International Journal of Information Security manuscript No.
(will be inserted by the editor)

Attention: There is an Inconsistency between Android Permissions and Application Metadata!

Huseyin Alecakir · Burcu Can · Sevil Sen

Received: date / Accepted: date

Abstract Since mobile applications make our lives easier, there is a large number of mobile applications customized for our needs in the application markets. While the application markets provide us a platform for downloading applications, it is also used by malware developers in order to distribute their malicious applications. In Android, permissions are used to prevent users from installing applications that might violate the users' privacy by raising their awareness. From the privacy and security point of view, if the functionality of applications is given in sufficient detail in their descriptions, then the requirement of requested permissions could be well understood. This is defined as description-to-permission fidelity in the literature. In this study, we propose two novel models that address the inconsistencies between the application descriptions and the requested permissions. The proposed models are based on the current state-of-art neural architectures called attention mechanisms. Here, we aim to find the permission statement words or sentences in app descriptions by using the attention mechanism along with recurrent neural networks. The lack of such permission statements in application descriptions creates a suspicion. Hence, the proposed approach could assist in static analysis techniques in order to find suspicious apps and to prioritize apps for more resource intensive analysis techniques. The experimental results show that the proposed approach achieves high accuracy.

Huseyin Alecakir
Middle East Technical University, Ankara, Turkey
E-mail: huseyinalecakir@gmail.com

Burcu Can
Research Group in Computational Linguistics
University of Wolverhampton, UK
E-mail: b.can@wlv.ac.uk

Sevil Sen
WISE Lab, Hacettepe University, Ankara, Turkey
E-mail: ssen@cs.hacettepe.edu.tr

Keywords Android · Security · Description-to-permission fidelity · Natural language processing · Recurrent neural networks · Attention mechanism

1 Introduction

With the developments in mobile technology, mobile devices have become an integral part of our lives. They provide many useful functionalities through mobile applications such as reading/writing e-mails, mobile banking, video conferencing. It is reported that there have been almost 2.5 million available applications in the official Android market (Google Play), and almost 2 million available applications on the official iOS market (Apple App Store) in the second quarter of 2019 [48]. With the increase in the number of mobile applications, mobile malware developers have also emerged in order to harm such devices and steal mobile users' information. According to McAfee Mobile Threat Report [35], mobile malware has continued to increase in scope and complexity in 2019.

A primary line of defense against such malicious attempts is to prevent them from entering market stores. Two common types of malware analysis and detection techniques are static and dynamic analysis. While in static analysis the code and the application package are analyzed without running the code, in dynamic analysis, runtime behaviours of applications are monitored. Both techniques have advantages and disadvantages compared to each other. For example, static analysis is not resilient to some evasion techniques such as obfuscation [46][47], dynamic code loading [55][10]; hence it might not detect new attacks nor even new variants of existing attacks. Mobile malware can also hide from dynamic analysis and may not trigger its malicious part. Moreover, dynamic analysis might not be affordable on some mobile devices due to the significant limitations of such de-

vices in terms of power consumption. Therefore, security solutions on mobile devices known as anti-malware systems mainly rely on static analysis. On the other hand, Google Play Protect is known to employ both techniques for application analysis on the Android official market [41].

One of the key points of the Android security mechanism is permissions. Android applications must request permission in order to access sensitive user data (such as contacts, call logs, and SMS) and some system features (such as camera, microphone, and Internet) [5]. Permissions required by the app must be listed in the application's manifest file. If the application requires a dangerous permission that could potentially affect the user's privacy or the device's normal operation, the permission must be granted explicitly by the user. The way Android asks users to grant dangerous permissions has changed with Android 6.0. Before Android 6.0, all dangerous permissions had to be granted during the installation time. In Android 6.0 and higher versions, users are asked to grant dangerous permissions at runtime.

Unlike traditional application distribution mechanisms, Android applications are distributed centrally, so Android markets beside application packages contain application metadata such as the definition of applications, user scores, and user reviews. Such metadata could be useful for security purposes. In recent years, studies that use metadata for malware detection have been introduced [34][12]. Another significant usage of such metadata is to discover inconsistencies between permissions and descriptions of applications [42][45][23]. From the privacy and security point of view, if the functionality of applications is given in sufficient detail in their descriptions, then the requirement of requested permissions could be well understood. This is named as description-to-permission fidelity in the literature [45]. For instance, if an application uses *SEND_SMS* permission, the developer must explicitly state in its description that why the application needs to send an SMS. Fortunately, metadata provided in Android markets provides us meaningful information to assess consistency between descriptions and permissions automatically. This new method for application analysis could be used as a complementary approach to other static and dynamic analysis techniques.

This study proposes a new approach for the description-to-fidelity problem. It could assist security researchers, market stores, end-users and developers in different ways. First of all, the underlying assumption here is that the use of dangerous permissions must be explained in application definitions, and the lack of this information in definitions creates a suspicion. Applications that are presumed to be suspicious as a result of static analysis techniques could then be analyzed by dynamic and manual analysis techniques. The proposed mechanism could help prioritizing applications to be analyzed by such resource-intensive analysis techniques, which could reduce the time and cost of application analysis in the markets. In that sense, the proposed approach could complement other static analysis techniques by adding the analysis of metadata. Even though market stores do not check the accuracy of app descriptions at the moment, the proposed approach can help automating this process by checking that dangerous permissions are not explained in the descriptions. Moreover, the proposed approach could be adapted to be applied on privacy policies. Google expects developers to be transparent about disclosing the collection, use, and sharing of personal and sensitive data, and limiting the use of such data to the purposes disclosed, and the consent provided by the user [27]. Such information is expected to be given in privacy policies and Google announced that it is going to remove applications which do not comply with Google's User Data Policy starting from March, 15 2017 [15]. Furthermore, the proposed approach could assist users before installing applications. It could also be used by developers in order to improve their descriptions and create user-understandable descriptions.

In this study, we use recurrent neural networks to detect whether a dangerous permission required by an application is explained in its description. To this end, we propose a model that uses gated recurrent unit (GRU) networks for representing description sentences and texts. Moreover, we aim to detect permission related words in descriptions using a cognitive inspired attention mechanism using neural networks which has shown superior performance in many natural language processing tasks recently such as in machine translation [11], document classification [56], semantic parsing [20, 24], and language modeling [19].

This current study makes the following contributions:

- We introduce two models based on deep recurrent neural networks to detect inconsistencies between requested permissions and descriptions. While the first model called *sentence-based* model aims to identify permission sentences in a description as other proposals in the literature [42][45][23], the *document-based* model based on hierarchical attention network aims to represent a description as a whole for the first time in the literature.
- Attention mechanism is firstly investigated for the use of description-to-permission fidelity problem. This mechanism is incorporated in the neural network architecture in order to learn the permission-related words in the app descriptions.
- A new dataset called DesRe for assessing description-to-permission fidelity problem is introduced and shared with the community [1]. This dataset contains labelled description sentences of applications for *READ_CONTACTS*, *RECORD_AUDIO*, and *STORAGE* permissions. Moreover, five reviews declared to be most helpful by other

[1] https://wise.cs.hacettepe.edu.tr/projects/security-risks/dataset/

users for each application are also shared for exploring the effects of user reviews in further studies.
- The proposed models are evaluated by using both the AC-NET dataset [23] and the newly proposed DesRe dataset and compared with other proposals in the literature. While the sentence-based model produces comparable results with the current state-of-the-art methods, the document-based model significantly outperforms the sentence-based proposals.

The remainder of the paper is organized as follows: Section 2 summarizes the related approaches in the literature. Section 3 introduces the new dataset called DesRe. Section 4 gives background information on neural networks and recurrent neural networks. Section 5 introduces the proposed methods based on GRUs and gives implementation details of these methods. The experimental results of the proposed approach are given and discussed in Section 6. Finally, Section 7 is devoted to concluding remarks and future work.

2 Related Work

Application screenshots and descriptions are the very first metadata that users meet before the installation of a mobile application. This fact makes application descriptions an indispensable part of the communication between application developers and users. Hence, app descriptions are expected to include enough information about the requirement of a requested dangerous permission, which is defined as description-to-permission fidelity [45].

The first work on assessing the description-to-permission fidelity proposes a framework called WHYPER [42] that uses natural language processing (NLP) techniques. It is proposed as a means to alleviate the shortcomings of a keyword-based approach such as confounding effects and semantic interference. Confounding effects result from words that can have different meanings. Semantic interference describes the usage of a permission without using a particular word. WHYPER [42] creates a semantic graph for each permission by using the application programming interface (API) documents and a lexical database called WordNet [39]. Using the semantics graphs, the model identifies whether the need of a permission is stated in the description or not. Watanabe et al. [52] proposes a keyword-based approach called ACODE. Since ACODE does not require labelling app descriptions, the keyword-based approach is claimed to be appealing for large datasets. By combining static analysis and text analysis, ACODE performs better than the keyword-based approach used for comparison in WHYPER [42] and produces comparable results with WHYPER. In addition, unlike other studies in the literature, it can be applied to different languages without much effort and modification.

Qu et al. [45] discusses the applicability of WHYPER, since some permissions might not have any API documents related. Furthermore, it is difficult to extract the complete semantic patterns of some permissions from API documents and this process is not fully-automated. Therefore, they propose a fully-automated framework called AUTOCOG [45], in which semantic information is obtained only from the descriptions. In AUTOCOG, the semantic relatedness between descriptions and permissions is measured using Explicit Semantic Analysis (ESA) [25]. Instead of using a dictionary-based corpus like WordNet [39] as done in WHYPER, ESA uses an extensive knowledge base (i.e., Wikipedia) in order to create vectorial representation of the text. While AUTOCOG performs much better than WHYPER, since it uses unsupervised learning, it could extract semantic relationships that may not actually exist, which may lead to false positives.

The closest work to the current study in terms of the applied technique has very recently been proposed by Feng et al. [23]. The framework called AC-Net also utilizes recurrent neural networks (RNNs) in order to learn and detect semantic relations. Labeled descriptions for 11 different permission groups are used for training. Predictions for learned permissions are generated as probability distributions in the model. Since RNNs are able to remember previous inputs, they are good at modelling sequential data. Although RNNs theoretically can manage to encode long sequences of input, practically, it is not easy to learn long-term dependencies using simple RNNs. This problem is defined as the vanishing gradient problem in RNNs. Gated Recurrent Units (GRU) [18] as used in AC-Net, offer a solution to RNNs' vanishing gradient problem by employing gates in order to adjust information flow in the network. Therefore, some data is allowed to flow within the network, whereas some of them are forgotten in the network. In this study as concurrently and independently developed from AC-Net, Gated Recurrent Units (GRU) are also used as RNN units as a solution to the same problem. Furthermore, in this study, an attention mechanism is used to learn the contribution of each word to the meaning of the description, thereby extracting the meaning of the description dependent on the permission-related words in the description. We hypothesize that each word in a description would have different contribution to the meaning and should not be handled equally as done in previous work. The results also support our hypothesis. We also define a hierarchical attention network that learns the weight of each sentence in the description, based on the meaning of each sentence within the description which is also learned out of each word in the sentence. Therefore, a two-level attention mechanism incorporated in order to extract the meaning of each description hierarchically in the proposed document-based model. Therefore, our work deviates from their work with these features.

Felt et al. [22] analyze users' behaviors and their awareness on the Android permission mechanism. Although 42% of users are not aware of the permission mechanism at all and, 42% of users are aware of the Android permission mechanism but do not look at the permissions requested during app installation, only 17% of the participants actually notice requested permissions. One way to benefit from this fact is to use the experience and attention of these cautious users. One of the most effective ways of doing this is to make use of user reviews. However, there are only a few studies on exploring the effects of user reviews in Android applications. Autoreb [32] makes application-level behavior inference based on security and privacy related user reviews. By doing so Autoreb has introduced the concept of review-to-behavior to the literature. The authors use text mining, information retrieval, and machine learning techniques to analyze user reviews and classify applications into four different categories: spamming, financial issues, over-privileged permission, data leakage. PACS [53] classifies applications into ten categories based on the application descriptions and user reviews using Support Vector Machines (SVM) [30] for detecting permission abuse in apps. It shows similarity to CHABADA [28] since it also uses descriptions to classify applications. Then, it builds maximum frequently used permission itemsets for each category by using Apriori Algorithm [49]. Using description and user reviews of a new application, PACS firstly finds the application's category and lists the permissions that are expected to be requested by the application. Any other permission requested by the application is considered as a suspicious request. Very recently, Nguyen et al. [40] investigate the relationship between security and privacy related application updates and user reviews. Results show that 60.77% of the security and privacy related reviews trigger a security and privacy related update. A very recent study called SmartPI [51] aims to find permission indications in user reviews by using unsupervised learning based on the assumption that user reviews are more representative than app descriptions. Firstly, representative words of permissions are extracted from apps, app descriptions, permission docs, API docs and user reviews. The list of words is enhanced with their synonyms [39] and their co-occurrences in descriptions. Then, both user reviews and permission-representative words are formed as feature vectors using word2vec [36]. Then, each review's similarity to the permission-representative words is calculated by using cosine similarity. According to the estimated similarities, funtionality-relevant user reviews are selected. Reviews are grouped into 10 (number of permissions) clusters using Biterm Topic Model (BTM) [17]. Finally, a review is mapped to a cluster, hence to a permission. The proposed study is compared with AUTOCOG [45], and shows slightly better results.

Some recent studies explore the use of privacy policy for enhancing the description-to-behaviour fidelity [59][58]. TAPVerifier employs a two-stage analysis in order to complement other studies such as AutoCog [45] and decreases their false positives: privacy policy analysis, code and permission analysis. While privacy policy analysis extracts the necessity of a requested permission from the app's privacy policy, code analysis extracts the permissions used in the code by using PScout [9], which maps API calls with permissions. The permissions of the third party libraries are also included in this study as one of the improvements on the previous study of the same authors [58]. Based on the observation that users cannot understand the purpose of permissions based only on descriptions, a recent study focuses on inferring this information from app's code and behaviours [50]. In the static analysis, two types of features are extracted from the code: app-specific features that include permission related APIs, Intents, Content Providers, and text-based features. Text-based features are extracted from identifiers (package, class, method, and variable names) in the code. TF-IDF vectors of the word roots in the identifiers that are obtained after pre-processing identifiers are taken as text-based feature vectors. All features are collected from custom code, then given to classifiers for assigning apps to one of the categories of purposes of the following two permission uses: contacts and location. The results show that text-based features are powerful enough for understanding the purpose of the permission's use and app-specific features are found to be supportive.

Because of the importance of application descriptions, there are also studies on automatic generation of application descriptions (AUTOPPG [60], DESCRIBEME [61]) and automatic creation of informative text (DREBIN [8]) by prioritizing security and privacy concerns. Recently, metadata of applications are also used for detecting malicious mobile applications. Martin et al. [34] propose a detection system called ADROIT that uses metadata besides permissions in order to discriminate malicious applications from benign ones. Another study [12] also uses metadata such as application category and description besides API calls and permissions for malware detection.

3 DATASET

Since we use supervised machine learning techniques in order to find inconsistencies between requested permissions and application descriptions, we have created an annotated description dataset and selected the permissions for this dataset with careful attention. Not only users need to understand the functionality of the selected permissions by reading app descriptions, but also such permissions need to have access to the critical resources (dangerous permissions [3]). So, they

are expected to be explicitly given in the descriptions. Furthermore, priorities are given to the permissions used in the previous studies [52, 42, 45, 23] for comparison. Based on these criteria, two permissions in our dataset are selected among the permissions used by all previous studies mentioned above: *RECORD_AUDIO* and *READ_CONTACTS*. In total, three permissions are included in our dataset due to difficulty of labelling all dangerous permissions manually. The last selected permission for our dataset is the STORAGE permission group. This group contains *READ_EXTERNAL_STORAGE* and *WRITE_EXTERNAL_STORAGE* permissions which give access to external storage. The reason of taking STORAGE as a permission group rather than one specific permission from that group is that any application granted for *WRITE_EXTERNAL_STORAGE* permission is also granted implicitly for *READ_EXTERNAL_STORAGE* permission [6]. The STORAGE permission group is one of the most requested permissions in applications [45, 23]. It is also found to be among the most comprehended permissions by users [22]. Moreover, it is the most mentioned permission in security related user reviews [40] which indicates that users are highly concerned about applications' access to external resources. Therefore, this permission group is particularly included in the dataset for analyzing the effects of reviews on description-to-permission fidelity in the future. The manually labelled dataset is called DesRe (DEScriptions and REviews of Android applications) and shared with the community [2].

There were only two datasets at the time we had started labelling: WHYPER [42] and AUTOCOG [45]. Since the WHYPER dataset was very limited for the training purpose of the current study (total 581 applications for three permissions) and only a part of the AUTOCOG dataset was able to be obtained from its authors, a new dataset is introduced in this study. The AC-Net dataset [23] which was independently introduced from our dataset is also used for comparison in the experimental results. Differently from the previous studies, AC-Net [23] labels the same 1417 applications for 11 permission groups. However some of the applications might not have requested these permissions. On the other, in this study, for each permission a group of applications is selected among the applications that have requested these permissions. Moreover, for each application, user reviews that are found to be most helpful by other users are added to the dataset. As an ongoing study, five user reviews that contain statements regarding the use of permissions have been labelling. Applications for each permission are downloaded from the official Android market by using the criteria below:

- Among the most popular free applications, which are installed at least 10,000 times.
- Among the applications with description that are at least 500 words long.

[2] https://wise.cs.hacettepe.edu.tr/projects/security-risks/dataset/

Table 1: Category Information of the DesRe Dataset

Permissions	# of Categories
READ_CONTACTS	30
RECORD_AUDIO	32
STORAGE	23

For each permission, at least 1000 applications from various categories (as shown in Table 1) are downloaded. After filtering out invalid applications (such as having non-English sentences in their descriptions), application descriptions are split into sentences by using Natural Language Toolkit (NLTK) [13] and are annotated manually by two people in order to indicate whether the requested permission is mentioned in the application description (to be more precise, description sentences) or not. Available datasets, namely WHYPER[42] and AC-Net [23] are also analyzed in the current study. Both datasets are investigated for *READ_CONTACTS* and *RECORD_AUDIO* permissions, but for the STORAGE permission group only AC-Net is analyzed since WHYPER [42] does not contain any data for this group. There were some wrongly labeled sentences and also similar expressions that are labeled either positive or negative within datasets of related studies. For example, in the WHYPER dataset [42] while the following sentence *"This is a simple voice recorder."* was marked as positive, the following one *"RecForge is a high quality sound recorder (far more better than default sound recorder)."* was marked as negative. There are also conflicts in labelling in different datasets. One of the most important ones of such conflicts is on tagging a sentence about recording video for the *RECORD_AUDIO* permission. While expressions that include recording video are considered as positive samples in AC-Net, WHYPER tagged them as negative samples. However, the *RECORD_AUDIO* permission is required to implement a video recording application [2]. In the current study, the Android Developer Guide [1] is taken as a reference in labelling for each permission. Furthermore, conflicts in labelling are re-reviewed by a third person.

Another conflict occurs in tagging phrases such as "share ... via social media accounts" as permission sentences for the READ_CONTACTS permission. While WHYPER and AC-Net tag such sentences as permission sentences, these apps are sending their simple data to other apps. As we know from the Intent mechanism of Android, an application that provides its users a sharing mechanism through the medium of external application does not need permissions which are already required by an external application [4][7]. In other words, the external application is responsible for accessing contacts data of its user. Therefore, it is tagged as statement

sentences in our dataset. Again, for such cases, the Android Developer Guide [1] is followed. The details of each dataset are summarized in Table 2. All datasets have include both permission sentences and statement sentences according to if they include the indication of permission or not in the sentence respectively. As it is seen in the table, in DesRe, there are more sentences and permission sentences (pSents in the table) compared to other datasets.

4 A Primer on Deep Learning

4.1 Multi-Layer Perceptrons

Neural networks are a type of parameterized function approximators. They have the ability of a highly non-linear mapping between the input and output. Here we introduce multi-layer perceptrons (MLP) and discuss how inference is performed in neural networks. Figure 1 demonstrates a typical MLP with two hidden layers. Each neuron is linked to another neuron in the following layer with a weight, which is illustrated with an arc.

In Figure 1, a typical fully-connected MLP is given. There are 4 neurons in the input layer. So the input layer is represented by a 4-dimensional vector; i.e. $\vec{x}$. In a typical fully-connected neural network, there are parametrized weight matrices $W^i \in \mathbb{R}^{d_i^{in},d_i^{out}}$ and a bias term $b^i \in \mathbb{R}^{d_i^{out}}$ for each layer i. Here, d_i^{in} is the dimensionality of the input vector and d_i^{out} is the dimensionality of the output vector in layer i.

An MLP with two hidden layers is formally defined as follows:

$$\begin{aligned} MLP(x) &= y \\ h^1 &= g^1(xW^1 + b^1) \\ h^2 &= g^2(h^1W^2 + b^2) \\ y &= h^2W^3 \\ x \in \mathbb{R}^{d^{in}}, W^1 \in \mathbb{R}^{d^{in},d^1}, &b^1 \in \mathbb{R}^{d^1}, W^2 \in \mathbb{R}^{d^1,d^2}, b^2 \in \mathbb{R}^{d^2} \end{aligned} \tag{1}$$

where g^1 and g^2 denote non-linear activation functions; h^1 and h^2 correspond to the first and second hidden layer vectors, respectively.

4.2 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a type of artificial neural networks mainly designed to handle sequential data. The primary difference with MLPs is that RNNs have shared parameters for each input feature that enables passing information from history to the future time steps.

RNNs take an arbitrarily sized sequential input and return a fixed-sized output vector at each time step. They encode the input in the current time step while combining the present with the output obtained from the previous time step thereby remembering the history. RNNs can be built in many ways, but in all forms they have a recurrent function. Here, we define an RNN function that takes a sequence of d^{in}-dimensional vectors $\vec{x}_{1:t}$ in t time steps and returns a d^{out}-dimensional output vector $\vec{y_t}$ in the tth time step. The mathematical definition of an RNN is given as follows:

$$\begin{aligned} RNN(x_{1:t}) &= y_t \\ h_t &= R(h_{t-1}, x_t) \\ y_t &= O(h_t) \\ y_t \in \mathbb{R}^{d^{out}}, &x_t \in \mathbb{R}^{d^{in}} \end{aligned} \tag{2}$$

At each time step t, RNN takes two input vectors: a state vector h_{t-1} that comes from the previous time step and an input vector x_t in the current time step. At each time step, the function R computes the current state vector h_t. Finally, the function O computes the output vector y_t at the tth time step. An illustration for an RNN is given in Figure 2.

5 Model

We propose two neural network models to infer the required permissions from the metadata of a mobile application to detect any inconsistencies between the requested permissions and the application data. In both models, we use descriptions to detect whether a permission required by an application is explained or not. The descriptions are textual data and all sequential by definition. The first model is sentence-based and the compositional meaning of each sentence is represented by a low dimensional vector which is learned out of the words that make the sentence. The second model is document-based and the compositional meaning of each description is obtained by the sentences that make the description, where the meaning of each sentence is also inferred analogously to the sentence-based model.

Recurrent neural networks (RNNs) [21] have shown superior performance on sequential data in the last decade. Unlike the feedforward neural networks, RNNs can make decisions based on the earlier input due to their internal memory. However, RNNS are incapable of learning long sequences because of vanishing gradients problem during backpropagation training. Long-short Term Memory Networks (LSTMs) [31] and Gated Recurrent Neural Networks (GRUs) [18] have been introduced to tackle with the vanishing gradients problem with extra gates in the neural architectures. Those gates allow only the relevant information to pass through the network and forget the irrelevant bits. Therefore, the history is filtered through the gates.

Table 2: Outline of Datasets

Dataset	READ_CONTACTS			RECORD_AUDIO			STORAGE		
	#Apps	#Sents	#pSents	#Apps	#Sents	#pSents	#Apps	#Sents	#pSents
Whyper	190	3379	235	200	3822	245	-	-	-
AC-NET	951	17,353	937	350	6371	319	1304	23,101	1338
DesRe	832	25,011	1740	1008	31,989	2224	801	25,909	764

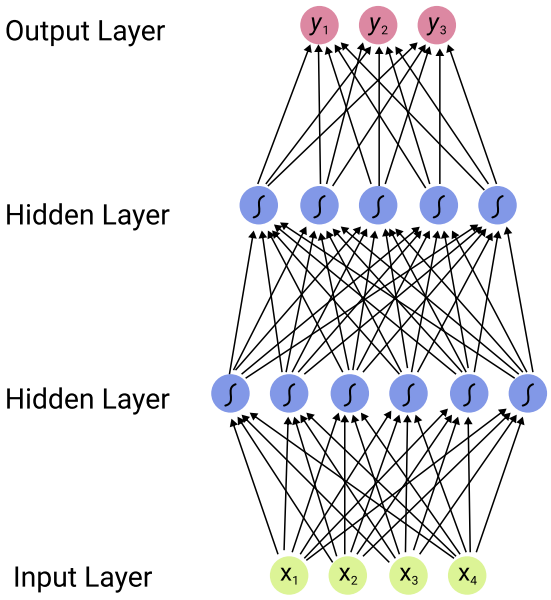


Fig. 1: A fully-connected multi layer perceptron (MLP) with two hidden layers where x denotes the input variables, y denotes the output variables, and h denotes the hidden variables in the network.

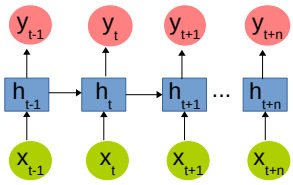


Fig. 2: A typical recurrent neural network (RNN) where x denotes the input variables, y denotes the output variables, and h denotes the hidden variables in the network.

Since we deal with application descriptions that involve long sequences, we utilize GRUs in this study. GRUs are slightly less complex compared to LSTMS with one less gate in their architecture. Therefore, their computational complexity is lower than that of LSTMs.

The overview of the proposed model is presented in Figure 3. The training is illustrated on the left side of the figure, whereas the right part shows the testing. The preprocessing tasks are applied to both training and testing data, which are described below.

5.1 Preprocessing

Prior to processing the sequential data, we preprocess application descriptions. Those preprocessing tasks involve sentence tokenization, word tokenization, punctuation removal, stopwords elimination, non-alpha characters removal, and stemming[3].

[3]We use Porter stemmer [44].

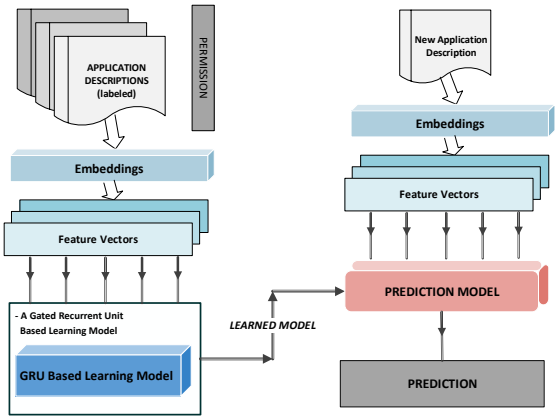


Fig. 3: Model Overview

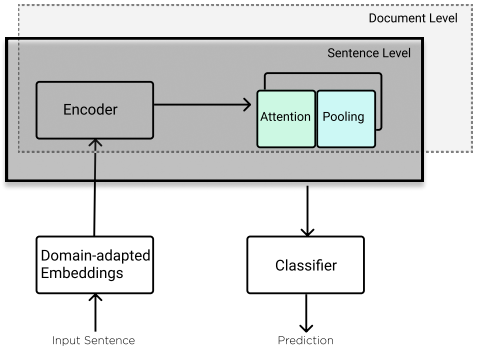


Fig. 4: A high-level description of both models.

5.1.1 Sentence Tokenization

In the sentence tokenization step, a document is split into sentences. There could be many types of symbols indicate the segmentation point. In English, a period does not always mark the end of a sentence; it can also be part of expressions such as shorthand notations and periods between numbers. Apart from the period, many other symbols mark the sentence splitting point such as punctuation marks (e.g. '.', '!', '?') and bullet points. We use the NLTK library[4] in Python for this task. NLTK is a standard library for many natural language processing tasks including such preprocessing tasks. The sentence tokenizer provided by NLTK uses an unsupervised algorithm to build a sentence boundary detection model. This method has been shown to work well for many European languages including English.

5.1.2 Text Cleaning & Word Tokenization

We follow standard text cleaning procedures step by step. We use regular expressions to remove URLs and e-mails addresses. Then, we use the Python demoji library to find or remove emojis from a blob of text. We filter out high-frequency words, i.e., stop words, to eliminate words that likely offer little meaning. We make use of 127 stop words provided by NLTK. After stop word elimination, we remove punctuations and non-alpha characters from sentences. Finally, we applied stemming techniques to reduce inflected or derived word forms to their root form. Porter's stemming algorithm has repeatedly been shown to be empirically very useful for NLP tasks. Finally, we make use of the NLTK library for the detection of word boundaries.

[4] https://www.nltk.org/

5.2 Word Representations

The syntactic and semantic features of every word within a sentence are represented by low dimensional representation vectors, which are learned using the distributional characteristics of words in a large document, thereby leading words with similar meanings to have similar representations in the space. We manually collected $421,223$ Android application descriptions from Google Play Store for this purpose. There are various neural language models proposed in the last years [37, 36, 14, 29]. In our proposed model, we trained the distributional word representations using Skip-gram model of Word2vec [36], which will be used as feature vectors in the model. Since those word embeddings are trained on application descriptions particularly, they can be considered as domain-adapted word embeddings. We also make use of pre-trained word vectors, which is provided by Mikolov et al. [38], trained on Common Crawl and Wikipedia using fastText.

5.3 Sentence-Based Encoder

Word embeddings of each sentence are fed into a GRU to have a compositional representation of the sentence. To this end, we use a bidirectional GRU, where one GRU processes the words in a sentence from the beginning till the end, and another GRU processes the words in a sentence from the end till the beginning in the reverse order as given below:

$$\begin{aligned} x_{it} &= W_e w_{it}, t \in [1, T] \\ h_{it} &= BiGRU(x_{it}), t \in [1, T] \end{aligned} \tag{3}$$

where w_{it} represents the word in the ith sentence at the time step t. Each sentence s_i contains T_i number of words. x_{it} is the embedding of the word w_{it}, and W_e denotes the embedding matrix for the corpus. We obtain the hidden representation of x_{it} using bidirectional GRU, and h_{it} denotes the hidden representation.

Then we apply a word-level attention mechanism [11, 54] to extract critical words that contribute to the meaning of a sentence significantly, which are particularly related to a permission statement. We adopt the word-level attention mechanism [57] for that purpose.

An attention network is normally built on a multi-layer perceptron (MLP). Therefore the mathematical model of the attention mechanism is given as follows:

$$\begin{aligned} u_{it} &= tanh(W_w h_{it} + b_w) \\ \alpha_{it} &= \frac{exp(u_{it}^\intercal u_w)}{\sum_i exp(u_{it}^\intercal u_w)} \\ s_i &= \sum_i \alpha_{it} h_{it} \end{aligned} \tag{4}$$

where W_w and b_w are MLP parameters (i.e. W_w corresponds to the weights of the hidden layer and b_w stands for the bias parameter). u_{it} is the hidden representations that are produced by the MLP for each h_{it}. The MLP has a softmax function applied in the output layer. Therefore, α_{it} corresponds to the softmax probabilities, where u_w denotes the word-level context vector. Here we use s_i for the compositional representation of the sentence which is the weighted sum of the hidden representations of word vectors. Therefore, each word is weighted by the attention mechanism to put emphasis on the words that are related to permissions.

AC-Net [23] highlights the importance of dimension reduction using pooling operations. We also validated it empirically. Besides computing sentence vectors through the attention network, we also add information extracted by global max and mean pooling.

Pooling is defined mathematically as follows:

$$s_i^{w/pooling} = s_i \circ h_{GMP}^s \circ h_{GAP}^s \tag{5}$$

where h_{GMP}^s and h_{GAP}^s refer to the hidden representations obtained by global max and mean pooling. We concatenate sentence hidden representation s_i with h_{GMP} and h_{GAP} to obtain the final representation $s_i^{w/pooling}$.

5.4 Document-Based Encoder

Our document-based model is built upon the model introduced by Yang et al. [57], which was proposed for document classification task. Hierarchical attention networks have shown great success in document classification [57] and sentiment analysis [62, 33].

The primary difference between the sentence-based and the document-based models is the inclusion of an extra layer of attention devised for sentence attention that learns the contribution of each sentence in the meaning of the full description as shown in Figure 6. Similarly, permission-related sentences are expected to have a higher weight compared to others. As indicated previously, we generate word context by weighted averaging of all the hidden representations of the tokens in an input sequence. Then, the context is passed into an MLP layer for the classification task. However, in the document-based model, we produce word contexts $s_i^{w/pooling}$ for each sentence in the document. After that, a bidirectional sentence encoder reads the word contexts $s_i^{w/pooling}$ to have the annotations of sentences h_i. We formally define the first level of the network in the sentence-based model as given in the previous section. The task of the second encoder is defined mathematically as follows:

$$h_i = BiGRU(s_i^{w/pooling}), i \in [1, L] \tag{6}$$

where $s_i^{w/pooling}$ represents the attention-based representation of sentences gathered from the first layer. A document contains L sentences. We obtain the hidden representation of $s_i^{w/pooling}$ using bidirectional GRU. Here, h_i denotes the hidden representation of each sentence in the description.

Then, we make use of a second-level attention mechanism to capture attention-based representations of the document, which is v_i. Therefore, this hierarchical network captures all the semantic aspects of the document. The sentence-level attention mechanism is defined formally as follows:

$$\begin{aligned} u_i &= tanh(W_s h_i + b_s) \\ \alpha_i &= \frac{exp(u_i^\intercal u_s)}{\sum_i exp(u_i^\intercal u_s)} \\ v_i &= \sum_i \alpha_i h_i \end{aligned} \tag{7}$$

where W_s and b_s are MLP parameters (i.e. W_s denotes the hidden weights of the MLP and b_s corresponds to the bias parameters). u_i is the hidden representations of h_i obtained from the single-layer MLP with a softmax output function. Here, α_i denotes the softmax probabilities and u_s denotes the sentence-level context vector. Here we use v_i for the compositional representation of the decsription which is the weighted sum of the hidden representations of the vectors of the description sentences.

Similar to the word-level attention mechanism, we apply max and mean pooling to the annotations of the sentences. Finally, we concatenate three types of information: attention-based representation of a description v_i, the vector obtained from global max h_{GMP}^d, and the vector obtained from global mean pooling h_{GAP}^d:

$$v_i^{w/pooling} = v_i \circ h_{GMP}^d \circ h_{GAP}^d \tag{8}$$

where h_{GMP}^d and h_{GAP}^d refer to the hidden representations gathered after applying global max and mean pooling. We concatenate the document hidden representation v_i with h_{GMP}^d and h_{GAP}^d to have the final representation $v_i^{w/pooling}$.

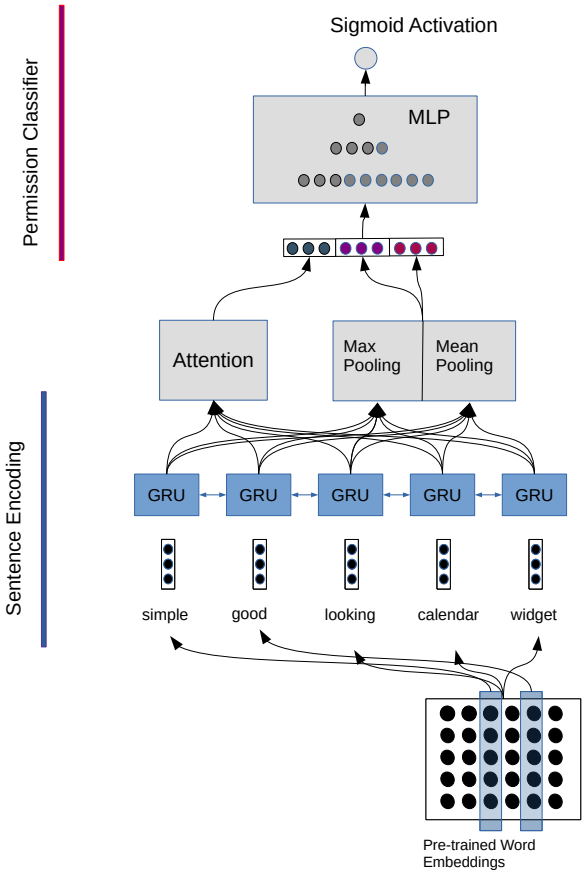


Fig. 5: The architecture of the sentence-based model.

5.5 Classification

In both encoder architectures, the classification is performed analogously. The concatenated final vector is fed into a multi-layer perceptron with a sigmoid activation function as given below:

$$\hat{y} = \begin{cases} \text{sigmoid}(MLP(s_i^{w/pooling})), & \text{if sentence-based.} \\ \text{sigmoid}(MLP(v^{w/pooling})), & \text{otherwise, for} \\ & \text{document-based.} \end{cases}$$

where $\hat{y}$ refers to the prediction output, which will be the permission score, 1 indicates that the application requires permission, and 0 indicates that the application does not require the permission.

5.6 Implementation Details

We implemented the model in DyNet library[5,6] that gives a dynamic framework for neural network models. The vector dimension of each pre-trained word embedding is 300. Therefore, a GRU network is created with an input size of 300. The dimensionality of the hidden layer in each GRU and attention matrix is 128. The MLPs have a hidden layer size of 128 and an output size of 1, where 1 indicates that the permission is stated in the sentence, and 0 indicates that the permission is not mentioned in the sentence. Dimensions in the proposed neural network architecture are determined empirically as a result of rigorous experiments.

We randomly initialize the model parameters with Glorot initialization [26]. We use Stochastic Gradient Descent as the trainer with momentum of 0.9 to get more stable gra-

[5] https://dynet.readthedocs.io/en/latest/tutorial.html

[6] The implementation will be publicly available if the paper gets accepted.

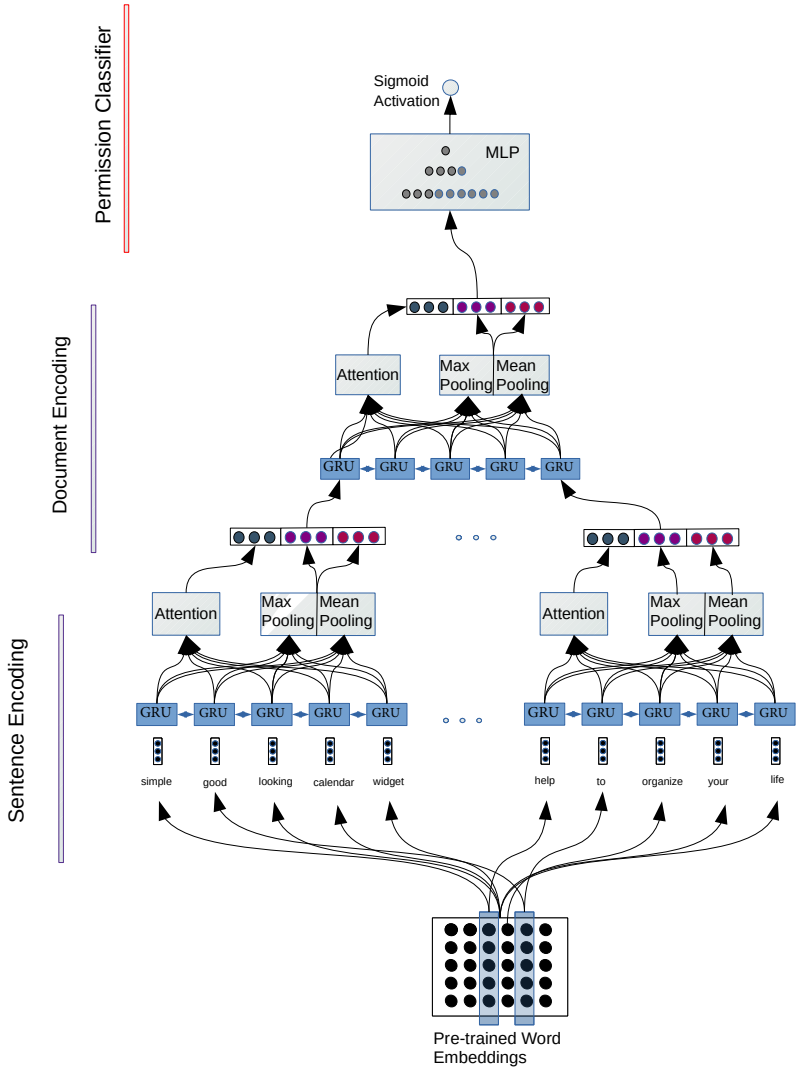


Fig. 6: The architecture of the document-based model.

dient trajectory. We apply gradient norm clipping to deal with the exploding gradient problem [43]. We use 10-fold cross-validation for training. The document-based model was trained for five epochs, and sentence-based was trained for a single epoch.

6 Experiments

6.1 Evaluation Metrics

There is a significant imbalance between the classes in AC-NET dataset [23]. For instance, only 522 of 24724 sentences are marked for the Camera permission. As an evaluation metric, standard accuracy is not appropriate because of the imbalance problem in the dataset. We would have obtained very high accuracy scores based on the classification results for the evaluation. However, the cost of errors will not be the same for different classes in such high imbalanced datasets. Therefore, in addition to the k-fold cross-validation, ROC-AUC and PR-AUC are used as the evaluation metrics in this study. Those metrics are known to be more representative compared to other metrics for domains with skewed class distribution and with unequal classification errors, and they are threshold agnostic. They are widely used in studies that require evaluation metrics insensitive to imbalanced class distribution [40, 16]. ROC-AUC is calculated as given be-

low:

$$ROC - AUC = \frac{\sum_{iepositive\_class} rank_i - \frac{N_p*(N_p+1)}{2}}{N_p * N_n} \quad (9)$$

where N_p and N_n denote the number of positive and negative samples, and $rank_i$ is the ranking of the ith positive sample.

PR-AUC is calculated as follows:

$$PR - AUC = \sum_n (R_n - R_{n-1}) P_n \quad (10)$$

where P_n and R_n are the precision and recall values and N is the number of samples. It is the precision-recall curve which is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds, like the ROC curve.

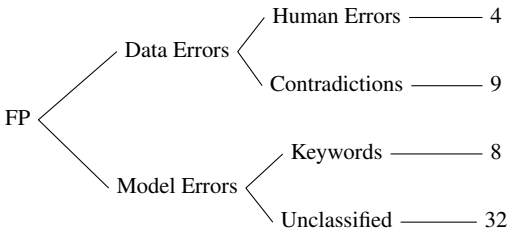
6.2 Results of the Sentence-Based Model

The results of the proposed sentence-based approach are given in Table 3. It is clearly seen that the accuracy increases by using the domain adapted word embeddings and an increase in PR-AUC values is observed in particular.

The proposed model is compared with AC-Net [23] in Table 4. Both models are trained by using their own domain adapted word embeddings. As it is seen in the results, the proposed approach produces competitive results with AC-Net [23]. Our architecture is different than AC-Net with an additional attention mechanism and a hierarchical network that can process the full description at once, whereas AC-NET can only process a description sentence at once. Moreover, the attention mechanism shows that we can find both permission-related words and permission-related sentences efficiently whereas AC-NET can only detect permission-related sentences. In their model, they do not utilize the contextual information while deciding whether an app description includes a permission or not. In contrast, our hierarchical model processes each description by processing all sentences at the same time to decide whether the description includes a given permission. Even in our sentence-based model, we utilize contextual information by looking at all words in a description sentence. While doing this, we also learn permission-related words along with their weights that indicate whether a word could be a permission indicator or not. In AC-NET, the final aim is only to detect whether a given sentence includes a permission or not. Therefore, we can extract more comprehensive information from the descriptions compared to their model. AC-Net [23] has already shown that learning semantic relations by using neural networks outperforms other related studies [42][45][52] in the literature considerably. Therefore, such studies are omitted in the table.

In order to assess our results further, false positives (sentences that are manually labelled as statement sentences but classified as permission sentences by our model) and false negatives (sentences that are manually labelled as permission sentences but classified as statement sentences by our system) are analyzed in detail. This analysis is performed only on READ_CONTACTS permission.

Fig. 7: Main reasons of false positives.



There are 24,720 sentences in AC-Net dataset and 944 of which are manually marked as permission sentences. In order to analyze the results of READ_CONTACTS permission, we randomly selected 2472 sentences as the test set and 87 of which are permission sentences. If the permission score is assigned as 0.5, as is seen from Table 5, 97.9% of these sentences are classified correctly as the statement sentences by our system.

6.2.1 Analysis of False Positives

Sentences which obviously deviate from statement sentences are analyzed in detail in this subsection. Therefore, sentences whose permission score is above 0.5 are analyzed. This is only 2.22% (53 sentences) of the statement sentences in the test set.

Figure 7 illustrates the types of false positive errors. Many of the false positive errors stem from our current model's abilities. We need more complex natural language models to infer the meaning of text. Furthermore, labelling errors in the dataset has a negative effect on model training and prediction steps.

Model Errors: Before proceeding to examine cases where our model is inadequate, it is important to note that it is not easy to categorize each error in the test set since it can lead to many types of subjective subcategories. For this reason, as it can be seen from the Figure 7, most of the cases are unclassified.

- We consider the permission-description problem as a kind of classification problem in this study. Consequently, our model is not able to learn syntactic structure of text, which has a pivotal role in semantics. As shown in Table 6, Sentence S1 is predicted as a permission sentence. Without knowing the syntactic structure of sentence and therefore the dependencies between phrases/words, which

Table 3: Evaluation scores of the proposed sentence-based model on the AC-Net dataset

Permission Group	Fasttext word embeddings		Domain adapted word embeddings	
	ROC-AUC	**PR-AUC**	**ROC-AUC**	**PR-AUC**
CONTACTS	0.98	0.73	0.98	0.73
MICROPHONE	0.97	0.46	0.98	0.50
CALENDAR	0.99	0.72	0.99	0.83

Table 4: Comparative results with AC-NET

Permission Group	AC-NET		Our Model	
	ROC-AUC	**PR-AUC**	**ROC-AUC**	**PR-AUC**
CONTACTS	0.97	0.75	0.98	0.73
MICROPHONE	0.96	0.50	0.98	0.50
CALENDAR	0.99	0.84	0.99	0.83
ACCESS_FINE_LOCATION	0.98	0.77	0.98	0.76
CALL_PHONE	0.99	0.63	0.99	0.65
CAMERA	0.98	0.76	0.98	0.76
GET_TASKS	0.95	0.48	0.94	0.51
READ_CALL_LOGS	0.99	0.71	1.0	0.77
READ_SMS	0.99	0.83	0.99	0.80
STORAGE	0.94	0.66	0.93	0.58
WRITE_SETTINGS	0.95	0.43	0.96	0.48

Table 5: Results of READ_CONTACTS permission.

		Predicted	
		Positive	Negative
Actual	Positive	68	19
	Negative	53	2332

captures "who is doing what to whom", it is not possible to determine the subject of "read contact data" event. This case demonstrates the need for better strategies for capturing semantic information, which is hidden in data.

– The preprocessing (especially stemming) has a significant impact on permission classification task. There are sentences such as S2 which is labelled as a statement sentence. Due to the stopword removal process, some of the important tokens, which may lead directly to a decision, are removed. In this example sentence S2, pronoun "your" is removed in the stopword removal step, and then the resulting sentence becomes "contact group new boss". Therefore, the resulting sentence has a different meaning than the original. Including syntactic information such as part-of-speech tag would mitigate these types of errors in the model, which remains as future work.

– Neural networks are good at learning correlations, and sometimes it can lead to undesirable situations. Our model learns the correlation between sequences of words and a given permission. If a word or combinations of words is seen mostly in positive examples, we can say that there is a strong positive correlation between sequences of words and a permission. In the AC-Net dataset, due to the erroneous labeling and the nature of the data, there is strong positive correlation between READ_CONTACTS permission with words such as "share" and "facebook" (or other social media sites). In the AC-Net dataset, some of the sentences consist of keywords such as "share" or/and "social" and they are labelled as permission sentences such as S6 in Table 6. Whenever we see a statement sentence, which has words such as "facebook" or "twitter", our model wrongly predicts this type of sentences as permission sentences. The case reported here accounts almost half of the false positives. This is exemplified in sentences S3, S4, and S5 in Table 6.

Data Errors: As shown in Figure 7, data labelling errors have contributed to the increase in false positives. Labelling errors is of two kinds: (1) errors which are caused by contradictory examples in permission and statement sentences; (2) errors which may be linked to human errors occurred during the labeling process. The common cases encountered in

Table 6: Examples of model errors.

S#	Sentence	PS	L
1.	”Find apps that can access to your personal information(GPS location#read contact data”	0.95	0
2.	”Contact your group to the new boss”	0.86	0
3.	”droid website: Twitter account: Steam Group: Google+ Beta Community (go here for beta access): osu”	0.95	0
4.	”Share on Facebook# Twitter and Google+Enjoy”	0.87	0
5.	”Save or Share Bible verse or plan or devotions easily on Facebook# Twitter# email# text etc”	0.82	0
6.	”Take a photo of your masterpiece and don’t forget to share it with your friends on Facebook and Twitter”	0.85	1

S# Sentence number.
PS Permission score calculated by our model.
L Manual label given by annotators.

these sentences and few examples for each case (presented in Table 7) are listed below:

- There are sentences such as S1 in Table 7 that is marked manually as statement sentences. However, these sentences are believed to explicitly specify the requirement for the READ_CONTACTS permission. Furthermore, there are sentences in the dataset that are labelled as permission sentences such as S2, but it seems that it is a labeling error too. Under these circumstances, these false positives result from human errors done during labelling the dataset. Labelling is a labor-intensive job and prone to such errors.
- In the dataset, we encounter with sentences about blocking calls/numbers such as S3 in Table 7 and about sharing on social media such as S6 in Table 7. Even though these sentences are annotated as statement sentences, there are very semantically similar sentences in the dataset, however they are annotated as permission sentences. S4 and S5 could be given as examples to sentences about blocking calls and sharing on social media respectively. Such annotation errors could result in confounding effects in learning, hence increase in both false positives and false negatives.

Our model considers the contextual information very well as seen in the example results. For example, the first sentence S1 from Table 8 is labelled as permission sentence with a high permission score due to the word *contact*. However, the second sentence S2 is labelled as a non-permission sentence with a very low permission score although those sentences also involve *contact*. Even though it seems like the words such as *contact* and *account* which are related to READ_CONTACTS permission have a confounding effect on the results at first view, our findings show that they do not have. Our model successfully estimates low prediction scores for the sentences containing expressions like *contact us*, *contact of this app*, *contact support*, *account management*, *official account*, *premium account* etc. The reason behind this is that our system is able to extract the contextual meaning of these expressions with the help of the recurrent neural networks that has an ability to process the sequential information effectively.

Table 7: Examples of labelling errors in the AC-NET dataset.

S#	Sentence	PS	L
1.	”Contact specific notifications for certain notification typesIf you’ve come from a Blackberry device and miss the features of BeBuzz / BerryBuzz then give LightFlow a try”	0.92	0
2.	”Buy furniture and home materials on Houzz using Android Pay for a simpler buying experience”	0.002	1
3.	”Block numbers from those you dont want to be able to contact you”	0.97	0
4.	”Block numbers# if needed”	0.62	1
5.	”Take a photo of your masterpiece and don’t forget to share it with your friends on Facebook and Twitter”	0.85	1
6.	”Share your progress and workouts with friends on social media”	0.86	0

S# Sentence number.
PS Permission score calculated by our model.
L Manual label given by annotators.

Table 8: Two contextual meanings of the word *”contact”*.

S#	Sentence	PS	L
1.	”Find & Merge contacts with duplicate phone or email”	0.99	1
2.	”If you have any questions or comments# please contact us at”	0.004	0

S# Sentence number.
PS Permission score calculated by our model.
L Manual label given by annotators.

6.2.2 Analysis of False Negatives

There are 943 sentences that are marked manually as permission sentences (labelled as 1) for READ_CONTACTS permission in the AC-Net dataset. There are 87 randomly selected permission sentences in our test split. As we mentioned earlier, the threshold for permission score is defined as 0.5, our model misses 21.8% (19 sentences) of the permission sentences. Figure 8 illustrates the types of false negative errors. The figure shows that many of the errors in false negatives stems from erroneous examples provided by AC-Net.

Fig. 8: Main reasons of false negatives.

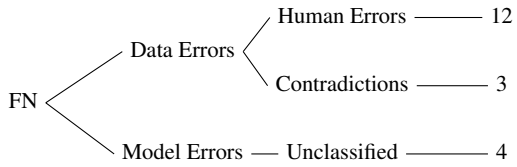


Table 9: Examples of false negatives.

S#	Sentence	PS	L
1	"You can recall a special memory from your photos and share them with your friends and family 1"	0.13	1
2	"Use of this application requires a Facebook or Zynga With Friends account"	0.94	0
3	"Buy furniture and home materials on Houzz using Android Pay for a simpler buying experience"	0.002	1
4	"But you need to pay for some items especially rare fishes and rare fish food"	0.005	1
5	"This application will deeply analyze your phonebook and bring it close to perfection"	0.109	1
6	"You will always know the true caller# even if they are not in your address book"	0.058	1

S# Sentence number.
PS Permission score calculated by our model.
L Manual label given by annotators.

Here, the sentences whose permission score is below 0.50 are analyzed. This is approximately 21.8% (19 sentences) of the permission sentences. Many of these sentences result from confounding effects due to labelling errors stem from contradictory keywords such as blocking calls and sharing on social media. Another similar effect results from labelling sentences for CONTACTS permission group, which contains READ_CONTACTS, WRITE_CONTACTS and GET_ACCOUNTS permissions, rather than only READ_CONTACTS permission. For example, while sentence S1 in Table 9 is a permission sentence, S2 is labelled as a statement sentence in the dataset. There are many cases like this in the dataset. Therefore, we label sentences only for READ_CONTACTS permission in our dataset in order to avoid confusion between applications' account data and users' contact data as mentioned before. There are also many false negatives resulting from wrongly labelled sentences in other topics such as S3, S4. There are also sentences which clearly indicate a need for the permission such as S5 and S6 but unable to be detected by our model.

Finally, the proposed method is trained and evaluated on the DesRe dataset introduced in this study. The results are given in Table 10. The results show a considerable increase in ROC-AUC. This is believed to be the result of consistent labelling based on the Android Developer Guide. Since the implementation of AC-Net is not publicly available, we were unable to run it on the DesRe dataset.

6.3 Results of the Document-based Model

In this section, the document-based model is evaluated for measuring the description-to-permission fidelity. The sentence-based models proposed for assessing the fidelity might not fit very well for this problem, since sentences may be irrelevant if we think them in isolation. However, our models have to make a decision, whether it is a permission related sentence or not, given the context (i.e. other sentences in the description).

It is not possible to compare sentence-based and document-models through examples. A sentence which is tagged as a statement sentence in the sentence-based model could be also a part of a permission involved document. Hence, an example based comparison is not feasible. We can still evaluate these two model by ROC-AUC and PR-AUC score metrics as given in Table 11. Here, we used the same test-train split in both models. Moreover, in order to compare both models, sentence-based results are grouped by application identifier, and select the sentence with maximum prediction score for each application, finally we use this score as the document prediction score. As it is seen in Table 11, there is substantial improvements in PR-AUC scores in the document-based model. Hierarchical attention network based document model can capture the document-permission correlations even in the low resource permissions such as GET_TASKS (see Table 12).

Please note that even though the dataset used for training the document-based model is much smaller than the dataset using for training the sentence-based model, it is a more balanced dataset. For instance, there are 1414 descriptions for the READ_CONTACTS permission, where 30.8% of them are tagged as permission documents. On the other hand, there are 24,720 sentences in the AC-Net dataset and only 3.82% of which are manually marked as READ_CONTACTS permission sentences. The statistics of all permissions in AC-Net is summarized in Table 12. It shows the percentages of permission sentences and documents for each permission in the AC-Net dataset, where the total number of documents and total number of sentences are 1,414 and 24,720, respectively.

In order to analyze the results of READ_CONTACTS permission, we randomly selected 142 documents as test set and 41 of which are permission documents. 0.3 is assigned for the permission prediction threshold according to the train set (which is defined empirically as a result of several experiments). As is seen from Table 13, 83% of these sentences are classified correctly as statement documents by the document-based model. In order to assess our results further, we try to analyze the false positive and false negative

Table 10: Evaluation scores of the proposed sentence-based model on the DesRe dataset

Permission Group	Fasttext word embeddings		Domain adapted word embeddings	
	ROC-AUC	**PR-AUC**	**ROC-AUC**	**PR-AUC**
READ_CONTACTS	0.98	0.81	0.98	0.81
RECORD_AUDIO	0.95	0.77	0.97	0.81
STORAGE	0.97	0.74	0.98	0.74

Table 11: Document classification results of the both proposed models on the AC-NET dataset

Permission Group	Document classification with Sentence-Based		Document classification with Document-Based	
	ROC-AUC	**PR-AUC**	**ROC-AUC**	**PR-AUC**
READ_CONTACTS	0.91	**0.84**	0.90	0.83
RECORD_AUDIO	0.95	0.74	0.96	**0.80**
READ_CALENDAR	0.98	0.84	0.99	**0.93**
ACCESS_FINE_LOCATION	0.95	0.87	0.96	**0.90**
CALL_PHONE	0.96	0.72	0.97	**0.80**
CAMERA	0.95	0.83	0.94	**0.84**
GET_TASKS	0.86	0.55	0.87	**0.67**
READ_CALL_LOGS	0.96	0.77	0.99	**0.87**
READ_SMS	0.97	0.89	0.98	0.89
STORAGE	0.87	0.84	0.87	0.84
WRITE_SETTINGS	0.89	0.64	0.92	**0.75**

Fig. 9: Main reasons of the false positives in document-based model.

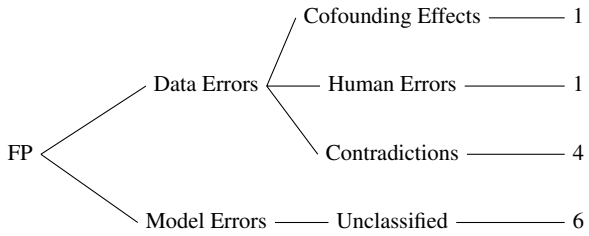
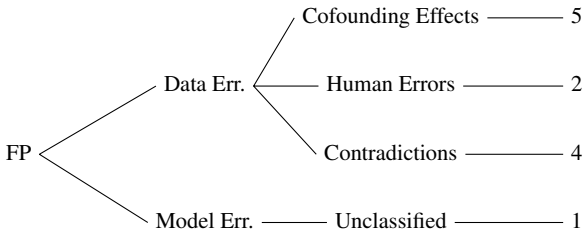


Fig. 10: Main reasons of the false negatives in the document-based model.



results which are reported in Table 13. Figure 9 illustrates the types of false positive errors. Half of the errors stem from our model's inadequacies, and rest of them are based on the dataset. The numbers on the figure correspond to the number of examples in the predicted results.

Figure 10 shows the summary statistics for false negative errors. The main reason behind most of the errors is the cofounding effects and contradictions in the dataset.

Additionally, we tested the document-based model on the DesRe dataset. The results are given in Table 14. As we previously mentioned for the sentence-based model, there is a considerable increase in ROC-AUC and PR-AUC owing to the consistent labelling based on the Android Developer Guide.

Table 12: Statistics of AC-Net.

Permissions	% of Labeled Sentences	% of Labeled Documents
READ_CONTACTS	3.82	30.8
RECORD_AUDIO	1.30	10.5
READ_CALENDAR	1.17	7.3
ACCESS_FINE _LOCATION	2.93	21.5
CALL_PHONE	1.31	8.0
CAMERA	2.12	16.1
GET_TASKS	1.39	10.1
READ_CALL_LOGS	0.80	6.8
READ_SMS	2.12	15.0
STORAGE	5.41	40.8
WRITE_SETTINGS	2.47	15.8

Table 13: Results of READ_CONTACTS permission in Document-based model.

		Predicted	
		Positive	Negative
Gold	Positive	29	12
	Negative	12	89

7 Conclusion and Future Work

In this paper, we investigate the use of natural language processing methods as well as recurrent neural networks to tackle the description-to-fidelity problem in Android applications. In order to do that, two models are introduced: sentence-based and document-based. Our sentence-based model is similar to the recent neural model AC-NET [23] since both use recurrent neural networks. However, our model also makes use of attention mechanism to capture contextual semantics. The attention mechanisms have shown a superior performance in almost all natural language processing tasks. We also incorporate attention mechanism in our proposed model to detect the permission-related words in a description sentence, thereby assigning different weights to the description sentences, which deviates our work from AC-NET and other works on the description-to-permission fidelity problem. In this document-based model, we use two-layered hierarchical attention mechanism to learn the description semantics, where one of them encodes the words in a sentence, and the latter encodes the sentences in a description. Thereafter, we correlate description semantics with permissions. The results are significantly improved with the hierarchical attention mechanism and it shows that our models could assist in prioritizing applications for more detailed analysis.

Another contribution of the study is to introduce a new annotated description dataset for three types of permissions, namely RECORD_AUDIO, READ_CONTACTS and STORAGE. Moreover, five reviews declared to be most helpful by other users for each application are also included in the dataset. We plan to investigate the effects of user reviews on both the sentence-based model and the document-based model, which is left as a future goal.

Acknowledgements We thank Muhammet Kabukçu and Beyza Çevik for their help in constructing the DesRe dataset.

Funding This study is supported by the Scientific and Technological Research Council of Turkey (TUBITAK-118E141).

Compliance with ethical standards

Conflict of interest Author Huseyin Alecakir declares that he has no conflict of interest. Author Burcu Can declares that she has no conflict of interest. Author Sevil Sen declares that she has no conflict of interest.

Ethical approval This article does not contain any study with human participants or animals performed by any of the authors.

References

1. (2019) Android developer guide. https://developer.android.com/, (Last accessed in May, 2019)
2. (2019) Camera api. https://developer.android.com/guide/topics/media/camera.html, (Last accessed in May, 2019)

Table 14: Evaluation scores of the proposed document-based model on the DesRe dataset

Permission Group	Fasttext word embeddings		Domain adapted word embeddings	
	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
READ_CONTACTS	0.93	0.94	0.94	0.95
RECORD_AUDIO	0.91	0.90	0.92	0.91
STORAGE	0.92	0.87	0.94	0.90

3. (2019) Dangerous permissions. https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions, (Last accessed in September, 2019)
4. (2019) Intent — android developers. https://developer.android.com/reference/android/content/Intent.html, (Last accessed in May, 2019)
5. (2019) Permissions overview. https://developer.android.com/guide/topics/permissions/overview, (Last accessed in May, 2019)
6. (2019) Read external storage permission. https://developer.android.com/reference/android/Manifest.permission#READ_EXTERNAL_STORAGE, (Last accessed in September, 2019)
7. (2019) Sending simple data to other apps — android developers. https://developer.android.com/training/sharing/send.html, (Last accessed in May, 2019)
8. Arp D, Spreitzenbarth M, Hübner M, Gascon H, Rieck K (2014) Drebin: Effective and explainable detection of android malware in your pocket. DOI 10.14722/ndss.2014.23247
9. Au KWY, Zhou YF, Huang Z, Lie D (2012) Pscout: analyzing the android permission specification. In: Proceedings of the 2012 ACM conference on Computer and communications security, ACM, pp 217–228
10. Aysan AI, Sakiz F, Sen S (2018) Analysis of dynamic code updating in android with security perspective. IET Information Security 13(3):269–277
11. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. 1409.0473
12. Ban T, Takahashi T, Guo S, Inoue D, Nakao K (2016) Integration of multi-modal features for android malware detection using linear svm. In: 2016 11th Asia Joint Conference on Information Security (AsiaJCIS), IEEE, pp 141–146
13. Bird S, Klein E, Loper E (2009) Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."
14. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. TACL 5:135–146, URL http://dblp.uni-trier.de/db/journals/tacl/tacl5.html#BojanowskiGJM17
15. Caira JRJ, Ey T (2020) Heads up, app developers: Google is getting serious about privacy and data security in apps. (Visited September 2020) [Online]. https://www.natlawreview.com/article/heads-app-developers-google-getting-serious-about-privacy-and-data-security-apps
16. Chawla NV (2005) Data Mining for Imbalanced Datasets: An Overview, Springer US, Boston, MA, pp 853–867. DOI 10.1007/0-387-25465-X_40, URL https://doi.org/10.1007/0-387-25465-X_40
17. Cheng X, Yan X, Lan Y, Guo J (2014) Btm: Topic modeling over short texts. IEEE Transactions on Knowledge and Data Engineering 26(12):2928–2941
18. Cho K, van Merrienboer B, Gülçehre Ç, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. CoRR abs/1406.1078, URL http://arxiv.org/abs/1406.1078, 1406.1078
19. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805
20. Dong L, Lapata M (2016) Language to logical form with neural attention. arXiv preprint arXiv:160101280
21. Elman J (1990) Finding structure in time. Cognitive science 14(2):179–211
22. Felt AP, Ha E, Egelman S, Haney A, Chin E, Wagner D (2012) Android permissions: User attention, comprehension, and behavior. In: Proceedings of the Eighth Symposium on Usable Privacy and Security, ACM, New York, NY, USA, SOUPS '12, pp 3:1–3:14, DOI 10.1145/2335356.2335360, URL http://doi.acm.org/10.1145/2335356.2335360

23. Feng Y, Chen L, Zheng A, Gao C, Zheng Z (2019) Acnet: Assessing the consistency of description and permission in android apps. IEEE Access 7:57829–57842, DOI 10.1109/ACCESS.2019.2912210
24. Finegan-Dollak C, Kummerfeld JK, Zhang L, Ramanathan K, Sadasivam S, Zhang R, Radev D (2018) Improving text-to-sql evaluation methodology. arXiv preprint arXiv:180609029
25. Gabrilovich E, Markovitch S (2007) Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'07, pp 1606–1611, URL http://dl.acm.org/citation.cfm?id=1625275.1625535
26. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: AISTATS
27. GooglePlay (2020) Privacy, security and deception. (Visited September 2020) [Online]. https://play.google.com/intl/en-US/about/privacy-security-deception/
28. 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, New York, NY, USA, ICSE 2014, pp 1025–1035, DOI 10.1145/2568225.2568276, URL http://doi.acm.org/10.1145/2568225.2568276
29. Grave E, Bojanowski P, Gupta P, Joulin A, Mikolov T (2018) Learning word vectors for 157 languages
30. Hearst MA (1998) Support vector machines. IEEE Intelligent Systems 13(4):18–28, DOI 10.1109/5254.708428, URL http://dx.doi.org/10.1109/5254.708428
31. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780, DOI 10.1162/neco.1997.9.8.1735, URL http://dx.doi.org/10.1162/neco.1997.9.8.1735
32. Kong D, Cen L, Jin H (2015) Autoreb: Automatically understanding the review-to-behavior fidelity in android applications. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, pp 530–541
33. Li Z, Zhang Y, Wei Y, Wu Y, Yang Q (2017) End-to-end adversarial memory network for cross-domain sentiment classification. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, AAAI Press, IJCAI'17, pp 2237–2243, URL http://dl.acm.org/citation.cfm?id=3172077.3172199
34. Martín A, Calleja A, Menéndez HD, Tapiador J, Camacho D (2016) Adroit: Android malware detection using meta-information. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp 1–8, DOI 10.1109/SSCI.2016.7849904
35. McAfee (2019) Mcafee mobile threat report. (Visited August 2019) [Online]. Available: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf
36. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781
37. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in Neural Information Processing Systems 26, Curran Associates, Inc., pp 3111–3119
38. Mikolov T, Grave E, Bojanowski P, Puhrsch C, Joulin A (2018) Advances in pre-training distributed word representations. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)
39. Miller GA (1995) Wordnet: A lexical database for english. Commun ACM 38(11):39–41, DOI 10.1145/219717.219748, URL http://doi.acm.org/10.1145/219717.219748
40. Nguyen DC, Derr E, Backes M, Bugiel S (2019) Short text, large effect: Measuring the impact of user reviews on android app security & privacy. In: Proceedings of the IEEE Symposium on Security & Privacy, May 2019, IEEE, URL https://publications.cispa.saarland/2815/
41. Oberheide J, Miller C (2012) Dissecting the android bouncer. SummerCon2012, New York 12
42. Pandita R, Xiao X, Yang W, Enck W, Xie T (2013) Whyper: Towards automating risk assessment of mobile applications. In: Proceedings of the 22Nd USENIX Conference on Security, USENIX Association, Berkeley, CA, USA, SEC'13, pp 527–542, URL http://dl.acm.org/citation.cfm?id=2534766.2534812
43. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, JMLR.org, ICML'13, pp III–1310–III–1318, URL http://dl.acm.org/citation.cfm?id=3042817.3043083
44. Porter MF (1997) Readings in information retrieval. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, chap An Algorithm for Suffix Stripping, pp 313–316, URL http://dl.acm.org/citation.cfm?id=275537.275705

45. 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, New York, NY, USA, CCS '14, pp 1354–1365, DOI 10.1145/2660267.2660287, URL http://doi.acm.org/10.1145/2660267.2660287
46. Rastogi V, Chen Y, Jiang X (2013) Droidchameleon: evaluating android anti-malware against transformation attacks. In: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, ACM, pp 329–334
47. Sen S, Aydogan E, Aysan AI (2018) Coevolution of mobile malware and anti-malware. IEEE Transactions on Information Forensics and Security 13(10):2563–2574
48. Statista (2019) Number of apps available in leading app stores as of 2nd quarter 2019. (Visited August 2019) [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
49. Toivonen H (2017) Apriori Algorithm, Springer US, Boston, MA, pp 60–60. DOI 10.1007/978-1-4899-7687-1_27, URL https://doi.org/10.1007/978-1-4899-7687-1_27
50. Wang H, Li Y, Guo Y, Agarwal Y, Hong JI (2017) Understanding the purpose of permission use in mobile apps. ACM Transactions on Information Systems (TOIS) 35(4):43
51. Wang R, Wang Z, Tang B, Zhao L, Wang L (2019) Smartpi: Understanding permission implications of android apps from user reviews. IEEE Transactions on Mobile Computing
52. 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), USENIX Association, Ottawa, pp 241–255, URL https://www.usenix.org/conference/soups2015/proceedings/presentation/watanabe
53. Wu J, Yang M, Luo T (2017) Pacs: Pemission abuse checking system for android applictions based on review mining. In: 2017 IEEE Conference on Dependable and Secure Computing, pp 251–258, DOI 10.1109/DESEC.2017.8073813
54. Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhutdinov R, Zemel R, Bengio Y (2015) Show, attend and tell: Neural image caption generation with visual attention. URL http://arxiv.org/abs/1502.03044, cite arxiv:1502.03044
55. Xue Y, Meng G, Liu Y, Tan TH, Chen H, Sun J, Zhang J (2017) Auditing anti-malware tools by evolving android malware and dynamic loading technique. IEEE Transactions on Information Forensics and Security 12(7):1529–1544
56. Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 1480–1489
57. Yang Z, Yang D, Dyer C, He X, Smola AJ, Hovy EH (2016) Hierarchical attention networks for document classification. In: HLT-NAACL
58. Yu L, Luo X, Qian C, Wang S (2016) Revisiting the description-to-behavior fidelity in android applications. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), IEEE, vol 1, pp 415–426
59. Yu L, Luo X, Qian C, Wang S, Leung HK (2017) Enhancing the description-to-behavior fidelity in android apps with privacy policy. IEEE Transactions on Software Engineering 44(9):834–854
60. Yu L, Zhang T, Luo X, Xue L, Chang H (2017) Toward automatically generating privacy policy for android apps. IEEE Transactions on Information Forensics and Security 12(4):865–880, DOI 10.1109/TIFS.2016.2639339
61. Zhang M, Duan Y, Feng Q, Yin H (2015) Towards automatic generation of security-centric descriptions for android apps. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, ACM, New York, NY, USA, CCS '15, pp 518–529, DOI 10.1145/2810103.2813669, URL http://doi.acm.org/10.1145/2810103.2813669
62. Zhou X, Wan X, Xiao J (2016) Attention-based LSTM network for cross-lingual sentiment classification. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Austin, Texas, pp 247–256, DOI 10.18653/v1/D16-1024, URL https://www.aclweb.org/anthology/D16-1024