey on the use of topic models when mining software repositories," *Empirical Softw. Eng.*, vol. 21, no. 5, pp. 1843–1919, 2016.

[38] C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis," in *Proc. 18th ACM Conf. Inf. Knowl. Manage.*, 2009, pp. 375–384.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst. 26*, 2013, pp. 3111–3119.

[40] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, Beijing, China, 2014, pp. 1188–1196.

[41] P. M. Vu, T. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Mining user opinions in mobile app reviews: A keyword-based approach (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automat. Softw. Eng.*, 2015, pp. 749–759.

[42] Y. Man, C. Gao, M. R. Lyu, and J. Jiang, "Experience report: Understanding cross-platform app issues from user reviews," in *Proc. 27th IEEE Int. Symp. Softw. Rel. Eng.*, 2016, pp. 138–149.

[43] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Comput. Linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[44] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2004, pp. 168–177.

[45] Opinion mining, sentiment analysis, and opinion spam detection, [Online]. Available: https://www.cs.uic.edu/ liub/FBS/sentiment-analysis.html#lexicon

[46] Part-of-speech tagging, 2019. [Online]. Available: https://www.nltk.org/book/ch05.html

[47] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit," *Psychol. Bulletin*, vol. 70, no. 4, pp. 213–220, 1968.

[48] M. Steyvers and T. Griffiths, "Probabilistic topic models," in *Handbook of Latent Semantic Analysis*, Mahwah, NJ, USA: Lawrence erlbaum, 2007, vol. 427, no. 7, pp. p. 427–448.

[49] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, 2009, Art. no. 15.

[50] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 145–151, Jan. 1991.

[51] P. J. Rousseeuw and M. Hubert, "Robust statistics for outlier detection," *Wiley Interdiscipl. Data Mining Knowl. Discov.*, vol. 1, no. 1, pp. 73–79, 2011.

[52] Q. Mei, X. Shen, and C. Zhai, "Automatic labeling of multinomial topic models," in *Proc. 13th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2007, pp. 490–499.

[53] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, "A decomposable attention model for natural language inference," in*Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 2249–2255.

[54] Gensim tool, [Online]. Available: https://radimrehurek.com/gensim/

[55] A. Islam and D. Z. Inkpen, "Semantic text similarity using corpus-based word similarity and string similarity," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 2, 2008, Art. no. 10.

[56] L. AlSumait, D. Barbará, and C. Domeniconi, "On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 3–12.

[57] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*. New York, NY, USA: Springer, 1992, pp. 196–202.

[58] S. E. Ahmed, "Effect sizes for research: A broad application approach," *Technometrics*, vol. 48, p. 573, 2006.

[59] Engaging users with app updates, 2019. [Online]. Available: https://developer.apple.com/app-store/app-updates/

[60] Discussion on the full screen mode of YouTube, 2019. [Online]. Available: https://www.reddit.com/r/jailbreak/comments/5tlkwu/question_full_screen_rotation_fix/, 2019.

[61] P. J. Guo, T. Zimmermann, N. Nagappan, and B. Murphy, "Characterizing and predicting which bugs get fixed: An empirical study of microsoft windows," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng.*, 2010, pp. 495–504.

[62] F. Thung, D. Lo, L. Jiang, Lucia, F. Rahman, and P. T. Devanbu, "When would this bug get reported?," in *Proc. 28th IEEE Int. Conf. Softw. Maintenance*, Trento, Italy, 2012, pp. 420–429.

[63] J. Cambronero, H. Li, S. Kim, K. Sen, and S. Chandra, "When deep learning met code search," in *Proc. ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2019, pp. 964–974.

[64] Y. Hong *et al.*, "Testing usability and acceptability of a web application to promote physical activity (iCanFit) among older adults," *JMIR Hum. Factors*, vol. 1, no. 1, 2014, Art. no. e2.

[65] G. O'Malley, G. Dowdall, A. Burls, I. J. Perry, and N. Curran, "Exploring the usability of a mobile app for adolescent obesity management," *JMIR Mhealth Uhealth*, vol. 2, no. 2, 2014, Art. no. e29.

[66] R. Likert, *A Technique for the Measurement of Attitudes*. New York, NY, USA: Science Press, 1932.

[67] W. Xie, F. Zhu, J. Jiang, E. Lim, and K. Wang, "TopicSketch: Real-time bursty topic detection from twitter," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2216–2229, Aug. 2016.

[68] M. Hasan, M. A. Orgun, and R. Schwitter, "A survey on real-time event detection from the twitter data stream," *J. Inf. Sci.*, vol. 44, no. 4, pp. 443–463, 2018.

[69] R. Arun, V. Suresh, C. E. V. Madhavan, and M. N. Murty, "On finding the natural number of topics with latent dirichlet allocation: Some observations," in *Proc. Adv. Knowl. Discov. Data Mining*, 2010, pp. 391–402.

[70] W. Zhao *et al.*, "A heuristic approach to determine an appropriate number of topics in topic modeling," *BMC Bioinf.*, vol. 16, no. 13, 2015, Art. no. S8.

[71] M. Gerlach, T. P. Peixoto, and E. G. Altmann, "A network approach to topic models," *Sci Adv.*, vol. 4, no. 7, 2017, Art. no. eaaq1360.

[72] W. J. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE Trans. Softw. Eng.*, vol. 43, no. 9, pp. 817–847, Sep. 2017.

[73] F. Palomba *et al.*, "Recommending and localizing change requests for mobile apps based on user reviews," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng.*, 2017, pp. 106–117.

[74] G. Grano, A. Ciurumelea, S. Panichella, F. Palomba, and H. C. Gall, "Exploring the integration of user feedback in automated testing of android applications," in *Proc. IEEE 25th Int. Conf. Softw. Anal., Evol. Reeng.*, 2018, pp. 72–83.

[75] G. Uddin and F. Khomh, "Opiner: An opinion search and summarization engine for APIs," in *Proc. 32nd IEEE/ACM Int. Conf. Automat. Softw. Eng.*, 2017, pp. 978–983.

[76] B. Lin, F. Zampetti, G. Bavota, M. D. Penta, and M. Lanza, "Pattern-based mining of opinions in Q&A websites," in *Proc. 41st Int. Conf. Softw. Eng.*, 2019, pp. 548–559.

[77] E. Platzer, "Opportunities of automated motive-based user review analysis in the context of mobile app acceptance," in *Proc. Central Eur. Conf. Inf. Intell. Syst.*, 2011, pp. 309–316.

[78] W. Maalej and H. Nabil, "Bug report, feature request, or simply praise? On automatically classifying app reviews," in *Proc. 23rd IEEE Int. Requirements Eng. Conf.*, 2015, pp. 116–125.

[79] S. A. Licorish, B. T. R. Savarimuthu, and S. Keertipati, "Attributes that predict which features to fix: Lessons for app store mining," in *Proc. 21st Int. Conf. Eval. Assessment Softw. Eng.*, 2017, pp. 108–117.

[80] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?," *IEEE Softw.*, vol. 32, no. 3, pp. 70–77, May/Jun. 2015.

[81] A. D. Sorbo *et al.*, "What would users change in my app? Summarizing app reviews for recommending software changes," in *Proc. 24th SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 499–510.

[82] X. Gu and S. Kim, "'What parts of your apps are loved by users?' (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automat. Softw. Eng.*, Lincoln, NE, USA, 2015, pp. 760–770.

[83] E. Noei, F. Zhang, and Y. Zou, "Too many user-reviews, what should app developers look at first?," *IEEE Trans. Softw. Eng.*, vol. 47, no. 2, pp. 367–378, Feb. 2021.

[84] M. R. Islam and M. F. Zibran, "Leveraging automated sentiment analysis in software engineering," in *Proc.14th Int. Conf. Mining Softw. Repositories*, 2017, pp. 203–214.

[85] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1352–1382, 2018.

[86] M. D. Hoffman, D. M. Blei, and F. R. Bach, "Online learning for latent dirichlet allocation," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 856–864.

[87] C. Gao, H. Xu, J. Hu, and Y. Zhou, "AR-tracker: Track the dynamics of mobile apps via user review mining," in *Proc. IEEE Symp. Serv.-Oriented Syst. Eng.*, 2015, pp. 284–290.

[88] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proc. 10th Working Conf. Mining Softw. Repositories*, 2013, pp. 41–44.

[89] B. Fu, J. Lin, L. Li, C. Faloutsos, J. I. Hong, and N. M. Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2013, pp. 1276–1284.

[90] C. Gao *et al.*, "Emerging app issue identification from user feedback: Experience on WeChat," in *Proc. 41th Int. Conf. Softw. Eng.*, 2019, pp. 279–288.

[91] M. Nayebi, B. Adams, and G. Ruhe, "Release practices for mobile apps–What do users and developers think?," in *Proc. IEEE 23rd Int. Conf. Softw. Anal., Evol., Reeng.*, Suita, Japan, 2016, pp. 552–562.

[92] M. Nayebi, H. Farrahi, and G. Ruhe, "Analysis of marketed versus not-marketed mobile app releases," in *Proc. 4th Int. Workshop Release Eng.*, 2016, pp. 1–4.

[93] A. Marcus, M. S. Bernstein, O. Badar, D. R. Karger, S. Madden, and R. C. Miller, "TwitInfo: Aggregating and visualizing microblogs for event exploration," in *Proc. Int. Conf. Hum. Factors Comput. Syst.*, 2011, pp. 227–236.

[94] S. B. Kaleel and A. Abhari, "Cluster-discovery of twitter messages for event detection and trending," *J. Comput. Sci.*, vol. 6, pp. 47–57, 2015.

[95] J. Li, J. Wen, Z. Tai, R. Zhang, and W. Yu, "Bursty event detection from microblog: A distributed and incremental approach," *Concurrency Comput. Pract. Experience*, vol. 28, no. 11, pp. 3115–3130, 2016.

[96] C. Li, A. Sun, and A. Datta, "Twevent: Segment-based event detection from tweets," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, Maui, Hawaii, USA, 2012, pp. 155–164.

[97] A statistical analysis of 1.2 million Amazon reviews, 2014. [Online]. Available: https://minimaxir.com/2014/06/reviewing-reviews/

[98] Y. Zhou, Y. Su, T. Chen, Z. Huang, H. C. Gall, and S. Panichella, "User review-based change file localization for mobile applications," 2019, *arXiv:1903.00894*.

[99] P. M. Vu, H. V. Pham, T. T. Nguyen, and T. T. Nguyen, "Phrase-based extraction of user opinions in mobile app reviews," in *Proc. 31st IEEE/ACM Int. Conf. Automat. Softw. Eng.*, 2016, pp. 726–731.