Listening to Users’ Voice: Automatic Summarization of Helpful App Reviews

Cuiyun Gao, Yaoxian Li, Shuhan Qi, Graduate Student Member, IEEE, Yang Liu, Xuan Wang, Zibin Zheng, and Qing Liao, Member, IEEE

Abstract—App reviews are crowdsourcing knowledge of user experience with the apps, providing valuable information for app release planning, such as major bugs to fix and important features to add. There exist prior explorations on app review mining for release planning; however, most of the studies strongly rely on predefined classes or manually annotated reviews. Also, the new review characteristic, i.e., the number of users who rated the review as helpful, which can help capture important reviews, has not been considered previously. In the article, we propose a novel framework, named SOLAR, aiming at accurately summarizing helpful user reviews to developers. The framework mainly contains three modules: the review helpfulness prediction module, topic-sentiment modeling module, and multifactor ranking module. The review helpfulness prediction module assesses the helpfulness of reviews, i.e., whether the review is useful for developers. The topic-sentiment modeling module groups the topics of the helpful reviews and also predicts the associated sentiment, and the multifactor ranking module aims at prioritizing semantically representative reviews for each topic as the review summary. Experiments on five popular apps indicate that SOLAR is effective for review summarization and promising for facilitating app release planning.

Index Terms—Review helpfulness, review summarization, topic modeling, topic sentiment, user reviews.

I. INTRODUCTION

The quality of mobile apps directly influences the user experience and concerns the benefits gained by developers. With more apps continuing to spring up, app owners face more challenges in providing good service to users and standing out from competitors. User reviews are valuable information from users and reflect instant user experience with apps, e.g., major bugs encountered by users and missing app features. Summarizing the useful information in user reviews can help developers pay attention to important user concerns and, thus, facilitate release planning of the apps.

Online reviews not only enhance user awareness but also serve as a reliable source of information about the quality of the app or the service of interest. Recently, user review mining has been extensively studied by both academic and industrial communities on prioritizing app reviews [1], [2], [3], classifying reviews into different categories [4], [5], [6], predicting the features favored/disliked by users [7], [8], or identifying emerging app issues [9], [10]. Most of the studies, however, strongly rely on predefined classes or manually annotated reviews, which may require huge manual labor. For example, Chen et al. [1] observed that manually annotating 2000 user reviews as informative or noninformative could cost 7.4 h. Popular apps, such as Facebook and WeChat, may receive tens of thousands of reviews each day [11]. Thus, an automatic and effective approach is necessary for summarizing user reviews.

Automatically summarizing user reviews is challenging. First, user reviews are generally short in length and contain massive noisy words, e.g., misspelled words, and abbreviations, so the context information is limited. Second, user reviews are mostly noninformative. According to [1], only 30% of the reviews provide informative user opinions for app updates, which increases the difficulty of extracting useful content from reviews. Third, reviews contain multiple and various topics for different apps, and the predefined granularities are difficult to cover all the topics of the apps. For example, Noei et al. [12] identified 23 common topics, such as searching and web browsing, while Di Sorbo et al. [5] summarized 12 topic clusters, including pricing and resources, etc., which are more general compared to Noei et al.’s definition [12]. Moreover, not all the topics require in-depth inspection by developers, and appropriate prioritization of the topics can be time-saving but still challenging. For example, ratings of user reviews are a commonly used index...
for prioritizing the extracted topics, but the ratings may not be aligned with the review texts [13].

To mitigate the above challenges, we design a novel framework, named SOLAR, an abbreviation for Summarization of Helpful App Reviews. In Chen et al. [1]'s work, the informative reviews are extracted by training on manually annotated reviews, which is rather labor-intensive and time-consuming. To alleviate the effort in filtering reviews according to the informativeness, we employ a new review characteristic, i.e., the number of users who rated the review as helpful, referred to as “helpfulness number” for convenience in this article. The helpfulness number of each review indicates the volume of users who consider that the delivered information is useful for them. In general, the reviews described in more detail or with real messages tend to be rated as more helpful [14]; thus, with the helpfulness number considered, the informative reviews could be captured, which constitutes the first process, i.e., review helpfulness prediction. Then, to mitigate the short-length nature of user reviews, we employ a biterm topic model (BTM) [15] for clustering topics, where BTM is especially designed for modeling topics of the short text corpus. For accurately estimating the sentiment associated with each topic, we adopt the topic model approach BST [16], which jointly models topics with sentiments. The topic modeling process is the second process, i.e., topic-sentiment summarization process. Finally, we propose a novel review ranking mechanism by involving multiple factors, including the semantic representativeness of the extracted topics and corresponding estimated sentiment, etc.

To validate the effectiveness of the proposed review summarization approach SOLAR, we conduct extensive experiments on five apps with a total of 11 659 reviews. Experimental results indicate that SOLAR achieves superior performance over the baseline approaches, increasing the precision and recall scores by at least 10.41% and 12.75%, respectively.

The contributions of this article are summarized as follows:
1) We propose a novel framework for automatically and accurately summarizing user reviews for facilitating release planning of mobile apps.
2) We propose to predict the informativeness of reviews based on a new review characteristic, i.e., the helpfulness number, and no manual labor is required. Novel multifactor review ranking approaches are also put forward for more accurate review summarization.
3) Experiments on real-world applications verify the effectiveness of the proposed framework. Our code and dataset are publicly available at https://github.com/monsterLee599/SOLAR.

Paper structure: The remainder of this article is organized as follows. Section II describes the motivation and background of our work. Section III presents our proposed framework. We introduce the experimental setup in Section IV and elaborate on the dataset, baseline models, and comparison results in Section V. Section VI provides some discussion about the proposed framework. Section VII illustrates the related work. We conclude and mention future work in Section VIII.

II. BACKGROUND AND MOTIVATION

A. User Review and the Informativeness

User reviews are an essential channel between app users and the developers, generally containing attributes such as user names, post dates, review texts, and ratings. Two examples of reviews for the Android Instagram app are illustrated in Fig. 1.1 Recently, Google Play releases a new characteristic, i.e., the number of users who rated the reviews as helpful, as shown at the top-right corner of each review. We refer to the new attribute as “helpfulness number” throughout this article. We can see that the first piece of review, as depicted in Fig. 1(a), has a greatly larger helpfulness number than the second piece of review, possibly because review 1 provides more detailed and clearer feedback, e.g., about the “Reel” feature, while review 2 only complains about the new update and does not detail the app issue. Thus, the attribute reveals the usefulness of the reviews to other readers and can be considered as an index of the review’s informativeness.

B. Topic Modeling

Topic modeling is a type of statistical model for uncovering the topics that occur in a collection of documents. One of the most popular topic modeling approaches is latent Dirichlet allocation (LDA) [17]. LDA assumes that each document is a mixture of topics, where a topic is a probabilistic distribution over words. LDA models each document \( r \) as a mixture of latent topics \( \theta_r \in \mathbb{R}^K \) following a multinomial distribution, where \( K \) is the number of topics. Each latent topic is described as a multinomial distribution \( \phi \in \mathbb{R}^V \) over the vocabulary, where \( V \) indicates the total number of unique words (i.e., vocabulary).

Although LDA has been proven successful in modeling formal and well-edited documents, such as news reports [18] and scientific articles [19], its performance will be inevitably compromised when processing short and ill-formed texts, such as app reviews and Twitter messages [15].

1The two examples were obtained on the same day from Google Play Store.
Biterm topic model [15] is specifically designed for modeling topics in short texts. Different from LDA, which captures the document-level word cooccurrence patterns, BTM directly models the word cooccurrence patterns in the whole corpus. The outputs of both LDA and BTM are two matrices: 1) Document-topic matrix $\Theta \in \mathbb{R}^{R \times V}$, where $R$ denotes the number of reviews; and 2) topic-word matrix $\Phi \in \mathbb{R}^{K \times V}$.

Joint sentiment/topic model (JST) [20] can detect the topic sentiment besides modeling topics. Unlike other machine learning approaches for sentiment classification, JST is unsupervised. Different from LDA and BTM, JST assumes that topics are associated with sentiment labels and words are associated with sentiment labels and topics. JST also produces two matrices but with three dimensions: 1) document-sentiment-topic matrix $\Theta \in \mathbb{R}^{R \times S \times K}$, where $S$ denotes the number of sentiment labels (e.g., $S = 3$ indicate that the sentiment labels include positive, neutral, and negative); 2) sentiment-topic-word matrix $\Phi \in \mathbb{R}^{S \times K \times V}$.

III. METHODOLOGY

In this section, we present an overview of the proposed framework SOLAR and then elaborate on each process of SOLAR. Fig. 2 presents the overall architecture of the proposed framework, which consists of four major steps. The first step preprocesses the raw user review data into a well-structured format to facilitate subsequent processes. In the second step, the helpfulness of each review instance is estimated and the reviews predicted as “helpful” are delivered to the next step. The third step jointly models the topics and associated sentiment for the helpful reviews. The last step prioritizes 1) topics and 2) reviews’ instances for each topic based on multiple factors, including semantic representativeness and sentiment. The prioritized reviews are regarded as the summary of reviews and will be provided to developers for managing app releases.

A. Preprocessing

We first remove emoticons and convert all the reviews into their lowercase. We then adopt rule-based methods in [21] to rectify repetitive words (e.g., “very very good” to “very good”). Finally, we lemmatize each word into the root form following the lemmatization method described in [22] (e.g., “was” to “be”).

B. Review Helpfulness Prediction

We extract 20 linguistic features that can potentially impact the helpfulness of review instances and differentiate helpful reviews from unhelpful ones. Tables I summarizes the set of 20 features that are grouped along five dimensions: stylistics, readability, lexicon, sentiment, and content.

Stylistics dimension refers to the stylistic features, including the numbers of words and sentences from word level, sentence level, and review level. The length information can influence the completeness of the information conveyed by reviews [2], [27], [28]. We use six features to quantify the stylistics dimension—namely, review-length, sentence-length, avg-sentence-length, 1-char-word-num, 2-char-word-num, and >2-char-word-num. All the features are calculated by counting words where the review-length is from review level, sentence-length and avg-sentence-length are from sentence level, and the other features are from character level.

Readability dimension refers to features that measure the readability of the user review. Readability, in general, is measured based on the syllables per word, the length of sentences, and the ratio of difficult words—it can estimate how many years of education are required for textual understanding [29]. To quantify the readability of the review text, we use the three readability measures proposed by previous work—namely difficult-word-num, flesch [23], and dale-chall [25].

Difficult words are defined as those with more than two syllables, which do not including proper nouns, familiar jargon or compound words, and difficult words do not contain common suffixes (e.g., “-es”) as a syllable [30]. We denote the number of words, syllables, difficult words, and sentences as Words, Syllables, Difficult Words, and Sentences, respectively. Based on the above definitions, the empirical formulas for calculating flesch [23] and dale-chall [25] are as follows:

$$\text{flesch} = 206.835 - 1.015 \times \frac{\text{Words}}{\text{Sentences}} - 84.6 \times \frac{\text{Syllables}}{\text{Words}}$$

$$\text{dale-chall} = 0.16 \times \frac{\text{Difficult Words}}{\text{Sentences}} + 0.05 \times \frac{\text{Words}}{\text{Sentences}}$$

where the constants in above formulas are from [23] and [25].

We also consider the number of misspelling words as one index for readability, denoted as misspelling-word-num. We define the misspelling words as those that are not found in Enchant English dictionary [26].

Lexicon Dimension refers to the features that are related to the word lexicons. Four features are involved to quantify the lexicon dimension—namely, noun-num, verb-num, adjective-num, subjective-num, and lexicon-diversity. We conduct part-of-speech tagging for each review text and count the respective numbers for nouns, verbs, and adjectives. The number of subjective words is counted based on the released subjective word list in [24]. The lexicon-diversity is the ratio of the number of unique words in a review text to the review-length.

Sentiment Dimension refers to the features reflecting user opinions. We consider three features for measuring the sentiment dimension—namely, polarity, sentiment-word-num, and rating-extremity. The polarity of a review text indicates whether the
expressed opinion is positive, negative, or neutral. We measure the polarity of the review text by computing the total positive score minus the total negative score of the review text, as illustrated in (3). The positive score and negative score are computed as the numbers of positive words (denoted as Positive Words) and negative words (i.e., Negative Words), respectively, where the negative words and positive words are determined based on the SentiWordNet database [31].

Sentiment-word-num is calculated as the ratio of the number of sentiment words to the total words. We also consider rating-extremity to measure how the review rating diverges from the average user rating, with the formula shown in (4).

\[
\text{polarity} = \frac{\text{Positive Words} - \text{Negative Words}}{\text{Words}} 
\]

\[
\text{rating-extremity} = |\text{Rating} - \text{Avg. Rating}|. 
\]

Content Dimension refers to features that capture the textual characteristics based on text mining techniques. Prior work showed that text mining techniques can help with informative review extraction [1]. Thus, we expect that analyzing the textual content of a review text can help distinguish helpful and unhelpful reviews. To quantify the content dimension, three features are included—namely, quality-related-word-num, uncertainty-degree, and unigram-tf-idf.

Based on functionalities and features, users make assessments about actual app quality. We rely on the dictionary of General Inquirer [32] to compute Quality-related-word-num. The General Inquirer is a well-established framework for content analysis. The advantage of adopting the dictionary-based approach is the validation of the dictionary as well as the resulting standardized classifications [33]. The quality-related words are determined by their similarity distances with the word “quality” in the General Inquirer dictionary. Examples of quality-related words are illustrated in Table II. Similarly, we determine the review uncertainty score by considering the “if”-related words in the General Inquirer dictionary, which generally denotes feelings of uncertainty, doubt, and vagueness [27] (exemplar words are “almost” and “may”).

Unigram tf-idf is a common technique for information retrieval and text mining, reflecting how important a word \( w \) is to a review text \( r \) in the collection \( R \). The Unigram tf-idf is calculated based on term frequency, i.e., \( tf \), and inverse document frequency, denoted as \( idf \).

\[
tf-idf(w, r, R) = tf(w, r) \cdot idf(w, R) 
\]

where \( tf(w, r) = \log(1 + \text{freq}(w, r)) \) and \( idf(w, R) = \log\left(\frac{\text{count}\{r \in D : w \in r\}}{|R|}\right) \).
**C. Bitern-Based Sentiment-Topic Modeling**

Based on the prediction results of review helpfulness, we expect the reviews classified as "helpful" to be informative for developers and employ them for the subsequent processes. In this section, we introduce the unsupervised model, named BST [16], for jointly modeling topics and sentiment of app reviews. Fig. 3(a) depicts the graph illustration of BST. The detailed modeling process can be referred to the work [16].

An example of the output sentiment-topic-word matrix $\Phi$ is shown in Fig. 3(b), where the number of sentiment labels is three and different sentiment labels $s = 1, 2, 3$ indicate positive, neutral, and negative, respectively. For each biterm $b$, BST models its topic distribution over the vocabulary and sentiment distribution over the three sentiment polarities. The sentiment-topic distribution of each review $r$ can be calculated as:

$$
P(z, s|r) = \sum_b P(z, s|b) \cdot P(b|r)$$

$$
P(z, s|b) = \frac{P(z, s)P(w_{ij}|z, s)P(w_{ij}|z, s)}{\sum_{z,s} P(z, s)P(w_{ij}|z, s)P(w_{ij}|z, s)}$$

$$
P(b|r) = \frac{n_r(b)}{\sum_{b} n_r(b)}$$  \hspace{1cm} (6)

where $n_r(b)$ is the frequency of the biterm $b$ in the review $r$. Similarly, we can infer the sentiment distribution of each review. We denote the computed topic distribution of each review $r$ as $P(z|r, s) = \{P(z_1|r, s), P(z_2|r, s), \ldots, P(z_k|r, s), \ldots\}$, where $k$ means the $k$th topic given a sentiment label $s$ and $\sum_k P(z_k|r, s) = 1$. The sentiment distribution of each review is indicated as $P(s|r, z) = \{P(s_1|r, z), P(s_2|r, z), P(s_3|r, z)\}$ where $s_1, s_2, s_3$ denote negative polarity, neutral polarity, and positive polarity, respectively.

**D. Multifactor Topic and Review Ranking**

The multifactor ranking step aims at prioritizing semantically representative reviews for each topic while ensuring the usefulness of the reviews for developers. We employ two procedures to rank reviews, i.e., topic ranking, and then review ranking. The prioritization scores of topics are utilized for review ranking. SOLAR finally outputs the prioritized reviews for each topic.

1) Topic Ranking: We prioritize the topics $z$ based on the corresponding features $F_z$, including mainly four aspects: topic proportion, topic sentiment, average rating, and freshness. The total score for each topic is calculated as follows:

$$
Score_z = \sum_{f \in F_z} \omega_f f
$$  \hspace{1cm} (7)

where $f \in F_z$ is the grading aspect of each topic, $\omega_f$ is the weight of the computed score for aspect $f$, and $\sum_{f \in F_z} \omega_f = 1$.

**Topic Proportion:** Topics covering more reviews generally indicate that the topics have received more attention in the recent period; thus, the topics tend to be more important. We define the proportion for topic $z$ as follows:

$$
f^z_{\text{Volume}} = \frac{\sum_{r \in R} \sum_s P(z, s|r)}{|R| \max_{r \in R} (P(z, s|r))}
$$  \hspace{1cm} (8)

where $R$ is the collection of reviews and $s$ indicates the sentiment label.

**Topic Sentiment:** Generally, the reviews predicted as negative tend to be more important for app developers than the positive reviews for app updating. Based on the output of Section III-C, we calculate the sentiment score of a topic as follows.

$$
f^z_{\text{Sentiment}} = \frac{\sum_{r \in R} P(s_1|z, r)}{|R| \max_{r \in R} (P(s_1|z, r))}
$$  \hspace{1cm} (9)

where $s_1$ indicates the negative polarity.

**Average Rating:** The rating reflects the overall attitude of users toward the app. Topics with poorer ratings should be paid more attention by developers. We calculate the average rating of the topic $z$ as follows:

$$
f^z_{\text{Avg.Rating}} = \frac{\sum_{r \in R} \text{Rating}_r}{|R| \max_{r \in R} (\text{Rating}_r)}
$$  \hspace{1cm} (10)

where $\text{Rating}_r$ denotes the user rating of the review $r$.

**Freshness:** Latest reviews can embody users’ newest opinions about the app, while the problems reflected in early-posted reviews tend to be relatively less important. We calculate the freshness feature of one topic as follows:

$$
f^z_{\text{Freshness}} = \frac{\sum_{r \in R} \text{Timestamp}_r}{|R| \max_{r \in R} (\text{Timestamp}_r)}
$$  \hspace{1cm} (11)

where $\text{Timestamp}_r$ indicates the post time of the review $r$.

2) Review Ranking: The review ranking process considers a set of features $F^r$ besides the prioritization scores of the topics. Other features include user rating, freshness, sentiment polarity, review length, and topic score. The overall score of one review $r$ is calculated as follows:

$$
Score_r = \sum_{f \in F^r} \omega_f f
$$  \hspace{1cm} (12)
where \( f \) is the scoring feature of each review, \( \omega_f \) is the weight defined for the feature \( f \), and \( \sum_{f \in F^r} \omega_f = 1 \).

**Rating:** User rating could directly express users’ experience during the app usage. Poor user ratings generally indicate that the users are discontent with the app usage, and the reviews may describe the problems they encountered or unsatisfied with app functionalities. We normalize the user rating for each review as the rating score \( f_{\text{Rating}} \)

\[
f_{\text{Rating}}^r = \frac{\text{Rating}_r}{\max(\text{Rating})}. \tag{13}
\]

**Freshness:** We also consider the post time of the reviews. Reviews uploaded more recently could be more important to developers for app release

\[
f_{\text{Freshness}}^r = \frac{\text{Timestamp}_r}{\max_{r \in R}(\text{Timestamp}_r)}. \tag{14}
\]

**Sentiment Polarity:** The reviews predicted with negative sentiment polarity are more important for app release than the reviews with positive sentiment. Based on the sentiment-topic-word matrix in Section III-C, we calculate the sentiment score of each review as follows:

\[
\begin{align*}
f_{\text{Negative}}^r &= P(s_1 | r, z_k) \tag{15} \\
f_{\text{Neutral}}^r &= P(s_2 | r, z_k) \tag{16} \\
f_{\text{Positive}}^r &= P(s_3 | r, z_k). \tag{17}
\end{align*}
\]

**Review length:** Reviews with longer lengths tend to deliver more detailed information about the user experience, and thus could be more useful for developers. The feature score is computed as follows:

\[
f_{\text{Length}}^r = -\log(\text{Length}_r) \tag{18}
\]

where \( \text{Length}_r \) indicates the number of words in the review \( r \).

**Topic:** The prioritization scores of the topics are also incorporated since the reviews that are more related to the topics with higher ranking scores \( \text{Score}^z \) would be more representative of the collected reviews

\[
f_{\text{Topic}}^r = \sum_z P(z | r) \times \text{Score}^z. \tag{19}
\]

### IV. EXPERIMENTAL SETUP

**A. Dataset**

1) **Dataset for helpfulness Prediction:** We crawled the apps ranked at the top 200 during August 2019 from Google Play, and collected 1,239,754 reviews for 364 apps\(^2\) in total. The collected information for each review instance includes the review text, author name, post date, helpfulness number, and developer’s reply. Since the helpfulness numbers for the reviews are updated per day, we remove the repetitive reviews and keep the most recent helpfulness number for each review instance. Consequently, the number of unique review instances is 571,823 and the distribution of helpfulness numbers is shown in Fig. 4. We check the distribution of helpfulness numbers by the Shapiro–Wilk test [35]. Shapiro–Wilk test is a typical test of normality in which the null hypothesis is that the input samples come from a normally distributed population. If the p-value computed by the Shapiro–Wilk test is smaller than 0.05, it means that the input distribution is significantly different from a normal distribution. The Shapiro–Wilk test result (p-value < 0.001) shows that the helpfulness number is normally distributed. For the convenience of model training, we determine that one review is helpful if its helpfulness number exceeds specific quantile \( q \). During experiments, we set \( q = 0.5 \) for ensuring that the numbers of helpful and unhelpful reviews in the collected corpus are equally distributed.

2) **Dataset for Validating Review Prioritization:** Following the previous work [4], we exploit the Android user reviews dataset made available by Chen et al. [36]. The dataset publishes user reviews for multiple releases of ~21 K apps, and the information for each review contains the review text, author name, posted date, user rating, and the app’s release it refers to. Additionally, each app in the dataset is associated with a metadata file describing its basic information such as the “updated” optional field that app developers can use to report the changes they made to the different app releases. According to Villarroel et al. [4], five apps in the CLAP dataset are considered in the work, as given in Table III. The five selected app releases received a total of 11,659 user reviews. Besides, the selected CLAP dataset has no overlapping with the dataset for helpfulness prediction.

---

\(^2\)Since the list of top 200 apps changed during the app crawling, the number of collected apps exceeded 200.
TABLE IV
EXAMPLES OF MANUAL EVALUATION

| ChangeLog of the eBay app | Review | Infor | Hit |
|---------------------------|--------|-------|-----|
| 1 Search refinement locking now available worldwide | Woot let me leave feedback | ✓ | ✗ |
| 2 Fixed bug where seller feedback would not load | When I click on view seller’s other items, nothing happens. | ✓ | ✓ |
| 3 It’s still can’t saved search refinement. | Excellent service and great value for money. | ✗ | ✓ |
| 4 Feedback issues | | | |

B. Implementation

The implementation details of the major phases of SOLAR, including review helpfulness prediction, sentiment-topic modeling, and multifactor topic and review ranking, are as follows. In the review helpfulness prediction phase, we conduct a tenfold cross validation for evaluating the trained classifier on distinguishing helpful and unhelpful reviews. In the sentiment-topic modeling phase, we group app reviews into \( K = 8 \) topics for each sentiment polarity. In the multifactor ranking phase, we take the top eight reviews for each topic based on the ranking scores for evaluation. Specifically, for the multifactor topic ranking, we experimentally set \( \omega_f \) for the topic’s aspects as 0.15, 0.2, 0.35, and 0.3 for the four grading aspects, i.e., topic proportion, topic sentiment, average rating, and freshness, respectively. For the review ranking, we experimentally define \( \omega_f \) for the review’s aspects as 0.2, 0.1, 0.1, 0.05, 0.05, 0.2, and 0.3 for the aspects including review rating, freshness, negative polarity, neutral polarity, positive polarity, review length, and topic, respectively.

In evaluation, we run each result ten times and computed the average for comparison. We asked three industrial developers who have more than three years of software development experience to manually check the consistency between the prioritized reviews and changelogs and also the informativeness. Each participant was paid 50$ for completing the evaluation.

We use the eBay app to explain our manual evaluation criterion. Table IV presents the changelogs of the app in our benchmark dataset (top) and five review examples (bottom). The criterion for determining reviews’ informativeness depends on whether the reviews describe issues in detail or provide helpful suggestions for developers. For example, the “feedback issues” mentioned in review 5 does not contain any useful information for developers, and is labeled as “noninformative.” The criterion for determining whether reviews hit any changelog is based on whether the reviews are semantically relevant to the changelogs.

For the review 1 in Table IV, although the review is related to “feedback,” it describes about “leave feedback” instead of loading feedback as the second changelog. So review 1 is labeled as inconsistent with the changelogs.

C. Evaluation Metrics

We evaluate the review prioritization results following the previous work [4], [10]. The metrics include Precision, Recall, and F1-Score.

\[
\text{Precision} = \frac{\#(G \cap T)}{\#(T)} \tag{20}
\]

\[
\text{Recall} = \frac{\#(G \cap T)}{\#(G)} \tag{21}
\]

\[
\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{22}
\]

where \( T \) and \( G \) denote the prioritized topics and changelogs, respectively.

We also involve the metric infor-score for measuring the informativeness of the prioritized reviews defined as

\[
\text{infor-score} = \frac{\#\text{informative reviews}}{\#\text{prioritized reviews}} \tag{23}
\]

where \#prioritized reviews and \#informative reviews indicate the numbers of prioritized reviews and informative reviews among the prioritized reviews, respectively.

D. Baseline Approaches

There exist many studies on prioritizing reviews for facilitating release planning. Not all the studies are comparable since some of them involve external knowledge such as source code or GitHub issues, and some require manual annotations for processing. In this article, we aim to review summarization without manual labeling or external source. To select the baseline approaches for comparison, we examine the related work from several aspects: With/without manually annotated data (abbreviated as MA data), accessibility of MA data, and with/without external knowledge reproducibility of source code.

We search the related work published in the recent seven years (i.e., 2014–2021) from Google Scholar.\(^3\) To ensure the quality of the papers, we exclude the papers with citations of fewer than five. Table V lists our examination results. As can be seen in Table V, all the related approaches require manually annotated data, which laterally reflects one advantage of SOLAR, i.e., no manual labor is involved. By removing the prior approaches with external knowledge or unavailability of MA data, we choose AR-Miner and IDEA as baseline approaches.

AR-Miner [1] is a typical framework for mining informative app reviews based on informative review extraction and topic modeling, where the prediction for the informativeness of app reviews requires training on manually labeled data. We adopt the well-trained model for extracting informative reviews of the benchmark dataset and then conduct review ranking.

IDEA [10] is one of the state-of-the-art online emerging app issue detection approaches. IDEA adapts an online topic modeling approach to track the changes in topics along with app versions and identifies the abnormal topics as emerging app issues. IDEA automatically labels each topic with the most semantically representative reviews. To ensure a fair completion, we restrict IDEA to predict the topics of the current app version without considering historical app versions.

V. EXPERIMENTAL RESULTS

In this section, we illustrate the experiment results of SOLAR by comparing with IDEA [10], and another competing approach,
AR-Miner [1], to assess its capability in prioritizing user reviews. Our experiments are aimed at answering the following research questions.

**RQ1:** What is the impact of different classifiers on the performance of review helpfulness prediction? Which features are more important for differentiating helpful reviews from unhelpful ones?

**RQ2:** What is the performance of SOLAR in app review prioritization compared with the baselines?

**RQ3:** What is the impact of the unhelpful reviews filtering process on the model performance?

**RQ4:** What is the impact of different numbers of topics on the performance of SOLAR?

### A. RQ1: Performance of Review Helpfulness Prediction

In RQ1, we explore the efficacy of different classifiers on the performance of review helpfulness prediction. Besides, we study the important features by explicitly considering the contributions of reviews from five dimensions, including stylistics, readability, lexicon, sentiment, and content.

In this study, we use SVM [34] as the default classifier to construct the model. We also use random forest [39] and expectation maximization naive Bayes (EMNB) [40] as the underlying classifiers for our baselines. The prediction results are illustrated in Table VI. From the table, we can observe that SVM can attain better overall performance in predicting review helpfulness than RF and EMNB. RF has higher precision than SVM (86.6 versus 85.5) but is worse than SVM in recall and F1-score. Besides, SVM and RF outperform EMNB concerning all the metrics. As shown at the bottom of Table VI, we can find that without considering any dimension of features reduces the classification performance. Specifically, the stylistics features are the most important for training the classifier regarding the F1-score metric, indicating that helpful reviews tend to present distinguishable text lengths and word lengths compared with unhelpful reviews. Besides, the readability and sentiment features also benefit the classification, which is reasonable. For example, review texts with better readability explain that the reviews are more carefully written and more likely to be helpful.

**Answer to RQ1:** In summary, the SVM-based classifier is effective in review helpfulness prediction. All the features utilized for classification are useful for distinguishing helpful and unhelpful reviews, among which the stylistics features are the most important.

### B. RQ2: Comparison With the Baselines

To evaluate the performance of SOLAR in review prioritization, we compare it with baseline approaches, including IDEA [10] and AR-Miner [1]. To ensure a fair comparison, we select the top eight reviews of each topic for SOLAR and baseline approaches.

Table VII presents the comparison results on the CLAP datasets. We can observe that SOLAR performs better than AR-Miner and IDEA in review prioritization. For example, the average results of SOLAR are 73.38, 78.25, and 74.42 in terms of Precision, Recall, and F1-Score, respectively, which outperform 10.41%, 12.75%, and 11.72% than AR-Miner, respectively. The results demonstrate that SOLAR can prioritize more topics containing the key terms in the changelogs, and the prioritized topics also reflect more app changelogs. SOLAR achieves consistently the best ranking performance regarding the F1-Score metric for all the studied apps except for eBay. For eBay’s reviews, AR-Miner presents slightly better performance than SOLAR, i.e., 89.50 and 84.58, respectively. The lower results of SOLAR may be attributed to that the eBay’s changelog only describes two changes and may not involve all the changes made in practice [10]. We also find that the reviews prioritized by SOLAR for each topic are more semantically coherent than those results output by AR-Miner, as depicted in Table VIII. We choose the two topics “seller feedback” and “search refined” since they
are semantically consistent with the changelogs of eBay. As can be seen in Table VIII, only 25.0%–37.5% of the reviews provided by AR-Miner are relevant to the corresponding topics; while the reviews prioritized by SOLAR are more semantically related. Among all the three approaches, IDEA shows the lowest average performance on our benchmark dataset. This may be because that IDEA is specifically designed for online app review analysis and may require reviews from multiple historical versions for effective review prioritization.

Regarding the infor-score metric, SOLAR significantly outperforms the baseline models by at least 28.49% on average, indicating that the reviews prioritized by SOLAR are more informative. Future research can utilize SOLAR to filter non-informative reviews for downstream tasks.

**Answer to RQ2:** In summary, SOLAR shows significantly better average performance than baseline approaches in review prioritization. Besides, more than 85% of the reviews prioritized by SOLAR are informative, outperforming the baselines by at least 28%.

**C. RQ3: Impact of Review Helpfulness Prediction on the Performance of SOLAR**

In this section, we study the impact of the review helpfulness prediction process on the performance of SOLAR. For analysis, we evaluate the performance of SOLAR without considering the review helpfulness information for filtering, namely SOLAR\textsubscript{no–filtering}. The results are depicted in Table IX. We can observe that the reviews prioritized by SOLAR are more informative than those output by SOLAR\textsubscript{no–filtering}, with an increased rate at 32.14% in terms of the infor-score metric. The advantage of SOLAR is consistent for the studied apps, except for the Timeregift app for which SOLAR\textsubscript{no–filtering} only shows marginally higher performance than SOLAR. Besides, the review helpfulness prediction process contributes greatly to the review prioritization performance. For example, SOLAR achieves Precision, Recall, and F1-Score at 73.38%, 78.25%, and 74.42% on average, respectively, outperforming SOLAR\textsubscript{no–filtering} by 9.59%, 4.19%, and 7.33%, respectively.

**Answer to RQ3:** In summary, the review helpfulness prediction process in SOLAR is beneficial for filtering non-informative reviews and providing more accurate review summaries.

**D. RQ4: Impact of Different Topic Numbers on the Performance of SOLAR**

During experimentation, we set the topic number $K = 8$ for each sentiment polarity. In this section, we analyze the impact of different topic numbers on the performance of SOLAR. Fig. 5 illustrates the performance changes along with varying topic numbers. We can observe that the values of Precision, F1-Score, and infor-score metrics present a downward trend with the growth of topic number, while the Recall metric shows an increasing trend. The results are reasonable. Larger topic numbers indicate that more reviews are prioritized, and thereby present higher chances to cover more changelogs, leading to an increasing Recall score. Meanwhile, more prioritized reviews would be irrelevant to the changelogs, thus lowering the other metric scores. As can be seen in Fig. 5, SOLAR achieves relatively better performance when the topic number is defined as 6 or 8. During experimentation, we set the topic number as 8 according to the F1-Score metric.

**Answer to RQ4:** In summary, SOLAR generally shows a downward trend with the growth of topic number. The model achieves relatively better performance when the topic number is defined as 6 or 8. According to the F1-Score metric, we set the topic number as 8.

**VI. DISCUSSION**

**A. Threat and Validity**

There are four major threats to the validity of our article.

1) **The diversity and freshness of available datasets:** We directly use the publicly released data of CLAP provided by their authors. The data include only five apps from Google Play Store. The limited categories and number...
of studied apps may influence the generalization of the proposed SOLAR. Besides, the helpfulness prediction model dataset was created in 2019, which seems a bit old. Since the recently published review data [41], [42] do not involve the helpfulness number, we train the helpfulness prediction model based on the old dataset. Moreover, the features of helpful reviews from different periods would be similar, so the freshness of the reviews would not be a great threat. We will conduct more experiments when appropriate datasets get publicly available.

2) Bias in manual evaluation: For checking the performance of SOLAR, we invite three industrial developers to evaluate the consistency between the prioritized reviews and changelogs and also the informativeness. The results of the human evaluation can be impacted by the participants’ experience. To mitigate the bias in human evaluation, we ensure that all three different participants evaluated each prioritized review. Besides, all the participants are industrial developers who have more than three years of software development experience.

3) Evaluation of baseline models: For comparison, we survey the recent studies on app review analysis, and chose two reproducible baselines: AR-Miner [1] and IDEA [10]. Since the original papers do not report the results on our benchmark datasets, we evaluate the baselines by carefully replicating the algorithms described in the original work of AR-Miner and restricting IDEA to prioritize reviews without considering historical app versions.

4) Weights in the multifactor topic and review ranking: In the multifactor ranking phase, the weights in (7) and (12) for respectively computing rankings scores of topics and reviews can impact the performance of the proposed approach. In this work, we experimentally set the weights for evaluation, indicating that the reported results of SOLAR may be suboptimal. In future work, we will build upon heuristic algorithms [43] to automatically determine the optimal weights.

B. Analysis on the Impact of Rating Normalization

During ranking reviews in Section III-D, we conduct normalization on the ratings, as shown in (13). In this section, we analyze the impact of rating normalization on the performance of SOLAR. The results are illustrated in Table X, where

---

\[ \text{TABLE VIII} \]

COMPARISON ON THE TOPIC GENERATED BY AR-MINER AND SOLAR FOR THE EBAY APP

| Approach | Topic 1<br>seller feedback | Topic 2<br>search refined |
|----------|-----------------------------|--------------------------|
| SOLAR    | ... can't view feedback ...  | ... inability to remember search settings ... |
|          | ... won't load any feedback for seller ... | ... have to refine search ... |
|          | ... can't see feedback sometimes | ... sick ... change the list results of a search ... |
| AR-Miner | ... can't read feedback ... from seller ... | ... search result STILL defaults to Best Match ... |
|          | ... wouldn't load when connected ... | ... removed search options ... |
|          | ... can't view message inbox ... | ... have to change everyday I search ... |
|          | ... can't write a message to seller ... | ... search results always ... to “best match” ... |
|          | ... can't view the descriptions or buy ... | ... doesn’t hold search settings ... |
|          | ... not log in to paypal to pay ... | ... Feedback won't load ... |
|          | ... have to refine search ... | ... inability to remember search settings ... |
|          | ... Won't let me search anything ... | ... hate the search suggestions ... |
|          | ... can't even get on to it | ... pictures don’t appear ... |

---

\[ \text{TABLE IX} \]

RESULTS OF SOLAR WITH AND WITHOUT THE REVIEW HELPFULNESS PREDICTION STEP

| App Name | Approach | inΦ-score | Precision | Recall | F1-Score |
|----------|----------|-----------|-----------|--------|----------|
| eBay     | SOLAR_non-normalized | 95.16 | 73.75 | 100.00 | 84.58 |
| SOLAR    | 95.16 | 73.75 | 100.00 | 84.58 |
| Viber    | SOLAR_non-normalized | 94.44 | 63.20 | 50.00 | 39.03 |
| SOLAR    | 97.81 | 66.25 | 60.00 | 62.29 |
| Barebooth | SOLAR_non-normalized | 88.51 | 67.50 | 68.00 | 67.25 |
| SOLAR    | 80.15 | 61.25 | 70.00 | 64.66 |
| Humboled | SOLAR_non-normalized | 43.93 | 89.82 | 73.50 | 79.65 |
| SOLAR    | 81.82 | 96.90 | 75.00 | 84.45 |
| Timorific | SOLAR_non-normalized | 74.59 | 73.75 | 85.00 | 77.93 |
| SOLAR    | 73.36 | 68.75 | 86.25 | 76.13 |
| Average  | SOLAR_non-normalized | 64.81 | 66.96 | 75.10 | 69.34 |
| SOLAR    | 85.64 | 73.38 | 78.25 | 74.42 |

---

\[ \text{TABLE X} \]

RESULTS OF SOLAR WITHOUT AND WITH RATING NORMALIZATION DURING REVIEW RANKING, WHERE SOLAR_non-normalized INDICATES THAT THE REVIEW RANKING PROCESS DOES NOT INVOLVE RATING NORMALIZATION

| App Name | Approach | inΦ-score | Precision | Recall | F1-Score |
|----------|----------|-----------|-----------|--------|----------|
| eBay     | SOLAR_non-normalized | 93.25 | 80.00 | 100.00 | 88.20 |
| SOLAR    | 95.16 | 73.75 | 100.00 | 84.58 |
| Viber    | SOLAR_non-normalized | 86.41 | 75.50 | 60.00 | 57.96 |
| SOLAR    | 97.81 | 66.25 | 60.00 | 62.29 |
| Barebooth | SOLAR_non-normalized | 78.89 | 61.25 | 75.00 | 65.10 |
| SOLAR    | 80.15 | 61.25 | 70.00 | 64.66 |
| Humboled | SOLAR_non-normalized | 81.82 | 82.83 | 75.00 | 78.41 |
| SOLAR    | 81.82 | 96.90 | 75.00 | 84.45 |
| Timorific | SOLAR_non-normalized | 72.41 | 72.50 | 87.50 | 79.10 |
| SOLAR    | 78.63 | 68.75 | 86.25 | 76.13 |
| Average  | SOLAR_non-normalized | 83.76 | 70.73 | 78.50 | 73.77 |
| SOLAR    | 85.64 | 73.38 | 78.25 | 74.42 |

---

[Online]. Available: https://github.com/monsterLee599/AR-Miner
SOLAR_{non-normalized} indicates the review ranking without rating normalization. The average results of SOLAR_{non-normalized} are 70.73, 78.50, and 73.77 in terms of Precision, Recall, and F1-Score on average, respectively. According to the experimental results, we can observe that SOLAR performs slightly better than SOLAR_{non-normalized} in Precision, F1-Score, and infor-Score. The results indicate that rating normalization during review ranking can help SOLAR to prioritize informative reviews.

C. Analysis on the Positive and Neutral Sentiment Polarity of Reviews

To explore the effects of positive and neutral sentiment polarity of reviews, we only consider the negative sentiment polarity of reviews (namely SOLAR_{only-neg}), and Table XI presents the results. The average results of SOLAR_{only-neg} are 74.73, 77.25, and 75.00 in terms of Precision, Recall, and F1-Score, respectively. Compared to SOLAR, we can only observe that considering the negative sentiment polarity has a slightly effect on the Precision, Recall, and F1-Score scores but has a significant impact on the infor-score. This result indicates that the positive and neutral sentiment polarity of reviews can help SOLAR to prioritize more informative reviews.

D. Analysis on the Number of Selected Reviews

In this section, we study the impact of different numbers of prioritized reviews on the performance of SOLAR. The results are illustrated in Fig. 6. We can observe that the values of Precision, Recall, and F1-Score present an increasing trend along with the growth of the review number, while the infor-score metric shows a downward trend. The results indicate that the number of prioritized reviews tends to cover more changelogs; however, they are also likely to include more noninformative reviews. To balance the informativeness and accuracy of prioritized reviews, we choose the number of prioritized reviews as 8 during experimentation.

VII. RELATED WORK

A. App Marketplace Analysis

The growth of smartphones and mobile applications makes the app marketplace a hotspot for researchers within and outside the software engineering community [16], [44]. Harman et al. [45] pointed out that app marketplaces provide a wealth of information in the form of pricing and customer reviews and thus can be treated as a new form of software repository. They also used data mining to analyze apps’ technical, custom, and business aspects in BlackBerry World. Chia et al. [46] discovered that the ratings used in app marketplaces are not reliable indicators of privacy risks of an app. Minelli et al. [47] proposed to leverage source code, usage of third-party APIs, historical data, along with data extracted from app marketplace to better comprehend apps. Martin et al. [48] introduced an approach to causal impact analysis to help app developers understand the impact of app releases. They also conduct a comprehensive survey on app marketplace analysis, including review mining.

B. App Review Mining

User feedback plays an essential role in serving as a major channel between developers and users, reflecting new feature requirements, enhancements in the user interface, and reporting serious app bugs [49]. For many years, researchers from academia and industry have explored mining app reviews for assisting different stages of app development and maintenance, such as prioritizing app reviews [1], [2], [3], [21], [50], predicting app feature liked/disliked by users [7], [8], classifying app reviews [4], [5], [6], and identifying emerging app issues [9], [10].

The booming user reviews inspired researchers to come up with heuristic approaches. Regarding prioritizing app issues, Chen et al. [1] proposed a computational framework that visualizes the most “informative” reviews which are identified by a topic model and an effective review ranking scheme. Gao et al. [2] pointed out that the issues presented in the level of phrase, i.e., a couple of consecutive words, can be more easily understood by developers than in long sentences. Then they designed a framework to track reviews over the release versions of the app and recommend phrase-level issues of an app to its developers. Malgaonkar et al. [50] studied recent works on app review prioritization and developed a multicriteria heuristic model for identifying and prioritizing informative reviews. For predicting

| App Name | Approach | infor-score | Precision | Recall | F1-Score |
|----------|----------|-------------|-----------|--------|----------|
| eBay     | SOLAR_{only-neg} | 89.84        | 83.75     | 100.00 | 90.72    |
|          | SOLAR    | 95.16       | 73.75     | 100.00 | 84.58    |
| Viber    | SOLAR_{only-neg} | 79.69        | 58.75     | 60.00  | 58.25    |
|          | SOLAR    | 97.81       | 66.25     | 60.00  | 62.29    |
| Barebone | SOLAR_{only-neg} | 79.29        | 63.75     | 70.00  | 66.25    |
|          | SOLAR    | 80.15       | 61.25     | 70.00  | 64.66    |
| Hmiltoned| SOLAR_{only-neg} | 81.82        | 92.40     | 75.00  | 82.55    |
|          | SOLAR    | 81.82       | 96.90     | 75.00  | 84.45    |
| Timerific| SOLAR_{only-neg} | 71.84        | 75.00     | 81.25  | 77.25    |
|          | SOLAR    | 73.26       | 68.75     | 86.25  | 76.13    |
| Average  | SOLAR_{only-neg} | 80.49        | 74.73     | 77.25  | 75.00    |
|          | SOLAR    | 85.64       | 73.38     | 78.25  | 74.42    |
app features liked/disliked by users, Gu et al. [7] and Guzman et al. [8] proposed to classify reviews into predefined categories and extracts aspects in sentences that include evaluation of aspect using natural language processing techniques. In order to classify reviews into different categories and prioritize emerging issues, Villarroel et al. [4] proposed a framework to categorize user reviews based on the information they carry out (e.g., bug reporting), cluster together related reviews (e.g., all reviews reporting the same bug), and automatically prioritize the clusters of reviews to be implemented. Gao et al. [9], [10] proposed an efficient and automated framework to identify emerging app issues based on online review analysis which achieves both high accuracy and real-time identification. Wu et al. [41] created a Chinese dataset from the Chinese Apple App Store and built a regression model to identify key features of app by analyzing app description and positive/negative user reviews. Haering et al. [51] focused on the gap between technically written bug reports with colloquially written app reviews, extracting issues from app reviews and matching them to bug reports. Henao et al. [42] proposed a framework for mining feature requests and bug reports from tweets and app store reviews via transfer learning.

In recent years, researchers are getting into analyzing the dynamic nature of user reviews. For example, Gao et al. [10] automatically capture app issues discussed in user reviews and detect the emerging ones for version modification. Besides employing user feedback for collecting user opinions, Guzman et al. [52] incorporate app-related twitters to facilitate the software evolution process. Nayebi et al. [53] propose the concept of “marketability” for open-source mobile apps and adopt analogical reasoning to guide unsuccessful marketable releases to be transited into successful ones.

Automatic review summarization is another challenging problem in app review mining because most app reviews are short, noisy, noninformative, and sometimes contain multiple and various topics for different apps [1]. Natural language processing approaches have been adopted to tackle this challenge. Previous research papers [5], [12] identified common topics in app reviews by different granularity, such as searching, web browsing, pricing, and resources. Mudambi et al. [13] found that not all the topics demand developers’ deep inspection. Besides, ratings of user reviews are a commonly used index, but the ratings may be aligned with the review texts. Therefore, accurate prioritization of the topics can be time-saving. Fu et al. [54] filtered reviews that expressed inconsistent sentiment with their ratings and then summarized the remaining topics. Iacob et al. [55] utilized latent Dirichlet allocation [17] and linguistic rules to generate summary for new feature requests. Aratijo et al. [56] proposed a BERT-based language model to automatically extract software requirements from app reviews.

C. Sentiment Analysis

The sentiment analysis techniques aim to detect the polarity (e.g., positive, neutral, or negative) of sentiment implied by texts [57], [58]. In recent years, many studies apply deep learning models, including reinforcement learning [59], emotional recurrent unit [60], and graph convolutional networks [61], [62], for sentiment analysis [63], [64], [65]. There exists other techniques [66], [67] proposed for sentiment analysis. For example, Valdivia et al. [66], [68] proposed weighted aggregation models for detecting and filtering neutral texts. Wang et al. [67] proposed a multilevel fine-scaled approach to handle ambivalence in the text. We will consider the issue of ambivalence [67] in our scenario in the future.

VIII. CONCLUSION

To maintain high-quality apps, developers often take a lot of effort to extract key information from large amounts of scribbled user reviews. In the work, we propose a novel framework, named SOLAR, focusing on automatically summarizing helpful user reviews for developers. SOLAR filters noninformative reviews based on a trained review helpfulness prediction model and groups topics jointly with corresponding sentiments by the topic-sentiment summarization module. We also propose a multifactor ranking module for prioritizing reviews for each topic. Extensive experiments verify the effectiveness of our proposed framework. In the future, we will conduct evaluation using app reviews across platforms and deploy SOLAR in industry.

REFERENCES

[1] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, and B. Zhang, “AR-miner: Mining informative reviews for developers from mobile app marketplaces,” in Proc. 36th Int. Conf. Softw. Eng., Hyderabad, India, P. Jalote, L. C. Briand, and A. van der Hoeck, Eds., 2014, pp. 767–778.
[2] 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. INSSRE IEEE Comput. Soc., 2015, pp. 35–45.
[3] F. Palomba et al., “Crowdsourcing user reviews to support the evolution of mobile apps,” J. Syst. Softw., vol. 137, pp. 143–162, 2018.
[4] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. D. Penta, “Release planning of mobile apps based on user reviews,” in Proc. 38th Int. Conf. Softw. Eng., Austin, TX, USA, 2016, pp. 14–24.
[5] A. D. Sorbo et al., “What would users change in my app? Summarizing app reviews for recommending software changes,” in Proc. ACM SIGSOFT FSE, 2016, pp. 499–510.
[6] W. Maaelej and H. Nabil, “Bug report, feature request, or simply praise? On automatically classifying app reviews,” in Proc. 23rd IEEE Int. Requirements Eng. Conf., Ottawa, ON, Canada, 2015, pp. 116–125.
[7] X. Gu and S. Kim, “What parts of your apps are loved by users?,” in Proc. 30th IEEE/ACM Int. Conf. Autom. Softw. Eng., Lincoln, NE, USA, 2015, pp. 760–770.
[8] E. Guzman and W. Maaelej, “How do users like this feature? A fine grained sentiment analysis of app reviews,” in Proc. IEEE 22nd Int. Requirements Eng. Conf., Karlshrona, Sweden, 2014, pp. 153–162.
[9] C. Gao et al., “Emerging app issue identification from user feedback: Experience on wechat,” in Proc. IEEE Int. Conf. Softw. Eng., Softw. Eng. Pract., 2019, pp. 279–288.
[10] C. Gao, J. Zeng, M. R. Lyu, and I. King, “Online app review analysis for identifying emerging issues,” in Proc. 40th Int. Conf. Softw. Eng., Gothenburg, Sweden, M. Chaudron, I. M. Cmkovic Chechik, and M. Harman, Eds., 2018, pp. 48–58.
[11] “App Annie,” [Online]. Available: https://www.appannie.com/
[12] F. 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.
[13] S. M. Mudambi, D. Schuff, and Z. Zhang, “Why aren’t the stars aligned? An analysis of online review content and star ratings,” in Proc. 47th Hawaii Int. Conf. System Sci., Waikoloa, HI, USA, 2014, pp. 3139–3147.
[14] T. L. Ngo-Ye and A. P. Sinha, “The influence of reviewer engagement characteristics on online review helpfulness: A text regression model,” Decis. Support Syst., vol. 61, pp. 47–58, 2014.
[15] X. Yan, J. Guo, Y. Lan, and X. Cheng, “A bitemr topic model for short texts,” in Proc. WWW. Int. World Wide Web Conf. Steering Committee/ACM, 2013, pp. 1445–1456.
[16] C. Gao et al., “Emerging app issue identification via online joint sentiment-topic tracing,” IEEE Trans. Softw. Eng., vol. 48, no. 8, pp. 3025–3043, Aug. 2022.
[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” in Proc. Adv. Neural Inf. Process. Syst., Natural Synthetic, Vancouver, BC, Canada, 2001, pp. 601–608.
