IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 21, 2019, accepted December 16, 2019, date of publication January 17, 2020, date of current version May 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2967139

Autonomous Permission Recommendation

HONGCAN GAO[1], CHENKAI GUO[1], (Member, IEEE), DENGRONG HUANG[1], (Member, IEEE), XIAOLEI HOU[1], YANFENG WU[2], JING XU[2], (Member, IEEE), ZHEN HE[2], AND GUANGDONG BAI[3], (Member, IEEE)

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

Corresponding authors: Chenkai Guo (guochenkai@nankai.edu.cn) and Jing Xu (xujing@nankai.edu.cn)

This work was supported in part by the Science and Technology Planning Project of Tianjin, China, under Grant 17JCZDJC30700 and Grant 18ZXZNGX00310, in part by the Tianjin Natural Science Foundation under Grant 19JCQNJC00300, and in part by the Fundamental Research Funds for the Central Universities of Nankai University under Grant 63191402.

ABSTRACT Modern smartphone operating systems (e.g., Android 6.0 and later versions) employ an ask-on-first-use policy to regulate app permissions. To assist users in policy decisions, relevant efforts have been focusing on leveraging contexts to capture users' privacy preferences. However, these techniques have various limitations, such as heavily relying on users' historical decisions on granting permissions, ignoring the fact that users are not experts on privacy protection, and hard to determine whether a permission shall be granted. To address this problem, we propose an autonomous permission recommendation system, AutoPer+, to automatically recommend users the permission decisions at runtime. The main insight of our proposed system is that the natural language description of an app reflects its functionality and its similarity to other apps, and thus can be used to analyze whether a permission is needed indeed by it, and the apps similar to it. First, we introduce a multi-topic model into app functionality mining, and design a topic-permission mapper for the proposed recommendation system. Then we propose a deep semi-supervised machine using Long Short-Term Memory (LSTM) neural networks to identify similar apps, by which we can explore privacy permission usage in a cluster of apps. Finally, we capture a trade-off between privacy and utility to present a systematic recommendation. In addition to the permission decision ("Allow" or "Deny"), the permission explanations are also provided for users to make decisions (called "Ask"). We evaluate the proposed system via extensive comparison experiments on 31,023 Android apps. The results show that our approach achieves an accuracy of 84.1%. Moreover, we conduct user studies via installing AutoPer+ in the participants' own Android devices. We receive positive responses from the participants, which implies AutoPer+ is potentially for real-world deployment for enhancing current permission recommendation.

INDEX TERMS Android, permission recommendation, deep semi-supervised machine, privacy and security.

I. INTRODUCTION

Nowadays, smartphones play a key role in people's daily lives. Mobile users have increasing choices to install the apps with various functionalities, due to their ready accessibility in the popular app markets. For instance, the number of apps in the Google Play store was placed at 2.7 million in June 2019, which increases by more than 1.7 million, compared with July 2013 [1]. However, with the explosive growth of mobile apps, privacy and security on mobile devices has been a common challenge, since personal data and resources can be accessed by apps. To control the apps' access to sensitive data, a permission-based mechanism is used in Android to inform users of privacy, and thus to protect users' privacy and security.

Earlier versions of Android (5.1 and below) implement an ask-on-install (AOI) policy that requires users to grant permissions at installation time. Massive research [2]–[4] has shown that the AOI policy is limited in reality, since few users read the requested permissions when installing an app, and even fewer correctly understand the consequences of granting permissions. This lead to the update of the permission mechanism since Android 6.0 – the ask-on-first-use (AOFU) policy where users are prompted at the first time

The associate editor coordinating the review of this manuscript and approving it for publication was Gautam Srivastava.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

to "allow" or "deny" access to sensitive resources, such as the location and contact list. Compared with AOI, AOFU provides transparency on permission usage, but requires much more user interaction since each authorization is prompted at runtime for user decision. However, due to the over-reliance on users, such permission mechanism has several limitations. The critical one is the dilemma that users face when making permission decisions for app requests. Therefore, it is urgent to propose effective measures to help users regulate the permissions.

Many studies have been investigated for Android permission management. Among them, a large proportion [5], [6], [7] focus on AOI permission mechanism, e.g., permission recommendation at installation time and potential privacy leaks detection. Based on AOFU policy, most existing research [8]–[11] relies on dynamically regulating permissions by user preferences and contextual information. Wijesekera *et al.* [8], [9] conducted a field study to explore and predict users' privacy preferences by analyzing their past decisions and behaviors. They also proposed Turtle Guard [10], an advanced permission manager, to help users make better decisions under various contexts. Similarly, Olejnik *et al.* [11] also used contextual cues to predict users' behaviors. These works have made attempts to address the shortcoming of current permission systems from different perspectives.

However, existing works are still limited, since they heavily rely on a large number of users' historical permission decisions, and ignore the fact that most of the users lack the consciousness and expertise on privacy protection. Actually, due to the over-reliance on users, the permission mechanism has a high risk of privacy leakage. One of the reasons lies in the fact that there is an increasing number of apps that have been known to request more permissions than they need, along with the explosively growing apps. For example, a photo-oriented app requires to access the SMS permission. More importantly, very few users have the expertise in understanding the personalized apps, e.g., the relationship between permissions and apps' functionalities, and the privacy permission usages in a set of similar apps, which are the most important factors for users to make permission decisions. Therefore, providing a systematic permission recommendation from different perspectives towards user understanding and app requirements in software development is of great importance.

In this paper, we focus on three challenges that users face when making permission decisions for app requests: (i) what is the relationship between a requested permission and an app; (ii) what is the correlation between a permission and a set of similar apps; and (iii) how to reconcile these two aspects to make a better permission decision. To solve these challenges, we make the first attempt to perform a systematic runtime-permission recommendation from various perspectives, and explore multi-dimensional explanations for the recommendations to uncover the reasons for users. Specifically, we first build a <multi-topic, permission> mapping by leveraging the popular LDA (Latent Dirichlet Allocation) model and machine learning techniques to mine apps' multiple functionalities. Then we proposed an LSTM-based semi-supervised machine to identify similar apps. Finally, we provide an autonomous permission recommendation to help users make permission decisions by capturing a trade-off between privacy and utility. The proposed approach has been implemented in a tool AutoPer+. In addition to offering multiple recommendations of "Allow", "Deny", and "Ask", AutoPer+ also provides comprehensive explanations to help users understand the rationale of the recommendation, as well as a feedback loop for users to audit and modify the granted permissions.

The main contributions of this paper can be summarized as follows:

- We conduct the first effort to perform a systematic runtime-permission recommender for an autonomous permission decision, multi-dimensional permission explanations, and a validated feedback mechanism.
- We propose a novel model, AutoPer+, that assists users in granting permissions by reconciling apps' functionalities and similar apps' permission usages.
- We introduce techniques of semi-supervised deep learning and NLP into AutoPer+, which are trained on 31,023 apps from the Google Play store.
- The experiments show the superiority of AutoPer+ compared to existing approaches in terms of effectiveness. Meanwhile, AutoPer+ is evaluated by users with varying expertise, which demonstrates our recommender is potential for real-world use.

Following lists the rest structure of this paper. Section II formalizes the research problems. Section III presents the proposed approach in detail. Section IV and Section V present the experimental setup and results. Section VI and Section VII discuss our work and specific threats, respectively. Section VIII introduces the related work. In the last part, Section IX concludes our work.

II. PROBLEM FORMALIZATION

AOFU permission mechanism divides the permissions into *"Dangerous"* permissions that guard sensitive data (e.g., camera, location, and call logs), and *"Normal"* permissions that do not directly threaten user privacy, such as vibration and Internet-related permissions. AOFU merely prompts users to grant *"Dangerous"* permissions at the first time when an app attempts to request sensitive data. More specially, *"Dangerous"* permissions contain 9 groups of 24 permissions, as shown in Table 1. Note that we adopt the concept of permission group in this work, similar to [12], [13], which means when one permission in a certain group is authorized, the other permissions in the group will be granted at the same time.

Given a corpus of apps $\mathcal{V}$, for each $v \in \mathcal{V}$, there is a set of requested permissions $\mathcal{U}$, with $\mu \in \mathcal{U}$. Each μ has a recommended decision η with a numerical value indicating whether it is allowed to authorize by our recommender, where

TABLE 1. Dangerous permissions.

Permission Group	Permissions
CALENDAR	READ_CALENDAR WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL USE_SIP PROCESS_OUTGOING_CALLS
SENSORS	BODY_SENSORS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

$\eta = 0$ represents denied, $\eta = 1$ represents allowed, and $\eta = 2$ represents no additional decision recommended. We denote the mapping between permission and recommendation as $\mathcal{S}(\mu) = \eta$. Additionally, since we also provide multi-dimensional explanations for our recommender to users, for each recommendation η, there is a set of permission explanations, denoted as $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$. We formally define three typical problems related to our permission recommendation as follows.

A. PROBLEM 1. DETERMINING $\mathcal{S}(\mu) = \eta$

To provide an autonomous permission recommendation, the proposed work not only takes the app's own functionalities into consideration, it also explores the privacy permission usage in similar apps. Therefore, the mapping $\mathcal{S}(\mu) = \eta$ is mainly influenced by two factors. One is the relationship between a permission μ and an app v's functionalities, denoted as $\mathcal{Q}(\mu, v) = \gamma$; the other is the correlation between a permission μ and a set of similar apps kv, with $kv \in \mathcal{KV}$, denoted as $\hat{\mathcal{Q}}(\mu, kv) = \tau$. Similar to η, both the results of γ and τ are also in one of three decisions (i.e., "Allow", "Deny", and "Ask"). Thus, each mapping $\mathcal{S}(\mu) = \eta$ comes from a combined result of $\mathcal{Q}(\mu, v) = \gamma$ and $\hat{\mathcal{Q}}(\mu, kv) = \tau$ with better performance. We will present our approach that captures the trade-off between γ and τ in Section III-E.

According to problem II-A, then the challenge falls on how to identify the following two problems: $\mathcal{Q}(\mu, v) = \gamma$ and $\hat{\mathcal{Q}}(\mu, kv) = \tau$.

B. PROBLEM 2: DETERMINING $\mathcal{Q}(\mu, v) = \gamma$

For problem 2, the relationship between a permission μ and an app v is decided by whether μ is needed by v from the perspective of functionality. That is, we determine the mapping $\mathcal{Q}(\mu, v)$ by identifying the multiple functionalities of an app.

Especially, the challenges of problem 2 are three-fold: (i) an app may have multiple functionalities, and then how to identify the relationship between an app and various functionalities; (ii) how to mine and define the correlation between a specific functionality and a corpus of permissions; and (iii) given an app with a requested permission, how to determine the mapping $\mathcal{Q}(\mu, v)$. Thus, the problem II-B can be defined as the following three stages.

Problem 2-1: Exploring various functionalities of an app. We use W to denote an app's functionalities. The mapping $\mathcal{R}_\infty(v, \omega)$ refers to the relationship between $\mathcal{V}$ and W. For each v, it has a set of $\mathcal{W} = \{\omega_1, \omega_2, \ldots, \omega_m\}$.

Problem 2-2: Analyzing the correlation between a functionality and a set of permissions $\mathcal{R}_\triangle(\omega, \mu g)$, where μg represents the *"Dangerous"* permission group including several permissions.

Problem 2-3: Given an app v and a requested permission μ, the mapping $\mathcal{Q}(\mu, v)$ is determined by the ranked result of $\mathcal{R}(\mu, v)$, while $\mathcal{R}(\mu, v)$ is determined by $\mathcal{R}_\infty(v, \omega)$ and $\mathcal{R}_\triangle(\omega, \mu g)$. We leverage the popular LDA topic model to address the Problem 2-1, and incorporate machine learning algorithms into the solution to the Problem 2-2, which will be described in detail in Section III-C.

C. PROBLEM 3: DETERMINING $\hat{\mathcal{Q}}(\mu, kv) = \tau$

For problem 3, the mapping $\hat{\mathcal{Q}}(\mu, kv) = \tau$ is determined by the correlation between a permission and a set of similar apps. While the privacy permission usage in a set of similar apps provides users a straightforward understanding of the correlation between permissions and apps, which also helps users to make decisions. Therefore, we give the following definition based on these considerations.

Permission Usage: Given a set of similar apps kv, the permission usage is the proportion of apps (in kv) that request the permission group.

To determine the mapping $\hat{\mathcal{Q}}(\mu, kv) = \tau$, the challenge is how to identify similar apps. Although the category is provided by the Google play to describe an app's information, the category is too coarse to decide a precise permission set. However, we observe that there are a few but very popular apps given fine-grained categories (almost 2,630 covered by 118 fine-grained categories) in the Google Play store. Therefore, we adopt a deep learning method to automatically learn the latent features from the apps with fine-grained categories, and classify the other apps only with coarse categories.

In this work, we leverage a BISLTM model to classify apps, and thus mine the mapping $\hat{\mathcal{Q}}(\mu, kv) = \tau$, with the output decision of "Allow", "Deny" or "Ask". The process will be introduced in Section III-D.

III. APPROACH

A. OVERVIEW

Our permission recommendation framework, AutoPer+, contains three main phases: <multi-topic, permission>

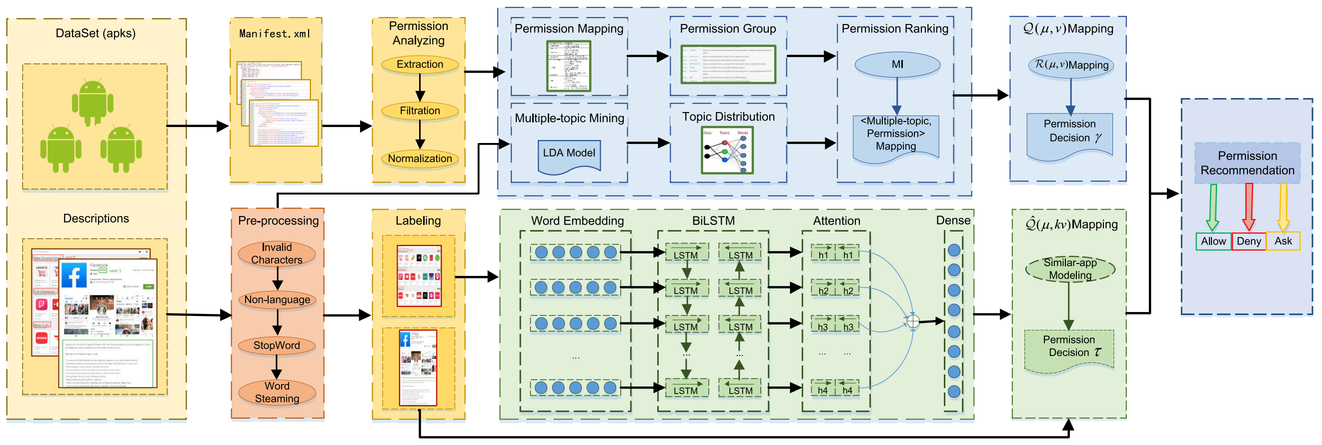


FIGURE 1. Framework overview.

mapping, similar-app mining, and decision recommendation, as shown in Figure 1. Since our model is based on description analysis, given a corpus of collected description instances, AutoPer+ first carries out a pre-processing to handle invalid words, filter non-English words, eliminate stop words, etc. Then the dataset is fed into two modules, <multi-topic, permission> mapping and similar-app mining, to construct different training models, respectively. In more detail, the goal of the first phase is to build a $\mathcal{Q}(\mu, v)$ mapping by which we can explore an app's multiple functionalities and map the relationship between functionalities and requested permissions, mainly including multi-topic mining, permission analysis, and <topic, permission> mapping. As an output, the result $\mathcal{Q}(\mu, v) = \gamma$ is provided for the further recommendation.

In the second phase, a deep semi-supervised model is designed to mine similar apps. To this end, our classifier is built based on two different datasets: **Level I** and **Level II**. We first collect all **Level II** apps with the tagged labels for further classification. Since the descriptions after pre-processing can not be directly used for model training, we have to convert the given description fragment into a vector representation. Hence, we exploit a word embedding component to carry out vectorization as well as keep the contextual semantics. After that, the produced **Level II** vector dataset is fed into a BILSTM with attention mechanism to train a multi-classification model. Subsequently, based on our prediction results of **Level I**, the analysis report of *permission usage* in similar apps is provided to determine $\hat{\mathcal{Q}}(\mu, kv) = \tau$.

Eventually, we recommend that whether a user adopts a permission request is a trade-off result between <multi-topic, permission> mapping and similar-app mining. As an output, the mapping $\mathcal{S}(\mu) = \eta$ figures out the permission recommendation with additional explanations $\mathcal{X}$.

B. PRE-PROCESSING

Since our approach is based on natural language descriptions provided by the developers in Android market, we first carry out a pre-processing to handle the collected raw dataset.

1) NON-ENGLISH REMOVAL

Due to the predominance of English in the Google Play store, our work only focuses on English text. Thus, after discarding invalid description instances, we exploit the tool called *langid.py* [14] to detect an app description's most likely languages and remove non-English descriptions. As a result, we obtain a total of 31,023 apps across all 30 categories.

2) WORD SEGMENTATION

Word segmentation is the step that divides the text into words. In English descriptions, words are typically identified by spaces and punctuation before and after the words. Therefore, we eliminate the stop words, which are commonly used in documentation but are considered meaningless for text classification (e.g., "is", "it", and "on"). Apart from that, we also remove punctuation like commas and quotes in this step.

3) WORD STEAMING

Word steaming is a necessary step in handling English language processing, since there are many variations of the same word such as "like", "likeness", and "liker". As a natural language processing technique, steaming is used to reduce derived words to their stems, by which we can also identify grammatical constructs and the relationship within the constructs. In practice, we use *NLTK* package to complete the task.

C. <MULTI-TOPIC, PERMISSION> MAPPING

1) MULTI-TOPIC MINING

In our work, the various functionalities of an app need to be identified by mining a set of description topics. To achieve this goal, we integrate LDA into the topic mining module. As an unsupervised topic generation technique, LDA generates a topic containing words that frequently occur together in the descriptions, which can be regarded as attributes of the topic. As a result, the text description of each app is divided into multiple topics with corresponding probabilities, which can be represented as various functionalities W of an app v, and the relationship within them $\mathcal{R}_\infty(v, \omega)$, respectively.

Specifically, in the implementation of LDA, an essential parameter is the number of topics. Since our apps cover 30 categories in the Google Play store, we set the parameter as 30. For each app, we filter out the topics with probabilities less than 5%. In addition, only the top 3 with the highest probabilities will be considered. As an example, we display the topics of three apps in Table 2.

TABLE 2. Examples of apps with LDA topics.

Package Name	Topic Name	Probability
media.music.musicplayer.mp3player	Music	0.653
	Media	0.157
	Communication	0.099
in.swiggy.android	Map	0.483
	Food and Drink	0.366
	Finance	0.081
photo.selfie.camera.hdcamera	Photography	0.665
	Beauty	0.203

From Table 2, we can observe that the app package "**media.music.musicplayer.mp3player**" is most related to three topics: "**Music**", "**Media**", and "**Communication**", with probabilities of 0.653, 0.157, and 0.099, respectively. While for package "**photo.selfie.camera.hdcamera**", two topics, i.e., "**Photography**" and "**Beauty**", are reserved. Note that the topic names, e.g., "**Music**" and "**Map**", are not generated by LDA; they are inferred from their attribute words.

2) PERMISSION ANALYSIS

In this step, we aim to obtain the requested *"Dangerous"* permissions in the apps. We first use *apktool* to decompile app installation files (.apk files), and then extract permissions from Android "*Manifest.xml*" files using *aapt* tool. In this process, we only focus on the 24 *"Dangerous"* permissions that access to sensitive data, and the *"Normal"* permissions (e.g., NFC, VIBRATE, and WAKE_LOCK) are ignored. As mentioned in Section II, given a set of requested permissions μ, we map them to the corresponding permission groups μg respectively, according to the mapping relationship (shown in Table 1).

In the collected apps, we observe that very few of them (less than 0.5%) request the permission group SENSORS. Therefore, SENSORS permission group is not considered in this step.

3) <TOPIC, PERMISSION> MAPPING

Our approach builds a <topic, permission> mapper to identify the correlation between topics and permission groups. As an example, the permission group CAMERA has a larger correlation with the topic "**Photography**" than the topic "**Map**". In order to precisely define the mapper, we use $\mathcal{R}_\triangle(\omega, \mu g)$ to represent the correlation of a functionality w and a permission group μg. A strong correlation between them indicates a high probability of permission being granted. Note that μg is the *"Dangerous"* permission group that the requested permission μ belongs to.

The permissions can be regarded as features that describe the behavior of restricting component access between apps, while the topics can be treated as classes that classify the functionalities of an app. Thus, the correlation of topics and permissions, i.e., the mapping $\mathcal{R}_\triangle(\omega, \mu g)$ can be evaluated by measuring the correlation of permissions/features and the topics/class variables. To achieve this goal, several alternative machine learning techniques, e.g., mutual information (MI), Pearson correlation coefficient (CorrCoef), and T-test, can be considered to build our model. Inspired by our previous experiment analysis [13], we leverage MI to achieve this goal with fairly better performance. Consequently, for each topic, we generate a rank list of eight permission groups according to the values of γ. Table 3 displays the μg rank lists of topic "**Map**", "**Media**", and "**Communication**".

TABLE 3. Permission group ranking.

Rank	Score	Map	Media	Communication
1	0.042	LOCATION	MICROPHONE	CONTACTS
2	0.033	STORAGE	STORAGE	MICROPHONE
3	0.021	MICROPHONE	SMS	SMS
4	0.017	CONTACTS	LOCATION	PHONE
5	0.017	CALENDAR	PHONE	CAMERA
6	0.014	CAMERA	CAMERA	STORAGE
7	0.012	PHONE	CONTACTS	LOCATION
8	0.009	SMS	CALENDAR	CALENDAR

By comparing the permission rank lists based on MI, it is obvious that the most relevant permission associated with "**Map**" topic is the permission group LOCATION (0.042), and the secondary is STORAGE (0.033), which are consistent with the user experience in real life, e.g., locating, downloading, and saving offline maps. On the other hand, in a "**Media**" app, it is straightforward to understand the purpose of permission group MICROPHONE; meanwhile, our results also show that CONTACTS permission group has a closer correlation with the topic "**Communication**" than the other permission group, such as CAMERA and LOCATION.

4) $\mathcal{Q}(\mu, v)$ MAPPING

Given an app v and a requested permission μ, the relationship between them γ is mainly influenced by two factors: the probability that an app belongs to a topic (i.e., $\mathcal{R}_\infty(v, \omega)$) and the correlation between a topic and a permission group (i.e., $\mathcal{R}_\triangle(\omega, \mu g)$). Thus, the mapping $\mathcal{R}(\mu, v)$ can be calculated by

$$\mathcal{R}(\mu, v) = \sum_{V} \mathcal{R}_\infty(v, \omega)\mathcal{R}_\triangle(\omega, \mu g) \tag{1}$$

where $\mathcal{R}_\infty(v, \omega)$ and $\mathcal{R}_\triangle(\omega, \mu g)$ are generated by multi-topic mining and <topic, permission> mapping, respectively.

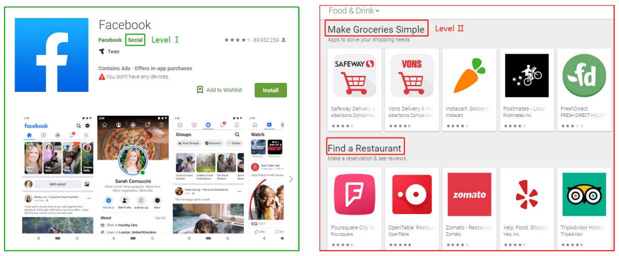
Finally, we rank the results of $\mathcal{R}(\mu, v)$ to generate a permission ranking list for each app, and recommend the permission decision $\mathcal{Q}(\mu, v) = \gamma$ based on the position of the requested permission in the ranking list. If the position lies in the top k_1, AutoPer+ turns into "Allow", meaning that our recommender grants permission to this request. If the position lies in the bottom k_2, AutoPer+ instructs "Deny", which is used to deny the permission request. Otherwise, AutoPer+ provides a request interface to prompt users to make decisions by themselves (called "Ask"). Based on our previous experimental results [13], the parameters settings here are $k_1 = 2, k_2 = 3$.

D. SIMILAR-APP MINING

In this step, we seek to analyze the usage proportion of a permission μ in a set of apps kv. Therefore, the challenge falls on how to mine similar apps kv. To address the challenge, we propose to measure the similarity between app descriptions by exploiting a semi-supervised deep learning algorithm, and design the search model as the following stages.

1) LABELING

To build the classification machine, the first step is to collect sufficient dataset with the marked labels for model training. However, there is no similarity score that can be regarded as the relationship between two apps, and the coarse category fails to decide a precise permission set, which make the labels hard to obtain.



(a) Level 1 (b) Level II

FIGURE 2. Dataset levels.

Actually, we can observe that although Google Play store does not provide a straightforward secondary category for each app, some popular and representative apps are tagged as fine-grained categories, which can be regarded as the training labels for classifier. According to problem II-C, the apps $\mathcal{V}$ can be divided into two levels. **Level I**: apps only with a coarse category (Figure 2(a)), and **Level II**: apps with both fine and coarse category (Figure 2(b)), each of which consists of a set of requested permissions and corresponding metadata. Therefore, we first crawl all **Level II** dataset including 118 labels for model training, and then tag the collected descriptions by labels for further classification.

2) EMBEDDING

Since the LSTM model only accepts numeric vectors as input, a particular vectorization phase is required to obtain such representation. As a hybrid neural network for text processing, word embedding [15] is an effective method to avoid dimension exploration while preserving original syntactic and sequence information.

We adopt Word2Vec [16], a popular unsupervised neural network method, to perform the word embedding. By converting words and phrases into a vector representation, Word2Vec can capture extra semantic features that help in text presentation. In more detail, Word2Vec generates word vector by two typical language models [17]: Bag-of-Words (CBOW) and Skip-Gram.

In the CBOW model, the goal is to predict a word based on the surrounding words, whereas the latter is to predict the context of words through a given target word. Since the goal of our work is to analyze continuous text descriptions, which is relevant to the semantics of the sentence, we employ Skip-Gram to construct Word2Vec model for better performance in the practical implementation. Specifically, using softmax function [18], the Skip-Gram model predicts conditional possibilities $p(c|w;\theta)$ with parameter θ is calculated as follows:

$$p(c|w;\theta) = \frac{exp(v_c \cdot v_w)}{\sum_{c \in C} exp(v_{c'} \cdot v_w)} \tag{2}$$

where c indicates the context of the given word w, and C refers to the set of overall available contexts. v_w and v_c refer to vector representations of w and c, respectively.

The loss function L_θ of the Skip-Gram training model with parameter θ can be represented as:

$$\begin{aligned}\mathcal{L}_\theta &= -\sum_{(c,w)\in D} \log p(c|w;\theta) \\ &= -\log \sum_{(c,w)\in D} (\log exp(v_c \cdot v_w) - \sum_{c' \in C} \log exp(v_{c'} \cdot v_w))\end{aligned} \tag{3}$$

To train our model, several parameters need to be manually set, e.g., embedding size of Word2Vec, hidden layer size of LSTM, and attention dimension, which will be discussed in Section V-B. Afterward, the dense vectors are fed into our BILSTM model for further classification.

3) BILSTM BUILDING

In our classification task, we model the text descriptions using Recurrent Neural Network with long short-term memory (LSTM) encoder [19]. LSTM has been proved to be an effective and scalable model for learning from sequence data and capturing long-range dependencies. The central idea behind the LSTM architecture is a memory cell, which can maintain its state over time, nonlinear gating units, and regulate the information flow into or out of the cell. The LSTM structure can be represented as:

$$i_t = \sigma(W^i x_t + U^i h_{t-1} + b^i) \tag{4}$$

$$f_t = \sigma(W^f x_t + U^f h_{t-1} + b^f) \tag{5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W^c x_t + U^c h_{t-1} + b^c) \tag{6}$$

$$o_t = \sigma(W^o x_t + U^o h_{t-1} + b^o) \tag{7}$$

$$h_t = o_t \odot tanh(c_t) \tag{8}$$

where σ denotes the *sigmoid (logistic)* function, and b denotes bias vectors. The input gate, output gate, and forget gate are denoted by i, f, o, respectively, while h denotes the hidden vector. W denotes input weight matrices and U denotes hidden-state weight matrices. c stands for the cell activation vectors.

Furthermore, to take advantage of two-directional hidden features (past features and future features), we adopt bidirectional LSTM networks [20] to explore a forward hidden state ($\overrightarrow{h_t}$) and a backward hidden state ($\overleftarrow{h_t}$) at time t. As a result, two-directional states are integrated into a final state, and can be computed as follows:

$$h_t = \overrightarrow{W^t}\,\overrightarrow{h_t} + \overleftarrow{W^t}\,\overleftarrow{h_t} + b_t \tag{9}$$

4) ATTENTION LAYER

Essentially, not all words in a description contribute equally to the representation. Thus, we capture the distinguished influences of words on the descriptions by leveraging a word attention mechanism [21], and thus obtain a dense vector considering the word weights. Specifically, we have

$$u_{ti} = tanh(Wh_{ti} + b) \tag{10}$$

$$a_ti = \frac{exp(score(u_{ti}^T u_w))}{\sum_{j=1}^{n} exp(score(u_{tj}^T u_w))} \tag{11}$$

$$s_t = \sum_i a_{ti}h_{ti} \tag{12}$$

where t refers to t-th description, and i refers to i-th word in the description. u_{ti} is a hidden representation of a continuous context vector h_{ti}, and b is a bias vector. Then we measure the importance of different words by leveraging the similarity between u_{ti} and context vector u_w. As a result, a normalized importance weight a_{ti} is obtained through the softmax function, and thus the sentence vector s_t is represented as a weighted sum of the word annotations.

5) DATA IMBALANCE

Data balance is an important factor in building a supervised model. However, our collected data is limited to achieve this goal. On one hand, the number of **Level II** dataset is insufficient containing 2,630 popular apps while marked by 118 fine-grained labels. On the other hand, the distribution of these apps is imbalanced, as it relies on their popularity and varies with different categories. Therefore, for better training, we leverage semi-supervised learning to alleviate the deviation of the weak supervision, as shown in Figure 3.

Our BILSTM structure is trained in two stages. In the first training stage, the classifier is trained (arrow ① shown in Figure 3) under labeled dataset (**Level II**) to predict (arrow ②) the unlabeled dataset (**Level I**). Afterward, we can get a pseudo-labeled dataset (arrow ③). In the second training stage, the input of the raw classifier is converted into a combination of the pseudo-labeled dataset and the labeled dataset (arrow ④). As a result, a fine-grained classifier is obtained in our semi-supervised training model.

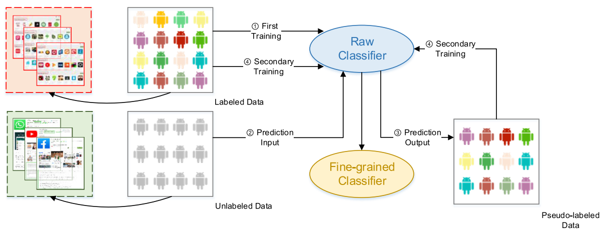


FIGURE 3. Semi-supervised learning.

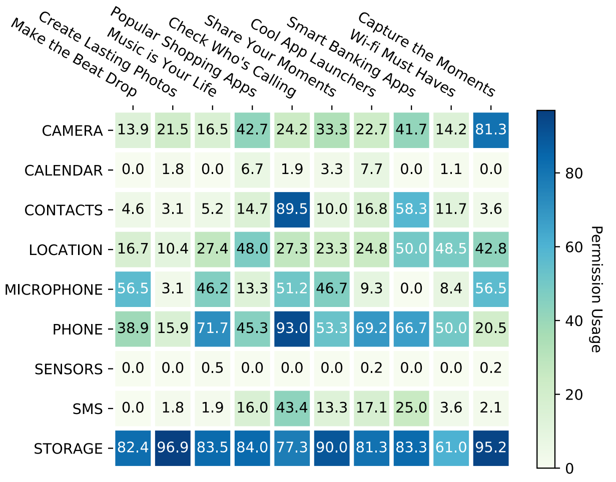


FIGURE 4. Permission groups usage distribution.

6) $\hat{\mathcal{Q}}(\mu, kv)$ MAPPING

The goal of this step is to look into the *permission usage* in various sets of similar apps. Our idea is based on the observation that if a permission μ is frequently requested by a cluster of apps kv, the correlation between μ and kv is often closer than other permission pairs (e.g., for the same purpose in the description), which is also more likely for users to grant access to some resources. For instance, the correlation of Map apps and the LOCATION-related permissions is closer than CAMERA and other permissions. Therefore, we focus on computing the *permission usage* of all permission groups in each similar-app cluster. In our study, the output of mapping $\hat{\mathcal{Q}}(\mu, kv)$ is "Allow" only when the percentage value is higher than or equal to ϱ_1 in each similar-app set kv, and if the percentage value is lower than or equal to ϱ_2, AutoPer+ turns into "Deny" mode, τ is "Ask" otherwise, where ϱ_1 and ϱ_2 are parameters and will be discussed in the experiments (Section V-A) later. More specifically, the following *permission usage* distribution matrix (Figure 4) displays 10 similar-app sets of samples (top) and their permission usages (right) in our collected dataset.

By comparing each similar-app set with its nine permission counterparts, we can observe that there is a significant difference in *permission usage* between different μg, which varies from cluster to cluster. Specifically, the permission group with the maximum number is STORAGE, which accounts for 96.9% of the total "**Create Lasting Photos**" apps, and the second one is group STORAGE in the "**Capture the Moments**" apps (95.2%). It is obvious that most apps apply for STORAGE-related permissions. Moreover, the *permission usage* of MICROPHONE and PHONE are higher in most clusters, i.e., "**Check Who's Calling**" and "**Music is Your Life**". Apart from that, our statistics also show that some "**Check Who's Calling**" apps more frequently request permission PHONE (93.0%), and "**Make the Beat Drop**" apps are more likely to request permission MICROPHONE (56.5%). In contrast, the number of the permissions related to CALENDAR and SMS is small compared with the other permissions. Similar to app functionalities mining, group SENSORS contains too few proportions in each app set (ranging from 0% to 0.5%), and therefore we discard the permission group in further study.

E. PERMISSION RECOMMENDATION $\mathcal{S}(\mu) = \eta$

In the last part of our study, we seek to provide a final permission recommendation based on our research results. Given an app v with a requested permission μ, AutoPer+ captures a trade-off between the above two recommendation results, i.e., $\mathcal{Q}(\mu, v)$ mapping and $\hat{\mathcal{Q}}(\mu, kv)$ mapping, based on different permission groups. Briefly speaking, if a permission group achieves better performance for recommendation in one of these two mappings, then the mapping result is the final decision indicating our recommendation. Thus, $\mathcal{S}(\mu) = \eta$ is a combined result of γ and τ with better performance. In our work, permission group CALENDAR and SMS are determined by $\hat{\mathcal{Q}}(\mu, kv)$ mapping, and the rest permission groups are the results of $\mathcal{Q}(\mu, v)$ mapping, which will be discussed in Section V-A. As the output, in addition to the permission decision ("Allow" or "Deny"), the permission explanations are also provided for users to make decisions by themselves ("Ask").

IV. EXPERIMENTAL SETUP

A. RESEARCH QUESTIONS

To evaluate the performance of AutoPer+, we designed an exploratory study to answer the following research questions (RQs):

RQ1: How accurate is AutoPer+ compared to different permission recommendation approaches?

RQ2: How well does the BILSTM model perform in detecting similar apps compared to state-of-art models?

RQ3: Are the permission decisions recommended via AutoPer+ actually useful for users?

B. DATASET

1) APP SELECTION

In this work, we reuse the dataset used in our previous work [13], which contains 28,850 Android apps, including their metadata, e.g., package name, category, number of downloads, average rating, requested permissions, and text description, from the Google Play store. Apart from that, we also collected other 2,630 apps with a fine-grained category **Level II** and 2,700 apps with a coarse-grained category **Level I** in June 2019 to train our model. When creating the new dataset, we removed the apps with less than 5,000 downloads. In addition, we also discarded the apps classified into different fine-grained categories, which are not feasible for training data. Afterward, we combined these two sets and removed the duplicate and invalid apps to construct the new dataset.

Finally, these collected apps are divided into **Level I** with the number of 29,275 apps, and **Level II** with 1,748 apps covering 118 fine-grained labels. We adopt the *cross-validation* method to monitor classification performance, where **Level II** data is split into two sets: a training set which is used to train the model, and a test set which is used to measure the classification performance. During learning, our algorithm is terminated when the network starts to overfit the training data. Table 4 displays the overview of our dataset.

TABLE 4. Dataset, where #TA denotes the total apps; #PA denotes the previous apps in [13]; #FA denotes the apps with fine-grained categories; #NA denotes the number of labels in Level II.

Categories	#TA	#PA	#FA	Level I	Level II	#NA
Overall	31,023	28,850	2,630	29,275	1,748	118

2) MANUAL ASSESSMENT

To evaluate the effectiveness of AutoPer+, we collected 264 apps (not included in Table 4) and manually inspected these samples to check whether the permission requested as expected. Among them, 173 apps came from our previous work [13]. Moreover, we added 91 new popular apps, each of which has been downloaded for more than 50,000 times. Thus, 264 apps are reserved for manual assessment.

During the manual assessment, three experienced Android developers are invited to examine these 264 apps, and the results are defined as the ground truth in our experiment. The experts manually examined the apps' information (e.g., category, comments, text descriptions, policies, and the relationship with similar apps) to make a comprehensive understanding of their functionalities and requested permissions. A permission is only needed if at least two experts agree on the necessity of the permission. Similar to our previous work [13], the manually recommended decision of each permission sample is also in one of three categories: "Allow", "Deny", and "Ask". During the study period, given an app with its information, each assessment takes 7 minutes averagely.

C. METRICS AND BASELINES

To evaluate the performance of AutoPer+, three well-known metrics for the measurement of classification task are used in the experiment, i.e., precision, recall, and F1-score. As a weighted harmonic mean of precision and recall, F1-score is an index of a comprehensive measure, which is described as

$$F1{-}score = \frac{2 \times precision \times recall}{precision + recall}. \tag{13}$$

TABLE 5. Performance of AutoPer+, where P denotes the precision; R denotes Recall; F denotes F1-score.

AutoPer+	Deny							Allow						
	TP	FP	FN	TN	P(%)	R(%)	F(%)	TP	FP	FN	TN	P(%)	R(%)	F(%)
CALENDAR	26	1	0	0	96.30	100.00	98.12	0	0	1	26	-	0.00	-
CAMERA	66	14	6	32	82.50	91.67	86.84	11	4	15	88	73.33	42.31	53.66
CONTACTS	6	3	16	101	66.67	27.27	38.71	81	15	1	29	84.38	98.78	91.01
LOCATION	7	2	3	123	77.78	70.00	73.69	98	8	7	22	92.45	93.33	92.89
MICROPHONE	31	11	10	29	73.81	75.61	74.70	15	1	3	62	93.75	83.33	88.23
PHONE	49	8	42	48	85.96	53.85	66.22	22	9	4	112	70.97	84.62	77.20
SMS	22	6	1	2	78.57	95.65	86.27	0	0	3	28	-	0.00	-
STORAGE	56	15	4	153	78.87	93.33	85.49	37	6	13	172	86.05	74.00	79.57

In order to validate the effectiveness of the proposed model in similar-app classification, we build several baseline models by following feature extraction methods and machine learning methods.

- **TF-IDF + RF (TR*)**: We leverage TF-IDF to extract description features, then the produced word vectors will be fed into Random Forest (RF).
- **Word2Vec + RF (WR*)**: We go through the same process for producing word vectors (Section III-D2), while the classification task is conducted by TR.
- **TF-IDF + MLP (TM*)**: Features are extracted by TF-IDF, and the classification task is conducted by a Multilayer Perceptron (MLP) classifier.
- **Word2Vec + MLP (WM*)**: Features are extracted by Word2Vec, and the classification task is conducted by a MLP classifier.
- **Word2Vec + BILSTM + Semi-super (WBS*)**: In our work, we propose the Word2Vec + BILSTM + Semi-super model to handle the challenge of contextual vector representation and weak labeling in classifying the similar-app task.

V. EXPERIMENTAL RESULTS

In this section, we present our experimental results to answer the research questions proposed in Section IV-A.

A. RQ1: PERFORMANCE OF AutoPer+

Table 5 summarizes the precision, recall, and F1-score of AutoPer+ with eight permission groups mentioned in Section II (excluding SENSORS permission group). Essentially, this work addresses a three-classification problem including categories of "Allow", "Ask", and "Deny". Similar to our previous work [13], we evaluate the performance of recommendation "Allow" and "Deny", since "Ask" mode only provides permission explanations without decisions.

As for the recommendation of "Deny", we can observe that AutoPer+ performs good results for most permission groups in terms of the same metric. Specifically, AutoPer+ achieves the best performance with respect to group CALENDAR, which matches the experts' decisions with the precision, recall, and F1-score being 96.30%, 100.00%, and 98.12%, respectively. Similar to CALENDAR, the AutoPer+ also performs a better result for CAMERA, achieving an F1-score of 86.84%. Whereas group CONTACTS has a lower recall, with the value of 27.27%. On the other hand, for the recommendation of "Allow", group CONTACTS and LOCATION achieve the F1-score of 91.01% and 92.89%, respectively, which perform much better than decision "Deny". Especially, permission LOCATION attains the best performance, with the F1-score of 92.89%. Generally, with regard to group STORAGE, AutoPer+ still retains a reasonable balance between precision and recall for the recommendation of both "Allow" and "Deny". As discussed before (Section III-D6), since there are very few numbers of CALENDAR and SMS permission in both training and assessment dataset, it is reasonable that TP and FP are zero in our experimental evaluation, and thus the corresponding precision and F1-score cannot be obtained. Overall, by computing all eight permission groups' metrics, our approach achieves fairly decent performance with an average accuracy of 84.1%, which indicates the effectiveness of the proposed approach in permission recommendation.

To evaluate the performance of the proposed approach, AutoPer+, we present a comparative study against our previous permission recommendation approach [13], which is an automatic permission recommender that only mines an app's multiple functionalities, denoted as AutoPer. Different from AutoPer, AutoPer+ takes both app's functionalities and *permission usage* into consideration, which is a combined result of these two approaches. Figure 6 shows the performance of these two approaches on "Deny" decision with permission group CALENDAR and SMS. It can be seen that AutoPer+ performs better than AutoPer in terms of all metrics. More specifically, AutoPer performs a worse result for group SMS, achieving an F1-score of 22.22%, well below the 86.27% threshold reached by AutoPer+. Especially, AutoPer+ performs the best for group CALENDAR, with the F1-score of 98.12%. Since the amount of requests for these two permissions is small, it is difficult for AutoPer to learn the relationship between topics and permissions during the process of functionality mining. As a comparison, the *permission usage* based on a fine-grained classification in AutoPer+ is more effective to build the relationship between apps and permissions. While for the rest

TABLE 6. Comparison Results, where P denotes the precision; R denotes Recall; F denotes F1-score.

CALENDAR	TP	FP	FN	TN	P(%)	R(%)	F(%)
AutoPer	16	1	10	0	94.11	61.54	74.42
AutoPer+	26	1	0	0	96.30	100.00	98.12
SMS	**TP**	**FP**	**FN**	**TN**	**%P**	**%R**	**%F**
AutoPer	3	1	20	7	75.00	13.04	22.22
AutoPer+	22	6	1	2	78.57	95.65	86.27

six permission groups, (e.g., CAMERA, CONTACTS, and LOCATION), the recommendation results of AutoPer+ for both "Allow" and "Deny" decisions come from AutoPer, which are thus not included in Figure 6. Although the results based on AutoPer are effective for most permission groups (6/8) in permission recommendation, AutoPer+ improves the metrics for group CALENDAR and SMS, which demonstrate the contribution of exploring *permission usage* in similar apps for permission recommendation.

In our model, the *permission usage* in similar apps indicates the correlation between the apps and permissions. To identify the proper parameters in AutoPer+ mentioned in Section III-D6 (i.e., ϱ_1 and ϱ_2), we compare the accuracy of different parameters for each permission group. Therefore, there are totally 55 effective combinations (10% increments) and the candidates with the best performance will be adopted by AutoPer+. In our model, the parameter combination (ϱ_1, ϱ_2) with the best performance for group CALENDAR and SMS is (30%, 70%), which means if the *permission usage* is lower than or equal to 30%, our model recommends "Deny"; if the number is higher than or equal to 70%, our model recommends "Allow"; otherwise, "Ask" mode is recommended.

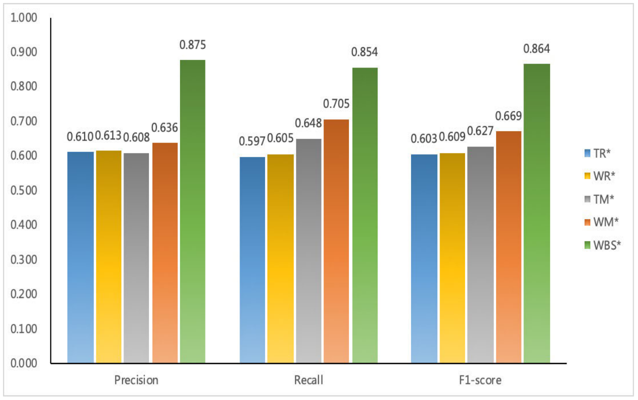


FIGURE 5. Performance of different approaches.

B. BILSTM VS ALTERNATIVE APPROACHES

Figure 5 presents the performance of different models (mentioned in Section IV-C) trained on the same dataset. Due to the imbalanced distribution of training sets (discussed in Section III-D5), we compare the performance of various models based on the top five fine-grained categories with the largest number in our experiments. We can observe that the deep BILSTM-based approach outperforms other baseline models in terms of precision, recall, and F1-score. In detail, F1-score enhancements with the semi-supervision tactic to TR*, WR*, TM*, and WM* are 0.261, 0.255, 0.237, and 0.195, respectively. Meanwhile, compared to TR* and TM*, the models with Word2Vec (WR* and WM*) both improve the performance in terms of three metrics. Results highlight that the Word2Vec performs better than TF-IDF in feature extraction. Especially, the proposed model on the same vector representation achieves the highest F1-score, with the value of 0.864. It also can be seen that based on the same feature representation, e.g., Word2Vec, MLP algorithm has a relatively comparable performance with the proposed model, with a more enhancement of F1-score (0.669), which is much better than RF (0.609). Whereas TR* achieves better performance with the precision of 0.610 than TM* (0.608), both of which use TF-IDF to extract the word vectors.

The practical benefits of the proposed model can be demonstrated by two reasons: one is the contextual and semantic word representation; the other is the advantages of semi-supervised learning implemented by BILSTM technique. For the first reason, Word2Vec captures the contextual information when transforming the word set to vector matrix, which is suitable for our structural semantic descriptions. Similarly, BILSTM-attention feeds a dense sequence of word vectors into the model to enhance ultimate performance as well. For the second reason, the BILSTM-attention model is employed to classify a description snippet into different fine-grained categories, which compresses the feature representations into a set of context-aware hidden vectors, and helps to build a better model.

Furthermore, in the comparison between other baseline algorithms, MLP outperforms RF with the F1-score improvement of 0.06. The reason lies in the nature of different algorithms, e.g., RF is a learning method that combines several weak classifiers into one strong classifier, while MLP is an artificial neural network model with multi-layer structures, which is more suitable for our dataset. Moreover, in the multi-classification problem, we also conduct the comparisons with the models based on Support Vector Machines (SVM) and Naive Bayes (NB). The results of these models are relatively poor than the above algorithms, which are thus not included in Figure 5.

In addition, during the training process, several parameters need to be manually set. Figure 6 shows the values of loss at different epochs. We can observe that when the epoch is set to 1,500, the loss gains the optimal performance. Similarly, we adopt various validation tests to set other parameters in our experiment. As a result, we set batch size as 64; embedding size of Word2Vec is 128; hidden layer size of LSTM is 128; attention dimension and sequence length are set as 100 and 400, respectively.

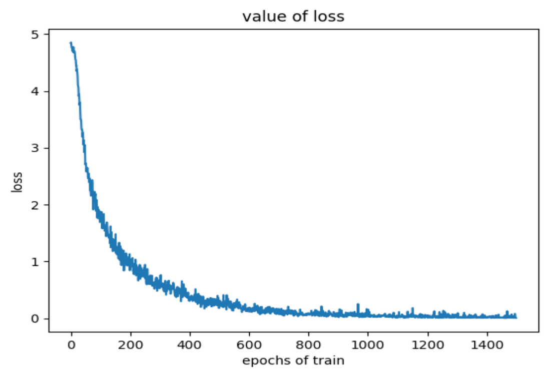
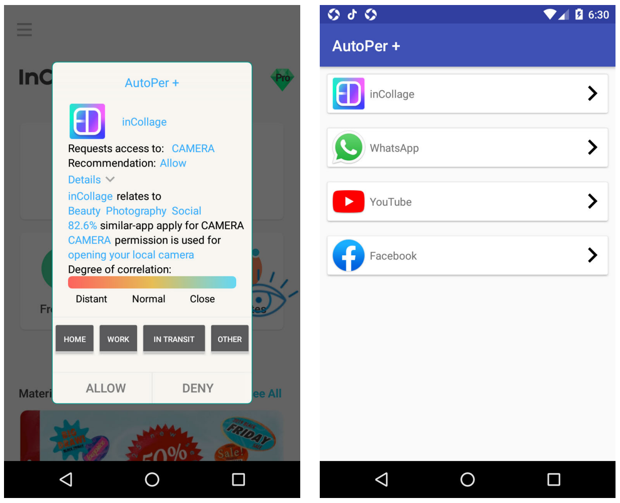


FIGURE 6. Value of loss.

C. EFFECTIVENESS OF RECOMMENDATIONS

To answer RQ3, we designed our approach into a new app tool AutoPer+, and distributed it to participants in real world scenarios. The newly designed app is extended to show not only a recommended permission decision (“Allow”, “Deny”, and “Ask”) with detailed explanations, but also a feedback interface for users to audit and correct their decisions, as shown in Figure 7.


AutoPer +
inCollage
Requests access to: CAMERA
Recommendation: Allow
Details
inCollage relates to
Beauty Photography Social
82.6% similar-app apply for CAMERA
CAMERA permission is used for
opening your local camera
Degree of correlation:

(a) Interface of Recommendation (b) Interface of Feedback

FIGURE 7. The interface of AutoPer+.

Here is an example of app “inCollage”, requesting access to CAMERA permission. From Figure 7 (left), we can notice that AutoPer+ allows “inCollage” to access CAMERA. Moreover, AutoPer+ provides a comprehensive recommendation explanation consisting of three main parts: (i) a series of app-related details, e.g., the multi-functionalities of app, and the purpose of the permission (indicated by $\mathcal{Q}(\mu, v) = \gamma$ in Section III-C4); (ii) the proportion of apps that request certain permission in a set of similar apps (indicated by $\hat{\mathcal{Q}}(\mu, kv) = \tau$ in Section III-D6); and (iii) a straightforward understanding of the correlation degree between the app and the permissions, such as keywords notification (i.e., *“Close”, “Distant”, and “Normal”*) and a progress bar-based design. Besides, Figure 7 (right) displays a feedback interface where users are prompted for reviewing and adjusting the previous settings. In detail, the screen presents information about apps that recently requested (i.e., *WhatsApp, YouTube, and Facebook*), including when the permission requests occurred and whether these permissions were granted.

We deployed AutoPer+ in 37 participants’ rooted Android devices. Some of them (14/37) are the users who also participated in our previous work [13]. We recruited the participants with Android 6.0 devices. They were asked to install the AutoPer+ app and to use at least two apps for the study. After two weeks, we collected their decisions concerning recommended permissions and investigated the satisfaction degree of the participants by filling a questionnaire online. The questionnaire contains the following four questions: (i) do you think the “Deny” recommendation protects your privacy, (ii) do you think it is reasonable to recommend “Allow”, (iii) how do you think the interpretations of apps and permissions help you to make decisions, and (iv) do you think the feedback mechanism is useful for your permission regulation. Each question is rated by users from 1 (completely useless) to 5 (very useful). Finally, all the questions are summarized as an average score.

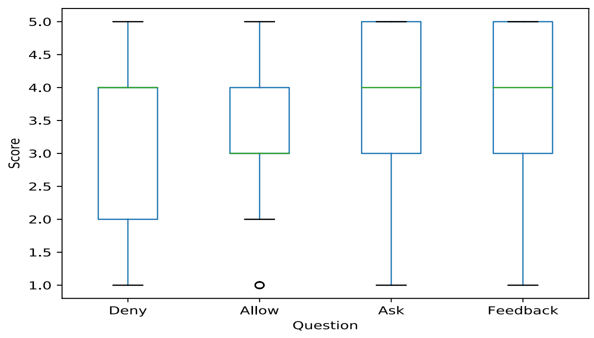


FIGURE 8. Distribution of scores.

The distribution of ratings is represented as a boxplot, shown in Figure 8. The majority of users (31/37) present positive opinions, with the score of more than or equal to 3. Especially, 3 participants scored 5 points, whereas 2 users scored less than 2 points. The results indicate that participants have quite different privacy concerns. To capture the rationales of collected scores, we requested them to make explanations about their scoring, where the participants that scored more than 3 primarily stated that the tool matches their needs or resolves their predicament when making decisions; one participant that scored less than 2 points stated that he wanted to make decisions by himself; the other one stated that the permission recommendation is a waste of time. Moreover, we also requested the 14 participants in our previous work [13]. They stated that they trusted AutoPer+ more, not only because of the increased explanations of similar apps, but also because of the tool’s feedback loop mechanism. Overall, the majority of recommendations were accepted, which indicates that the tool effectively recommends permissions for users.

VI. DISCUSSION

A. OPTION "ASK"

AutoPer+ aims to protect users' privacy behaviors by providing runtime-permission decisions. Inspired by prior works [8], [9], [11], [22], users' privacy preferences may change under varying contexts (e.g., user's location, permission's request time, and the status of devices). Our work, however, is based on static description analysis, rather than relying on dynamic features. Hence, we additionally provide an "Ask" interface without recommendations to mitigate this limitation. Thus, while protecting users' privacy (Decision "Deny") and data availability (Decision "Allow"), we also consider the impact of the environment on users' behaviors (Decision "Ask"). Although we do not provide recommendations on "Ask" mode, multiple explanations are presented to prompt users to make decisions by themselves. From the feedback of users in Section V-C, we can observe that the option "Ask" is well-received by the participants. But in the long term, dynamic analysis paired with static text descriptions may be a better option.

B. EXPERIMENTAL COMPARISON

Existing research has explored Android permission recommendation from two perspectives: users' preferences and security. For users, the effectiveness of a recommendation system ultimately depends on the degree of user acceptance. It is obvious that the research based on user preferences and historical habits will perform a better result on user satisfaction and acceptance. Thus, it is no need to make a comparison with any previous work from user acceptance. Additionally, the result of research focusing on security is simply either "yes" or "no". While in our work, due to the characteristics of our test dataset and the consideration of the "Ask" option, we perform three decisions. Although it is also not feasible to make a quantitative comparison with these binary classification models, we compare AutoPer+'s performance with our previous work [13] on the same dataset, and build several baseline approaches to verify the accuracy of the proposed algorithms, which clearly demonstrate the effectiveness of our approach.

C. SIMILAR-APP IDENTIFICATION

The reason why the coarse-grained category provided by the Google Play store is not feasible to identify the functionality and similar apps can be demonstrated as follows. On one hand, to satisfy the diverse needs of users, an app usually contains multiple functionalities, thus a specific category in the Google Play store is hard to adequately represent the functionalities of the app (i.e., *Facebook*). On the other hand, the apps with similar functionality may be classified into different categories. For instance, translation-oriented apps, such as *Microsoft Corporation* and *Google LLC*, are classified into categories of "business office" and "tool" respectively. Thus, it is unreachable to explore multiple functionalities just relying on a coarse-grained category. Similarly, a coarse-grained classification is not effective to identify similar apps. In our work, based on the **Level II** apps with a fine-grained category, we incorporate a deep learning machine into similar-app identification, which serves as an effective way to classify apps into fined-grained clusters. Moreover, to train our model, both **Level I** and **Level II** apps were collected from the Google Play store. Although the training data is limited to the apps with "category" in Google Play, AutoPer+ is generic to the apps with description texts, even some app stores do not provide categories, since our fine-grained classifier is a supervised model that takes the app's description as input.

VII. THREATS TO VALIDITY

Several threats may make impacts on the results of our study. First, the original training data is not large enough to reliably train an advanced machine learning classification model. Although we adopt a semi-supervised model to alleviate the deviation, a longer-term data collection method will be considered for a more accurate recommendation. Second, our evaluation for the recommender mainly comes from two aspects: manual analysis and participants' feedback. Due to the empirical nature of the evaluation, it might have introduced biases, as users' decisions would be affected by many factors, e.g., the limitations of permission understanding and users' privacy preference. However, such biases are common and unavoidable in evaluating privacy tools for real scenarios. For manual analysis, we reduce negative influence by synthesizing the opinions of all three evaluation experts, where a permission is decided by the most voted decision. Apart from that, to compensate the participants' bias, instead of simply providing a decision, we also present extra explanations for the decisions. Third, as discussed in Section III, AutoPer+ focuses on the 24 *"Dangerous"* permissions divided by the AOFU system. As a result, the *"Normal"* permissions will be granted by the system automatically. Essentially, although these *"Normal"* permissions are considered low-risk permissions for user's privacy, some of them are related to sensitive resources to some extent, e.g., Internet-related permissions. Hence, we plan to implement AuotPer+ for more data types, such as ACCESS_WIFI_STATE and READ_HISTORY_BOOKMARKS. Last, in the current training phase of our model, such as Word2Vec and BILSTM, the parameters, e.g., dropout rate and epoch are set still relying on traditional comparisons. We will carry out more advanced parameter optimizing approaches in the future for better classification performance.

VIII. RELATED WORK

The research on Android permission can be categorized into three lines of work: permission rationales and privacy security, permission requirements discovery, and learning-based permission recommendation and prediction.

A. PERMISSION RATIONALES AND PRIVACY SECURITY

Prior research [2]–[4], [7] has shown the ineffectiveness of AOI in protecting user privacy is raised by the lack of user attention and comprehension. To bridge the gap between apps and users, various works have attempted to study the problem of permission rationales. Some research discussed the factors that affect users in making the installation decisions [5], [23], [24] and explained how apps access to and share private data to enhance the user experience [25], [26]. In particular, research [27] attempted to explore the primary reason that text descriptions fail to prompt the use of sensitive resources, which are useful in improving users' cautiousness in privacy protection. For AOFU prompts, Bonné *et al.* [12] found the users' security decisions in the runtime system rely on various factors, e.g., app functionalities, app categories, and permission types. In addition, Liu *et al.* [28] conducted the first study on runtime-permission-group rationales to understand the patterns of permission-explaining behaviors by analyzing natural language rationales. Furthermore, Scoccia *et al.* [29] conducted a large-scale empirical study to investigate how users perceive the new run-time permission system of Android by inspecting user reviews. They also provided a set of insights to improve the Android run-time permission system.

Moreover, permission has been widely used to deal with security and privacy tasks, especially malicious application detection. For instance, Wang *et al.* [30] explored the risk induced by permission for detecting malicious apps. Sharma and Gupta [31] proposed a novel approach, RNPDroid, to categorize the risks into four levels, i.e., high, medium, low, and none. Furthermore, Shrivastava and Kumar [32] proposed an algorithm that combines permission vectors to identify benign and malware app permissions, and conducted a systematic literature survey on permission-based malware detection [33], which provide useful guidance for the malware and benign permissions requirement.

Different from exploring the runtime-permission rationale messages and malicious application detection, our work aims to help users automatically regulate permission requests by proposing a novel permission recommender.

B. PERMISSION REQUIREMENTS DISCOVERY

Existing research [34]–[36] has explored the space of permission requirements discovery from the perspective of app descriptions, which mainly focuses on the permission over-privilege issue. Specifically, Pandita *et al.* [34] proposed a framework, WHYPER, to perform semantic analysis to app text descriptions using NLP techniques. However, the scope of the WHYPER is limited to fixed vocabularies, API documents, and synonyms of keywords. To address this problem, Qu *et al.* [35] proposed AutoCog to extract semantics from descriptions without using API documents. Moreover, Gorla *et al.* [36] proposed the CHABADA framework applying an unsupervised clustering algorithm to extract text descriptions and identify API outliers, which attempts to check implemented app behavior against advertised app behavior.

All these efforts provide useful resolutions for permission requirements engineering. However, due to the development of Android mechanism and the Internet, prior works fail to achieve the desired privacy protection, where the diverse expressions within text descriptions make the rigorous semantic analysis ineffective. Similar to prior works, we also explore permission requirements from the perspective of app description. The difference is our work aims to explore the relationship between permission and app as well as similar apps by leveraging techniques of topic mining and deep learning.

C. PERMISSION RECOMMENDATION

As for AOFU, machine learning has exhibited its great power on building permission recommendation models [11], [22]. Various techniques for feature mining and learning have been used to predict users' preferences under contextual information. Nissenbaum [37] proposed the theory of "contextual integrity" suggesting that permission models should focus on sensitive resources. As a follow-up work, Barth *et al.* [38] attempted to systematize Nissenbaum's theory and extend the theory to smartphones. Based on prior works [37], [39], Wijesekera *et al.* [8], [9] conducted a field of study to explore how often the resources are accessed in practice and how much a contextualized permission model could improve dynamic permission granting. Moreover, they also performed a study using machine learning to analyze and predict user privacy decisions under contextual circumstances [22]. Closed to the work, Olejnik *et al.* [11] presented an advanced permission mechanism, SmarPer, to match users' privacy preferences with their historical behaviors by using contextual cues. Furthermore, Tasi *et al.* [10] designed TurtleGuard, a novel contextually-aware permission manager, to help users vary privacy preferences.

Existing studies increase the protection of contextual information by proposing new permission models. However, these learning-based permission recommendations require users' historical decisions. In contrast, our work helps users make decisions by mining existing app description, which is implemented and evaluated in practical scenarios.

IX. CONCLUSION

In this paper, we proposed AutoPer+, an autonomous permission recommendation model by reconciling app's multi-functionalities and privacy permission usage in similar apps. The AutoPer+ contains three main modules: a <multi-topic, permission> module for identifying the relationship between app's functionalities and requested permissions, an attentive BILSTM module for classifying the similarity apps, and a combination module for capturing the advantage of utility and privacy. The results of extensive experiments show that our tool achieves better performance in permission recommendation compared to conventional methods. Furthermore, we also deployed our tool to the devices of 37 participants in real scenarios. Instead of solely offering multiple recommendations (''Allow'', ''Deny'', or ''Ask''), a systematic explanation and a feedback mechanism are also designed to assist users to make as well as audit permission decisions. Moreover, the permission recommendations by AutoPer+ achieve considerable positive approvals by participants, further illustrating the effectiveness of the proposed approach. For future work, we plan to expand AutoPer+ by introducing contextual analysis of a dynamic recommendation.

ACKNOWLEDGMENT

(Hongcan Gao and Chenkai Guo contributed equally to this work.)

REFERENCES

[1] J. Clement. *Number of Available Applications in the Google Play Store From December 2009 to Jun. 2019*. Accessed: Jul. 3, 2019. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[2] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. A. Wagner, ''Android permissions: User attention, comprehension, and behavior,'' in *Proc. Symp. Usable Privacy Secur. (SOUPS)*, Washington, DC, USA, Jul. 2012, p. 3.

[3] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. M. Sadeh, and D. Wetherall, ''A conundrum of permissions: Installing applications on an Android smartphone,'' in *Proc. Financial Cryptogr. Data Secur. FC Workshops USEC WECSR*, Kralendijk, Bonaire, Mar. 2012, pp. 68–79.

[4] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, ''Permission evolution in the Android ecosystem,'' in *Proc. 28th Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Orlando, FL, USA, 2012, pp. 31–40.

[5] C. Gibler, J. Crussell, J. Erickson, and H. Chen, ''Androidleaks: Automatically detecting potential privacy leaks in Android applications on a large scale,'' in *Proc. 5th Int. Conf. Trust Trustworthy Comput. (TRUST)*, Vienna, Austria, Jun. 2012, pp. 291–307.

[6] S. Biswas, W. Haipeng, and J. Rashid, ''Android permissions management at app installing,'' *Int. J. Secur. Appl.*, vol. 10, no. 3, pp. 223–232, Mar. 2016.

[7] S. Biswas, K. Sharif, F. Li, and Y. Liu, ''3P framework: Customizable permission architecture for mobile applications,'' in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl.* Springer, 2017, pp. 445–456.

[8] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. A. Wagner, and K. Beznosov, ''Android permissions remystified: A field study on contextual integrity,'' in *Proc. 24th USENIX Secur. Symp.*, Washington, DC, USA, vol. 15, Aug. 2015, pp. 499–514.

[9] P. Wijesekera, A. Baokar, L. Tsai, J. Reardon, S. Egelman, D. Wagner, and K. Beznosov, ''The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences,'' in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 1077–1093.

[10] L. Tsai, P. Wijesekera, J. Reardon, I. Reyes, S. Egelman, D. A. Wagner, N. Good, and J. Chen, ''Turtle guard: Helping Android users apply contextual privacy preferences,'' in *Proc. 13th Symp. Usable Privacy Secur. (SOUPS)*, Santa Clara, CA, USA, Jul. 2017, pp. 145–162.

[11] K. Olejnik, I. Dacosta, J. S. Machado, K. Huguenin, M. E. Khan, and J.-P. Hubaux, ''SmarPer: Context-aware and automatic runtime-permissions for mobile devices,'' in *Proc. IEEE Symp. Secur. Privacy (SP)*, San Jose, CA, USA, May 2017, pp. 1058–1076.

[12] B. Bonné, S. T. Peddinti, I. Bilogrevic, and N. Taft, ''Exploring decision making with Android's runtime permission dialogs using in-context surveys,'' in *Proc. 13th Symp. Usable Privacy Secur. (SOUPS)*, 2017, pp. 195–210.

[13] H. Gao, C. Guo, Y. Wu, N. Dong, X. Hou, S. Xu, and J. Xu, ''AutoPer: Automatic recommender for runtime-permission in Android applications,'' in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Hlwaukee, WI, USA, Jul. 2019, pp. 107–116.

[14] M. Lui and T. Baldwin, ''langid. py: An off-the-shelf language identification tool,'' in *Proc. ACL Syst. Demonstrations*, 2012, pp. 25–30.

[15] O. Levy and Y. Goldberg, ''Neural word embedding as implicit matrix factorization,'' in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf.*, Montreal, QC, Canada, Dec. 2014, pp. 2177–2185.

[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, ''Distributed representations of words and phrases and their compositionality,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ''Efficient estimation of word representations in vector space,'' in *Proc. 1st Int. Conf. Learn. Represent. (ICLR)*, Scottsdale, AZ, USA, May 2013.

[18] Y. Goldberg and O. Levy, ''Word2vec explained: Deriving Mikolov et al.'s negative-sampling word-embedding method,'' 2014, *arXiv:1402.3722*. [Online]. Available: http://arxiv.org/abs/1402.3722

[19] S. Hochreiter and J. Schmidhuber, ''Long short-term memory,'' *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] A. Graves and J. Schmidhuber, ''Framewise phoneme classification with bidirectional LSTM and other neural network architectures,'' *Neural Netw.*, vol. 18, nos. 5–6, pp. 602–610, Jul. 2005.

[21] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, ''Hierarchical attention networks for document classification,'' in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 1480–1489.

[22] P. Wijesekera, J. Reardon, I. Reyes, L. Tsai, J.-W. Chen, N. Good, D. Wagner, K. Beznosov, and S. Egelman, ''Contextualizing privacy decisions for better prediction (and protection),'' in *Proc. CHI Conf. Hum. Factors Comput. Syst. (CHI)*, Montreal, QC, Canada, 2018, p. 268.

[23] W. Klieber, L. Flynn, A. Bhosale, L. Jia, and L. Bauer, ''Android taint flow analysis for app sets,'' in *Proc. 3rd ACM SIGPLAN Int. Workshop State Art Java Program Anal. (SOAP)*, Edinburgh, U.K., Jun. 2014, pp. 5:1–5:6.

[24] W. Enck, P. Gilbert, B. Chun, L. P. Cox, J. Jung, P. D. McDaniel, and A. Sheth, ''TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones,'' in *Proc. 9th USENIX Symp. Operating Syst. Design Implement. (OSDI)*, Vancouver, BC, Canada, Oct. 2010, pp. 393–407.

[25] M. Harbach, M. Hettig, S. Weber, and M. Smith, ''Using personal examples to improve risk communication for security & privacy decisions,'' in *Proc. Conf. Hum. Factors Comput. Syst. (CHI)*, Toronto, ON, Canada, Apr./May 2014, pp. 2647–2656.

[26] P. G. Kelley, L. F. Cranor, and N. M. Sadeh, ''Privacy as part of the app decision-making process,'' in *Proc. ACM SIGCHI Conf. Hum. Factors Comput. Syst. (CHI)*, Paris, France, Apr./May 2013, pp. 3393–3402.

[27] T. Watanabe, M. Akiyama, T. Sakai, and T. Mori, ''Understanding the inconsistencies between text descriptions and the use of privacy-sensitive resources of mobile apps,'' in *Proc. 11th Symp. Usable Privacy Secur. (SOUPS)*, Ottawa, Canada, Jul. 2015, pp. 241–255.

[28] X. Liu, Y. Leng, W. Yang, W. Wang, C. Zhai, and T. Xie, ''A large-scale empirical study on Android runtime-permission rationale messages,'' in *Proc. IEEE Symp. Vis. Lang. Hum.-Centric Comput. (VL/HCC)*, Lisbon, Portugal, Oct. 2018, pp. 137–146.

[29] G. L. Scoccia, S. Ruberto, I. Malavolta, M. Autili, and P. Inverardi, ''An investigation into Android run-time permissions from the end users' perspective,'' in *Proc. 5th Int. Conf. Mobile Softw. Eng. Syst. (MOBILESoft)*, 2018, pp. 45–55.

[30] W. Wang, X. Wang, D. Feng, J. Liu, Z. Han, and X. Zhang, ''Exploring permission-induced risk in Android applications for malicious application detection,'' *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 11, pp. 1869–1882, Nov. 2014.

[31] K. Sharma and B. B. Gupta, ''Mitigation and risk factor analysis of Android applications,'' *Comput. Electr. Eng.*, vol. 71, pp. 416–430, Oct. 2018.

[32] G. Shrivastava and P. Kumar, ''Android application behavioural analysis for data leakage,'' *Expert Syst.*, to be published.

[33] G. Shrivastava, P. Kumar, D. Gupta, and J. J. Rodrigues, ''Privacy issues of Android application permissions: A literature review,'' *Trans. Emerg. Telecommun. Technol.*, to be published.

[34] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, ''WHYPER: Towards automating risk assessment of mobile applications,'' in *Proc. 22th USENIX Secur. Symp.*, Washington, DC, USA, Aug. 2013, pp. 527–542.

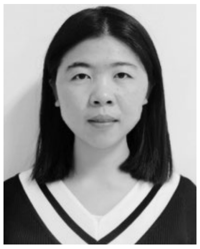
[35] Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, ''AutoCog: Measuring the description-to-permission fidelity in Android applications,'' in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Scottsdale, AZ, USA, Nov. 2014, pp. 1354–1365.

[36] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, ''Checking app behavior against app descriptions,'' in *Proc. 36th Int. Conf. Softw. Eng. (ICSE)*, Hyderabad, India, May/Jun. 2014, pp. 1025–1035.

[37] H. Nissenbaum, "Privacy as contextual integrity," *Wash. L. Rev.*, vol. 79, p. 119, 2004.
[38] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum, "Privacy and contextual integrity: Framework and applications," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, Berkeley, CA, USA, May 2006, pp. 184–198.
[39] A. P. Felt, S. Egelman, M. Finifter, D. Akhawe, and D. A. Wagner, "How to ask for permission," in *Proc. 7th USENIX Workshop Hot Topics Secur. (HotSec)*, Bellevue, WA, USA, Aug. 2012, p. 7.

HONGCAN GAO received the M.S. degree from the Hebei University of Technology, in 2017. She is currently pursuing the Ph.D. degree with the College of Computer Science, Nankai University. Her research interests include software analysis on mobile apps and software security.

CHENKAI GUO (Member, IEEE) received the Ph.D. degree from Nankai University, in 2017. He is currently an Assistant Professor with the College of Computer Science, Nankai University. His research interests include software analysis on mobile apps, information security, and intelligent software engineering.

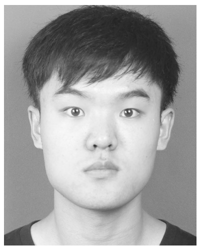


DENGRONG HUANG (Member, IEEE) received the B.E. degree from Nankai University, in 2017. She is currently pursuing the master's degree with the College of Computer Science, Nankai University. Her research interests include mobile apps analysis and intelligent software engineering.

XIAOLEI HOU received the B.E. degree from the Hebei University of Technology, in 2018. He is currently pursuing the master's degree with the College of Computer Science, Nankai University. His research interests include recommendation systems and machine learning.

YANFENG WU received the B.E. degree from Nankai University, in 2017. He is currently pursuing the Ph.D. degree with the College of Artificial Intelligence, Nankai University. His research interests include deep learning, software engineering, and speaker recognition.

JING XU (Member, IEEE) received the Ph.D. degree from Nankai University, in 2003. She is currently a Professor with the College of Artificial Intelligence, Nankai University. Her current research interests include intelligent software engineering and medical data analysis. She received the Second Prize with the Tianjin Science and Technology Progress Award, in 2017 and 2018.



ZHEN HE received the B.E. degree from Nantong University, in 2018. He is currently pursuing the master's degree with the College of Artificial Intelligence, Nankai University. His main research interests include video processing and machine learning.

GUANGDONG BAI (Member, IEEE) 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 currently a Senior Lecturer with The University of Queensland. His research interests include cyber security, software engineering, and machine learning.