# Detecting User Significant Intention via Sentiment-Preference Correlation Analysis for Continuous App Improvement

Jianmao Xiao[1], Shizhan Chen[1], Qiang He[2](✉), Hongyue Wu[1], Zhiyong Feng[1], and Xiao Xue[1]

[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
{zt_xjm,shizhan,hongyue.wu,zyfeng,jzxuexiao}@tju.edu.cn
[2] School of Software and Electrical Engineering, Swinburne University of Technology, Hawthorn, VIC 3122, Australia
qhe@swin.edu.au

**Abstract.** Detecting users' significant intentions (e.g., new features wanted) timely and precisely is crucial for developers to update and maintain their apps in the competitive mobile app market. Sentiment and preference mining from crowd reviews provide an opportunity to proactively collect app users' intentions, e.g., bug fixing and feature refinement. However, users' sentiment and preferences often change over time due to either internal factors (e.g., new bugs) or external factors (e.g., new competitors). This makes it difficult for app developers to grasp users' sentiment and preferences in time. In this paper, we propose a novel and automated framework named DSISP for detecting users' significant intentions effectively via sentiment-preference correlation analysis. DSISP first employs sentiment analysis and NLP (Natural Language Processing) techniques to obtain sentence-level sentiment scores and fine-grained user preference features from app reviews in different time slices. Then, the temporal correlation between user sentiment and preferences is analyzed, which can be used to monitor users' sentiment tendency and preferences in time. Finally, DSISP identifies users' dramatically-changing sentiment (e.g., sentiment valley) to detect users' significant intentions. We evaluate the feasibility and performance of DSISP by using real-world app reviews and app official changelogs. The experimental results show that DSISP can detect users' significant intentions effectively and efficiently, with a precision of 0.962 on average. It can help app developers keep track of how their users' intentions evolve over time so that they can improve their apps correspondingly and continuously.

**Keywords:** Significant intention · Sentiment analysis · Preference feature · Evolution and maintenance

E. Kafeza et al. (Eds.): ICSOC 2020, LNCS 12571, pp. 386–400, 2020.
https://doi.org/10.1007/978-3-030-65310-1_27

## 1 Introduction

App stores are digital distribution platforms that allow users to submit ratings, feedback and comments on apps, which explicitly or implicitly expresses their potential sentiment and preferences for apps [1,2], e.g., their satisfaction with particular features, the vulnerabilities encountered or requests for new features. Sentiment represents a user's approval of an app, and preference indicates its intention of the app. Users' preferences can be obtained by mining crowd review features that reflect users' opinions. Keeping track of users' sentiment and preference features timely and precisely can help app developers update and improve their apps, e.g., in terms of fixing bugs, or adding new features, etc. [3,4].

User reviews are direct feedback from users that have experienced the apps. In recent years, researchers have proposed several approaches to extract useful information from crowd reviews for maintaining and evolving mobile apps [3,5]. These approaches are mainly designed for user reviews classification [6,7], clustering [8,9], and summarization [1,3,10]. The extracted information represents crowd-sourced knowledge from the users' perspective and can be used to identify users' intention [11] or detecting app emerging issues [12], etc.

The abovementioned studies are mainly focused on reducing the effort in extracting software aspects or user preferences without considering the changes in users' sentiment and preference over time. In fact, due to app updates or changes in the external environment (e.g., new competitors), users' sentiment and preferences will change dynamically over time. For example, when an app crashes, is injected with ads, or breaches users' privacy, users' complaints will increase immediately. Their sentiment will also turn negative rapidly.

Users' up-to-date sentiment and preferences indicate their instant experiences with apps in use. When the users' sentiment and preferences are not grasped by app providers in time, it may lead to the loss of users and reduce the users' stickiness. For example, Facebook Messenger lost a large number of users in August, 2014 because it was found to contain severe privacy issues (e.g., accessing the photos and contact numbers on users' phones)[1]. Such issues had already been pointed out by users in their reviews a few months ago before that. Therefore, detecting and understanding of users' intentions timely is necessary and critical. However, the problem of how to effectively and timely detect users' significant intentions from app reviews have not been studied systematically.

In this paper, we propose a novel framework named DSISP (Detecting users' Significant Intentions via Sentiment and Preference analysis) for detecting users' sentiment and preferences by analyzing their reviews. DSISP takes user reviews as input and employs sentiment analysis technique [13] to calculate users' sentiment scores within different time slices. Then, it employs NLP and collocation finding technology [14] to mine fine-grained preference features from user reviews. After that, it analyzes the temporal correlation between users' sentiment and preferences. Finally, DSISP uses a SVI (Sentiment Valley Identification algorithm) and

[1] http://www.businessinsider.com/facebook-messenger-app-store-reviews-arehumiliating-2014-8.

Twitter-LDA [15] to identify users' significant sentiment and detect their significant intentions, respectively. In summary, the contributions of this paper are as follows.

– We present a method for aggregating crowed users' sentiment for each preference with automated sentiment analysis on app reviews. Then, we analyzed how these preferences are temporal correlation to users' sentiment.
– We propose a framework named DSISP to automatically detect users' significant intentions by analyzing the temporal correlation between users' sentiment and preferences. The source code of DSISP and review data are published on GitHub[2].
– We verify the effectiveness of DSISP based on the changelogs of six apps (include three open-source Android apps) in different app categories.

The remainder of the paper is structured as follows. Section 2 introduces the related work. Section 3 outlines the overall picture of DSISP and details each step involved in its procedure. Section 4 reports the experimental results. Section 5 concludes this paper and points out the future work.

## 2 Related Work

Currently, a number of approaches have been proposed to mine and analyze app reviews with the goal of deriving important information to help developers update their apps [7,8,16]. For example, Pagano et al. [16] investigated the correlation between app reviews and ratings. Harman et al. [17] proposed the concept of app store mining and identified the correlation between user ratings and app download rankings. These studies provide a basis for developers to understand user behaviors and adjust their app deployment strategies. However, there are several limitations which prevent app developers from using the information in the reviews effectively. For example, an app store generates a lot of app reviews every day - the Facebook app receives more than 10,000 reviews on Google Play every day[3]. Besides, reviews vary in quality. Manual analysis of a large number of such reviews is time-consuming and labor-intensive. To address this issue, automatic feature extraction is proposed to mine user needs [3,8,10].

Chen et al. [18] devised AR-MINER, an approach for filtering and ranking informative reviews using a semi-supervised learning based approach. They demonstrated that, on average, 35% of reviews contain informative contents. Based on AR-MINER, L. Villarroel et al. [7] proposed a method named CLAP, which employs classification and clustering algorithms to automatically prioritize the user reviews to be implemented when planning the subsequent app release. Palomba F et al. [8] proposed a method named CHANGEADVISOR to analyze the structure, semantics and sentiment of sentences in user reviews, extract useful information from user reviews and suggest changes to software components for

[2] https://github.com/ztxjm123/DSISP.
[3] App Annie. https://www.appannie.com/en.

developers. Zhou Y et al. [9] proposed an automated method named RISING that supports continuous integration of user feedback through classifying, clustering, and linking user reviews to the source code. Their experimental results show that RISING outperforms CHANGEADVISOR in clustering and positioning accuracy, thus producing more reliable results.

To better understand users' review contents and reduce the information gap between developers and users, most studies tend to artificially customize specific rules or concepts for mining user review features. Guzman E et al. [19] proposed a method for classifying app reviews into several categories related to software maintenance. Specifically, they divided user reviews into bug reports, functional advantages, functional defects, user requests, etc., which can provide developers with a detailed suggestion based on user reviews. Di Sorbo et al. [3,10] proposed a user intention classification method SURF to systematically define specific aspects of an app (such as UI, file download, etc.) that need to be maintained. It can effectively help developers plan app update tasks in the future.

Compared with directly extracting or clustering user review topics, the above-mentioned approaches further refined user review information and partitioned it into specific categories. However, little attention has been paid to how to mine users' significant intentions from app reviews in a timely manner, which are essential for developers to update and maintain their apps. To this end, this paper focuses on extracting users' sentiment and preference features in a continuous period. This will allow app developers or app vendors to track users' behaviors timely, and alert them to users' significant intentions promptly.

## 3 Methodology

DSISP aims to help developers to keep track of users' significant intentions which may be considered in app maintenance and improvement tasks for developers. It employs data mining and sentiment analysis techniques to automatically analyze users' sentiment and preferences for apps over a period of time. Figure 1 overviews the framework of DSISP. First, DSISP extracts users' sentiment and fine-grained preferences from their reviews through sentiment analysis and NLP, respectively. This obtains the user sentiment and produces a list of fine-grained preference features. Then, the review sentiment and preference evolution are analyzed based on time series. Afterwards, DSISP establishes the temporal correlation between sentiment and preference features. Finally, it employs a sentiment valley identification algorithm to detect users' significant intentions.

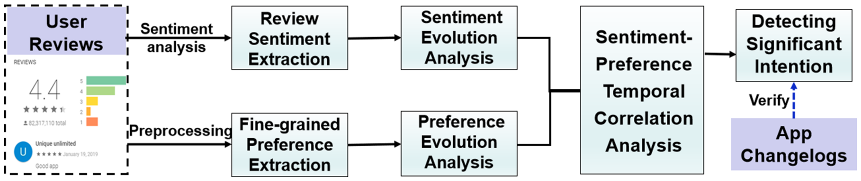

User Reviews
REVIEWS
4.4
82,317,110 total
Unique unlimited
Good app
Sentiment analysis
Preprocessing
Review Sentiment Extraction
Fine-grained Preference Extraction
Sentiment Evolution Analysis
Preference Evolution Analysis
Sentiment-Preference Temporal Correlation Analysis
Detecting Significant Intention
Verify
App Changelogs

**Fig. 1.** Overview of the DSISP

## 3.1 Review Preprocessing

App reviews are generally submitted via mobile terminals and written using mobile keyboards. They often contain lots of noise data, such as misspelled words, non-English words and non-informative words, etc. Such noise data impacts the fine-grained preference feature extraction from user reviews. The review data needs to be preprocessed first.

**Multi-language Filtering and Lemmatization.** We use the Langid tool[4] to filter non-English comment information from user reviews. Then, we use the Wordnet[5] lemmatizer from NLTK[6] to achieve word stem for reducing the number of features that need to be inspected later.

**Noun, Verb, and Adjective Extraction.** We use the part of speech (POS) tagging functionality of NLTK to extract the nouns, verbs, and adjectives in the reviews as these parts of a speech are most likely to express the users' fine-grained preference features. We manually inspected 1,040 reviews to validate this assumption.

**Noise Word Filtering.** This step aims to reduce the non-informative words from user reviews, such as emotional words (e.g., "bad" and "nice"), abbreviations (e.g., "asap"), and useless words (e.g., "someone"), etc. We use wordMapper [20], a dictionary of nearly 300,000 vocabularies related to app reviews, to reduce the impact of non-information words. It contains common spelling errors, abbreviations and abbreviated words in user reviews and their corrections. Based on this dictionary, we add extra words related to app reviews to the dictionary, such as"you're→you are", "app", "developer names", etc., summarized by two researchers from 1,040 user reviews. These predefined stop words are filtered out together with the stop words provided by NLTK.

After the preprocessing, most of the noise data has been removed from the user reviews. However, the preprocessing also shortens the length of the review texts at the same time. Thus, some of the review texts may contain too little information to be useful for extracting preference features. Therefore, in our work, we select reviews with rich vocabulary information - reviews with 4 words or fewer are discarded.

## 3.2 Review Sentiment Extraction

Sentiment analysis is the process of assigning a positive or negative quantitative value for each review [21]. We use SentiStrength [13], a sentiment analysis tool, to perform user review sentiment analysis. Compared with other tools, SentiStrength provides several advantages: it is designed for short informal texts with abbreviations and slang (features commonly observed in app review). It employs linguistic rules for negations, amplifications, booster words, emotions, which are particularly well suited for processing user reviews.

[4] https://github.com/saffsd/langid.py.
[5] https://wordnet.princeton.edu/.
[6] http://nltk.org/.

With SentiStrength, we assign each user review a positive $RS^{+}$ and negative $RS^{-}$ sentimental score, both ranging from 1 (neutral) to 5 (extremely positive or negative). The higher absolute value of the sentence score is taken as the final score of the review sentence, because the larger absolute value can reflect the actual sentiment of the sentence more accurately. In addition, it is worth noting that the emoticons, polarity words, etc., in user reviews would impact their sentiment. For instance, "love" is assigned a score of [3, −1] and "!" a [1, −1] score. Therefore, we analyze the user review sentiment scores directly without data preprocessing discussed in Sect. 3.1.

## 3.3 Fine-Grained Preference Extraction

Compared with ratings provided by users, user reviews offer finer-grained information and have become a rich source to help detect users' preferences [22]. Most of the reviews contain users' opinions on various aspects of the app (i.e., user preference), such as functional features or app security. For example, let us consider the review sentence "Uploading pictures with the app is necessary!". The functional feature (i.e., user preference feature) "Uploading picture" in this sentence expresses the user's intention directly.

We use the collocation search algorithm of NLTK to extract the fine-grained features in user reviews. Collocation can be expressed as a set of words that often co-occur [23]. It can include two or more words [14], but does not require words that are always adjacent. In user reviews, preference features can often be described as collocation phrases since they usually appear more frequently and represent a specific meaning about the app, e.g., app features or used experience. Given a set of collocation phrases, we use the grammatical relationship collocation extraction algorithm based on n-gram distance to find a collocation of two words in user reviews. Assume a collocation phrase $(w_i, w_j)$, $w_i$ is the base word, and $w_j$ is the collocation word. Both $w_i$ and $w_j$ belong to the review corpus after the preprocessing discussed in Sect. 3.1. We evaluate whether the review collocation phrase is reasonable based on the following three conditions [24]:

$$\begin{array}{lll} \text{strength} = \frac{\text{freq}_i - \bar{f}}{\sigma} \geq k_0, & & \text{(C1)} \\ \text{spread} \geq U_0, & & \text{(C2)} \\ p_j^i \quad \geq \bar{p}_i + \left(k_1 \times \sqrt{U_i}\right), & & \text{(C3)} \end{array} \tag{1}$$

where $freq_i$ represents the frequency of collocation phrase $(w_i, w_j)$ appear in user reviews, $\bar{f}$ is the average frequency of all collocation phrases in review corpus. In addition, let us define

$$spread = U_i = \frac{\sum_{j=1}^{10}\left(p_i^j - \bar{p}_i\right)^2}{10} \tag{2}$$

where $p_i^j$ is the appearance times of the collocation phrase $(w_i, w_j)$ in the distance $j$, the distance range of English words is defined as [−5, 5], similar to [24], $\bar{p}_i = \frac{1}{10}\sum_{j=-5}^{5} p_i^j (j \neq 0)$ is the average appearance times of collocation phrase

$(w_i, w_j)$ in all distances. $k_0$, $k_1$, and $U_0$ are custom thresholds. In our work, we set $k_0 = 1$, $k_1 = 1$, and $U_0 = 10$, similar [24].

In fact, the *strength* in $C1$ of formula (1) is to calculate the $z-score$ of $freq_i$, so as to filter out collocation phrases that appear less frequently in users reviews. The *spread* in $C2$ is the variance of collocation phrases at various distances, the greater the *spread*, the more reasonable the collocation phrase. The $C3$ further filters out collocation phrases that are $k_1$ times of $p_i$ based on the distance distribution of collocation $(w_i, w_j)$. So as to get the most reasonable collocation phrase.

A large number of collocations can be mined from user reviews, since users might use different words to express the same preference feature, i.e., the collocation phase has a synonym phenomenon. Therefore, we use the synonym dictionary Wordnet to merge different collocations. For example, if we have the following collocation phrases, <*picture view*>, <*view photographs*> and <*see photo*> with a frequency of 20, 10, and 5 respectively, we will select the most frequent occurrence collocation phrase (i.e., <*picture view*>) as the final merged preference feature. After implementing synonym merging, the top 10 collocation phrases with the highest frequency are selected as the final fine-grained preference features.

**Preference Feature Score Acquisition.** We compute the sentiment score for a preference feature based on the following principles: 1) If preference feature $PF_i$ appears in review sentence $j$, its sentiment score is equal to the positive or negative score of the sentence in which it is located; 2) If there are both positive $Po^+$ and negative $Ne^-$ scores in review sentence $j$, the preference feature sentiment score $PFS_i$ is calculated as:

$$PFS_i = \begin{cases} Po^+, |Po^+| > |Ne^-| \\ Ne^-, \text{else} \end{cases} \tag{3}$$

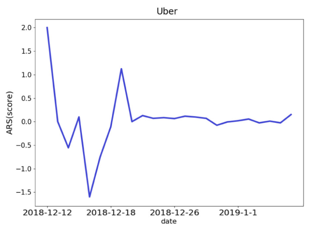
That is, the largest absolute value is selected as the feature score since it best expresses the user's sentiment toward the preference feature.

## 3.4 Sentiment and Preference Evolution

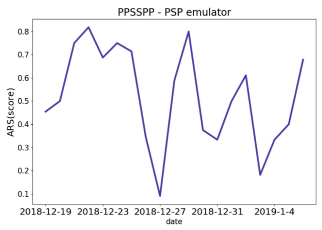
**Time Series Sentiment Evolution.** Assuming the users have made $n$ reviews on an app during time slice $T_i$. A total of sentiment scores can be obtained, denoted as $RS(\text{ score }) = \{RS_{i1}, RS_{i2}, \ldots, RS_{in}\}$. It is worth noting that at a fixed time, the number of user reviews for the same app is not fixed. Therefore, we take the average sentiment score $ARS(\text{score})$ as the users' sentiment score calculated as follows:

$$ARS(\text{ score }) = \frac{1}{n} \sum_{\substack{j=1 \\ i \in T(T_1, T_2, \ldots, T_m)}}^{n} RS_{i,j} \tag{4}$$

where $T(T_1, T_2, \ldots, T_m)$ represents $m$ consecutive but non-overlapping time slices with equal length. For example, each $T_i$ is 5 days or a week, $RS_{i,j}$ is



(a)



(b)

**Fig. 2.** Sentiment evolution of Uber and PPSSPP-PSP

the $j^{th}$ review sentiment score of the app in the $T_i$, $n$ is the number of reviews within $T_i$.

Figure 2 shows the sentiment evolution trends of Uber and PPSSPP-PSP. The history of users' sentiment score changes from Dec. 2018 to Jan. 2019 is visualized by line charts. We can see that the users express various changing trends for different apps over time. This phenomenon also validates our assumption above - users' sentiment changes over time. In addition, apart from a stable sentiment trend, users' sentiment often rises or falls rapidly during different time slices, resulting in peaks and valleys, e.g., Uber.

**Time Series Preference Evolution.** To calculate fine-grained preference feature score during different time slices and grasp the tendency of different preference features over time, for each app, we construct a $TSPFS$ (Time-Series Preference Feature Score) matrix to represent the distribution of fine-grained preference feature scores by all the users on each feature within different time slices.

**Table 1.** Time-series preference feature score matrix ($TSPFS$)

| | $T_1$ | $T_2$ | $\dots$ | $T_m$ |
|---|---|---|---|---|
| $PF_1$ | $PFS_{1,1}$ | $PFS_{1,2}$ | $\dots$ | $PFS_{1,m}$ |
| $PF_2$ | $PFS_{2,1}$ | $PFS_{2,2}$ | $\dots$ | $PFS_{2,m}$ |
| $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ |
| $PF_n$ | $PFS_{n,1}$ | $PFS_{n,2}$ | $\dots$ | $PFS_{n,m}$ |

Table 1 presents a $TSPFS$ matrix. A row of the matrix indicates that $n$ fine-grained preference features (i.e., $PF_1, PF_2, \dots, PF_n$), and columns are $m$ consecutive but non-overlapping time slices with equal lengths (i.e., $T_1, T_2, \dots, T_m$). $T_i$ represents the $i^{th}$ time slice. $PFS_{i,j}$ is the overall score of preference feature $j$ within the $i^{th}$ time slice, and is calculated as follows:

$$PFS_{i,j} = \sum_{\substack{k=1 \\ i \in T(T_1, T_2, \dots, T_m)}}^{n} FRS_{i,j,k} \tag{5}$$

where $n$ indicates the frequency that feature $j$ appears within the $i^{th}$ time slice, $FRS_{i,j,k}$ is the score of preference feature $j$ in review $k$ in slice $T_i$.

Figure 3 shows the evolution of users' preference features toward Uber, which includes the preference features and its proportions, feature scores. Take "customer service ($-19$): 38.45%" in Fig. 3 for example. It is a preference feature, where $-19$ is the preference feature score, and 38.45% is the proportion of preference features "customer service" of all the preference features at that time. Given a larger proportion of the feature and the absolute value of the feature score, the preference feature can better reflect the user's intention within that time slice.

As we can observe from Fig. 3, users' preferences are constantly changing across different time slices. For example, users' preferences include features "customer service", "waiting time" and "credit card". Their scores are $-19$, $-6$ and 4, respectively between December 12, 2018, and December 15, 2018. However, from December 16, 2018, to December 19, 2018, users' preference feature "credit card" disappeared, and the feature scores of "customer service", "waiting time" changed. This means that the degree of preference changed. The similar can be observed in other time slices. Here, we mainly focus on the new features with low feature scores since they are more likely to indicate users' real intention.

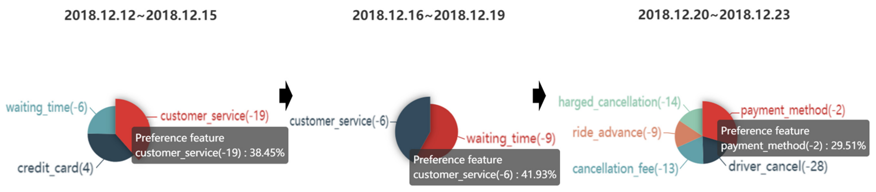


**Fig. 3.** Time series preference feature evolution of Uber

## 3.5 Sentiment-Preference Feature Correlation Analysis

As discussed before, users' sentiment change over time due to the app updates, security issues, etc. Accordingly, peaks and valleys appear in users' sentiment trend. Figure 4 shows the temporal correlation between users' sentiment and preference features. $(T_1, T_2, T_3, \ldots, T_m)$ represents consecutive but non-overlapping time slices with equal lengths. During different time slices, we can mine users' preference features based on users' sentiment (e.g., valley or peak) and implement sentiment-preference correlation analysis. In this study, we focus on the users' sentiment in valleys since they are more likely to indicate users' preference features. $(t_1, t_2, t_3, \ldots, t_m)$ represents the corresponding sentiment valley time points in each time slice. Through these valley time points, we can mine users' preference features and obtain their significant intentions.

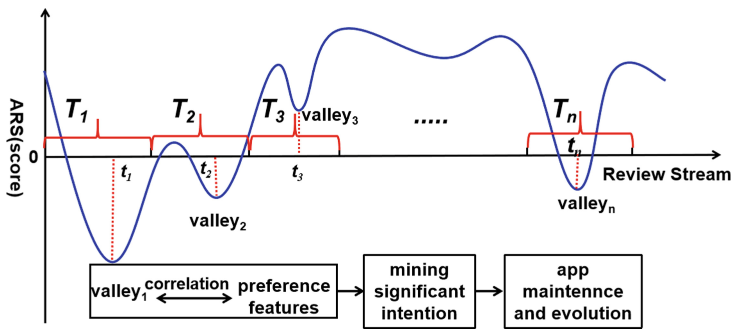

ARS(score)
0
T1
T2
T3
Tn
valley3
.....
t1
t2
t3
tn
Review Stream
valley2
valleyn
valley1 correlation preference features
mining significant intention
app maintennce and evolution

**Fig. 4.** Temporal correlation between user sentiment and preference features

## 3.6 User Significant Intention Detection

Users' intentions are in association with their sentiment. In order to detect users' significant intentions, the first step is to identify their significant sentiment (i.e., valley). Given the review sentiment scores and fine-grained preference features obtained with the methods discussed in Sects. 3.2 and 3.3, we partition a fixed time slice $v_t$, such as 5 or a week, etc., and find the sentiment valleys during that time slice. The specific process for identifying user sentiment valleys is shown in Algorithm 1.

We detect users' significant intentions around sentiment valleys. Similar to twitter texts, user reviews are short texts. Pagano and Maalej [16] found that 80.4% of users' reviews contain 160 characters or fewer, making Twitter-LDA [15] a good candidate for analyzing topics in user reviews. Therefore, we use Twitter-LDA to summarize the fine-grained preference features (i.e., collocation phrases) during the sentiment valley time slice as users' significant intentions (i.e., high-level preference features, referred to as $HLpf$ hereafter). Table 2 shows three most common $HLpf$ topic distribution of Uber app with their sentiments under sentiment valley within the time slices December 12–15, 2018. We can observe that these topics mainly represent the users complain about the worst customer service, credit card and waiting for time problems of Uber. These can well reflect the users' significant intentions.

**Table 2.** Topic distribution within the sentiment valley of Uber

| Topic | $PFS_{i,j}$ |
|---|---|
| customer_service, customer_disgusting, service_sometimes, service_worst, contact_customer | −19 |
| credit_card, credit_cost, card_adding, card_discounted, card_inconvenient | 4 |
| waiting_time, waiting_outside, amount_time, driver_waiting, driver_outside | −6 |

**Algorithm 1.** User Sentiment Valley Identification (SVI)

```
Input: $D$: the set of user reviews which include sentiment score and date
$v_t$: the number of days in a time slice
Output: $valley\_list$: the list of valleys
1: $initialization : status \leftarrow unknown$
2: for each $d \in D$ do
3:     if $d.status \leftarrow unknown$ then
4:         if $d.sentiment\_score > d{+}1$ then
5:             $status \leftarrow downhill$
6:         else
7:             $status \leftarrow uphill$
8:         end if
9:     end if
10:    if $d.status \leftarrow downhill$ then
11:        if $d.sentiment\_score < (d{+}1).sentiment\_score$ then
12:            if $d.date\_valley\_list[-1].date < v_t$ then
13:                if $valley\_list$ [-1]$.sentiment\_score > d.sentiment\_score$ then
14:                    $valley\_list$[-1]$\leftarrow d$
15:                else
16:                    add $d$ to $valley\_list$
17:                end if
18:            end if
19:        end if
20:    end if
21: end for
22: return $valley\_list$
```

## 4 Experiments and Results

### 4.1 Experiment Preparation

**Review Dataset.** We select the testing apps based on the following three criteria: i) there are adequate user reviews; ii) they are from different categories to ensure the generalization of the testing apps; iii) there are detailed changelogs.

Finally, we select six testing apps from Google Play and collected their changelogs from App Annie. Table 3 lists the testing apps with app name, category, total reviews and review time, etc. Overall, we obtain 48,278 reviews between November, 2018 and April, 2019 for all testing apps. Among them, they are include 3 open-source apps, because we want to verify whether DSISP can detect the users' significant intentions when the app with fewer user reviews but more modifications than closed-source apps. In addition, the review_time is before the app update_time. This can judge whether the user intentions detecting by DSISP are processed by the developer in time, thereby verifying the feasibility and efficiency of DSISP.

**Changelogs.** We evaluate the performance of DSISP using apps' official changelogs as ground truth. The changelogs reflect the actual modifications made by

**Table 3.** The subject apps

| AppName | Category | Reviews | Review_time | Changelog_version | Update_time |
|---|---|---|---|---|---|
| YouTube Music | Music & Audio | 5,875 | 2018.11.9–2019.1.24 | 2019.04.01 | 2019.4.1 |
| Uber | Maps & Navigation | 7,890 | 2018.12.12–2019.1.6 | 3.332.10005 | 2019.1.9 |
| Facebook | Social | 33,288 | 2018.12.19–2019.1.7 | 2019.01.08 | 2019.1.8 |
| PPSSPP-PSP emulator (open) | Action | 1,100 | 2018.12.6–2019.3.17 | 1.8.0 | 2019.3.18 |
| AnySoftKeyb oard (open) | Tools | 59 | 2018.12.12–2019.3.21 | 2019.03.22 | 2019.03.22 |
| Tutanota (open) | Communication | 66 | 2018.12.22–2019.4.4 | 3.50.11 | 2019.4.25 |

the developer when maintaining and updating an app. Table 4 shows several changelogs of PPSSPP-PSP emulator under version 1.8.0. As we can see that the changelog records include bugs fixing (e.g., Graphics fixes), or new feature added (e.g., Allow putting PSP storage on custom paths like SD cards), etc. Although the changelogs may not cover all the modifications to the releases, they represent a lower bound and the prominent part of the changes [12]. Hence, It is suitable for validating the users' significant intentions detected by DSISP.

**Table 4.** The Changelog of PPSSPP-PSP emulator under V1.8.0

What's New in V 1.8.0:
* Speed improvements in EDF2, FF4
* Graphics fixes in a number of games (lighting, missing geometry, etc)
* Change default Backend to OpenGL (Vulkan still recommended)
* Fix control issue in Sonic Rivals and Rock Band
* Allow putting PSP storage on custom paths like SD cards
...

**Performance Metrics.** We employ the following three performance metrics to verify the effectiveness of DSISP. The $Precision_{SI}$ indicates the precision of detecting users' significant intentions. $Recall_{SI}$ indicates whether the detected significant intentions reflect the changes mentioned in the changelogs. $F_{hybrid}$ balances between $Precision_{SI}$ and $Recall_{SI}$.

$$\begin{aligned} Precision_{SI} &= \frac{S(C \cap SI)}{S(C)}, \; Recall_{SI} = \frac{S(C \cap SI)}{S(SI)} \\ F_{hybrid} &= 2 \times \frac{Precision_{SI} \times Recall_{SI}}{Precision_{SI} + Recall_{SI}} \end{aligned} \tag{6}$$

$S(C)$ represents the changelog records, $S(SI)$ is the users' significant intentions detected by DSISP, and $S(C \cap SI)$ represents the number of detected significant intentions which mentioned in changelogs. During our evaluation, we experimentally set the parameters of topic $k = 10$ by empirically, $v_t = 5$. More other $v_t$ values will be discussed in Sect. 4.2.

## 4.2 User Significant Intention Detection Result

Table 5 reports the results of $Precision_{SI}$, $Recall_{SI}$, and $F_{hybrid}$ achieved by DSISP. We can observe that DSISP has obtained a very high $Precision_{SI}$ with an average value of 0.962, while its $Recall_{SI}$ reaches 0.629 ($F_{hybrid} = 0.755$). For the closed-source apps, the $Precision_{SI}$ has reached 1.0, indicating that the users' significant intentions detected by DSISP cover all changelog information, i.e., the users' preferences in the sentiment valley are genuinely reflect the users' significant intentions.

Furthermore, we also found that except for YouTube Music ($Recall_{SI} = 0.313$) and Uber ($Recall_{SI} = 0.398$) app, the $Recall_{SI}$ of other apps is higher than 0.7. We manually analyzed the reviews of YouTube Music and Uber app, and found that this is due to the fact that there are much more reviews for popular apps under the sentiment valley. As a result, the users' significant intentions mined by DSISP not only cover the changelogs but also include some other modification information which not mentioned in changlogs. Therefore, it led to a lower $Recall_{SI}$. For open-source apps, due to the more frequent modifies and updates by developers, the modify records contained in changlog are also more, so the $Recall_{SI}$ value is higher than the closed-source apps as a whole, which indirectly proves that DSISP can efficiently mine users' significant intentions.

In addition, we also analyzed the effect of different $v_t$ on the efficiency of DSISP. $v_t$ indicates the size of time slice during the sentiment valleys are mined. Table 5 shows the results of DSISP when $v_t = 5$, 10 and 15, we can observe that with $v_t$ increases, the $Precision_{SI}$, $Recall_{SI}$ and $F_{hybrid}$ are showing diversified changes, such as increasing, decreasing, or unchanged, this indicates that different time granularities will have a direct impact on mining users' significant intentions by DSISP. The average optimal $Precision_{SI}$ (i.e., 0.962) and $F_{hybrid}$ (i.e., 0.755) are achieved while $v_t = 5$. This is also the value that we set in our experiment for detecting users' significant intentions mentioned above. More over, the developers can also dynamically set other $v_t$ values as needed.

**Table 5.** $Precision_{SI}$, $Recall_{SI}$, and $F_{hybrid}$ achieved by DSISP

| AppName | $v_t = 5$ | | | $v_t = 10$ | | | $v_t = 15$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | $Precision_{SI}$ | $Recall_{SI}$ | $F_{hybrid}$ | $Precision_{SI}$ | $Recall_{SI}$ | $F_{hybrid}$ | $Precision_{SI}$ | $Recall_{SI}$ | $F_{hybrid}$ |
| YouTube Music | 1.000 | 0.313 | 0.606 | 1.000 | 0.333 | 0.500 | 1.000 | 0.338 | 0.506 |
| Uber | 1.000 | 0.398 | 0.569 | 1.000 | 0.398 | 0.569 | 1.000 | 0.398 | 0.569 |
| Facebook | 1.000 | 0.717 | 0.835 | 1.000 | 0.717 | 0.835 | 1.000 | 0.717 | 0.835 |
| PPSSPP-PSP emulator (open) | 1.000 | 0.710 | 0.830 | 1.000 | 0.761 | 0.864 | 1.000 | 0.761 | 0.864 |
| AnySoftKey-board (open) | 1.000 | 0.785 | 0.880 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 | 0.750 |
| Tutanota (open) | 0.772 | 0.850 | 0.809 | 0.409 | 0.818 | 0.545 | 0.272 | 0.857 | 0.413 |
| Average | 0.962 | 0.629 | 0.755 | 0.859 | 0.629 | 0.677 | 0.837 | 0.637 | 0.656 |

Efficient detection performance can provide reliable suggestions for developers to update and maintain their apps in future.

## 5 Conclusion and Future Work

Timely and effectively detecting users' sentiment and preferences is crucial to capturing users' significant intentions, which is paramount for app developers and app vendors in mobile app maintenance and evolution. In this paper, we proposed DSISP, a framework for automatically detecting users' significant intentions from users' reviews. DSISP produces two summaries at different granularity levels about app reviews. These summaries can help app developers to analyze and quantify users' intentions about individual app features and to use this information to identify new requirements or to plan future releases. Moreover, DSISP can keep track of users' up-to-date sentiment and preferences and analyze how these preference features are temporally correlated with users' intentions. The experimental results show that DSISP can effectively and efficiently detect users' significant intentions, with a precision of 0.962 and a $F_{hybrid}$ of 0.755 on average.

In the future, we will leverage multi-dimensional user feedback information to enhance DSISP, such as email records between users and developers, app reviews on social media, etc. We will also employ the app issues and commits to mining whether users' feedback bugs can be reflected at the source code level with the aim to help app developer maintain and improve their apps.

**Acknowledgements.** This work is supported by the National Key R&D Program of China grant No.2017YFB1401201, the National Natural Science Key Foundation of China grant No.61832014 and the Shenzhen Science and Technology Foundation (JCYJ20170816093943197).

## References

1. Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C.A., Canfora, G., Gall, H.C.: How can i improve my app? Classifying user reviews for software maintenance and evolution, pp. 281–290 (2015)
2. Dąbrowski, J., Letier, E., Perini, A., Susi, A.: Finding and analyzing app reviews related to specific features: a research preview. In: Knauss, E., Goedicke, M. (eds.) REFSQ 2019. LNCS, vol. 11412, pp. 183–189. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-15538-4_14
3. Di Sorbo, A., et al.: What would users change in my app? Summarizing app reviews for recommending software changes, pp. 499–510 (2016)
4. Palomba, F., et al.: Crowdsourcing user reviews to support the evolution of mobile apps. J. Syst. Softw. **137**, 143–162 (2018)
5. Liu, Y., Liu, L., Liu, H., Wang, X.: Analyzing reviews guided by app descriptions for the software development and evolution. J. Softw. Evol. Process. **30**(12) (2018)
6. Messaoud, M.B., Jenhani, I., Jemaa, N.B., Mkaouer, M.W.: A multi-label active learning approach for mobile app user review classification. In: Douligeris, C., Karagiannis, D., Apostolou, D. (eds.) KSEM 2019. LNCS (LNAI), vol. 11775, pp. 805–816. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29551-6_71

7. Villarroel, L., Bavota, G., Russo, B., Oliveto, R., Di Penta, M.: Release planning of mobile apps based on user reviews, pp. 14–24 (2016)
8. Palomba, F., et al.: Recommending and localizing change requests for mobile apps based on user reviews, pp. 106–117 (2017)
9. Zhou, Y., Su, Y., Chen, T., Huang, Z., Gall, H.C., Panichella, S.: User review-based change file localization for mobile applications. IEEE Trans. Softw. Eng., 1 (2020)
10. Di Sorbo, A., Panichella, S., Alexandru, C.V., Visaggio, C.A., Canfora, G.: SURF: summarizer of user reviews feedback, pp. 55–58 (2017)
11. Huang, Q., Xia, X., Lo, D., Murphy, G.C. : Automating intention mining. IEEE Trans. Softw. Eng., 1 (2018)
12. King, I.: Online app review analysis for identifying emerging issues. In: The 40th International Conference (2018)
13. Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., Kappas, A.: Sentiment strength detection in short informal text. J. Assoc. Inf. Sci. Technol. **61**, 2544–2558 (2010)
14. Cohen, K.B., Dolbey, A.: Foundations of statistical natural language processing. Language **78**(3), 599 (2002)
15. Zhao, W.X., et al.: Comparing Twitter and traditional media using topic models. In: Clough, P., et al. (eds.) ECIR 2011. LNCS, vol. 6611, pp. 338–349. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20161-5_34
16. Pagano, D., Maalej, W.: User feedback in the AppStore: an empirical study, pp. 125–134 (2013)
17. Harman, M., Jia, Y., Zhang, Y.: App store mining and analysis: MSR for app stores, pp. 108–111 (2012)
18. Chen, N., Lin, J., Hoi, S.C.H., Xiao, X., Zhang, B.: AR-miner: mining informative reviews for developers from mobile app marketplace, pp. 767–778 (2014)
19. Guzman, E., Elhalaby, M., Bruegge, B.: Ensemble methods for app review classification: an approach for software evolution (N), pp. 771–776 (2015)
20. Vu, P.M., Pham, H.V., Nguyen, T.T.: Mining user opinions in mobile app reviews: a keyword-based approach (2015). arXiv: Information
21. Kucuktunc, O., Cambazoglu, B.B., Weber, I., Ferhatosmanoglu, H.: A large-scale sentiment analysis for yahoo! Answers, pp. 633–642 (2012)
22. Ma, Y., Chen, G., Wei, Q.: Finding users preferences from large-scale online reviews for personalized recommendation. Electron. Commer. Res. **17**(1), 3–29 (2016). https://doi.org/10.1007/s10660-016-9240-9
23. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python: Analyzing Text with The Natural Language Toolkit. O'Reilly Media, Inc., Sebastopol (2009)
24. Smadja, F.: Retrieving collocations from text: Xtract. Comput. Linguist. **19**(1), 143–177 (1993)