The Journal of Systems & Software 184 (2022) 111118

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Sharing runtime permission issues for developers based on similar-app review mining☆

Hongcan Gao [a,1], Chenkai Guo [a,1], Guangdong Bai [c], Dengrong Huang [a], Zhen He [b], Yanfeng Wu [b], Jing Xu [b,*]

[a] College of Computer Science, Nankai University, Tianjin 300350, China
[b] College of Artificial Intelligence, Nankai University, Tianjin 300350, China
[c] School of Information Technology and Electrical Engineering, The University of Queensland, St Lucia, QLD 4072, Australia

ARTICLE INFO

Article history:
Received 17 July 2020
Received in revised form 6 September 2021
Accepted 26 September 2021
Available online 12 October 2021

Keywords:
Android
Runtime permission
Permission-Related Issues (PRIS)
User reviews
Machine learning

ABSTRACT

The Android operating system introduces an ask-on-first-use permission policy after 6.0 version to regulate access to user data, which raises Permission-Related Issues (PRIS for short). Relevant research has been conducted to identify the PRIS through investigating users' opinions towards runtime permissions. These efforts mainly focus on helping users understand and be aware of permissions, but neglect to assist developers in discovering permission requirements. In this paper, we propose a novel framework named PRISharer, which mines potential permission issues from the reviews of similar apps to assist developers in discovering possible permission requirements at runtime. PRISharer first builds a deep fine-grained classifier to identify similar apps, and then employs sentiment analysis based keywords extraction to mine permission-related reviews from similar apps' reviews. Finally, the <category, permission, issues> mappings based on a multi-label learning method are generated to provide a PRIS profile for developers. The results of comparative experiments on more than 12 million reviews of 17,741 Android apps demonstrate that PRISharer achieves (i) superior performance in terms of F1-score for PRIS analysis, with an average improvement of 24.4%, (ii) the best recall (89.3%) in extracting permission-related reviews and (iii) 82.4% positive responses by expert developers, through which the effectiveness of PRISharer is well verified.

1. Introduction

The Android operating system enforces a permission-based mechanism to control third-party apps' access to various sensitive resources. It plays a significant role in protecting users' privacy and security. Starting from version 6.0 (API level 23), Android introduced a dynamic permission model, i.e., ask-on-first-use (AOFU), where apps can request for *"dangerou"* permissions (e.g., LOCATION and CONTACTS) at runtime, while the *"normal"* permissions (e.g., BLUETOOTH and VIBRATE) are granted automatically. Specifically, the *"dangerous"* permissions are divided into nine groups, each of which contains different numbers (1–7) of permissions.

Prior research (Au et al., 2012; Vidas et al., 2011) has shown that developers do not always correctly regulate permissions. For example, Au et al. (2012) state that "developers themselves make errors in specifying permission"; Vidas et al. (2011) claim that "the install time permission requirements are determined by the developer and may not accurately reflect permissions that the application actually requires for proper operation". Such problem leads to the increasing permission related issues (PRIS), such as requesting unnecessary permissions and missing necessary permissions. These issues may affect user experiences and cause users to uninstall the app. Therefore, how to mine the effective PRIS feedback from users, and support developers to discover permission requirements during the software development process is of great significance. Although the AOFU model provides greater flexibility in granting permissions, it imposes more burden of permission regulation, such as the frequency and time of permission requests.

Regarding Android permission system, the research literature on PRIS is limited (Alepis and Patsakis, 2019; Scoccia et al., 2019, 2018), especially for runtime permissions. Existing research provides an understanding of PRIS from different perspectives. Scoccia et al. (2018) conducted a large-scale study to investigate how app users perceive the Android permission system based on review mining, and provided several insights into future improvement on permission system. Different from analyzing users' reviews, Alepis and Patsakis (2019) presented a thorough

☆ Editor: Neil Ernst.

* Corresponding author.
E-mail address: xujing@nankai.edu.cn (J. Xu).

[1] Equal contribution.

https://doi.org/10.1016/j.jss.2021.111118
0164-1212/

analysis to unravel several permission issues at the API level, which includes transformation attacks, usage of other application's resources, privilege escalation, etc. Furthermore, instead of identifying PRIS, Scoccia et al. (2019) also focused on analyzing the process of developers by creating and fixing PRIS of four types: over-permission, under-permission, missing checks, and multiple requests in proximity.

Existing studies may be limited when assisting developers in discovering permission requirements in practice, since they ignore the relationship between the PRIS and the app under development. That is, it is imperative for the developer to identify the possible specific PRIS related to the app under development, which can be provided as valuable guidance for better PRIS management. To achieve this goal, we make the first attempt to explore mining PRIS from apps similar to the app under development, bringing at least three-fold potential benefits. First, app reviews, serving as important feedback for developers to observe user experience and discover various PRIS issues, have become increasingly prevalent in software development. Second, PRIS varies from category to category. For example, it is normal for a calling app to request for contact permissions, while it will be regarded as a suspicious permission request for a camera app. Thus, the PRIS from similar apps benefits developers to discover the possible problems of the current app. Third, by mining a set of similar apps, more potential permission information can be provided for developers, such as a similar-app list and various permission usages, which is beneficial for further software improvement. Based on the above observations, sharing PRIS from a set of similar apps' reviews is an effective way to assist developers with software improvement.

In this paper, we propose a novel framework PRISharer (**P**ermission-**R**elated **I**ssues **Sharer**) that shares runtime permission issues for developers based on similar-app reviews mining. Different from enhancing the understanding and awareness of PRIS for users, PRISharer focuses on the challenge that developers need to face during the app development process. More specifically, given a current app κ, PRISharer first leverages bidirectional long short-term memory (BiLSTM) (Graves and Schmidhuber, 2005) and DeepFM (Guo et al., 2017) to learn the similarity between apps from their meta-information, i.e., titles, descriptions, and policies, thus detecting a set of apps that similar to κ. Then PRISharer extracts potential permission-related sentences from candidate similar apps' reviews based on proposed multi-layer keywords. Afterwards, a model pre-trained with support vector machine (SVM) is used to handle the multi-label classification problems, and generate a <category, permission, issues> mapping for each similar-app group. Additionally, PRISharer provides a radar chart to visualize the summarization results. Instead of only providing potential permission problems, PRISharer also presents a detailed comprehension including a fine-grained category, top similar apps, and corresponding permission usages for developers.

We evaluate PRISharer's performance on a large dataset consisting of more than 12 million reviews from 17,741 apps in the Google Play store. First, we evaluate the effectiveness of PRISharer in identifying PRIS compared with existing state-of-the-art approaches, achieving a higher F1-score of 0.95. Second, we illustrate the performance of similar-app identification based on different feature combinations, and thus observe to what extent the features can help with the effectiveness. Third, we conduct a comparative study on permission-related review sentence extraction. The experimental results show that PRISharer based on multi-layer keywords can obtain more sentences and achieve better performance, with a recall of 0.893. Besides, we also push our review issues to the app developers, and achieves 82.4% positive responses, further indicating our analysis framework is effective.

Our contributions in this paper can be summarized as follows:

- We propose a novel framework (PRISharer) to overcome the gap between the PRIS and the app under development, with a focus on assisting developers to discover permission requirement for further improvement.
- We introduce techniques of BiLSTM and FM into similar-app identification based on various metadata features, so as to build a supervised app classification machine.
- We construct multi-layer permission keywords and leverage various keywords-based strategies to extract permission-related review sentences.
- The experiments show the effectiveness and usefulness of PRISharer in permission-related sentence extraction, similar-app identification, and PRIS sharing.

The rest parts of this paper are structured as follows. Section 2 introduces the proposed PRISharer framework; Section 3 presents the similar-app identification, followed by the detailed PRIS analysis in Section 4. Section 5 and Section 6 describe the experimental setup and results, respectively; Section 7 introduces the related work; Section 8 presents the threats to validity; Section 9 concludes our work.

2. PRISharer framework

Fig. 1 shows the overall architecture of PRISharer, which consists of three main phases: similar-app identification, PRIS mining, and visualization. Given a target app App_k and a corpus of candidate apps η (arrow ①) shown in Fig. 1), PRISharer first builds a fine-grained classification model to identify the apps in η that are similar to App_k. In this step, the input is the multi-metadata of each app pair (App_k, App_i) (arrow ②), including descriptions, titles and policies, and the output is the similarity score of each pair (arrow ③). As a result, a ranking app list based on similarity prediction is obtained (arrow ④). In the second phase, we aim to mine PRIS from a large corpus of similar apps' reviews. To this end, we collect the reviews from similar apps and input them into the PRIS mining module (arrow ⑤). In this step, we first carry out pre-processing and sentiment analysis to refine these reviews (arrow ⑥) and arrow ⑦). Then, diverse sentence extraction strategies are leveraged to extract the PRIS sentences based on the proposed multi-layer keywords (arrow ⑧). After that, these sentences are continuously fed into a trained multi-label classifier to generate PRIS (arrow ⑨). In the last phase, a visual mapping <category, permission, issues> is systematically analyzed and shared for developers (arrow ⑩).

3. Similar-app identification

Although the coarse category provided by Google play can be used to describe an app's genre, it is not effective to identify PRIS among the apps with the same coarse category. For example, the apps `Free WiFi App` and `Alarm Clock` are classified into the same coarse category `Tools`, whereas they have different permission requirements. Hence, we take a more fine-grained category into our work. Considering both of the two types of apps, we give the following definitions:

Level I apps: the apps with a coarse-grained category, such as `Business`, `Finance`, and `Social`.

Level II apps: the apps with a fine-grained category, such as `Light Up the Night` and `Lock Down Your Phone`, which are the fine categories included in the coarse category `Tools`.

Specifically, we observe that although the number of available apps in the Google Play Store in 2019 is more than 2.6 million,[2]

[2] https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

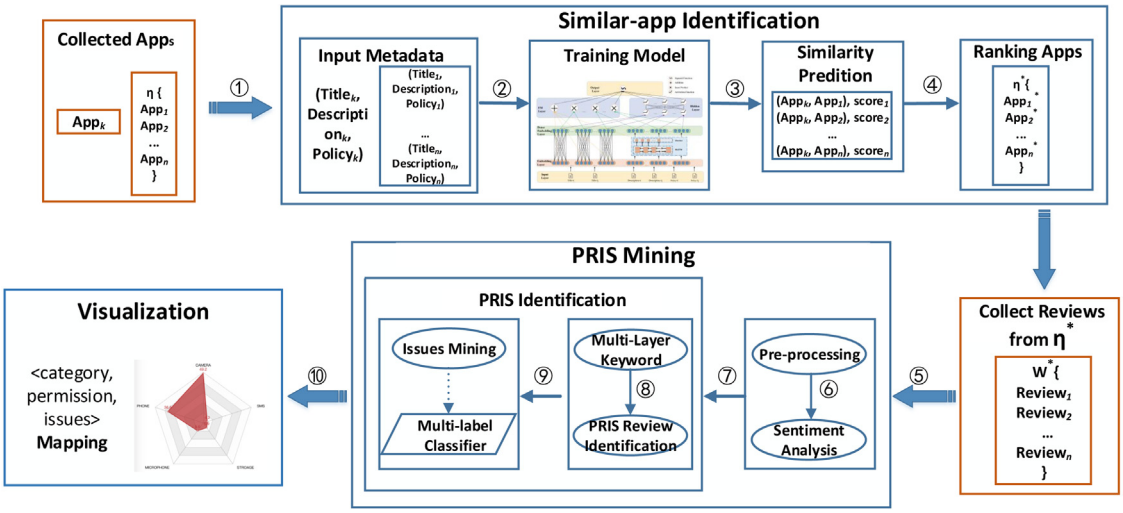


Fig. 1. PRISharer framework.

there are only 2,329 apps (see details in Section 5.2.1) that are assigned with a fine-grained category,[3] which can be used to build the training dataset for further classification. Moreover, the metadata information of the collected apps with fine-grained categories is available in GitHub.[4] Obviously, the apps with "fine-grained" categories are insufficient to mine PRIS, so we use typical intelligent techniques to find similar apps.

In the first step, we seek to identify apps similar to the current app among candidate apps. Inspired by deep learning techniques of BiLSTM and DeepFM, we propose an automatic similar-app classifier based on apps' various components. Fig. 2 is a detailed presentation of the similar-app classification model in Fig. 1. Given a pair of apps (App_k, App_η), the input of the model is the metadata combinations of two apps, i.e., titles ($Title_k$, $Title_\eta$), descriptions ($Description_k$, $Description_\eta$) and policies ($Policy_k$, $Policy_\eta$). In detail, we first adopt word embedding to compress the raw features into a low dimensional representation (Embedding Layer). Then, the long text fields (descriptions and policies) are further fed into an attention-based LSTM encoder for better feature presentations (Dense Embedding Layer). After that, an FM layer and a DNN hidden layer are used to capture the interactions among various features. In the last step, the similar-app model combines the results of FM and DNN to predict the ultimate results (Output Layer). Specifically, to build the training model, we propose a new labeling method to measure the similarity between a pair of apps, which can be treated as our training labels.

3.1. Labeling

In this subsection, we propose a new labeling method to measure the similarity between **Level I** and **Level II** apps. Our idea is based on the observation that apps classified into the same fine-grained category have greater similarity than apps in the different fine-grained categories. For example, the coarse category `Tools` (**Level I**) includes several fine categories (**Level II**, e.g., `Flashing` and `Screen`). Clearly, a flashing-oriented app is more similar to the apps in the same fine-grained category rather than the apps in `Screen`. Therefore, based on two levels of apps, there are totally three different similarity distributions between two apps: the apps in different **Level I**, the apps in the same **Level I** but

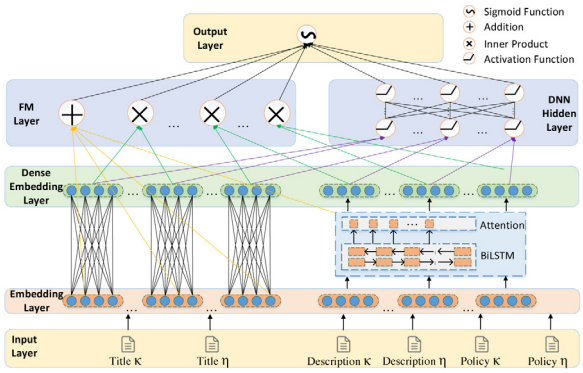


Fig. 2. Similar-app classifier.

different **Level II**, and the apps in the same **Level II**. These three types of app pairs are labeled with a similarity score of 0.25, 0.50, and 0.75 respectively, and thus can be regarded as training labels for similarity learning. As a result, a large corpus of labels are generated in this step, and we select 1,272, 1,535, and 2,844 app pairs as our training sets.

3.2. Embedding layer

To capture the similarity between apps, we take various textual fields, i.e., titles, descriptions, and privacy policies, into consideration. Among them, an app's description offers the most information to identify similar apps (Liu et al., 2018b), by which we can mine different functionalities of an app (Gao et al., 2019). Apart from that, the title and privacy policy also supply useful information for describing apps' behaviors (Yu et al., 2017). For the natural languages provided by developers, our goal is to compress these textual features into vectors as well as keep the rich contextual semantics. Therefore, we apply pre-processing techniques to handle the collected raw natural languages. In detail, we first split each review into several sentences with some common separators. Second, similar to prior works (Gao et al., 2019; Tao et al., 2020), PRISharer only focuses on English texts, thus we employ *langid* tool (Lui and Baldwin, 2012) to detect the most likely languages of reviews, and then discard the non-English reviews. Afterwards, we eliminate the regular English stop words, punctuation, and meaningless symbols, because these characters

[3] https://play.google.com/store/apps.
[4] https://github.com/GHCan/PRISharer.

frequently occur but unlikely assist in analyzing opinions. More importantly, since there are many variations of the same word in English language expressions, e.g., "eating", "eaten", and "ate" are the variations of "eat", we employ NLTK package (Loper and Bird, 2002) to perform a word stemming task. After the pre-processing, we adopt word embedding (Levy and Goldberg, 2014) to compress the raw feature input into a low dimensional vector representation.

We use Word2Vec (Mikolov et al., 2013b), a neural probabilistic language model, to implement the embedding task. Word2Vec is widely used for extracting low-dimensional vector representations of words in natural language processing, including CBOW model (Mikolov et al., 2013a) and Skip-gram model (Goldberg and Levy, 2014). To effectively identify the similarity around the key words of textual features, we adopt Skip-gram model to generate dense real-value vectors with contextual-semantic information by training the input corpus. As a result, the outputs of this step are denoted as:

$$O(i) = \mathbf{W}_i \tag{1}$$

where $\mathbf{W}_i \in \eta^{d_m \times n}$ is a matrix to be learned; d_m is the dimension of each vector; n is the size of the fields. Afterwards, the original input vectors with different length can be transformed into embedding features of the same size.

3.3. BiLSTM-based encoder

PRISharer aims to handle cover both short text fields (titles) and long text fields (descriptions and policies). After projecting these two fields to a dense vector, we employ a LSTM-based encoder Hochreiter and Schmidhuber (1997), to pre-train long text fields for better feature presentations. As an improved recurrent neural network, LSTM has been proven an effective model for handling long-range dependencies in previous studies Sak et al. (2014), Xingjian et al. (2015), Sundermeyer et al. (2012). It introduces three control gates, i.e., input gate, forget gate and output gate, and can be represented as:

$$\begin{aligned} i_v &= \sigma\left(W_i\left[x_v, h_{v-1}\right] + b_i\right) \\ f_v &= \sigma\left(W_f\left[x_v, h_{v-1}\right] + b_f\right) \\ o_v &= \sigma\left(W_o\left[x_v, h_{v-1}\right] + b_o\right) \\ c_v &= f_v \odot c_{v-1} + i_v \odot g_v \\ g_v &= \tanh\left(W_c\left[x_v, h_{v-1}\right] + b_c\right) \\ h_v &= \vec{o}_v \odot \tanh\left(c_v\right) \end{aligned} \tag{2}$$

where c_v denotes the cell vectors; W_i, W_f, W_o, and W_c are the weight matrix of input gate i_v, forget gatef_v, output gate o_v, and modulate gate g_v, respectively. h_v denotes the hidden state at the v-th view, while $\odot$ is the element-wise multiplication and $b \in R$ is the bias vector of LSTM cell.

Specially, the BiLSTM considers the dependency among inputs of both two directions, i.e., forward and backward directions, to grasp the past features and future features, respectively. For each time step v, the vectors of both forward $\vec{h}_v$ and backward $\overleftarrow{h}_v$ directions are concatenated as the final output H_v:

$$H_v = \overleftarrow{h_v} || \overrightarrow{h_v} \quad h_v \in R^{2d} \tag{3}$$

where $\|$ is the operator of concatenating two-direction context representations, and d is the dimension of the LSTM cells in the hidden state. Based on these two independent LSTMs, both the left and right contextual representations are preserved .

The BiLSTM encoding layer introduces the features in a specific text with the same weights. However, due to the structural complexity of the natural language, not all words in the textual fields (descriptions and policies) contribute equally to the vector representations. Hence, a level of word attention mechanism is introduced to distinguish the importance of informative words in each description and policy fragment, by which we can obtain a dense vector with attentive weighted words.

Practically, the hidden states h_t produced by BiLSTM layer is fed into a multi-layer perceptron (MLP) to obtain a hidden representation g_t, which can be computed using the following equation:

$$g_t = \tanh\left(W_g h_t + b_g\right) \tag{4}$$

where W_g is a weight matrix of the MLP corresponding to hidden state h_t, and b_g is a bias vector. Then a normalized attention weight a_t can be computed using the following softmax equation:

$$a_t = \frac{\exp\left(g_t^T g_w\right)}{\sum_t \exp\left(g_t^T g_w\right)} \tag{5}$$

where g_w is a word level context vector, which can be regarded as a high level representation to distinguish the importance of the words. Note that the bigger the a_t is, the more important the g_t is.

Finally, the attention-based hidden state representation is computed by a weighted sum of word annotation h_t:

$$I = \sum_t a_t h_t \tag{6}$$

3.4. DeepFM Compound

To capture the interactions among various features, i.e., short title vectors, long description and policy vectors, we leverage a widely used model Factorization Machines (FM) proposed by Rendle (2010), which is defined as follows:

$$y_{FM} = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{(i,j)\in\mathcal{R}_\mathcal{X}} \left\langle V_i, V_j \right\rangle x_i x_j \tag{7}$$

where $w_o \in \mathcal{R}$ is the global bias; w_i is the weight of the ith variable x_i (order-1 feature); V_i and V_j are the latent feature vectors with k dimension, and $V \in \mathbb{R}^{k\times m}$, where m is the number of features and k is a given parameter. The additional unit $\langle V_i, V_j \rangle$ is the pairwise (order-2 feature) interaction between x_i and x_j, where $\langle ., . \rangle$ is the inner product of two vectors. Here, $\mathcal{X} = \{(i,j)|0 < i \le m, 0 < j \le m, j > i, x_i \ne 0, x_j \ne 0\}$.

Next, we adopt a deep neutral network (DNN) to train the higher order relationships, which can be calculated as:

$$y_{DNN} = \sigma\left(W^l l + b^l\right) \tag{8}$$

where σ is the sigmoid activation function; W^l denotes the weight parameters; b^l is the bias item; l is computed using a ReLU activation function.

In the last step, our similar-app model combines the results of FM and DNN to predict the ultimate result.

$$\hat{y} = \sigma\left(y_{FM} + y_{DNN}\right) \tag{9}$$

Thus, for a target app κ, a ranking list η_s for similar apps based on the predicted similarity is obtained in this step.

4. PRIS mining and sharing

After identifying similar apps, the next step of PRISharer is to mine the permission-related issues among the reviews of these similar apps, as shown in Fig. 3. Given a large set of user reviews (arrow ①), the pre-processing step leverages Natural Language Processing (NLP) techniques to split the reviews into sentences, discard no-English reviews, modify words, etc. Thus the review sentences after pre-processing can be obtained (arrow ②). Second, the sentiment analysis step further divides review

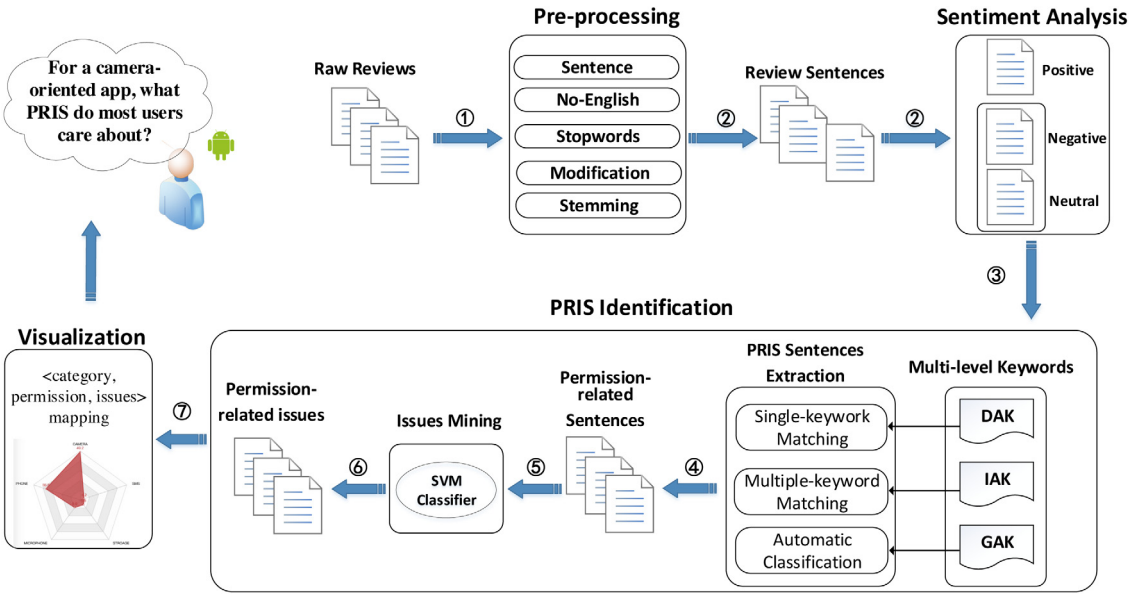


Fig. 3. PRIS mining and sharing.

sentences into positive, negative and neutral expressions, and only the "non-positive" sentences are reserved and input to the next PRIS identification (arrow ③). In this module, we first build a multi-level keyword including DAK, IAK and GAK for further sentence extraction. Then, based on the kinds of keywords, different sentence extraction strategies are employed to extract permission-related sentences (arrow ④). Next, these sentences are fed into a pre-trained classifier to generate permission-related issues for each permission group (arrow ⑤ and arrow ⑥). Finally, the systematical <category, permission, issues> mappings are provided with a radar chart to share the PRIS results (arrow ⑦) (see Fig. 3).

4.1. Pre-processing

Similar to textual fields processing, for each user review, we also first carry out a pre-processing step including sentence segmentation, non-English removal, word filtration, and stemming, to refine the collected raw natural languages (See details in Section 3.2). Besides, we further perform a word modification. Different from the metadata provides by developers in Google Play, user reviews usually contain some words that cannot be correctly recognized, e.g., abbreviations and misspelled words, which are hurdles for discrimination of reviews. Therefore, we leverage library PyEnchant to modify misspelled words, and replace the abbreviations with correct forms based on an abbreviation list collected by Liu et al. (2018b).

4.2. Sentiment analysis

Sentiment analysis determines whether the expressions indicate positive or negative opinions towards the app, which offers developers an intuitive and effective way to understand the users' feelings. Prior research (Ha and Wagner, 2013; Maalej and Nabil, 2015) has shown positive reviews cannot reflect security issues, and thus this work mainly focuses on the review expressions that convey user's negative and neutral sentiments.

To capture the sentiment information, we employ the VADER Sentiment Analyzer[5] in NLTK to classify the pre-processed sentences as positive, negative, or neutral. This analyzer covers a list of lexicons related to text expressions along with their sentiment measurements, which has been widely used in NLP tasks. Similar to the previous work (Tao et al., 2020), we discard the positive expressions using traditional threshold values. Moreover, prior works (Khalid et al., 2014; Pagano and Maalej, 2013) demonstrate the low-rated reviews suggest a lower user satisfaction, and most of them include requirement-related problems, e.g., functionality bugs or security issues. Furthermore, we also consider the reviews with lower-rating that are negatively correlated with the user satisfaction. We observe that although some expressions are labeled as neutral, they indicated the negative feedback. For instance, a raw review "This app is leaking GPS location" with a score of 1.0 expresses a privacy leakage opinion, while the expression is identified as neutral information. Based on the insights, we can conclude that the neutral review with a lower rating is more likely to reveal a negative opinion, and thus is helpful to extract the permission-related issues. To identify such sentences, for the reviews with neutral sentiments, we inspect corresponding rating scores by users, and retain the reviews with scores below 3.5.

4.3. PRIS identification

The purpose of this step is to mine permission-related issues from review sentences, including the following three stages:

- Multi-layer keyword extraction for further permission tracking;
- Permission-related sentence matching based on various keyword lists;
- A machine learning training model for permission-related issues mining.

4.3.1. Multi-layer keyword extraction

To extract the permission-related reviews, we build multi-layer keywords covering nouns and verbs, which include three different types of keyword patterns. For better understanding, we define them as Direct-association Keyword (DAK), Indirect-association Keyword (IAK), and Generated-association Keyword (GAK), respectively. As shown in Table 1, DAK refers to the keyword that has strong and direct associations with permission issues, e.g., *"permission"*, while IAK represents the keyword that has relative weak and indirect associations with permission issues, e.g., *"camera"*, *"contacts"*, *"location"*. To obtain DAK and

[5] https://pypi.org/project/vaderSentiment/.

IAK, we first select several permission-related keywords from the literature (Cen et al., 2014; Tao et al., 2020; Pletea et al., 2014) which provides security-related keywords, and then manually collect more keywords. Note that all selected keywords produced at this stage will be classified into DAK or IAK, and be tagged as a noun or verb according to the respective syntactic category called part of speech. We use part-of-speech (POS) tagging (Das and Petrov, 2011) in NLTK to label the word in sentences with noun or verb in our work.

During the manual selection, two authors of this paper are involved in distinguishing multi-layer keyword tasks. Both of them have more than two years of research experience in Android permission. Specifically, they examined the apps' metadata information (e.g., permission-related explanations in descriptions and policies) and user reviews to make a comprehensive understanding of permission expressions. A keyword is only selected when both two authors agree on the necessity of the word to potential sentence extraction.

Different from DAK and IAK, GAK is constructed by leveraging word co-occurrence and synonym algorithm. The central collection idea behind is that if a word occurs frequently with a given word, it may be regarded as a "neighborhood" of the given word. Clearly, it is useful to identify such keywords for further opinion mining. Similar to co-occurrence words, we also take synonyms into consideration. Table 1 lists the overall keywords distributed in three layers.

4.3.2. Permission-related review identification

The next stage is to identify permission-related sentences among candidate review sentences. To this end, we perform the following three extraction strategies according to the layers of keywords.

For DAK layer, we simply employ a keyword-based matching. As described above, the keywords in DAK are most related to permission issues, which could be widely used to describe the users' opinions when a permission requests. Thus, if a sentence contains the word in DAK layer, it will be selected by PRISharer.

For IAK layer, we employ multiple-keyword pattern matching to retrieve sentences based on the candidate combinations between nouns and verbs, e.g., *"request for contacts"*, *"access to camera"*. This strategy is due to our observation that although single keyword in IAK represents an indirect association with permission issues, the combinations of nouns and verbs can effectively increase the association degree, and thus can be used to represent permission issues. For example, the word *"camera"* commonly occurs in the reviews of the photography-based app, while the phrase *"apply the camera"* reveals a behavior related to permission. Hence, we group nouns and verbs to generate various multiple-keyword patterns, and then use multiple pattern matching algorithms to obtain more samples. Similarly, since GAK is generated by IAK, we perform the same strategy for GAK.

Different from keyword-based matching, we also explore a larger set of potential sentences based on a supervised machine. In detail, we first manually analyzed a sample of the review sentences, and classified them into a taxonomy by analyzing whether these sentences are related to permission or not (See details in Section 5.3). Then, by employing the pre-trained classifier, we can distinguish more permission-related sentences from reserved samples. Since the original sentences are hard to be directly used for modeling similarity, we first convert the given text instances into vector representations. Afterwards, learning and NLP techniques are used to capture more candidate datasets. In this step, we use SVM to achieve this goal due to its superior performance (discussed in Section 6.3).

4.3.3. Issues mining

This stage aims to mine issues from extracted review sentences. Inspired by the categories of permission opinions proposed by Scoccia et al. (2018), we first labeled a set of sentences with their respective category to build a ground truth, and then put them into a supervised machine for further automatic classification.

During the labeling task, two authors in this paper were assigned the same samples. They first discussed the same issues criteria to reduce bias before labeling, then analyzed and revised the disagreements after the manual step. A sentence was determined only two authors agree on the issues of the sentences. As a result, we reclassified the original categories, including two main aspects. The first aspect is we discarded two categories described in Scoccia et al. (2018), i.e., Permission Praise (PP) and Minimal Permissions (MP), both of which are related to positive opinions. Additionally, we added a new category, i.e., Privacy Intrusion (PI), based on our observations. For example, an illustrative case: *"Changes permissions after installation, Not nice."* expresses the permission leak issue. As a result, we finally determine a list containing nine permission-related issues. Table 2 shows these categories with their respective examples. Note that a sentence can be assigned to more than one category, e.g., *"Too many permissions required, it crashes if you refuse them too"*, which is marked with TMP and FN.

Next, we trained an automatic classification model by learning the labeled sentences with specific categories. Generally, several alternative machine learning techniques, such as Support Vector Machine (SVM) and Logistic Regression (LR), can be considered in our model. Among them, we choose SVM algorithm to train our model, since it achieves fairly better performance compared with other algorithms, which will be discussed in Section 6.1. Thus, based on the trained classifier, we automatically classify the extracted sentences into corresponding clusters.

4.4. Visualization

In the last step, we generate systematical <category, permission, issues> mappings by summarizing the results generated from the above three phases. Inspired by the previous works (Chen et al., 2014; Tao et al., 2020), we also visualize the results with a radar chart for a better comparison between different attributes. As for a radar chart, the categorical variables will be assigned to the vertices of a polygon according to the category frequency labeled in sentences, by which developers can identify the dominant permission issues. In addition, more sharing results such as original reviews and the information related to similar apps will be presented in the radar. The visualization details will be discussed in Section 6.1.4.

5. Experimental setup

5.1. Research questions

To evaluate the performance of PRISharer, we design empirical studies to address the following research questions.

RQ1: How accurate is PRISharer in identifying PRIS compared to state-of-the-art approaches?

RQ2: What is the performance of the proposed similar-app model in classifying apps based on different combinations of meta-information features?

RQ3: Does PRISharer effectively extract permission-related review sentences based on multi-layer keywords?

RQ4: Are the issues shared via PRSharer actually useful for developers in the process of software development?

Table 1
Multi-layer keywords on PRIS with definitions.

Category	Definition	Noun	Verb
Layer I	DAK	Permission	–
Layer II	IAK	calendar, camera, contacts, location, microphone, phone, sensors, SMS, storage, message, call, audio, data,	access, read, write, receive, edit, send, grant, deny, ask for, steal, intrude, crash
Layer III	GAK	privacy, security, leak, confidentiality, breach, policy, over-authorization, risk, malware	request, demand, modify, invade, bypass, track, invoke, invasive, freeze, disable

Table 2
Details of nine categories of PRIS.

Dataset	Description	Example
PC	Permissions Complaint	"I didn't like that it asks permission to be able to access my contacts and be able to make calls."
TMP	Too Many Permissions	"Too many permissions demanded."
UP	Unclear Permissions	"Why would you need microphone permissions for this game?"
PB	Permission-related Bug	"You can't give the app permissions (android) due to screen overlay bug."
RPR	Repeated Permission Requests	"It keeps asking for location access and demanding I change screen overlay permissions [...]"
SP	Settings Permissions	"If you go to the app and remove permission for them then relaunch the app it asks for permission [...]"
BRT	Bad Request Timing	"Asked me to allow to make phone calls and to give permission to my photos [...]"
FU	Functionality Unavailable	"App forces you to give access to phone and contacts in order to use it."
PI	Privacy Intrusion	"Permissions are too intrusive."

5.2. Selected subjects

The selected subjects include app collection and review selection.

5.2.1. App collection

In our experiments, we first collected the relevant metadata of 19,118 **Level I** apps, covering 31 categories, from the Google Play store in June 2019. Each crawled entity contains a package name, a marked coarse category, a set of requested permissions, a long text description, a privacy policy, and the number of downloads, etc. Apart from that, we also collected the same information of the apps with a fine-grained category (**Level II** mentioned in Section 3.1), with the total amount of 2,630. Moreover, we discarded non-popular apps with less than 5,000 downloads from **Level I** apps (2,357/19,118 = 12.3%) and **Level II** apps (301/2,630 = 11.4%), respectively.

Afterwards, we removed the duplicate apps that appeared in both of the two sets from **Level I**. As a result, we get 17,741 entities, where **Level I** with the number of 15,412 apps covering 31 coarse-grained categories, and **Level II** with 2,329 apps covering 118 fine-grained categories. After the process, we labeled the apps according to the measurements mentioned in Section 3.1.

5.2.2. Review selection

For each entity reserved in the above section, using an open-source crawler, we collected its user reviews based on the app's package name from July 2019 to September 2019. Each review subject includes a unique reviewer id, a long review text, the creation date, and an associated rating assigned by the user, i.e., a score ranging from one to five. Since Google Play store returns only a limited number of reviews for each request at a given time, we crawled the reviews of the collected apps for multiple times to obtain an extensive dataset.

However, the distribution of crawled reviews is unbalanced, due to the factors such as app genres, the popularity of apps, the app's active users, etc. For example, `Clash of Clans` is a social app with more than 30 million reviews, while some newly released apps (affiliated to the coarse category `Dating and Parenting`), have an average of fewer than 1,000 reviews. Here, we reserved up to 5,000 reviews for each app. After merging reviews and discarding duplicates, a total of 17,741 apps with 12,048,536 valid reviews are obtained. The overall statistics of our dataset can be concluded as Table 3.

Table 3
Dataset. Where Apps denote the total number of collected apps; #L1Apps denotes the number of the app with a coarse-grained category; #NL1 denotes the number of #L1Apps's categories; #L2Apps denotes the number of the app with a fine-grained category; #NL2 denotes the number of #L2Apps's categories.

Categories	Apps	#L1Apps	#NL1	#L2Apps	#NL2	Reviews
Overall	17,741	15,412	31	2,329	118	12,048,536

5.3. Ground truth

To build a fair ground truth, we manually annotated a collection of review sentences based on prior work (Scoccia et al., 2018), which selected 1,000 reviews from the potential permission-related samples and determined an informative label for each review. The reason why we select sentences from potential permission-related reviews is to balance the dataset. If we randomly label the sentences from all the reviews, there will be a large number of irrelevant samples, due to the high discard rate in user reviews, which makes the prediction an inefficient task. Moreover, since our work focuses on the evaluation at sentence level, we divided the original reviews into sentences. Afterwards, we manually marked each review sentence for subsequent sentence extraction and automatic classification, including the following two aspects:

On one hand, to measure the effectiveness of extracted permission sentences, we first discarded the positive opinions (annotated with a "+" sign), as discussed in Section 4.2. Then we split the reviews into sentences with separators including ".", "?", "…", etc., and filtered out invalid sentences. Afterwards, we labeled these available sentences to describe whether they are related to permission issues. In this step, we totally inspected a set of 2379 sentences out of 998 reviews (valid data 998/1,000) as our ground truth.

On the other hand, similar to permission-related sentences, we also build ground truth for permission-related issues to verify the performance of PRIS identification. As discussed in Section 4.3.3, after discarding two positive categories (PP and MP) and adding a new category (PI), nine categories are summarized in this work. In this step, we totally classified a set of 749 review sentences. The detailed statistics are concluded as Table 4.

Apart from that, due to the limitation of data amount and data distribution, the prediction result may suffer from the challenge of overfitting. To alleviate the challenge, on the one hand, we

Table 4
Overview of annotated sentences.

Dataset	Description	Number
TReviews#	Original reviews	998
TSentences#	Sentences out of reviews	3,019
NReviews#	Non-positive reviews	667
LSentences#	Labeled sentences	2,379
LPSentences#	Labeled permission-related sentences	749
Trainingset#	PRIS training dataset	11,826

prefer classical machine learning models rather than complicated learning model (e.g., neural network) with numerous parameters. On the other hand, typical up-sampling method (Chawla et al., 2002) is adopted in the pre-processing of data, through which the amount of practical training sentences is concretely augmented. In detail, we first select different sentences with diverse content around a target sentence. Then we can achieve a new vector by computing the average value of the selected sentences. At last, 11,826 samples (including 6,741 original samples and 5,085 augmented samples) are achieved for the practical model training. Note that the PRIS identification is a typical multi-label classification problem, where each issue category is treated as an independent binary classification task. Thus, the original training dataset contains a total of 6,741 ($749 * 9$) samples, as described in Section 4.3.3.

5.4. Evaluation metrics

In this part, we use three widely-used metrics, i.e., precision, recall, and F1-score, to evaluate the performance of PRISharer in permission-related issue identification task (RQ1) and permission-related sentence extraction task based on multi-layer keywords (RQ3), both of which can be considered as a supervised classification predication. For example, precision in RQ1 refers to the ratio of correctly identified permission-related samples by PRISharer to the total number of permission-related samples, while recall refers to the radio of identified permission-related samples by PRISharer to the total number of permission-related samples. Lastly, F1-score is a harmonic mean of precision and recall values to measure the overall prediction performance. Specifically, since RQ3 is a multi-label classification task, we also adopt micro-average to measure the overall classification performance.

Additionally, since the similarity prediction (RQ2) belongs to a regression task, we employ two well-known regression metrics, i.e., Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), to evaluate the similar-app prediction results of the proposed model, which can be defined as follows:

$$MAE = \frac{1}{M}\sum_{r,q}|\hat{R}_{r,q} - R_{r,q}| \tag{10}$$

$$RMSE = \sqrt{\frac{1}{M}\sum_{r,q}(\hat{R}_{r,q} - R_{r,q})^2} \tag{11}$$

where $R_{r,q}$ denotes the similarity between current app R and app q, with $q \in Q$; $\hat{R}_{r,q}$ represents the predicted similarity; M is the number of tested samples. A smaller MAE and RMSE indicates a better performance of predicted values.

5.5. Baselines

Since both RQ1 and RQ3 belong to the classification tasks, their main stages are feature transformation and classification. To validate the effectiveness of PRISharer in solving the classification problems, two types of numerical representation methods, TF–IDF and Word2Vec (Due to the limitation of data amount, we do not consider the sophisticated embedding methods, such as Bert (Devlin et al., 2018) and GPT Radford et al. (2019)), are adopted to covert the review sentences into numerical representation. Then, four types of machine learning methods, namely Support Vector Machines (SVM), LightGBM, Logistic Regression (LR), and Random Forest (RF) are chosen as classification baselines. Specially, LightGBM is a novel Gradient Boosting Decision Tree (GBDT) algorithm which mainly contains gradient-based one-side sampling (GOSS) and exclusive feature bundling (EFB).

By assembling the above representation methods and classification algorithms, we can build eight baseline approaches (2*4), including TF-IDF+SVM, TF-IDF+LightGBM, TF-IDF+LR, TF-IDF+RF, Word2Vec+SVM, Word2Vec+LightGBM, Word2Vec+LR, and Word2Vec+RF, respectively. Here, for example, TF–IDF+SVM refers to the reviews are represented by TF–IDF, and the classifier is implemented by SVM. Based on the comparative results, we adopt the best configuration to perform review sentences.

5.6. Evaluation setting

All the collected reviews are set as language corpus for our task, which are first preprocessed as described in Section 3.2. For TF–IDF, we count the frequency of words and reviews to compute the TF and IDF, respectively. The embedded vector can be represented by the product of TF and IDF. Noting that the TF–IDF here hardly belongs to a feature extraction method since it merely reflects the emerging frequency of the words. However, we can use it to transform the words into numerical representation, so as to feed the data into downstream classifier. The numerical representation is not a vector, but the sentences composed by words can be represented as a vector which is composed of the numerical values of words. For Word2Vec, we choose the Skip-Gram (Goldberg and Levy, 2014) to implement the embedding in practice. Given a targeted word in the corpus, the Skip-Gram models a conditional probability using a softmax function to predict the emerging of surrounding words. According to eight experiments described in Section 5.5 and their results, we found that the performance based on Word2Vec is overall better than that of TF–IDF. Therefore, the PRISharer adopts the Word2Vec as the embedding module and only the Word2Vec-based results are displayed when comparing with other methods.

To fairly estimate the performance of the learning models, we employ popular K-fold cross validation in the model evaluation. Specially, we set K to 10, considering the limited amount of samples in our collected datasets. As common, the dataset proportion for training, validating and testing are set as 8:1:1. The dataset is first split into 10 subsets, and one of them is used as testing set; one of the other sets is randomly selected as validating set; the remaining sets are treated as training sets. Then, conduct the training, validating and testing, and iterate this process until all the 10 possible combinations have been evaluated.

6. Experimental results

In this section, we present the experimental results to answer the proposed research questions in Section 5.1.

6.1. RQ1: The performance of PRISharer in identifying PRIS

6.1.1. Comparison with R-PRISharer

Table 5 summarizes the precision, recall, and F1-score of PRISharer using the four different classification models, i.e., SVM, LightGBM, LR, and RF, discussed in Section 5.5. To compare the performance of sentence-based analysis and that of review-based analysis, we also present a comparative study against the automatic classification that takes reviews as input (denoted by R-PRISharer). The results are summarized in Table 5.

Table 5
Performance comparison. Notice that the values in Table 5 are a micro-average measurement of nine categories displayed in Table 2.

Model	Model	Precision	Recall	F1-score
R-PRISharer	SVM	0.94	0.92	0.93
	LightGBM	0.67	0.57	0.62
	LR	0.91	0.84	0.87
	RF	0.96	0.88	0.92
PRISharer	**SVM**	**0.91**	**0.99**	**0.95**
	LightGBM	0.90	0.83	0.86
	LR	0.82	0.89	0.85
	RF	0.96	0.93	0.94

We can observe that PRISharer achieves the best performance in all metrics when using SVM, with the precision, recall, and F1-score of 0.91, 0.99, and 0.95, respectively (values in bold). Among the four algorithms, it has F1-score ranging from 0.85 to 0.95, and precision from 0.82 to 0.96. More specifically, SVM achieves a better balance between precision and recall than other approaches, such as RF (0.94) and LightGBM (0.86). However, the performance of LR is relatively worse, with an F1-score of 0.85. This may be because the linear classifier is limited in dealing with nonlinear features, and it is not flexible enough to recognize complex textual patterns. Note that in the process of experimental implementation, to solve the multi-label PRIS identification problem, we treat each issue category as an independent binary classification task.

Similar to PRISharer, R-PRISharer based on SVM demonstrates better performance (an average F1-score of 0.93) than the other three approaches, especially LightGBM (the lowest F1-score of 0.62). Among them, the RF-based model has a comparable performance with SVM, which is much better than LR and LightGBM. The results highlight the effectiveness of SVM in modeling review texts.

In most cases, R-PRISharer under-performs PRISharer in issue prediction. For a given algorithm, e.g., LightGBM, R-PRISharer achieves an F1-score of 0.62, obviously below the 0.86 reached by PRISharer. This indicates that comparing with review-based analysis, our sentence-based approach is able to identify more permission-related issues. The benefits of sentence analysis can be ascribed to the noise reduction in opinion expressions, which filters out the irrelevant descriptions about permission for better prediction. For example, one user points out: *"It does not require a lot of permissions. The sounds are especially relaxing and pleasant, and being able to adjust and mix them is great. The images are nice, too. I have tried many sleep apps, but this is the best."* Although this review contains five sentences and is marked with permission opinion, only the first sentence is related to permissions. Hence, it is necessary to split these reviews into various sentences for a more precise classification. Overall, the effectiveness of PRISharer in identifying PRIS among similar-app reviews is further confirmed by the fairly decent performance.

6.1.2. Comparison with state-of-the-art approaches

To evaluate the performance of PRISharer in identifying PRIS, we compare the proposed framework with state-of-the-art approaches. Since there are few previous works that use learning method to mine the PRIS, we implement the typical two state-of-the-art relevant works (i.e., Ngram (Scoccia et al., 2018) and AR-miner (Chen et al., 2014), although they do not directly target at the PRIS mining) and extend them through different selection of common parameters, prediction granularity and classifier. Eventually, 26 baselines participate in the performance comparison. Among them, the Ngram-based models adopt techniques of lemmatization, n-grams, bag-of-words and classifier configuration to classify the user reviews; while AR-miner based models leverage LDA (Latent Dirichlet Allocation) to mine the informative reviews. In detail, for Ngram-based methods, we set three different parameters (N = 1, 2, 3, denoted by Uni-gram, Bi-gram, and Tri-gram, respectively), and four downstream classifiers (SVM, LightGBM, LR and RF). In addition, we implement two levels of prediction granularity (sentence-level and review-level) for each baseline model, which are represented by *"S-"* and *"R-"*, respectively. Here, we choose the SVM-based PRISharer which achieves the highest prediction performance as the representative model for our work. Table 6 summarizes the comparative results of PRISharer with the baseline approaches.

Table 6
Comparison results of identifying PRIS.

Models	Precision	Recall	F1-score
S-Uni-gram+SVM	0.89	0.86	0.88
S-Uni-gram+LightGBM	0.64	0.83	0.70
S-Uni-gram+LR	0.88	0.85	0.86
S-Uni-gram+RF	0.88	0.87	0.87
S-Bi-gram+SVM	0.97	0.84	0.85
S-Bi-gram+LightGBM	0.82	0.79	0.81
S-Bi-gram+LR	0.61	0.31	0.39
S-Bi-gram+RF	0.80	0.86	0.83
S-Tri-gram+SVM	0.74	0.86	0.79
S-Tri-gram+LightGBM	0.88	0.83	0.85
S-Tri-gram+LR	0.29	0.34	0.23
S-Tri-gram+RF	0.66	0.84	0.70
R-Uni-gram+SVM	0.88	0.82	0.85
R-Uni-gram+LightGBM	0.66	0.76	0.69
R-Uni-gram+LR	0.77	0.66	0.70
R-Uni-gram+RF	0.82	0.81	0.81
R-Bi-gram+SVM	0.87	0.83	0.85
R-Bi-gram+LightGBM	0.87	0.77	0.81
R-Bi-gram+LR	0.32	0.31	0.26
R-Bi-gram+RF	0.87	0.73	0.79
R-Tri-gram+SVM	0.84	0.85	0.84
R-Tri-gram+LightGBM	0.83	0.80	0.80
R-Tri-gram+LR	0.21	0.28	0.21
R-Tri-gram+RF	0.80	0.83	0.80
S-AR-Miner	0.54	0.96	0.69
R-AR-Miner	0.66	0.88	0.76
PRISharer	0.91	0.99	0.95

From Table 6, PRISharer achieves significantly better performance in identifying PRIS in terms of F1-score, with an average improvement of 24.4%. For Ngram-based approaches, the S-Unigram based SVM achieves a higher F1-score (0.88). Given the same downstream classifier, the performance of S-Ngram performs better than R-Ngram in most cases, indicating the advantages of sentence-level analysis. Moreover, we can also observe that SVM-based models achieve the best F1-score under the same N parameter. Differently, the performance of LR-based models is unstable, varying from 0.21 to 0.86.

For AR-Miner-based models, the achieved prediction performance is relatively poorer than most of the Ngram-based models, not to mention the PRISharer. The reason may lie in that LDA-based AR-Miner aims to extract informative topics from raw user reviews, but lacks the improvement over PRIS domain, especially for the short sentence-level analysis.

6.1.3. PRISharer's Performance

Table 7 shows the detailed results for nine categories based on different models. In general, PRISharer achieves a high prediction rate for all categories. All these classification algorithms achieves an F1-score higher than 0.900 for PC, SP, BRT, FU, and PI. However, for issue UP, they achieve a relatively lower F1-score, varying from 0.784 to 0.915. Specifically, based on SVM, FU and PI are the issues that attain the best performance, with an F1-score of 1. Since fewer samples are labeled as these two issues in our testing set, it is easy to get an extreme value, and if these samples

Table 7
Comparison results of nine categories.

	Metrics	PC	TMP	UP	PB	RPR	SP	BRT	FU	PI
Precision	SVM	0.949	0.931	0.864	0.927	0.943	0.977	0.986	1	1
	LightGBM	0.947	0.922	0.806	0.864	0.893	0.892	0.986	0.993	0.993
	LR	0.948	0.908	0.852	0.916	0.899	0.945	0.979	0.993	0.980
	RF	0.904	1	0.911	0.922	1	0.953	0.993	1	1
Recall	SVM	1	1	0.973	0.944	0.983	1	1	1	1
	LightGBM	0.969	0.869	0.784	0.832	0.847	0.928	0.972	0.980	1
	LR	0.984	0.975	0.883	0.813	0.907	0.968	0.986	1	1
	RF	0.977	0.983	0.919	0.879	0.990	0.984	0.986	0.993	1
F1-score	SVM	0.974	0.964	0.915	0.935	0.963	0.988	0.993	1	1
	LightGBM	0.958	0.895	0.795	0.848	0.870	0.910	0.979	0.986	0.997
	LR	0.966	0.941	0.867	0.861	0.903	0.957	0.982	0.997	0.990
	RF	0.958	0.991	0.915	0.900	0.941	0.969	0.989	0.997	1

Table 8
Sharing results of PRISharer.

APP	Details
App name	Camera FV-5 Lite
Fine-grained category	Capture the Moments
Similar apps	Cameringo Lite. Filters Camera; Noah Camera; HedgeCam 2: Advanced Camera
Permission usage	STORAGE; CAMERA; PHONE
PRIS sharing	<Photography, camera, Permission-related Bug >; <Photography, phone, Unclear Permissions>; <Photography, microphone, Too Many Permissions>

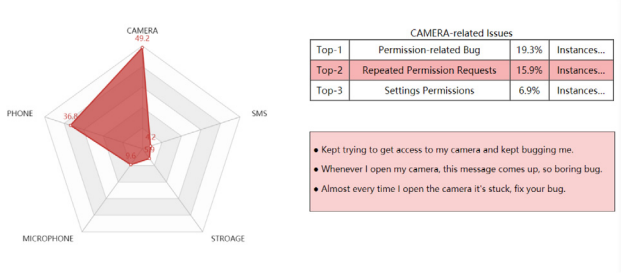


Fig. 4. Visualization results.

are identified correctly, the highest F1-score will be achieved. In our experiments, we adopt SVM as our training model due to the higher precision and recall for all categories. Note that all these models are trained over TF–IDF numerical representation, since it performs better for review transformation, discussed in Section 6.3.

6.1.4. Case study

We use `Camera FV-5 Lite`, a professional camera app, as an example to show the detailed recommendation made by PRISharer, shown in Table 8. Given the app's meta-information as input, PRISharer generates (a) a fine-grained category, (b) a list of candidate apps ordered by their similarity, (c) the usage of nine permission groups in similar apps, and (d) PRIS sharing mappings.

The results given by PRISharer provide several useful feedback for developers. First, the issues mined by PRISharer reflect that most users complained the issues about CAMERA-related bug and suspected its request of PHONE permission. To clear these, the developers could add explanations on the rationale in description text when releasing the app. Second, PRISharer reports the permissions that users are concerned on, i.e., CAMERA, PHONE, and MICROPHONE, suggesting that developers should pay more attention to these three permissions than others. Third, the information related to similar apps can be exploited to help understand the app's permission requirements. For instance, the most similar apps and top permission usage are helpful for developers to obtain more permissions-related knowledge.

To give an intuitive impression about how the proposed model shares PRIS for developers, we also visualize the above results with a radar chart containing a polygon as discussed in Section 4.4. We choose the radar chart to present since it facilitates the comparison between different attributes in a single diagram. As displayed in Fig. 4, each vertex represents a permission with an issue percentage reflected by user reviews to help developers identify the dominant permissions. Clearly, the results mean that, given a photo-oriented application, the CAMERA issues are most concerned by users. Moreover, by clicking the vertex, a diagram will be expanded to assist developers to get insight into the specific issue for each permission. Additionally, if developers click a cell in the last column, they can see the review instance lists. With visualization, developers can quickly tell the sharing feedback, manifest and focus on their goals.

Table 9
Comparison results of different feature combinations.

Models	Features	MAE	RMSE
TM	Title	0.126	0.155
TDM	Title and Description	0.070	0.092
TPM	Title and Policy	0.018	0.134
DM	Description	0.083	0.111
PDM	Policy and Description	0.072	0.096
PM	Policy	0.102	0.134
PRISharer	Title, Policy, Description	0.009	0.080

6.2. The performance of similar-app model based on different features

To measure the contribution of different features in identifying similar apps, we build several baselines based on various feature combination methods. Table 9 summarizes the performance of alternative combinations.

We can observe that PRISharer achieves higher performance than other candidates, with the lowest MAE and RMSE of 0.009 and 0.080, respectively. Especially, comparing with other models, the improvement for MAE ranges from 0.117 to 0.009, while for RMSE, the improvement ranges from 0.012 to 0.075. In general, as the number of features increases, the experiments achieve lower MAE and RMSE. For example, the models based on two features, i.e., TDM, TPM, PDM, perform better than the models based on a single feature, i.e., TM, DM, and PM. The results indicate that the deep mining of multi-features benefits the performance.

More importantly, we can also observe that for a given feature, the description feature contributes the most to the performance improvement. For instance, comparing with TM and PM, DM performs better in terms of both MAE and RMSE, with the value of 0.083 and 0.111. This is not surprising, as descriptions, which are pieces of long text samples, always cover more informative

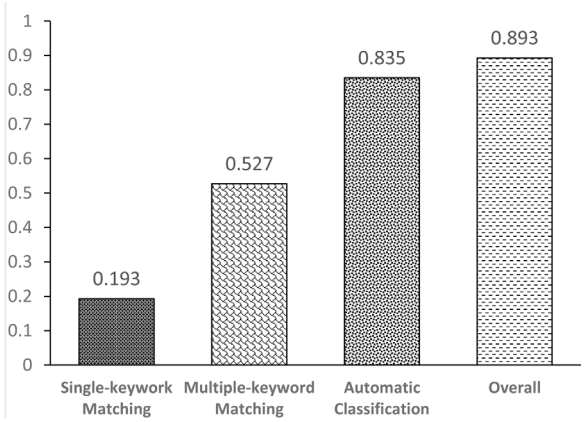


Fig. 5. Recall of different sentence extraction strategies.

expression to the app, and thus contribute more in capturing the similarity between apps. Moreover, comparing with descriptions and policies, the integration of title features achieves limited improvement for the results. In general, the results further indicate that feature addition of app meta-information is able to enhance the prediction performance, and app descriptions are more important than other features for identifying similar apps.

Different from prior works (Liu et al., 2018b; Gao et al., 2019) that modeled the similarity via unsupervised training methods, such as metadata-based clustering and similarity computing, we leverage a supervised neural network to learn the similarity between apps. Although the number of **Level II** apps is limited, we propose a novel similarity measurement to amplify the training labels based on the coarse-grained category and fine-grained category provided by the Google Play store. The results presented in Table 9 show that PRISharer is able to handle more features interaction and predict the similarity better. However, more features, such as sensitive APIs and code segments, will be explored for future work.

6.3. The performance of permission-related sentence extraction

Fig. 5 summarizes the recall of permission-related sentences trained over different keywords, where the overall approach considers all strategies, i.e., single-keyword matching, multiple-keyword matching, and automatic classification. In the process of review extraction, the metric recall, namely, the ratio of correctly identified review sentences is the most important metric, since our final goal is to identify PRIS among potential sentences. From Fig. 5, it is obvious that the overall approach presents the best performance, whereas single-keyword matching performs poorly, with the recall of 0.893 and 0.193 respectively. Besides, we also have the following observations.

As for extraction performance, comparing with automatic classification, both keyword-based matching strategies demonstrate low recall, and multiple-keyword matching (0.527) exhibits higher recall than single-keyword matching (0.193), which means the number of correctly selected sentences based on DAK is significantly less than the strategy based on IAK and GAK respectively. Practically, the results can be explained by the following three reasons. First, the high discard rate of keyword-based matching strategy is reasonable, since a sentence is extracted only when the keyword with a correct format appears in the reviews. Second, unlike DAK, the combinations of IAK generate more candidate keywords, and thus obtain a larger number of potential sentences. Third, the best performance of the overall approach is not surprising, since its number of correctly extracted sentence is a cumulative result of keyword-based matching and automatic classification. As described in Section 4.3, these strategies are exploited to obtain more permission-related sentences. The reason why we consider the number of sentences is that our work aims to identify the opinions from users reviews, the larger of the sentence data, the more effective for the opinion extraction. Thus, both keyword-based matching strategy and supervised classification strategy are applied to extract the potential sentences. As a result, PRISharer aligns a good balance between sufficient potential samples and better extraction performance.

Table 10
Comparison results of different models.

Feature extraction	Classification	Precision	Recall	F1-score
TF–IDF	SVM	0.905	0.835	0.868
	LightGBM	0.948	0.801	0.868
	LR	0.924	0.767	0.838
	RF	0.891	0.829	0.859
Word2Vec	SVM	0.875	0.815	0.844
	LightGBM	0.872	0.825	0.848
	LR	0.844	0.286	0.429
	RF	0.866	0.751	0.805

Moreover, as for automatic classification, we also conduct comparisons with several commonly assembled models, mentioned in Section 5.5. Table 10 gives a performance overview obtained through eight alternative approaches. We can observe that the TF-IDF+SVM model outperforms than other baselines in terms of F1-score and recall. Given a fixed feature extraction method, the LR classifier performs worse compared with other algorithms, while there is only a slight difference between SVM and LightGBM. For example, among the cases using Word2Vec as the feature extraction method, SVM achieves an F1-score of 0.844, which is slightly lower than LightGBM (0.848). Specifically, these two classifiers achieve the same F1-score (0.868) based on TF–IDF, while SVM has a higher Recall (0.835).

Meanwhile, from the comparison among different feature extraction methods, we can conclude that TF–IDF performs better in review numerical representations. Surprisingly, Word2Vec that considers the contextual semantics does not benefit the ultimate performance. The possible explanation is that although Word2Vec helps to learn the contextual information, most of the review sentences are short texts due to the segmentation in the preprocessing step, which is different from descriptions and policies. More importantly, the lengths of sentences are unbalanced, such that Word2Vec cannot effectively capture the information between words. In contrast, TF–IDF takes the global information, such as the word frequency and inverse document frequency, into account, which is more suitable for the review representation in this work.

6.4. The usefulness for developers

To verify the usefulness of our shared issues in the real world, we conduct a usefulness of PRISharer study for developer. To this end, we contacted the students who graduated from our lab and are engaged in different software development companies. Up to now, we have received the responses from these students covering seven companies in three cities. Thus, we asked them to contact the colleagues in the mobile department of their companies to ask if the colleagues would like to participate in our research investigation. If the colleague agrees, we will send him/her an introduction about the PRISharer, a sharing result of app `Camera FV-5 Lite` (Displayed in RQ1) pushed by PRISharer and a related questionnaire. Eventually, we receive the answers of 17 developers with at least three years of app development

Table 11
Feedback results of developers.

Questions	Score ≥ 4	Score $= 3$	Score ≤ 2	Average
Q1	14	1	2	4.00
Q2	13	1	3	3.88
Q3	15	0	2	4.06

experience. The questionnaire is designed by the following four questions:

(1) Do you think the detailed <category, permission, issues> mapping shared by PRISharer is useful for further improvement of your project? (2) Do you think the information related to similar apps shared by PRISharer helps you to understand the PRIS when developing an app?

(3) Do you think the visualization assists you easily in understanding the existing problems?

(4) What do you think could be improved with PRISharer?

If the Q4 is answered, we will send the developer an extra rating request for their concern of the nine issues discussed in Table 2.

For Q1, Q2 and Q3, each question is rated on a scale of 1–5, where 1 is "not at all helpful" and 5 is "extremely helpful". Table 11 summarizes the feedback results of these three questions. We can observe that most developers (14/17) held positive opinions for Q1, with an average score of more than or equal to 4, which indicates the usefulness of PRISharer in software development. For example, one developer stated that: "I will explain why we apply for PHONE permission, and be more aware of CAMERA permission when developing a photography app." Similarity, both Q2 and Q3 received positive response, with the average scores of 3.88 and 4.06, respectively. In addition, for these questions, we also received some negative feedback, where the developers thought they were experts at permission regulation or they did not want to waste time on these questions. The overall results (82.4% positive responses on average) highlight that PRISharer is effective and promising for developers to understand permission issues and discover permission requirements.

With respect to the open-ended question Q4, we received 13 responses. Among which, most developers (7/13) suggested we should provide more details about user experience, such as functionality, usability, and layout, while some suggestions mentioned about the effects of permission issues (4/13), since they wanted to know whether the issues would lead to the uninstall of the app or other negative effects. In addition, two developers concerned about the precision of sharing. According to the 13 responses of Q4 and the extra request to the 13 developers for issue rating. We can conclude that the major issues that developers concerned are TMP, UP, FU, and PB, indicating that the developers paid more attention to the problems in development process and they were more inclined to improve the quality of apps from the issues that are concerned most by users. In contrast, SP and RPR are not their focus when developing an app. The collected suggestions offer practical guidance to us on how to improve our future work.

7. Related work

In this section, we discuss the related work in three research fields: (a) app store data mining, (b) runtime permission, and (c) factorization machine. We integrate these existing approaches with our work, and distinguish our study from them as well.

7.1. App store data mining

In recent years, there is an increasing number of studies concentrate on mining app store data for discovering and analyzing Android requirement engineering, which mainly contain two types of features, namely app metadata provided by app developers (e.g., descriptions, permissions, policies, and titles) and app reviews (e.g., review texts and scores) provided by app users.

On one hand, among the app metadata, the description field has been widely recognized as a reasonable resource for exploring various tasks, such as over-privilege detection (Gorla et al., 2014; Pandita et al., 2013; Qu et al., 2014; Watanabe et al., 2015), similar-app identification (Gao et al., 2020; Liu et al., 2018b), and domain knowledge analysis (Hariri et al., 2013), since descriptions provide the most useful and representative information to discover app functionalities and characteristics. Other than descriptions, permissions and policies can also be analyzed for requirement discovery. For example, privacy policy-based mobile security analysis (Breaux et al., 2014; Breaux and Schaub, 2014; Slavin et al., 2016; Yu et al., 2017; Zimmeck et al., 2016) and runtime permission-based recommendation (Gao et al., 2019). On the other hand, since app reviews reflect how users consider the app, most efforts focus on mining useful opinions from reviews to recommend valuable information for developers. Vasa et al. (2012) made an exploratory study to discover the factors that affect user reviews, such as time of release, topics, and several properties including quality and constructiveness. Hoon et al. (2013) employed an LDA model considering the use of time series on user reviews to analyze topics of reviews. Closed to this work, Phong Minh Vu et al. Vu et al. (2015a) proposed a keyword-based framework, MARK, to mine users' useful opinions in app reviews, they also developed a semi-automatic supporting tool to analyze the reviews (Vu et al., 2015b). Both Di Sorbo et al. (2016, 2017) proposed SURF to recognize and classify the informative reviews for improving mobile applications. Furthermore, app reviews can also be exploited for bug reports (Maalej and Nabil, 2015), testing purposes (Grano et al., 2018), and discovering development requirements (Guo et al., 2019; Liu et al., 2018a; Scoccia et al., 2018).

All these efforts provide useful resolutions to handle various tasks by mining app store data, especially for google play store. Similar to these works, we seek to share PRIS for developers based on various kinds of app store data on the Google Play store, however, our work differs from them in two ways. First, we not only consider the app metadata, but also explore the reviews of these apps, covering four categorical fields and three textual fields. Second, the goals of app metadata and user reviews are different. In this work, app metadata is fed into a deep learning network to model the similarity between app, while user reviews are exposed to explore the PRIS task. By mining these two types of datasets, PRISharer overcomes the gap between developers and users.

7.2. Runtime permission

Various works have attempted to explore runtime permission, most of which mainly focus on runtime permission recommendation. In particular, Bonné et al. (2017) conducted a large field of study to explore user behaviors under the runtime permission system, showing that users' decisions rely on their expectations of the permission usage. In addition, Tan et al. (2014) and Liu et al. (2018a) made investigations into Android runtime rationales based on the iOS system and Android system, respectively. Since the authorizations of dangerous permissions are decided by users under AOFU, some works focus on the permission decision recommendation and prediction. Wijesekera et al. (2018) leveraged machine learning to predict user privacy decisions under dynamic circumstances. Olejnik et al. (2017) proposed a tool, SmarPer, to collect users' historical behaviors using contextual cues, and thus to analyze and predict users' privacy preferences. Furthermore, Tsai et al. (2017) presented an advanced contextually-aware permission manager, to assist users to regulate permissions with

varying privacy preferences. Besides, Liu et al. (2018b) made the first attempt to help developers explain an app's permission to users by mining potential permission explanations from similar-app descriptions.

Although studies about Android permissions are commonly done, few efforts focus on PRIS at runtime in the literature. Tao et al. (2020) proposed SRR-Miner, which summarizes security-related issues from user reviews of mobile applications. It reveals a series of security issues including several permission issues for users, but not focuses on runtime permission. The closest work to our work is the investigation by Scoccia et al. (2018) into the runtime permissions. They developed a semi-automatic classification to cluster user reviews according to specific permission concerns, and to mine how users perceive the runtime permission system from reviews. Although a number of improvement points are provided to the runtime permission system, they do not connect the PRIS with real developers.

Existing studies provide useful permission-related resolutions to users or developers. Similar to prior works, we also explore permission-related knowledge at runtime, while the difference is that our work aims to help developers be aware of potential problems when developing apps by mining similar-app PRIS.

7.3. Factorization machines

Factorization Machines (Rendle, 2010) have exhibited great power to enhance the effectiveness of feature interactions in software engineering, especially in the recommendation field. Xiao et al. (2017) proposed Attentional Factorization Machine (AFM) to enhance the interpretability of FM recommendations by discriminating the importance of feature interactions. Ying et al. (2018) introduced the attention mechanism to a two-layer hierarchical network, which takes both user's long-term and short-term preferences into account, for the next item recommendation. Moreover, Guo et al. (2019) proposed a knowledge-based deep factorization machine (KDFM), applying the topical attention and FM attention techniques to recommend service for mobile apps. Liang et al. (2020) proposed a hierarchical neural network approach named MV-AFM, which handles the feature interactions from different views of app recommendation.

In this work, we make the first attempt to introduce FM in the similar-app identification model. As discussed in Section 3.4, after apps' long and short textual features are fed into the deep similar-app machine, we employ FM to capture the feature interactions for better prediction.

8. Threats to validity

In this part, we discuss several threats that may impact the validity of our study.

Dataset. On the one hand, user reviews are solely collected in Google Play store, and the potential sentences are insufficient due to the high discard rate of the reviews. Although we use diverse collection strategies based on multi-layer keywords to collect more samples, the original usable sentences is still not large enough. The threat impacts the effect of both the data augmentation and final prediction. To alleviate it, we will extend the datasets from more app markets and relevant forums. On the other hand, the ground truth is built by manual labeling, which is influenced by one's domain experience and expert knowledge. To alleviate the bias, the labeling criteria is carefully discussed by all the authors, and the labeled samples are subject to several rounds of inspection by two authors.

Applicability. The proposed similar-app model is not limited to the Google Play store which is just one of our application scenarios. The proposed framework can be adapted for other stores, provided that the store can provide multi-dimensional metadata information for apps, such as Amazon app store and AppBrain. In fact, app source code is also an important feature for the intelligent classification but is not considered in our approach currently due to the challenge of source code extraction. We plan to extend the code feature by techniques of decompilation and anti-aliasing in future work.

Survey. To verify the usefulness of PRISharer in the real world, we recruited some participants and collected their survey responses. We acknowledge the possible limitations in this study. First, the scope of the participants may be limited. The participants contribute to our study since they received the invitation from their colleagues. We restrained from identifying ourselves to them and requested them to focus on evaluating whether our framework makes sense to developers, in order to avoid bias in their responses. Second, the number of survey responses is relatively small, which may impact the generalizability of our findings. This is caused by our limited connection to developers who are relevant to the permission issues. In future work, we plan to alleviate the limitation by resorting to crowdsourcing platforms. Third, we only send the sharing result of a typical app "Camera FV-5 Lite" in the developer survey since the typical result represents the general sharing for other apps and it can effectively save time for completing the survey. Indeed, there may be decision bias for "only one app", which motivates us to balance the efficiency and fairness of the developer study in the future.

9. Conclusion

In this paper, we propose PRISharer, a novel framework designed to deep explore user opinions of runtime permission issues for mobile apps. Instead of permission-related issues sharing, PRISharer provides analysis for rich relevant information including fine-grained category, similar app lists and corresponding permission usages. Specially, we make the first attempt to (1) learn a similar-app model from the app's multiple meta-information features with techniques of BiLSTM and DeepFM, (2) extract permission-related sentences based on multi-layer keywords, and (3) summarize the <category, permission, issues> mappings by employing machine learning techniques. Extensive experiments demonstrate PRISharer achieves superior performance in issue identification task. Comparative studies also illustrate the effectiveness of similar-app classification and permission-related extraction. Additionally, a developer survey with considerable positive approvals further indicates the usefulness of the proposed approach. In the future work, to further improve the analysis effects, we plan to extend the scope of both data collection and developer survey. Meanwhile, more app features will be explored to enhance the training of prediction model.

CRediT authorship contribution statement

Hongcan Gao: Conceptualization, Methodology, Writing – original draft. **Chenkai Guo:** Conceptualization, Methodology, Writing – review & editing. **Guangdong Bai:** Conceptualization, Writing – review & editing. **Dengrong Huang:** Data curation, Software. **Zhen He:** Data curation. **Yanfeng Wu:** Data curation. **Jing Xu:** Conceptualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China (Grant No. 62002177), Science and Technology Planning Project of Tianjin, China (Grant No. 17JCZDJC30700, 18ZXZNGX00310 and 20YDTPJC01810), Tianjin Natural Science Foundation, China (Grant No. 19JCQNJC00300).

References

Alepis, E., Patsakis, C., 2019. Unravelling security issues of runtime permissions in android. J. Hardw. Syst. Secur. 3 (1), 45–63.

Au, K.W.Y., Zhou, Y.F., Huang, Z., Lie, D., 2012. Pscout: analyzing the android permission specification. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 217–228.

Bonné, B., Peddinti, S.T., Bilogrevic, I., Taft, N., 2017. Exploring decision making with Android's runtime permission dialogs using in-context surveys. In: Thirteenth Symposium on Usable Privacy and Security. {SOUPS} 2017, pp. 195–210.

Breaux, T.D., Hibshi, H., Rao, A., 2014. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. Requir. Eng. 19 (3), 281–307.

Breaux, T.D., Schaub, F., 2014. Scaling requirements extraction to the crowd: Experiments with privacy policies. In: 2014 IEEE 22nd International Requirements Engineering Conference. RE, IEEE, pp. 163–172.

Cen, L., Si, L., Li, N., Jin, H., 2014. User comment analysis for android apps and CSPI detection with comment expansion. In: PIR@ SIGIR. Citeseer, pp. 25–30.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: synthetic minority over-sampling technique. J. Artificial Intelligence Res. 16, 321–357.

Chen, N., Lin, J., Hoi, S.C., Xiao, X., Zhang, B., 2014. AR-miner: mining informative reviews for developers from mobile app marketplace. In: Proceedings of the 36th International Conference on Software Engineering, pp. 767–778.

Das, D., Petrov, S., 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 600–609.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Di Sorbo, A., Panichella, S., Alexandru, C.V., Shimagaki, J., Visaggio, C.A., Canfora, G., Gall, H.C., 2016. What would users change in my app? summarizing app reviews for recommending software changes. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, pp. 499–510.

Di Sorbo, A., Panichella, S., Alexandru, C.V., Visaggio, C.A., Canfora, G., 2017. Surf: summarizer of user reviews feedback. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion. ICSE-C, IEEE, pp. 55–58.

Gao, H., Guo, C., Huang, D., Hou, X., Wu, Y., Xu, J., He, Z., Bai, G., 2020. Autonomous permission recommendation. IEEE Access 8, 76580–76594.

Gao, H., Guo, C., Wu, Y., Dong, N., Hou, X., Xu, S., Xu, J., 2019. AutoPer: Automatic recommender for runtime-permission in android applications. In: 2019 IEEE 43rd Annual Computer Software and Applications Conference. COMPSAC, vol. 1, IEEE, pp. 107–116.

Goldberg, Y., Levy, O., 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722.

Gorla, A., Tavecchia, I., Gross, F., Zeller, A., 2014. Checking app behavior against app descriptions. In: Proceedings of the 36th International Conference on Software Engineering. ACM, pp. 1025–1035.

Grano, G., Ciurumelea, A., Panichella, S., Palomba, F., Gall, H.C., 2018. Exploring the integration of user feedback in automated testing of android applications. In: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering. SANER, IEEE, pp. 72–83.

Graves, A., Schmidhuber, J., 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. 18 (5–6), 602–610.

Guo, H., Tang, R., Ye, Y., Li, Z., He, X., 2017. DeepFM: a factorization-machine based neural network for CTR prediction. arXiv preprint arXiv:1703.04247.

Guo, C., Wang, W., Wu, Y., Dong, N., Ye, Q., Xu, J., Zhang, S., 2019. Systematic comprehension for developer reply in mobile system forum. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering. SANER, IEEE, pp. 242–252.

Ha, E., Wagner, D., 2013. Do android users write about electric sheep? examining consumer reviews in google play. In: 2013 IEEE 10th Consumer Communications and Networking Conference. CCNC, IEEE, pp. 149–157.

Hariri, N., Castro-Herrera, C., Mirakhorli, M., Cleland-Huang, J., Mobasher, B., 2013. Supporting domain analysis through mining and recommending features from online product listings. IEEE Trans. Softw. Eng. 39 (12), 1736–1752.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Hoon, L., Vasa, R., Schneider, J.G., Grundy, J., et al., 2013. An Analysis of the Mobile App Review Landscape: Trends and Implications. Tech. Rep., Faculty of Information and Communication Technologies, Swinburne University of Technology.

Khalid, H., Shihab, E., Nagappan, M., Hassan, A.E., 2014. What do mobile app users complain about? IEEE Softw. 32 (3), 70–77.

Levy, O., Goldberg, Y., 2014. Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems. pp. 2177–2185.

Liang, T., Zheng, L., Chen, L., Wan, Y., Philip, S.Y., Wu, J., 2020. Multi-view factorization machines for mobile app recommendation based on hierarchical attention. Knowl.-Based Syst. 187, 104821.

Liu, X., Leng, Y., Yang, W., Wang, W., Zhai, C., Xie, T., 2018a. A large-scale empirical study on android runtime-permission rationale messages. In: 2018 IEEE Symposium on Visual Languages and Human-Centric Computing. VL/HCC, IEEE, pp. 137–146.

Liu, X., Leng, Y., Yang, W., Zhai, C., Xie, T., 2018b. Mining android app descriptions for permission requirements recommendation. In: 2018 IEEE 26th International Requirements Engineering Conference. RE, IEEE, pp. 147–158.

Loper, E., Bird, S., 2002. NLTK: the natural language toolkit. arXiv preprint cs/0205028.

Lui, M., Baldwin, T., 2012. langid. py: An off-the-shelf language identification tool. In: Proceedings of the ACL 2012 System Demonstrations, pp. 25–30.

Maalej, W., Nabil, H., 2015. Bug report, feature request, or simply praise? on automatically classifying app reviews. In: 2015 IEEE 23rd International Requirements Engineering Conference. RE, IEEE, pp. 116–125.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013a. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013b. Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems. pp. 3111–3119.

Olejnik, K., Dacosta, I., Machado, J.S., Huguenin, K., Khan, M.E., Hubaux, J.P., 2017. SmarPer: Context-aware and automatic runtime-permissions for mobile devices. In: 2017 IEEE Symposium on Security and Privacy. SP, IEEE, pp. 1058–1076.

Pagano, D., Maalej, W., 2013. User feedback in the appstore: An empirical study. In: 2013 21st IEEE International Requirements Engineering Conference. RE, IEEE, pp. 125–134.

Pandita, R., Xiao, X., Yang, W., Enck, W., Xie, T., 2013. {WHYPER}: Towards automating risk assessment of mobile applications. In: Presented As Part of the 22nd {USENIX} Security Symposium, {USENIX} Security 13, pp. 527–542.

Pletea, D., Vasilescu, B., Serebrenik, A., 2014. Security and emotion: sentiment analysis of security discussions on GitHub. In: Proceedings of the 11th Working Conference on Mining Software Repositories, pp. 348–351.

Qu, Z., Rastogi, V., Zhang, X., Chen, Y., Zhu, T., Chen, Z., 2014. Autocog: Measuring the description-to-permission fidelity in android applications. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 1354–1365.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., 2019. Language models are unsupervised multitask learners. OpenAI Blog 1 (8), 9.

Rendle, S., 2010. Factorization machines. In: 2010 IEEE International Conference on Data Mining. IEEE, pp. 995–1000.

Sak, H., Senior, A.W., Beaufays, F., 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

Scoccia, G.L., Peruma, A., Pujols, V., Malavolta, I., Krutz, D.E., 2019. Permission issues in open-source android apps: An exploratory study. In: 2019 19th International Working Conference on Source Code Analysis and Manipulation. SCAM, IEEE, pp. 238–249.

Scoccia, G.L., Ruberto, S., Malavolta, I., Autili, M., Inverardi, P., 2018. An investigation into android run-time permissions from the end users' perspective. In: Proceedings of the 5th International Conference on Mobile Software Engineering and Systems. ACM, pp. 45–55.

Slavin, R., Wang, X., Hosseini, M.B., Hester, J., Krishnan, R., Bhatia, J., Breaux, T.D., Niu, J., 2016. Toward a framework for detecting privacy policy violations in android application code. In: Proceedings of the 38th International Conference on Software Engineering. ACM, pp. 25–36.

Sundermeyer, M., Schlüter, R., Ney, H., 2012. Lstm neural networks for language modeling. In: Thirteenth Annual Conference of the International Speech Communication Association.

Tan, J., Nguyen, K., Theodorides, M., Negrón-Arroyo, H., Thompson, C., Egelman, S., Wagner, D., 2014. The effect of developer-specified explanations for permission requests on smartphone user behavior. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 91–100.

Tao, C., Guo, H., Huang, Z., 2020. Identifying security issues for mobile applications based on user review summarization. Inf. Softw. Technol. 122, 106290.

Tsai, L., Wijesekera, P., Reardon, J., Reyes, I., Egelman, S., Wagner, D., Good, N., Chen, J.-W., 2017. Turtle guard: Helping android users apply contextual privacy preferences. In: Thirteenth Symposium on Usable Privacy and Security, {SOUPS} 2017, pp. 145–162.

Vasa, R., Hoon, L., Mouzakis, K., Noguchi, A., 2012. A preliminary analysis of mobile app user reviews. In: Proceedings of the 24th Australian Computer-Human Interaction Conference. ACM, pp. 241–244.

Vidas, T., Christin, N., Cranor, L., 2011. Curbing android permission creep. In: Proceedings of the Web, Vol. 2, pp. 91–96.

Vu, P.M., Nguyen, T.T., Pham, H.V., Nguyen, T.T., 2015a. Mining user opinions in mobile app reviews: A keyword-based approach (t). In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 749–759.

Vu, P.M., Pham, H.V., Nguyen, T.T., Nguyen, T.T., 2015b. Tool support for analyzing mobile app reviews. In: 2015 30th IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 789–794.

Watanabe, T., Akiyama, M., Sakai, T., Mori, T., 2015. Understanding the inconsistencies between text descriptions and the use of privacy-sensitive resources of mobile apps. In: Eleventh Symposium on Usable Privacy and Security, {SOUPS} 2015, pp. 241–255.

Wijesekera, P., Reardon, J., Reyes, I., Tsai, L., Chen, J.W., Good, N., Wagner, D., Beznosov, K., Egelman, S., 2018. Contextualizing privacy decisions for better prediction (and protection). In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. ACM, p. 268.

Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S., 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. arXiv preprint arXiv:1708.04617.

Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems. pp. 802–810.

Ying, H., Zhuang, F., Zhang, F., Liu, Y., Xu, G., Xie, X., Xiong, H., Wu, J., 2018. Sequential recommender system based on hierarchical attention networks. In: The 27th International Joint Conference on Artificial Intelligence.

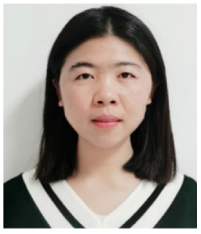
Yu, L., Luo, X., Qian, C., Wang, S., Leung, H.K., 2017. Enhancing the description-to-behavior fidelity in android apps with privacy policy. IEEE Trans. Softw. Eng. 44 (9), 834–854.

Zimmeck, S., Wang, Z., Zou, L., Iyengar, R., Liu, B., Schaub, F., Wilson, S., Sadeh, N., Bellovin, S., Reidenberg, J., 2016. Automated analysis of privacy requirements for mobile apps. In: 2016 AAAI Fall Symposium Series.

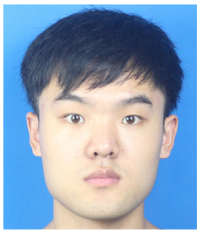
Hongcan Gao is a Ph.D. candidate in the College of Computer Science of Nankai University. She received the master's degree from Hebei University of Technology in 2017. Her research interests include software analysis on mobile apps and software security.

Chenkai Guo received the Ph.D. degree from Nankai University in 2017. He is an assistant professor in the College of Computer Science, Nankai University. His research interests include software analysis on mobile apps, information security, and intelligent software engineering.

Guangdong Bai received the bachelor's and master's degrees in computing science from Peking University, China, in 2008 and 2011, respectively, and the Ph.D. degree in computing science from the National University of Singapore in 2015. He is now a Senior Lecturer in the University of Queensland. His research interests include cyber security, software engineering and machine learning.



Dengrong Huang (M'87) is a master candidate in the College of Computer Science of Nankai University. She received the B.E. degree from Nankai University in 2017. Her research interests include mobile apps analysis and intelligent software engineering.



Zhen He is a master candidate in the College of Artificial Intelligence of Nankai University. He received the B.E. degree from Nantong University in 2018. His main research interests include video processing and machine learning.

Yanfeng Wu is a Ph.D. candidate in the College of Artificial Intelligence of Nankai University. He received the B.E. degree from Nankai University in 2017. His research interests include deep learning, software engineering and speaker recognition.

Jing Xu received the Ph.D. degree from Nankai University in 2003. She is a professor in the College of Artificial Intelligence, Nankai University. She received the Second Prize of the Tianjin Science and Technology Progress Award twice, in 2017 and 2018, respectively. Currently, her research interests include intelligent software engineering and medical data analysis.