AN RNN-BASED APPROACH FOR DISCOVERING INCONSISTENCIES BETWEEN PERMISSIONS AND METADATA IN ANDROID APPLICATIONS

ANDROID UYGULAMALARINDA İZİNLER İLE META VERİLER ARASINDAKİ TUTARSIZLIKLARI KEŞFETMEK İÇİN RNN TABANLI BİR YAKLAŞIM

MUHAMMET KABUKÇU

ASSOC. PROF. SEVİL ŞEN AKAGÜNDÜZ
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

2019

ÖZET

ANDROID UYGULAMALARINDA İZİNLER İLE META VERİLER ARASINDAKİ TUTARSIZLIKLARI KEŞFETMEK İÇİN RNN TABANLI BİR YAKLAŞIM

Muhammet KABUKÇU

Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Doç. Dr. Sevil ŞEN AKAGÜNDÜZ
İkinci Danışman: Öğr. Üyesi Dr. Burcu CAN BUĞLALILAR
Eylül 2019, 88 sayfa

Mobil cihazların sürekli el altında olması sebebiyle, İnternet erişimi için mobil cihazların etkin kullanımı her geçen gün artmaktadır. Günümüzde, mobil cihazların çoğu Android işletim sistemini kullanmaktadır. Mobil cihazlarda genellikle mobil uygulamaları kullanarak ihtiyaçlarımızı karşılamaktayız. Bu durum, ihtiyaçlarımız için özelleştirilmiş çok sayıda mobil uygulamayı beraberinde getirmektedir. Bunun sonucu olarak da uygulamaları keşfetmek ve indirebilmek için kullanabileceğimiz uygulama marketleri ortaya çıkmıştır. Android'in resmi uygulama marketi Google Play ve Apple'ın resmi uygulama marketi App Store gibi uygulama marketleri, uygulama geliştiricilerin uygulamalarını tüm dünyadaki kullanıcılara sunmaları için pratik bir ortam sağlamaktadır. Bu marketler, uygulamanın yanında meta veri dediğimiz uygulama tanımı, kullanıcı yorumları, uygulama skoru gibi uygulama hakkında bilgi verecek diğer kaynakları da içermektedir. Uygulama marketleri kullanıcılara bu hizmetleri sağlamakla beraber, bu durumun kaçınılmaz bir sonucu olarak, kötü niyetli uygulama geliştiricilere zararlı veya güvenli olmayan uygulamalarını geniş bir kitleye sunma fırsatı da sağlamış olmaktadırlar.

Uygulama marketleri kendilerini ve kullanıcılarını kötü amaçlı ve zararlı uygulamalardan uzak tutmak için bazı güvenlik tedbirlerine sahiptirler. Bunun yanında, cihaz üzerinde de alınmış güvenlik önlemleri bulunmaktadır. Android İşletim Sistemi'nde izinler, kullanıcıların farkındalıklarını arttırarak kullanıcının gizliliğini ihlal edebilecek uygulamaları yüklemelerini önlemek için kullanılmaktadır. Bir uygulamayı kurarken (veya uygulama çalışırken), kullanıcı, uygulamanın istediği tehlikeli izinleri görebilmektedir. Bu izinler, kritik sistem kaynaklarına veya hassas kullanıcı verilerine erişen uygulama programlama arayüzlerini kullanmak için uygulamalar tarafından talep edilmektedir. Gizlilik ve güvenlik açısından, bir uygulamanın işlevselliği uygulama açıklamasında yeterince ayrıntılı olarak belirtilirse, istenen izinlerin gerekliliği kullanıcı tarafından anlaşılabilir. Bu, literatürde uygulama tanımı-talep edilen izin uyumluluğu olarak tanımlanmaktadır.

Bu çalışmada, doğal dil işleme teknikleri ve tekrarlayan sinir ağları kullanılarak istenen izinler ve uygulama meta verileri arasındaki tutarsızlıkları belirlemek için uygulama tanımı-talep edilen izin uyumluluğu problemine yönelik yeni bir yaklaşım önerilmiştir. Uygulama açıklamalarının yanı sıra, kullanıcı yorumlarının da bu gibi tutarsızlıkları keşfetme üzerindeki etkisi incelenmiştir. Deney sonuçları, önerilen yaklaşımın uygulama meta verisinden izin ifadelerinin çıkarılmasında yüksek doğruluk elde ettiğini ve bunun kullanıcının veri gizliliği ve güvenliği için kullanılabileceğini göstermektedir.

Anahtar Kelimeler: Android, mobil güvenlik, mobil uygulamalar, uygulama izinleri, uygulama tanımları, kullanıcı yorumları, uygulama tanımı-talep edilen izin uyumluluğu, derin öğrenme, doğal dil işleme, tekrarlayan sinir ağları

ABSTRACT

AN RNN-BASED APPROACH FOR DISCOVERING INCONSISTENCIES BETWEEN PERMISSIONS AND METADATA IN ANDROID APPLICATIONS

Muhammet KABUKÇU

Master of Science, Computer Engineering Department
Supervisor: Assoc. Prof. Sevil ŞEN AKAGÜNDÜZ
Co-Supervisor: Assist. Prof. Dr. Burcu CAN BUĞLALILAR
September 2019, 88 pages

Since mobile devices are increasingly on hand today, users have become more heavily involved with their use in accessing the Internet. Today, most mobile devices use the Android operating system. On mobile devices, users' needs are generally met through the use of mobile applications, and this brings along a large number of mobile applications customized for our needs. Applications, more commonly referred to as "apps", are usually downloaded from an Application Store. These application stores came into existence in order for users to discover what is available through a single location, and to download any apps they may want. Application stores offer a wide range of apps, customized for almost everyone's various needs. Stores such as Android's official market store, known as "Google Play," and Apple's official market store, known as "App Store", provide a practical outlet for developers to present their applications to users worldwide. In addition to the apps, these markets include other resources known as metadata, which provide information about each app such as the application description, user comments, and the corresponding application score. However, as an inevitable consequence of their design and function, application stores also provide developers of malicious software the opportunity to introduce harmful or unsafe applications to a wide and largely unsuspecting audience.

Application stores utilize certain security precautions in order to keep the store clean and to steer genuine store users away from harmful content. In addition, there are also certain security precautions installed on handheld devices. With the Android mobile operating system, "permissions" are used in order to prevent users from installing apps that might violate the user's privacy by raising their awareness. When installing an app (or when an app is running), users are notified of any permission requests from apps that are perceived as being dangerous (i.e., permissions to access critical system resources or privacy-sensitive user data). These permissions are requested by apps in order to use application programming interfaces (APIs) that access critical system resources or sensitive user data. From a privacy and security perspective, if the functionality of an app is sufficiently detailed in its description, the need for the requested permissions can be readily understood by the user. This is defined as description-to-permission fidelity in the literature.

In the current study, a novel approach for the description-to-permission fidelity problem is proposed in order to identify inconsistencies between requested permissions and application metadata by using natural language processing techniques and recurrent neural networks. Besides application descriptions, the effect of user reviews on discovering such inconsistencies is also investigated. The experimental results show that the proposed approach achieves a high degree of accuracy in detecting permission expressions from application metadata, and could therefore be applied for the protection of user privacy and security.

Keywords: Android, mobile security, mobile applications, application permissions, application descriptions, user reviews, description-to-permission fidelity, deep learning, natural language processing, recurrent neural networks

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisors Assoc. Prof. Sevil ŞEN AKAGÜNDÜZ and Assist. Prof. Burcu CAN BUĞLALILAR who have always encouraged me and guided me with their valuable contributions and criticisms at all stages of my dissertation.

Besides I would like to thank my thesis committee members for insightful comments for this thesis.

In addition, I would like to thank everybody who supported and contributed to this study. Especially, I would like to thank Hüseyin ALEÇAKIR, with whom I studied together in the same project, and also all members of Hacettepe University Wireless Networks and Intelligent Secure Systems Lab (WISE Lab) that I had the opportunity to meet during this study.

Finally, I thank my beloved family for their continued support throughout my educational life.

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) with the project number 118E141.

CONTENTS

TABLES

FIGURES

SYMBOLS AND ABBREVIATIONS

Abbreviations

aapt	**A**ndroid **A**pplication **P**ackaging **T**ool
Adam	**Ada**ptive **M**oment Estimation
API	**A**pplication **P**rogramming **I**nterface
APK	**A**ndroid **P**ackage (**K**it)
App	**App**lication
AUC	**A**rea **U**nder **C**urve
CBOW	**C**ontinuous **B**ag of **W**ords
DesRe	**Des**criptions and **Re**views of Android Applications
DEX	**D**alvik **EX**ecutable
DPR	**D**escription-to-**P**ermission **R**elatedness
DVM	**D**alvik **V**irtual **M**achine
ESA	**E**xplicit **S**emantic **A**nalysis
FOL	**F**irst **O**rder **L**ogic
GloVe	**Glo**bal **Ve**ctors
GRU	**G**ated **R**ecurrent **U**nit
kNN	**k**-**N**earest **N**eighbors
LDA	**L**atent **D**irichlet **A**llocation
LSTM	**L**ong **S**hort-**T**erm **M**emory
MLP	**M**ulti**l**ayer **P**erceptron
NLG	**N**atural **L**anguage **G**eneration
NLP	**N**atural **L**anguage **P**rocessing
OS	**O**perating **S**ystem
POS	**P**art-**O**f-**S**peech
PR	**P**recision **R**ecall
RNN	**R**ecurrent **N**eural **N**etwork
ROC	**R**eceiver **O**perating **C**haracteristic

ROC-AUC	**A**rea **U**nder the **R**eceiver **O**perating **C**haracteristic **C**urve
SVM	**S**upport **V**ector **M**achines
tf-idf	**t**erm **f**requency-**i**nverse **d**ocument **f**requency
URI	**U**niform **R**esource **I**dentifier
WSD	**W**ord **S**ense **D**isambiguation

1. INTRODUCTION

The number of Internet users has increased rapidly in recent years. The daily number of new users going online reached one million for the first time in January 2018 [2]. According to the Global Digital Report 2019 [2], there are a total of 4.39 billion active Internet users worldwide (or 57% of the total population), with 3.99 billion being mobile users. In January 2019, from a total of 3.26 billion social media users, 3.48 billion accessed social media via mobile devices. This number is 297 million higher than recorded the previous year. In 2018, a total of 194 billion mobile applications were downloaded worldwide, with users spending US$101 billion on mobile applications.

Another statistical study [3] showed that in Turkey, as of 2019, there were 76.34 million mobile subscriptions (93% of a total population of 82.44 million), and that 77% of the adult population were smartphone users. The total number of active mobile Internet users in Turkey was reported to be 56.03 million. When it came to social media usage, the study showed a total of 52.00 million social media users, with 44.00 million of those being active users of mobile social media apps. In 2018, a total of 2.87 billion mobile apps were downloaded in Turkey, with users spending US$360.5 million on mobile apps.

All of these statistics demonstrate the significance of the Internet, and the evolution of the way that it connects daily life across the global population. Each day, people are becoming more heavily involved with mobile device usage of the Internet. The most important factor in this picture of such widespread usage are the mobile devices that many people have to hand virtually all the time.

1.1. Mobile Applications and Application Stores

In using mobile devices, users generally have their needs met through the usage of mobile applications or apps. Mobile apps make users' lives easier; a fact supported by the vast number of mobile apps available that have been customized to the needs of users. Apps are downloaded and installed to mobile devices from central platforms referred to as Application Stores. It is reported that there were almost 2.5 million applications available on the Android official market store (Google Play), and almost 2 million on the iOS official market (Apple App Store) during the second quarter of 2019 [4].

Android is currently the most widely used mobile operating system (OS). Recent statistics show that from May 2018 to May 2019, 75.27% of the global mobile operating system market share belongs to Android, with 22.74% belonging to Apple's iOS [5]. Android also has a higher share of the Turkish market with 82.29%, whereas iOS has only 16.91% [6]. Figure 1.1. illustrates the global market shares of the leading mobile operating systems from 2009 to 2019 [7]. Detailed market share data can be seen in Appendix A. The numbers emphasize the importance and necessity of Google Play which, as the official application store of the Android OS, was first launched on October 22, 2008 under the name Android Market. In addition to enabling users to search for and download apps, application stores also include content such as games, music, movies, and e-books.

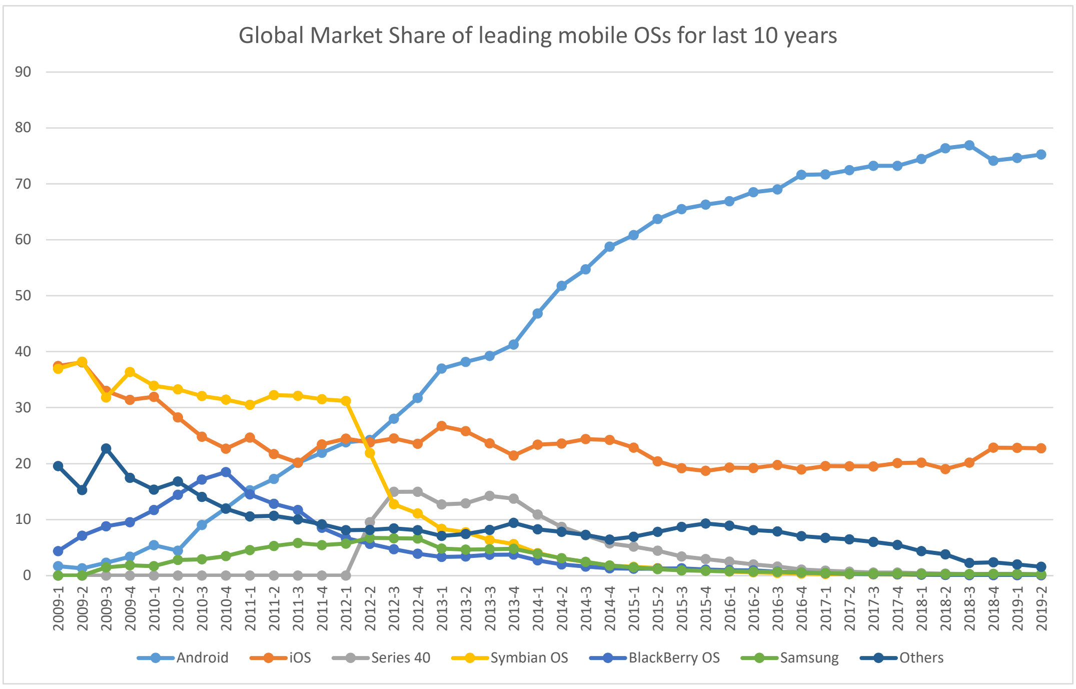


Figure 1.1. Global market share of leading mobile OSs between 2009 - 2019

1.2. Security of Mobile Applications and Application Stores

Application stores enable users to discover and download applications and other content as previously mentioned. While application stores provide users with these services, they also offer a practical way for application developers to deliver their apps to a global user audience. However, as an inevitable consequence of their design and function, application stores also provide developers of malicious software the opportunity to introduce harmful or unsafe applications to a wide and largely unsuspecting audience. The exponential increase seen in the number of mobile apps has also motivated malware developers to attempt to illegitimately acquire sensitive user data. According to the McAfee Mobile Threat Report [8], malware applications in the mobile world has continued to increase in both scope and complexity during 2019.

Application stores take certain security precautions in order to distance themselves and their users from the detrimental impact of malicious activities. Since Android OS is the most widely used mobile OS, Google Play is also utilized by the vast majority of mobile users. Under the circumstances, the security of Google Play is of vital significance to the industry at large. For an application to be offered through Google Play, application developers need to comply with certain Google Play policies. Google performs an analysis on applications for policy violations (e.g., hate speech, violence, sensitive events, impersonation, intellectual property) prior to authorizing the release of applications onto the market [9]. For example, image analysis is performed in order to detect applications that include images considered to be of an adult nature (known as adult content). Applications are also analyzed as to whether or not they include code recognized as being malicious. All of these analyses are collectively known as Google Play Protect [9] (which started out as Google Bouncer [10] and was rebranded and enhanced to become Google Play Protect in 2017 [11, 12]). As from 2015, in addition to the automated analysis systems, an internal team of human reviewers started analyzing applications manually for policy violations [13].

The aforementioned precautions are just some of the security-based methods employed on the application store side. Just as the Google Play Protect analyzes applications within the Play Store for malicious applications, it can also be activated on users' devices too. It scans applications prior to installation on a mobile device, and also scans them after installation. If permissions of a mobile app are granted by the user, Google Play Protect uploads the scan results to Google Play and applies them for the protection of other users. There are also other actions that can be taken on the user side. Apps can be analyzed during or after installation on a mobile device by applying security solutions such as mobile anti-virus programs. Harmful applications that can access sensitive resources may already be installed on mobile devices. Therefore it is these circumstances that reveal the vital importance of security solutions being applied both to the market and the user sides in order to protect users prior to an application being installed.

The permission mechanism is one of the most significant Android security mechanisms since it permits or prohibits applications' from accessing critical resources. Applications require permission approval by device users in order to access critical system resources or sensitive user data. According to the device's Android version or application's target Software Development Kit (SDK) version, permission requests considered dangerous are presented to the user for approval during the application installation process, or when the application is running (when a critical resource receives an access attempt by an app) (Section 2.1.1.). Permissions are requested by apps in order to use certain Application Programming Interfaces (APIs) (Section 2.1.2.) that access critical system resources or sensitive user data. The mechanism assumes that users are able to determine whether or not the listed permissions are genuinely required for the functionality of the application. For example, if a word-processing application demands access to location data, the user should reconsider whether or not to proceed to install the application.

1.3. Description-to-Permission Fidelity Problem

The permission mechanism assumes that users are sufficiently competent and able to determine whether or not an application actually needs the requested permission in order to fulfill its expected functionality. However, users are not always that careful or they may not be adequately discerning to make the necessary inference about perceived irregularities between requested permissions and expected functionality. In order to evaluate the awareness of users with regard to the Android permission mechanism, Felt et al. (2012) [14] conducted a survey and laboratory study. Their study showed that 42% of participant users were not aware of the permission mechanism at all, and only 17% of the participants actually noticed requested permissions during the installation of applications. Furthermore, just 3% of the participant users actually understood and recalled the exact functionality of an application seeking the requested permissions. Consequently, it can be concluded that users are inclined to install potentially harmful applications by blindly or inadvertently accepting, although not needed, the permissions requested by the malicious application.

Android applications are distributed centrally through application markets. Application markets provide application metadata such as the description of an application, user scores, and user comments with application packages. Application metadata could therefore be used for the benefit of users' security and privacy. In recent years, application metadata has been used for malware detection [15, 16]. There have also been studies on detecting applications that request permissions beyond their needs by defining correlations between application description topics and the required API usage [17]. Another important usage of metadata is to discover inconsistencies between requested permissions and application descriptions [18–20]. From a privacy and security perspective, if the functionality of an application is sufficiently detailed in the application description, the need for the requested permission can be much better understood by the user. This is termed as *description-to-permission fidelity* [19] and this is what is expected from an application developer. If there is no identifier found in the application description (or in the rest of the application metadata such as application screenshots) indicating the need for a requested permission considered to be dangerous, the application may be legitimately considered as a suspicious application.

In the current study, a novel approach is defined in order to detect text parts in application metadata that reveal the necessity of a requested permission that is considered as being potentially dangerous. As previously mentioned, users do not adequately benefit from the permission mechanism in order to adequately protect their security and privacy [14]. Rather than permissions, users may benefit more from application descriptions, since they are non-technical and therefore more likely to be considered understandable by users. At this stage, the work within the scope of this thesis can be considered as a supporting tool to the permission mechanism. The approach and tool proposed in the current study can therefore be used as follows:

- To warn mobile device users if an application description does not explain the necessity of a requested permission (user benefit).
- To encourage application developers to better express their application functionality for requested permissions prior to uploading an application to an application market (developer benefit).

- To prevent the upload of applications which have insufficient descriptions for requested permissions from a security and privacy perspective (application market benefit).

1.4. Major Contributions of the Thesis

A novel approach for the description-to-permission fidelity problem is proposed in the current study in order to identify inconsistencies between requested permissions and application metadata through the use of natural language processing (NLP) techniques and recurrent neural networks (RNNs). The basic assumption was that the need for permissions considered to be dangerous must be adequately defined in the application description, and that the absence of permission needs in the description creates a suspicion. Applications detected as suspicious can then be analyzed using more complex techniques. The experimental results show that the proposed approach achieves a high degree of accuracy in detecting permission expressions from application metadata, and could be used for the protection of users' privacy and security. In addition to application descriptions, the effect of user reviews on discovering inconsistencies was also investigated in the study.

In order to achieve the objectives of the thesis:

- Dataset was created with 2,641 applications containing description texts for three different permissions (READ_CONTACTS 832 apps, RECORD_AUDIO 1008 apps, and STORAGE 801 apps). The dataset is called DesRe[1] (DEScriptions and REviews of Android applications).
- A novel model which uses state-of-the-art NLP techniques and RNNs was proposed in order to discover inconsistencies between application permission requests and application metadata.
- The effects of user reviews on assessing application fidelity was investigated.
- An extensive evaluation of the proposed model against other state-of-the-art models is presented, and the results discussed.

[1]Instructions to use the DesRe dataset is presented in the dataset web page: `https://wise.cs.hacettepe.edu.tr/projects/security-risks/dataset/`

1.5. Structure of the Study

The structure of the thesis is as follows:

Chapter 2 presents the essential background knowledge. It starts off by presenting the background to the Android application, and also provides an overview of the Android permission mechanism. Background information on NLP is then given, followed by a brief introduction to RNNs.

Chapter 3 provides an overview of previous published studies that assessed the description-to-permission fidelity of Android applications. Specific focus is given to studies employing NLP techniques and RNNs by using application descriptions for Android application security. In addition to studies that use application descriptions, studies using other metadata are also investigated. This chapter also presents other prominent studies on the topic, ending with a discussion on related studies.

Chapter 4 describes the proposed model which uses state-of-the-art NLP techniques and RNNs in order to establish inconsistencies between application permission requests and application metadata. An overview of the model is presented, as well as two different versions of the model; a sentence-based model, and a description-based model.

Chapter 5 introduces the dataset used in the study. Additionally, datasets from related studies which are also used in the current study are also described. Metrics used for the evaluation of the study's experiments are defined. Finally, the experimental results are presented and then discussed.

Chapter 6 concludes the thesis with a brief summary of the work undertaken and its contributions made to the field of Android application security. The chapter and thesis ends with a discussion on the limitations of the study, and suggestions as to future topics that could be studied based on the context of the current study.

2. BACKGROUND

In this chapter, background information is given in order to follow the context and approaches presented in subsequent the sections of this study. First of all, basic information about Android applications and the Android permission mechanism is given in Section 2.1.. Then, in Section 2.2., word embeddings are explained, and then recurrent neural networks (RNNs) are introduced briefly in Section 2.3..

2.1. Introduction to Android Applications

An Android application can be implemented using programming languages such as Java [21], C++, and Kotlin [22]. Android Software Development Kit (SDK) tools are used in order to build and implement Android application projects [23]. In the build operation, implemented source code, resource files and dependent libraries are processed by compilers. The source code is then converted to a Dalvik Executable (DEX) [24] file, which is then executed by Dalvik Virtual Machine (DVM) and the remainder of the inputs are converted into compiled resources. The DEX file is then combined with the compiled resources into an APK file by the APK Packager, which is then signed. The APK file is an archive used to install the application to a mobile device using the Android OS.

The Android OS is a multiuser Linux environment that regards each application as a different user through use of unique user IDs [23]. Each user ID is known only by the OS, and is used to set permissions for all files within the application. Each application has its own process and own virtual machine (an instance of DVM). This means that each application runs within an isolated environment, within its own security sandbox, without adversely affecting other applications, operating systems or users. This creates a secure environment in which an application can only access the resources and system components it has the necessary permission for. The primary purpose of this type of secure environment is to protect users' security and privacy.

2.1.1. Android Permission Mechanism

An Android application needs to obtain permission approval in order to access sensitive user data (e.g., SMS messages or images) or critical system features (e.g., a mobile device's camera or microphone) [25]. The permissions required by the app must be listed in the application's manifest file. With respect to the resources the application requests access to, permissions can be granted either by the OS automatically or manually by the device user.

The aforementioned manifest file, which is strictly named as "AndroidManifest.xml," is used by Android SDK, the Android OS and also by Google Play. It is positioned at the root directory of the project. The manifest file defines the application ID (which is actually a package name such as "com.android.chrome"), permissions requested by the application, application components (e.g., activities, services, broadcast receivers and content providers), and the required hardware and software features. Permissions needed by the application are declared by using the <uses-permission> tag in the manifest file.

When a permission is requested by an application, either the Android OS requests that the device user grants permission, or the Android OS automatically grants the permission [25]. The Android OS decides which option to apply based on the requested permission type, i.e., who the protection level of the permission belongs to. There are three different protection levels; normal permissions, signature permissions, and dangerous permissions. Normal permissions permit the application access to data or system features outside of its sandbox, but access to them by an app does not present much of a risk. Some examples for normal permissions are INSTALL_SHORTCUT or VIBRATE. Normal permissions are automatically granted by the Android OS during the installation of an app. Signature permissions are also granted automatically by the Android OS during the installation of an app. However, these permissions are only used if the requesting application is signed with the same certificate as the app that declared the permission, e.g., BIND_PRINT_SERVICE. On the other hand, requested dangerous permissions are listed for user's prompt, meaning that the user has to explicitly approve the requested permission. As these permissions relate to data that have the potential to violate the user's privacy or present a risk to the usual operation of the system, they are considered to be potentially dangerous. For example, READ_CONTACTS, READ_CALENDAR and RECORD_AUDIO are considered to be dangerous permissions.

Only dangerous permissions are presented to the device user, and are prompted as permission groups. Dangerous permissions and permission groups are listed in Appendix B. Permissions are listed to the user for approval at the time of an application's installation where the device's Android version is 5.1.1 or lower, or the application's target SDK version is API level 22 or lower while running on any version Android OS (Section 2.1.2.). In this case, all dangerous permissions are presented to the user with two options, either to accept all of them or to reject all of them. If the device user opts to reject, the app's installation is canceled by the OS. However, if the mobile device is running on Android 6.0 OS or above, and the application's target SDK version is API level 23 or above, then no permissions are requested at the point of an application's installation. When the device user runs the application, if a source that requires a granted dangerous permission is attempted to be reached by the application, a dialog box appears and presents a permission group header to the device user. If a device's user does not approve the request, the related source is not used, but the application remains running, but only performs operations for which it uses granted permissions and permitted sources. In order to change permission approvals one-by-one, the device user needs to open and adjust certain system settings.

2.1.2. Application Programming Interfaces (APIs)

Application Programming Interfaces (APIs) allow the capabilities of an application/service/-platform (e.g., Google, Ubuntu, Android) to be used by another application. The rules and limitations defined by the provider must be followed in order to utilize these capabilities. In order to develop Android applications, developers need to interact with the underlying Android operating system, and Android SDK provides Android Platform APIs for this purpose. Android SDK is a set of development tools, and aside from APIs it also includes relevant documentation for APIs (API docs), libraries, and tutorials for the Android OS, etc. Application developers need to use Android SDK in order to develop Android-compatible mobile applications. The Android API Package includes resource classes that are used by mobile apps, and it also defines application permissions for system features. On the other hand, media APIs includes classes to play or record, i.e., to manage media in audio and video.

Each Android OS release comes with an upgraded set of APIs. Actually, each Android release has a version, a dessert (code) name[1], and an API Level. Detailed Android release chronology is presented in Appendix C. Whilst different API Levels or different Android versions may have the same corresponding code name, for each Android version there is a specific API level. API levels are defined with an increasing integer value. API updates are released in a way so that they remain compatible with previously released versions.

[1]Starting from Android 10, released in August, 2019, Google no longer names Android releases after dessert names and uses just numbers.

2.2. Word Embeddings

In 1954, a linguist called Zellig S. Harris [26] posed the question of whether or not languages have distributional structures. According to Harris [26], languages can be structured in terms of certain features such as social intercourse, historical change, and distribution, and that this structure is detectable. Harris inspected language by evaluating the distributional structure of various language elements, together with their occurrence with other language elements, disregarding certain independent features. Even though most humans believe that they choose random words when conversing, in fact they choose words according to certain linguistic rules, from a set of sentence parts that are then presented together. Even the order in which these parts occur is distributionally structured

In order to process textual data, different mathematical representation methods can be applied to text. One approach to represent words is one-hot-encoding. Word representations are created by using vocabulary size vectors filled with zeros and using one for a corresponding word. The drawback here is that since the vectors are high-dimensional sparse vectors, it is computationally too costly in fact to use them. Also as the number of words grows, representation size also grows. Another approach to representing words is through the use of distributional representations. The idea uses co-occurrences of terms in a corpus as a means to representing them. To find similarities between two texts, co-occurrence of sentence parts can be applied, because sentence parts that have similar meanings happen to occur in similar contexts. To extract relations of words or texts, contexts are held as weighted vectors with respect to frequency counts or term frequency-inverse document frequency (tf-idf) [27] scores of words in texts. The meaning of any given word can therefore be represented as a vector of that word's relatedness to each context.

In 2013, Mikolov et al. [28] introduced the "word2vec" representation model as a prediction-based unsupervised method. The key principle behind the model is to use each entry in a vector as a hidden feature in order to represent the words. This feature can represent a semantic relation or syntactic relation. There are two different models used by word2vec to create word embeddings, and these are 'continuous bag of words' (CBOW) and 'skip-gram.' In CBOW, the word which stays in the center is attempted to be predicted using the words that surround it. CBOW is considered to work better on smaller rather than larger datasets . On the other hand, in the skip-gram model, the word at the center of the window is used in order to predict the surrounding words. The skip-gram model works better on larger rather than smaller datasets. The skip-gram model requires more computational power than the CBOW model. Using these models, word embeddings can be created using text data as input. There are also pretrained word2vec embeddings that have been trained using these models on Google news data. GloVe [29] is another word representation model in which co-occurrences of words are used for training. In GloVe, one can find embeddings with vector sizes of 25, 50,100, 200, and 300.

Another model for creating word embeddings is fastText [30]. FastText uses bag of character n-grams as input to the neural network, which consist of the substrings of any given word. For example, for the word *network*, with "n" equal to three, $<net, etw, two, wor, ork>$ tri-grams are used by fastText as input to the neural network and representations are created for each. The fastText representation of the word *network* is obtained by adding these vectors. As a result, fastText achieves a good level of performance on word representations.

In the current study, pretrained fastText word vectors were used as word embeddings with a vector dimension size of 300. These vectors were trained using Common Crawl [31] and Wikipedia [32] data, with character n-grams of length 5, i.e., 5-grams.

2.3. Recurrent Neural Networks

The human brain processes or learns from many data types sequentially. These data types include song lyrics, textual data, video data, and audio data, which are all examples of sequential data. Recurrent neural networks (RNNs) achieve superior levels of performance in modeling sequential data. This ability comes through the help of a mechanism that allows information to flow from one time step to the next. A basic RNN Unit is presented in Figure 2.1.. Therefore, RNNs are able to remember previous inputs so that they can make decisions based on both previous inputs and the current input. On the contrary, feed forward neural networks process data only at a given time step, and thereby independently from the input given to previous time steps.

Neural networks are trained using the backpropagation method. A forward pass is performed through the network and predictions are calculated. After that, predictions are compared with ground truths and as a result of this process, error values are subsequently derived. These error values are calculated by using loss functions and are regarded as to how weakly the network performs, i.e., its performance strength. Taking these error values into account, internal weights of the network can then be adjusted. The adjustment value is called the gradient and is calculated through a backpropagation algorithm with respect to the effect of the gradients of the previous layer. Gradients are values that tell us how much to change the weights in order to efficiently decrease the cost (i.e., to decrease the number of errors) of the network. The goal with backpropagation is to update each of the weights in the network so that the actual output is closer to that of the target output, thereby minimizing the error for each output neuron, as well as that of the network as a whole.

When performing backpropagation, by the time this information flows through states, each node in a layer calculates its gradient based on the effects of gradients in the layer before. As a result of this, in RNNs, the effect of the information to next state vanishes exponentially as information flows layer over layer. In other words, RNNs may experience difficulties in retaining information from previous layers. This constraint is called the vanishing gradient problem [33], and its result is defined as short-term memory problem in RNNs.

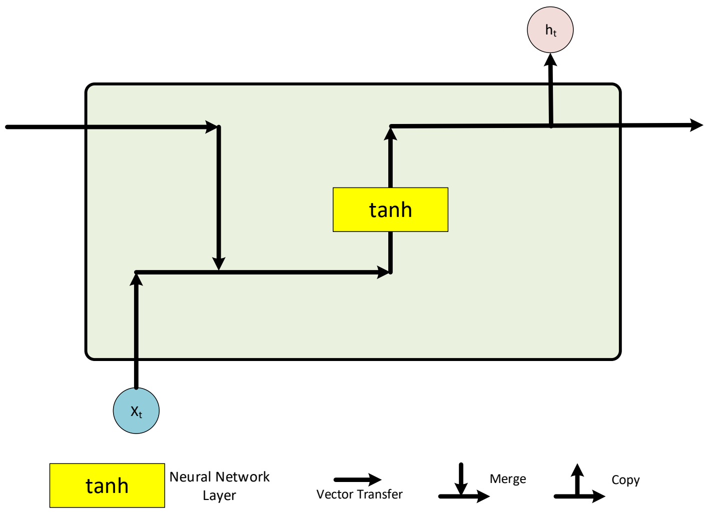


Figure 2.1. A basic RNN unit

"Neurons that fire together, wire together."(Hebbian Theory)

Long Short Term Memory (LSTM) networks offer a solution to RNNs' short-term memory problem by employing gates to adjust information flow in the network. By doing so, LSTMs aims to pass only the relevant information to proceeding sequences and forget rest of it to make better predictions. An LSTM unit is presented in Figure 2.2..

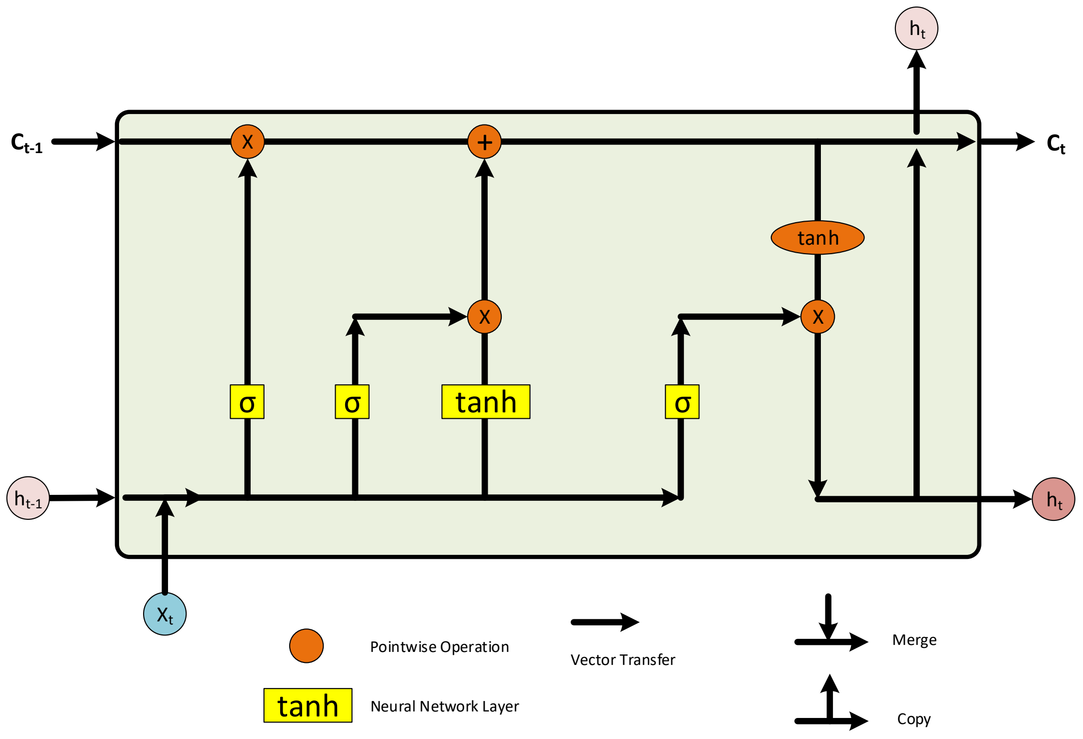


Figure 2.2. An LSTM unit

2.4. Conclusion

In this section, the essential background knowledge is presented that is subsequently referred to throughout the remainder of the thesis. As the current study focuses primarily on the usage of neural networks for mobile application security, a general overview of the field is given from an application security perspective using basic terms and definitions. Additionally, word embeddings and terms considered significant to Android applications are presented.

3. RELATED WORK

Mobile users review application descriptions in order to see whether or not an application meets their needs. This makes application descriptions indispensable in the communication between application developers and mobile device users. From the privacy and security perspective, if an application's functions are presented to the mobile device user in sufficient detail within the application description, then the reasons why the requested permissions are needed could be much better understood, which is termed 'description-to-permission fidelity' [19]. However, since users are not always that careful, or sometimes unable to infer a level of corroboration between the requested permissions and the application description [14], there is considerable potential in the idea of studying application descriptions and requested permissions as a means towards the protection of users' privacy and security [15, 17–20, 34].

The purpose of the current study is therefore to examine the levels of consistency between application metadata and requested permissions. As a first step, prominent studies in this context were analyzed in detail. The key studies are listed as follows in accordance with their date of publication:

- The first prominent study in the area presented the WHYPER [18] framework (2012), which aimed to assess description-to-permission fidelity of a mobile application by using NLP techniques for sentence structure analysis. The framework used WordNet [35] to create a semantically-related vocabulary set of a requested permission.
- Then, a fully-automated framework called AUTOCOG [19] was introduced in 2014, which uses application descriptions rather than API documents to extract semantic information.
- Another important study was ACODE [34] (2015), which was a largescale study that combined static code analysis and text analysis.
- The most recent study related to the area of assessing description-to-permission fidelity is AC-Net [20] (2019), which utilizes RNNs in order to learn and detect semantic relations.

There have also been other studies relating to assessing the fidelity of applications. A set of leading studies and their approaches are briefly summarized in the following subsections, followed by a discussion of the prominent studies.

3.1. WHYPER Framework

A primitive way of assessing description-to-permission fidelity is to perform a basic keyword-based search on a mobile app's application description. This takes a set of words related to a given permission and checks as to whether or not any of the keywords appear in the application description text. One problem experienced with keyword-based searches is gathering all the keywords related to the permission. Another problem is the detection of semantic relatedness of words or texts. By its very nature, keyword-based searches look for exact terms specified in the query. A word can have different meanings in different texts and that can have confounding effects on the results of keyword-based searches. Another disadvantage of the keyword-based search arises when semantic inferences are used within sentences. Here, semantic inference means describing the usage of a permission without use of a particular word. Consequently, keyword-based searches overlook semantic relationships between the words in the search text and the keywords, adversely affecting the search results. In order to overcome the impact of these disadvantages, NLP techniques can be employed.

The first published work on assessing the description-to-permission fidelity proposed a framework called WHYPER [18] which employs NLP techniques. WHYPER creates a semantic graph for each permission by using API documents and a lexical database called WordNet [35]. WHYPER also creates first order logic (FOL) representations of description sentences. Using the semantic graph and FOL representations, the model identifies whether or not there is a need for a permission according to the application description.

Permissions are requested by applications in order to be able to use APIs that require access to critical system resources or to sensitive user data. In WHYPER, for a requested permission (e.g., READ_CONTACTS), a semantic graph is defined as an illustration of resources managed by actions. In creating the semantic graph, noun phrases (as resources) and verb phrases (as actions) are extracted from the related API documents (e.g., android.media.AudioRecord for the RECORD_AUDIO permission). For each resource and action, related words from WordNet [35] (also known as synsets) are added to the semantic graph. As a result, a set of noun phrases for resources and a set of verb phrases as actions are held for each permission's semantic graph. These semantic graphs then together form the semantic model.

Another input to the WHYPER framework is the application description. Taking the description text of a mobile application, WHYPER extracts grammatical relationships of words and part-of-speech (POS) tags of each word within sentences using Stanford Parser [36, 37]. Then, WHYPER creates a FOL representation of each sentences. Taking these representations and the semantic model of a specific permission as its input, the WHYPER framework detects sentences that indicate a legitimate need for the requested permission. Detected sentences are called *permission sentences*. The design of this framework is summarized in Figure 3.1..

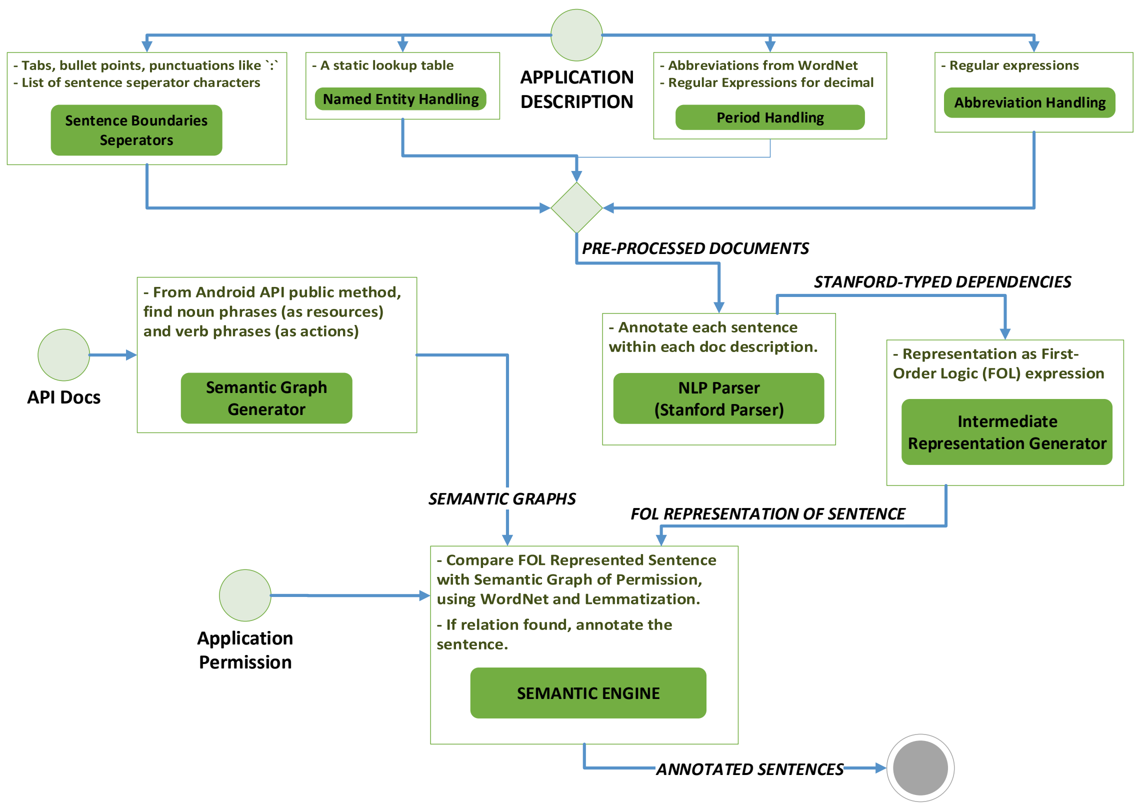

APPLICATION DESCRIPTION
- Tabs, bullet points, punctuations like `:`
- List of sentence seperator characters
Sentence Boundaries Seperators
- A static lookup table
Named Entity Handling
- Abbreviations from WordNet
- Regular Expressions for decimal
Period Handling
- Regular expressions
Abbreviation Handling
PRE-PROCESSED DOCUMENTS
STANFORD-TYPED DEPENDENCIES
API Docs
- From Android API public method, find noun phrases (as resources) and verb phrases (as actions)
Semantic Graph Generator
- Annotate each sentence within each doc description.
NLP Parser (Stanford Parser)
- Representation as First-Order Logic (FOL) expression
Intermediate Representation Generator
SEMANTIC GRAPHS
FOL REPRESENTATION OF SENTENCE
Application Permission
- Compare FOL Represented Sentence with Semantic Graph of Permission, using WordNet and Lemmatization.
- If relation found, annotate the sentence.
SEMANTIC ENGINE
ANNOTATED SENTENCES

Figure 3.1. Design of the WHYPER framework

In the scope of the current study, semantic graphs were created for three specific permissions. These per-missions are READ_CONTACTS, RECORD_AUDIO, and READ_CALENDAR.

The framework was tested for 9,953 manually labeled sentences extracted from the description texts of 581 free-of-charge popular applications (collected in January 2012). The results (see Table 3.1.) show that WHYPER is able to find out permission sentences with an average recall of 81.5% and an average precision of 82.8%. The WHYPER results were also compared with keyword-based search results, with the WHYPER results showing a 40% improvement on precision .

Table 3.1. Performance of WHYPER

Permission	**#App**	**#S**	**#SI**	**Precision**	**Recall**	**F-Score**	**Accuracy**
READ_CONTACTS	190	3,379	204	91.2	79.1	84.7	97.9
READ_CALENDER	191	2,752	288	83.7	85.1	84.4	96.8
RECORD_AUDIO	200	3,822	259	75.9	79.7	77.4	97

$^{\#S}$ Number of sentences

$^{\#SI}$ Number of sentences identified as permission sentences

3.2. AUTOCOG Tool

In their study called AUTOCOG, Qu et al. (2014) [19] took the use of NLP techniques one step further in extracting permission sentences from application descriptions. Their study defined the following limitations with regards to the WHYPER framework:

- Since API documents cannot contain all possible patterns of related words that are correlated with a certain permission, there is a need for additional resources to obtain additional semantic information. lack of API documents for certain permissions prevents the framework from being used for applications that request these permissions.
- Manual work is required in WHYPER in order to extract patterns for semantic graphs from API documents. Without some level of automation, extending the use of WHYPER to other permissions does not seem feasible.

Due to these limitations in WHYPER, Qu et al. [19] proposed a fully-automated framework called AUTOCOG, in which semantics are gathered only from application descriptions. In AUTOCOG, the semantic relatedness of a description and a permission are measured using Explicit Semantic Analysis (ESA) [38, 39]. Instead of using a dictionary-based corpus like

WordNet [35], as performed in WHYPER, ESA uses a known large knowledgebase (i.e., Wikipedia) in order to create vectorial representations of the texts. The key point here is that dictionary-based approaches do not propose a solution to the word sense disambiguation problem. On the other hand, approaches such as ESA work on the meaning of a word or document rather than its vocabulary form. The word sense disambiguation problem is caused by different meanings of a word in different contexts. In Wikipedia-based ESA, to extract relations of words or texts, a concept is created for each article and these concepts are held as weighted vectors with respect to term frequency-inverse document frequency (tf-idf) [27] scores of words in articles. The meaning of any given word can then be represented as a vector of that word's relatedness. This vector can be considered as a projection of an input article into Wikipedia-based concepts. The input text is represented by a vector and the semantic relation between two texts is estimated by calculating the cosine similarity of vectors.

Using ESA, two models are developed by AUTOCOG, the Description Semantics Model and the Description-to-Permission Relatedness (DPR) Model. The Description Semantics Model is used in order to understand the meanings of application descriptions, and is able to measure semantic relatedness of two texts. The DPR Model is used to extract noun phrases and np-counterparts[1] that have a statistically positive correlation with a given permission in related description documents. The design of the AUTOCOG tool is summarized in Figure 3.2..

In the learning phase, AUTOCOG extracts noun phrases from descriptions and creates a semantic relatedness score matrix among them using ESA. After filtering out less frequent noun phrases, the remaining noun phrases with a semantic relatedness score higher than a predefined threshold are then grouped together. By using these groups and their relatedness scores, a relatedness dictionary is created between noun phrases. To relate permissions with noun phrases, just considering the number of times permissions and noun phrases are used together will lead to an increase in false positives. Considering these concerns, a special metric is defined so as to relate permissions with noun phrases. Then, using a final threshold value (500) over relatedness results, correlations between noun phrases and permissions can be defined by the DPR Model. In a final step, AUTOCOG extends noun phrases by pairing them with related np-counterparts. As a result, each permission has a list of related noun phrases and np-counterpart pairs.

[1]np-counterpart is defined in AUTOCOG as verb-phrases and noun-phrases that are detected as governor-dependant pairs with noun-phrases residing at leaf node of Stanford Parser output. In addition to this, if the noun-phrase contains the own relationship, relative possessive is also an np-counterpart.

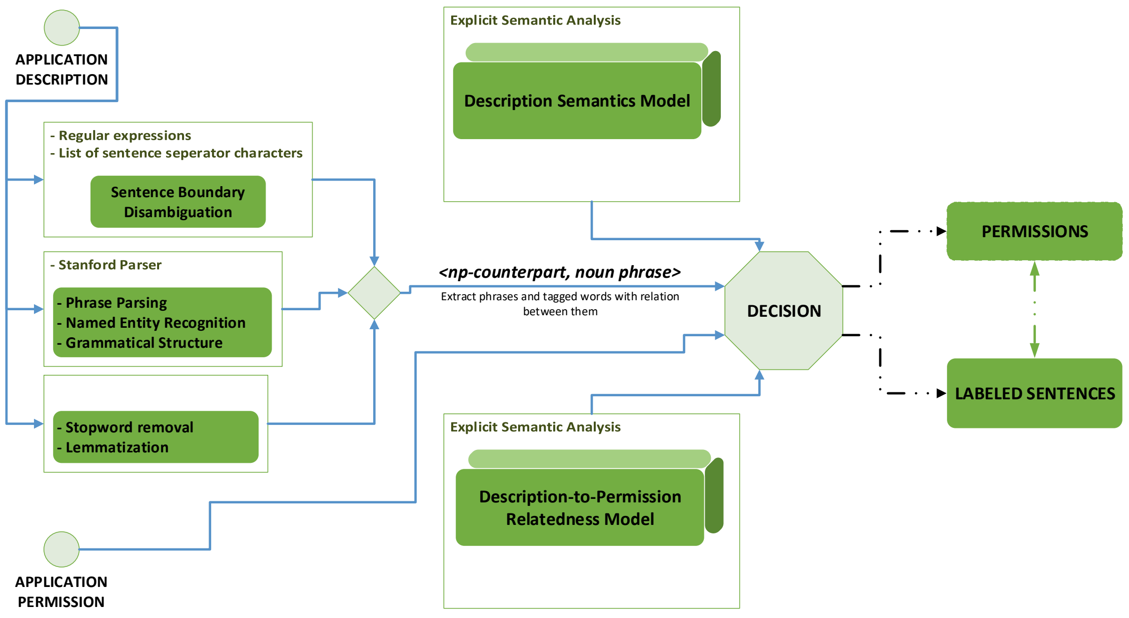

APPLICATION DESCRIPTION
- Regular expressions
- List of sentence seperator characters
Sentence Boundary Disambiguation
- Stanford Parser
- Phrase Parsing
- Named Entity Recognition
- Grammatical Structure
- Stopword removal
- Lemmatization
APPLICATION PERMISSION
Explicit Semantic Analysis
Description Semantics Model
<np-counterpart, noun phrase>
Extract phrases and tagged words with relation between them
DECISION
Explicit Semantic Analysis
Description-to-Permission Relatedness Model
PERMISSIONS
LABELED SENTENCES

Figure 3.2. Design of the AUTOCOG tool

Given an application description and a requested permission, AUTOCOG extracts noun phrase np-counterpart pairs from the description text and then identifies whether or not a need for the requested permission is stated in the description text.

A large application dataset (37,845 application descriptions collected from Google Play Store in August 2013) was then used in order to train the Description Semantic Model and the DPR Model. Evaluations were performed for 11 different permissions using 1,785 applications downloaded from the Google Play Store in May 2014. The results show that AUTOCOG achieves[1] an average recall of 92% and average precision of 92.6%. Test results for the studied permissions are presented in Table 3.2..

Another evaluation test was performed with 45,811 application descriptions downloaded from the Google Play Store. As the conclusion of the test, it was observed that only 9.1% of the descriptions had indications for all the requested permissions by the application.

[1]In WHYPER, number of sentences is used to calculate experiment results. On the other hand AUTOCOG evaluates results in terms of number of descriptions. Results of the WHYPER framework in the terms of number of descriptions for average precision, recall, F-score, and accuracy are %85.5, %66.5, %74.8, and %79.9 respectively.

Table 3.2. Performance of AUTOCOG

Permission	**Precision**	**Recall**	**F-Score**	**Accuracy**
READ_CONTACTS	95.2	91.7	93.4	92.6
READ_CALENDAR	94	92.9	93.5	94.4
RECORD_AUDIO	92.1	91.4	91.8	89.5
GET_ACCOUNTS	89.5	87.2	88.3	94
WRITE_CONTACTS	93.4	90.5	91.9	93.3
CAMERA	90.5	91.8	91.2	91.3
WRITE_EXTERNAL_STORAGE	89.8	91.4	90.6	92.7
ACCESS_FINE_LOCATION	95	93.4	94.2	95.3
ACCESS_COARSE_LOCATION	98	92.5	95.1	96.7
RECEIVE_BOOT_COMPLETED	89.5	91.1	90.3	92.7
WRITE_SETTINGS	90.3	97	93.5	94
AVERAGE	**92.6**	**92**	**92.3**	**93.2**

While AUTOCOG performs much better than WHYPER, since it uses unsupervised learning, it could equally extract semantic relationships that may not actually exist, which may in turn lead to false positives.

3.3. ACODE Framework

While WHYPER is proposed as a means to alleviate the shortcomings of a keyword-based approach (confounding effects that a word can have different meanings and semantic inference that describes the usage of a permission without using a particular word), Watanabe et al. (2015) [34] proposed a keyword-based approach called ACODE (Analyzing COde and DEscription) due to being considered lightweight in performance terms for larger datasets, since ACODE does not require the labeling of application descriptions.

Their basic keyword-based search takes a set of words related to a given permission and checks whether or not any of the keywords appear in the description text. Being a lightweight method makes keyword-based search expedient for operations on large data volumes. Using this advantage of the method, ACODE performed experiments on 200,000 applications (100,000 from Google Play and 100,000 from third-party application stores) without any need for the manual labeling of description texts. By doing so, ACODE sets itself apart from other prominent studies that also analyze application descriptions [18–20].

In order to extract keywords for a permission, ACODE performs static code analysis. The objective of the static code analysis is to find out whether or not the application code has code parts that use the requested permission. As such, ACODE extracts the AndroidManifest.xml file from the application package file using the Android Asset Packaging Tool (aapt) [40]. In finding declarations of a permission from the AndroidManifest.xml file, ACODE searches for APIs or Uniform Resource Identifiers (URI) that require this permission in order to access critical system resources or sensitive user data in the application code. They use PScout [41], which maps permissions with APIs and URIs, in order to check whether or not the declared permissions in the manifest file are in fact needed. After that, using Apktool [42], ACODE disassembles the application's DEX file in order to search for a function call to separate unused code parts from those actually used. Since ACODE does not label description sentences manually, descriptions of applications that have used function calls are assumed as being relevant description texts. Keywords for the keyword-based search are selected by evaluating relevance weights of terms from relevant and non-relevant descriptions texts. Once the keywords have been identified for dangerous permissions, the keyword-based search is performed in order to find out whether or not an application description includes access to sensitive user data or access to critical system resources.

By combining static code analysis with text analysis, ACODE performs better than a basic keyword-based search used for comparison with WHYPER in [18] and produces comparable results with WHYPER. The comparative results of ACODE with the WHYPER Framework and a basic keyword-based method is presented in Table 3.3.. Other than being lightweight and not requiring manual text annotation, since ACODE is a keyword-based approach, it can be applied to different languages without too much effort or level of change.

Table 3.3. Comparison of ACODE with WHYPER and basic keyword-based search

Permission	Framework	Accuracy
READ_CONTACTS	ACODE	0.84
	WHYPER	0.89
	Keyword Search	0.74
READ_CALENDAR	ACODE	0.89
	WHYPER	0.92
	Keyword Search	0.74
RECORD_AUDIO	ACODE	0.74
	WHYPER	0.86
	Keyword Search	0.76

3.4. AC-Net Framework

The closest work to the current study in terms of the applied technique was recently proposed by Feng et al. (2019) [20]. The framework, called AC-Net, utilizes RNNs in order to learn and detect semantic relations. Application descriptions are sequential data and since RNNs are able to remember previous inputs, they are good at modeling sequential data. However, RNNs suffer from vanishing gradient problem due to using the backpropagation method of learning. This problem is defined as short-term memory problem in RNNs. Gated Recurrent Unit (GRU) [43], as used in AC-Net, offers a solution to RNNs' short-term memory problem by employing gates in order to adjust information flow in the network. In AC-Net, labeled application descriptions for 11 different permission groups are used for the purposes of training. Predictions over test data for learned permissions are generated as probability distributions in the model. The design of this system is summarized in Figure 3.3..

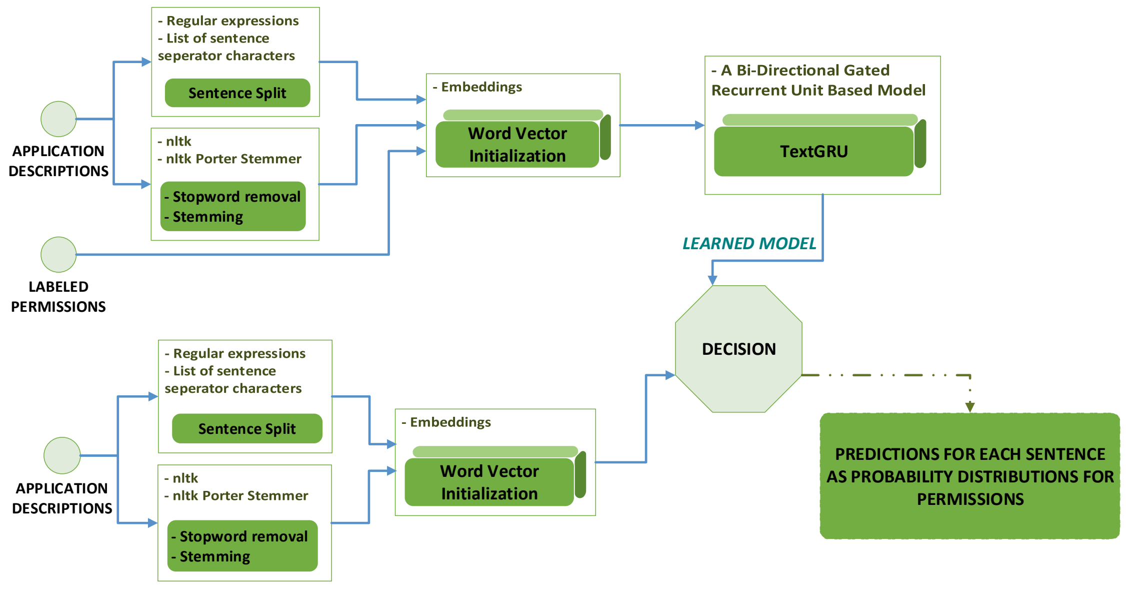


Figure 3.3. Design of the AC-Net framework

In the scope of the AC-Net study, embeddings were created by using 69,941 application descriptions (containing 783,199 sentences and 39,800 unique words) using word2vec [28]. Experiments were also performed using GloVe [29] and Common Crawl [44] word embeddings. Comparative results for different embeddings showed that newly created embeddings have a positive effect on the results. One possible explanation for this effect is that application descriptions which are domain-specific inputs are used to creating embeddings. As a result of this, relations between words in this specific context are likely to be learned and presented better by newly-created embedding vectors.

The system was evaluated by using approximately 25,000 sentences extracted from description texts of 1,415 popular applications (collected in December 2015). The results of AC-Net on three permissions[1] that were also evaluated by the previous related studies are presented in Table 3.4.. The results show that AC-Net was able to identify permission sentences with 24.5% more accuracy than the previous studies.

[1]AC-Net performed the evaluation using permission groups rather than permissions. CONTACTS permission group includes READ_CONTACTS, WRITE_CONTACTS, and GET_ACCOUNT permissions. CALENDAR permission group includes READ_CALENDAR and WRITE_CALENDAR permissions. MICROPHONE permission group includes RECORD_AUDIO permission.

Table 3.4. Performance of AC-Net

Permission Group	ROC-AUC					PR-AUC				
	AC-Net	Key-Based	WHYPER	AUTOCOG	ACODE	AC-Net	Key-Based	WHYPER	AUTOCOG	ACODE
CONTACTS	0.97	0.72	0.6	0.68	0.76	0.74	0.41	0.28	0.41	0.43
CALENDAR	0.99	0.98	0.65	0.89	0.95	0.84	0.71	0.28	0.5	0.71
MICROPHONE	0.96	0.8	0.59	0.79	0.79	0.49	0.44	0.24	0.4	0.51

3.5. Other Related Studies

There are also other studies on assessing the fidelity of applications. Gorla et al. (2014) [17], in their study called CHABADA, applied clustering techniques in examining the relations between requested permissions and application metadata. Processing application descriptions collected from the Google Play Store, topics for each application are first specified by means of Latent Dirichlet Allocation (LDA) [45]. By using k-Means [46] on specified topics, application clusters are then generated. Sensitive APIs used by applications within each cluster are then extracted. If an application uses an API different from the set of APIs extracted for its topic, it is considered as a potentially harmful application. The approach does not require a labeled training dataset, since it is an unsupervised method.

Felt et al. [14] analyzed users' behaviors and their awareness on Android permission mechanism. Although 42% of users were not aware of the permission mechanism at all, and 42% of users were aware of Android permission mechanism but did not look at the permissions requested during app installation, only 17% of the participants actually noticed the requested permissions. One way to benefit from this would be to make use of the experience and attention of these more cautious users. One of the most effective ways of doing so would be to utilize data from the user reviews. However, only a few studies exist on exploring the effects of user reviews in Android applications.

AUTOREB [47] makes application-level behavior inferences based on security- and privacy-related user reviews. In doing so, AUTOREB introduced the concept of review-to-behavior fidelity to the literature. They used text mining, information retrieval, and also machine learning techniques in order to analyze user reviews and classify applications into four different categories (Spamming, Financial Issues, Over-Privileged Permission, Data Leakage). Their experimental results showed that the introduced system is capable of predicting security behaviors of applications with accuracy as high as 94.04% through utilizing user reviews as input data.

User reviews constitute a favorable direct communication channel between mobile device users and application developers. Developers update their apps with respect to user reviews on application markets in order to make their apps more useful or more preferable to users. These updates are sometimes implemented on user interfaces, sometimes they aim to improve the performance of an app, and sometimes the updates are related to improvements in security and privacy. Nguyen et al. (2019) [48] were inspired by this, and recently investigated the relationship between security- and privacy-related application updates and user reviews. They extracted privacy-related user reviews using a keyword-based approach. They also extracted security- and privacy-related changes in the code by way of static analysis. Based on these extractions, the user reviews were then able to be mapped to the corresponding application updates. Their results showed that 60.77% of the security- and privacy-related reviews triggered a security and privacy update.

In research by Wu et al. (2017) [49], PACS was presented as another study that benefits from application descriptions and user reviews. In order to achieve that, it collects application descriptions, user reviews, and Android package (APK) files from applications in the market store. First, PACS decompiles the APK files and extracts the requested permissions of applications from their AndroidManifest.xml files. Then, PACS classifies the applications into 10 categories based on application descriptions and user reviews using Support Vector Machines (SVM) [50]. Similar to CHABADA [17], PACS uses descriptions for classifying applications. Then, it builds maximum frequently used permission item sets for each category by using Apriori Algorithm [51]. Using description and user reviews of a new application, PACS first finds the application's category and then lists the permissions that are expected to be requested by the app. Any additional permissions requested by the app are then considered as suspicious requests.

Due to the importance of application descriptions, there have also been studies conducted on the automated generation of application descriptions (AutoPPG [52], DescribeMe [53]), and the automated creation of informative text (DREBIN [54]) by prioritizing security and privacy concerns. DescribeMe [53] aimed at generating descriptions by taking into account the use of sensitive APIs by an application. First, the security behavior graphs of an app are created by static analysis, then natural language generation (NLG) techniques are employed in order to generate descriptions by traversing and parsing these graphs. DREBIN [54] transfers the attributes it collects from applications to a vector space and determines whether or not an application is malicious or benign by using SVM [50]. In addition, it creates a text describing why the application is classified as malicious. This text is created by adding phrases that match with attribute sets to predefined sentence patterns. The system also provides users with a confidence level value about the decision made.

AutoPPG [52] generates privacy policy for Android apps by using static code analysis and NLP techniques. It aims to generate easy-to-understand texts that inform device users about how their information will be collected, used, and ultimately disclosed. First, API documents are analyzed in order to identify personal information used by APIs. Then, by employing static code analysis, invoked sensitive APIs and their mappings to personal information are extracted. The information under which conditions these APIs are invoked and how the personal information is processed (e.g., stored, transferred) are also obtained in this step. Finally, NLP techniques are employed to generate privacy policies.

A recent study claimed that descriptions created by DescribeMe, AutoPPG, and also by DREBIN were too technical for end-users, therefore they proposed a system called PERSCRIPTION [55] through which personalized security-specific descriptions are created by learning users' security concerns. PERSCRIPTION collects approved and unapproved permissions by users and ranks them according to their proportion (allowed or denied) to detect security and privacy concerns of a specific user. By using the permission ranks and reordering the permission sentences' list collected from DREBIN, PERSCRIPTION creates a description for a given application by using its application category (defined in the application store) and permission requests. The overall aim being to provide device users with the most relevant sentences about the permissions that are of most concern to the user. PERSCRIPTION also benefits from certain psychological approaches in serving to users' linguistic preferences. Security and privacy concerns of mobile device users are determined based on changes made by users to the default installation settings of previously installed apps. According to the category distribution of apps previously installed, using the Big Five Model [56], the user's language preference is determined. Then, based on this information, a description of the application is created in a language style that the user can more readily understand and in a way that takes the user's security and privacy concerns into consideration. In this way, it is aimed to help users to understand security risks more easily and thereby to make more accurate decisions in time by becoming more conscious about the security and privacy of their mobile device and personal data.

Recently, metadata of applications has also been utilized for detecting malicious mobile applications. Martin et al. [15] proposed a detection system called ADROIT that uses metadata with permissions in order to discriminate malicious applications from the benign. In order to achieve that aim, classifiers are generated with the help of text mining methods. The researchers performed experiments by using six different classification algorithms (Random Forest, KNN, Decision Tree, AdaBoost, Naive Bayes, and Bagging). First, applications are labeled as malware or benignware by leveraging antivirus engines. Then, by using application metadata (e.g., description, developer ID and organization, user ratings, and permission list from the manifest), words that are expressed as different patterns are extracted. In order to do that, a corpus is created with the collected descriptions, and then this corpus is cleaned by using NLP techniques (e.g., removing stopwords and stemming). Then, the resulting set is converted to a term-document matrix by using the terms that have a frequency exceeding a predefined threshold. This matrix is then used in order to generate classifiers by using each of the six aforementioned algorithms. In another study [16], also uses metadata such as application category and description, besides API calls and permissions for the detection of malware in mobile apps. As a means of characterizing Android malware, Yang et al. [57] proposed a topic-specific approach based on application descriptions that uses information flow.

3.6. Discussion

As the purpose of the current study is to examine consistency between application metadata and the permissions requested by mobile device applications, this section presents a detailed discussion of the aforementioned prominent studies in this context [18–20, 34].

The keyword-based approach [34] is considered to be the most primitive approach for examining the reliability of mobile apps by comparing application metadata with requested permissions. This is considered to be a lightweight method, thereby making it possible to be applied to a large-sized database, and without the need for the manual labeling of text data. The method proposed by [34] makes the framework easily adaptable to other languages. Watanabe et al. [34] used a large-sized dataset containing 200,000 applications in order to assess description-to-permission fidelity. In doing so, `ACODE` produced results that were comparable to other studies that focused on description analysis. However, the approach encounters problems in the gathering of all the relevant words related to a requested permission. Another important challenge of keyword-based approaches is that they cannot detect semantic inferences within description texts.

The WHYPER framework offers the use of API documents in order to collect keywords for requested permissions. It also uses synsets of words from WordNet, which holds semantic relatedness of words, in order to compare descriptions against the requested permissions. Although two shortcomings of keyword-based searching (confounding effects that a word can have different meanings, and semantic inference that describes the usage of a permission without using a particular word) are addressed, WHYPER seems to fail at certain points. Since there are certain permissions that may not have related API documents, the task of collecting all the related words, i.e., obtaining complete semantic patterns, may not be achievable. Another weakness is due to use of words in synsets, which is devoid of context information, and can thereby result in the detection of unrelated sentences as permission sentences. This, naturally, can adversely affect the number of false positives. There is also a defined element of manual work in using the framework, which requires a considerable amount of effort for such a system.

AUTOCOG resolves the automation and missing API document problems by using only descriptions within its automated structure. By using ESA, relations of words are attempted to be found without losing context information. While AUTOCOG performs much better than WHYPER, it is also more susceptible to extracting relationships which are actually unrelated in context due to being an unsupervised approach.

The most recent study, AC-Net, which was developed during the same time interval as the current study, uses a specific implementation of RNN called GRU Networks to assess description-to-permission fidelity. Since RNNs are good at modeling sequential data such as application descriptions, hence AC-Net achieves better predictions than the previous studies by using this approach. Different from the previous studies, the AC-Net model extracts not just binary results for sentences as to whether or not they include explanation of a particular permission, but generates the degree of description-to-permission fidelity of the application. Created word embeddings in the scope of the AC-Net study enables a GRU-based RNN to learn from a domain-specific input. This seems to be the factor that resolves AUTOCOG's context problem, and as a result decreases the number of false positives.

As a conclusion, using RNNs for assessing the description-to-permission fidelity is the most suitable approach seen within the related studies. Since application descriptions are sequential data, and RNNs are good at modeling sequential data, the approach seems very promising for the issue of description-to-permission fidelity. Having the same idea in mind, both `AC-Net` and the current study, which were conducted independently and mutually unaware, utilized different specifications of RNNs in order to assess description-to-permission fidelity of mobile applications. While `AC-Net` uses GRU Networks, the proposed model of the current study uses Long Short-Term Memory (LSTM) Networks, which utilizes a more complicated gate structure in order to remember the previous data, and thereby resolve the short-term memory problem associated with RNNs. As another contribution, rather than only using application descriptions, the current study's proposed model also utilizes user reviews for assessing description-to-permission fidelity.

4. MODEL

This chapter presents the proposed approach which uses state-of-the-art NLP techniques and RNNs in order to discover inconsistencies between mobile application permission requests and application metadata. The aim is to detect indications of permission needs (expressions that reveal the need for a required dangerous permission) within application descriptions and user reviews. There are two different models implemented within the scope of the approach:

- A **Sentence-Based Model** was used to process each sentence within application descriptions in order to detect permission expressions within these sentences. Each sentence that was found was then labeled manually and the model trained using these manually labeled sentences. Once the model was trained, test data was then fed into the model on a sentence by sentence basis. Each sentence was classified as a permission sentence, with those that reveal a need for a requested dangerous permission being labeled as "1," or as a statement sentence and labeled as "0." Further details of the sentence-based model are described in Section 4.1..
- A **Description-Based Model** was designed so as to evaluate the effect of user reviews during the test phase. The model was trained using only application descriptions, with user reviews excluded at the training phase, but included at the test phase. Two different evaluation tests were performed for the description-based model. The first evaluation used only application descriptions at both the training and the test phases in order to define ground truth data. The second evaluation was performed in order to see the effects of user reviews beside descriptions. For the training phase, only application descriptions were used, but for the test phase, if the trained model did not find a permission expression in the application description, then the test was repeated for that application using user reviews of the relevant application as input. This description-based model is described in more detail in Section 4.2..

Within the context of the current study, each model was trained separately for each studied permission. This means that a binary classification was performed for each studied permission. The main reason for having chosen binary classification over multiclass classification was to avoid the burden of retraining the entire model each time a new permission was added to the system. Since three permissions were studied (RECORD_AUDIO, READ_CONTACTS, and STORAGE), three separate sentence-based models and three separate description-based models were created and trained independently from each other.

The real world problem aimed at being resolved in the current study was to extract indicators of a requested permission from application metadata (namely application descriptions and user reviews) in order to assist mobile device users in protecting their security and privacy. A neural network model was designed in order to perform this task. Descriptions and user reviews are both textual data and are sequential. Recurrent neural networks (RNNs) [58] are known to be successful on handling sequential data because unlike feed-forward neural networks, RNNs have an internal memory. RNNs use this memory in order to process input sequences and to remember previous inputs, thereby enabling RNNs to make decisions based on earlier inputs to the system.

Neural networks are trained using backpropagation. In performing backpropagation, when information flows through states, each node in a layer calculates its gradient using the effects of gradients in the previous layer. As a result, in RNNs, the effect of the information to next state vanishes exponentially as information flows over layers. In other words, RNNs may have difficulties in retaining information from previous layers. This constraint is known as the vanishing gradient problem [33], and its result is defined as short-term memory problem in RNNs. Long Short Term Memory (LSTM) networks [59] offer a solution to tackle RNNs' short term memory problem by employing gates so as to adjust information flow within the network. In doing so, LSTMs only pass relevant information to proceeding sequences and disregard other data in order to arrive at better predictions.

In the current study, LSTM networks [59] were used for the encoding of text data. In order to process textual data through an LSTM, the application descriptions were preprocessed. The preprocessing task involved sentence tokenization, word tokenization, punctuation removal, stopwords elimination, non-alpha character removal, and stemming[1] (Porter stemmer was used [60]). Once the model had been trained, the same preprocessing tasks were also applied for the test data.

Once the textual data has been preprocessed, syntactic and semantic features of each word within a sentence are represented by low-dimensional dense representation vectors. Within the current study, fastText [61] pretrained word embedding vectors for English were used for

[1]Applying stemming to input text did not have considerable effect on the results. The reason behind is, since fastText uses character 5-grams to encode words, stems and subwords were already encoded into the word embedding vectors.

word representations (Section 2.2.). These representation vectors were constructed (learned) using the distributional characteristics of words from largescale documents, and as a result, leading words with similar meanings were found to have similar representations.

4.1. Sentence-Based Model

An overview of the sentence-based model is presented in Figure 4.1., with the training phase on the left side and the test phase on the right. For the training phase, after preprocessing, each description sentence was fed into the model using fastText word embedding vectors. At the end of the training phase, the LSTM learned to create the compositional representation of a sentence. Also, the implemented binary classifier learned classifying sentences into either permission sentences or statement sentences.

In the test phase, a new application description sentence was given as an input to the trained model. The same preprocessing operations were applied and embedding vectors fetched for each word. Then, the model classified the new sentence as a permission sentence where it included an indicator for the studied permission.

The architecture of the sentence-based LSTM model is illustrated in Figure 4.2.. For each word within each description sentence, fastText embedding vectors were fetched, starting from the first word to the last word.

$$s_i = LSTM(x_{1:n}, i) \tag{1}$$

The *i*th sentence in currently processed application description is $sentence_i = \{x_1, \cdots, x_n\}$, and the embedding vector for the *j*th word within the sentence is x_j. The output of the LSTM is the compositional vector representation of the input sentence as in s_i in Equation 1. This vector representation, is then fed into a multilayer perceptron (MLP) for the purposes of classification. The prediction result, as in o_i in Equation 2, is the output of the sigmoid activation function. The given result is 1 (representing a permission sentence) or 0 (representing a statement sentence).

$$o_i = \text{sigmoid}(MLP(s_i)) \tag{2}$$

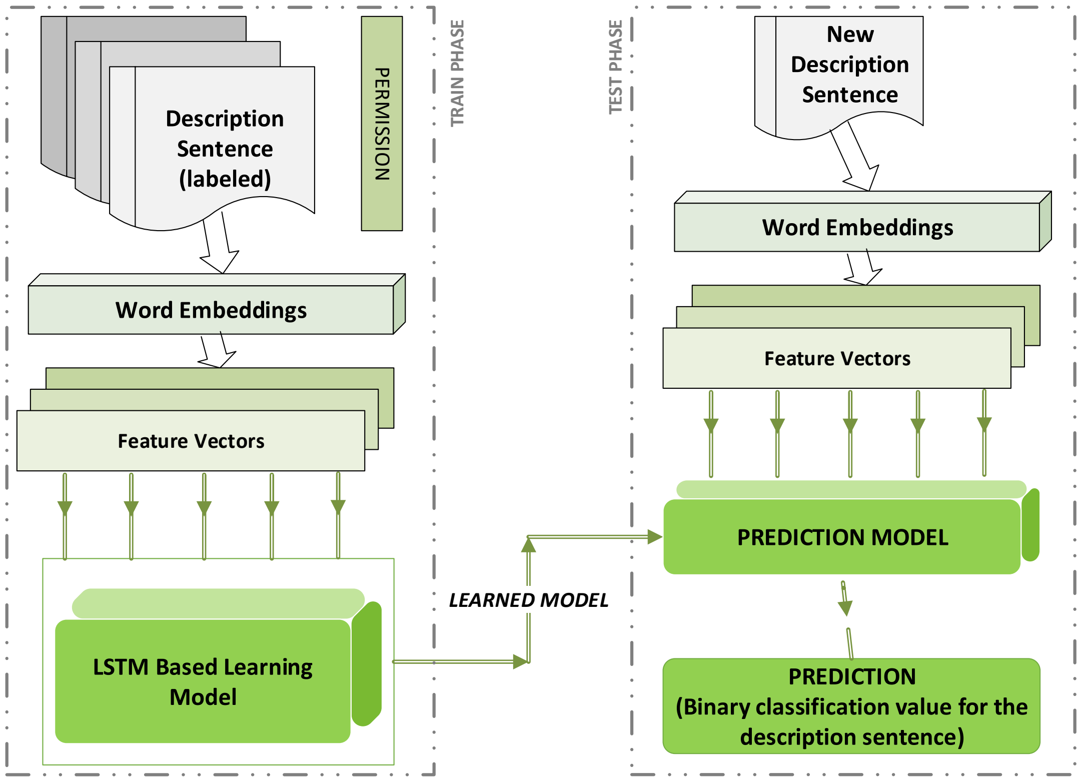


Figure 4.1. Overview of the sentence-based model

Since the model performs binary classification, binary cross-entropy was used as the loss function. The loss function is presented in Equation 3. y and $\hat{y}$ stands for the target output and prediction result, respectively.

$$L(y, \hat{y}) = -(y * \log(\hat{y}) + (1 - y) * \log(1 - \hat{y})) \tag{3}$$

In the test phase, a new application description sentence was given as an input to the trained model. The same preprocessing operations were applied and embedding vectors fetched for each word. Then, the model classified the new sentence as a permission sentence where it included an indicator for the studied permission. Prediction result of the sentence-based model is a value between 1 (representing a permission sentence) and 0 (representing a statement sentence).

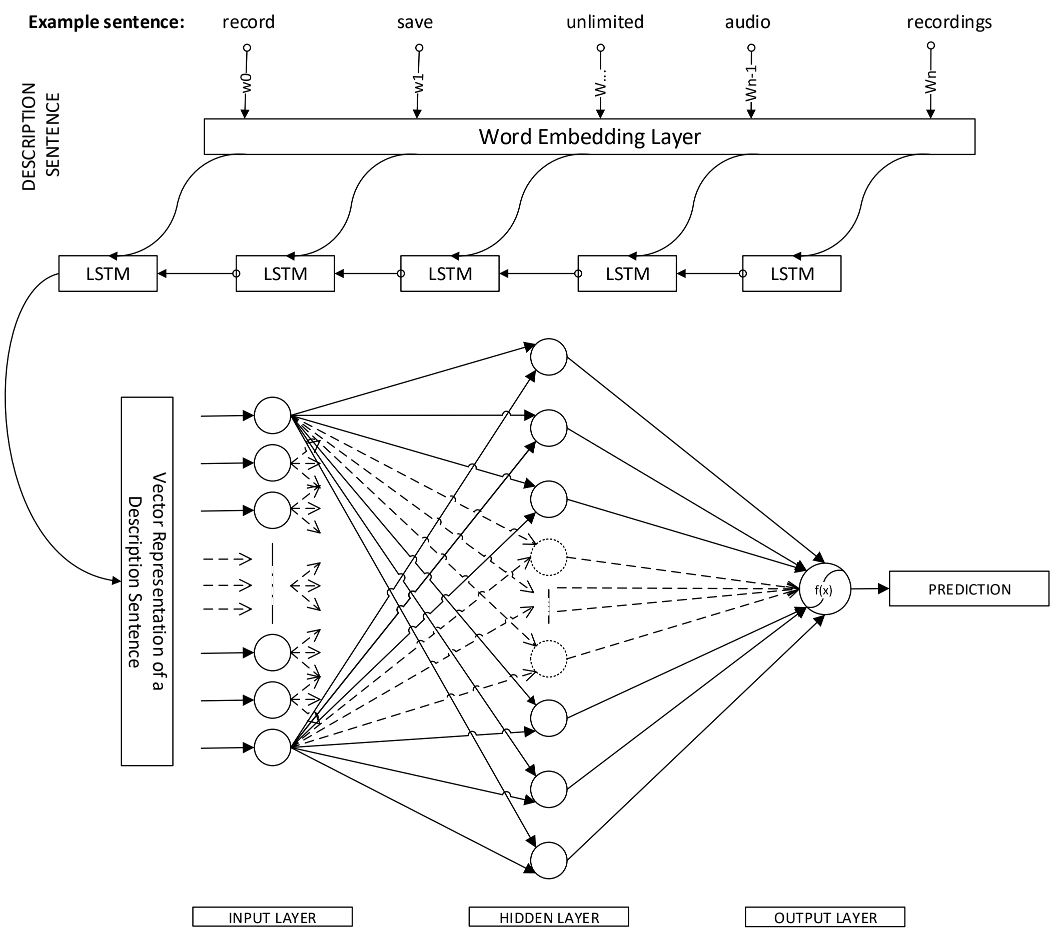
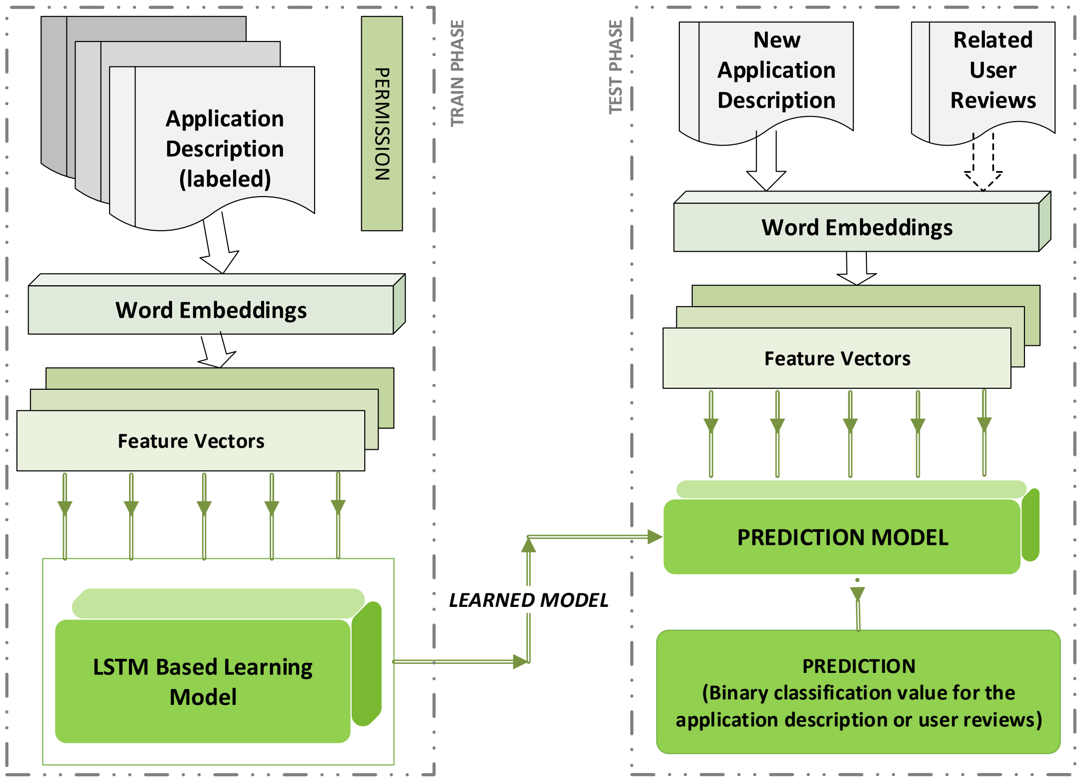


Figure 4.2. Architecture of the sentence-based LSTM model

4.2. Description-Based Model

An overview of the description-based model is shown in Figure 4.3., with the training phase presented on the left side and the test phase on the right. The objective of the description-based model is to evaluate the effect of user reviews on assessing description-to-permissions fidelity. For this purpose, two evaluation tests were performed. For both tests, the model was trained using only application descriptions. The aim of the first test was to find ground truth results using only application descriptions at the test phase. On the other hand, the purpose of the second evaluation was to find the effects of user reviews, therefore both application descriptions and user reviews were given, as follows, as inputs in the test. If the prediction



Application Description (labeled)
PERMISSION
TRAIN PHASE
Word Embeddings
Feature Vectors
LSTM Based Learning Model
LEARNED MODEL
TEST PHASE
New Application Description
Related User Reviews
Word Embeddings
Feature Vectors
PREDICTION MODEL
PREDICTION (Binary classification value for the application description or user reviews)

Figure 4.3. Overview of the description-based model

result of the description was smaller than 0.5, then the test was repeated for that application using user reviews as input to the trained model.

In the datasets, each sentence in the application descriptions was labeled. In order to define the label of each application description, the sentence labels were taken into consideration. If an application description contained at least one sentence labeled as a permission sentence, then the label of the application description was defined as “1,” else it was defined as “0” (statement sentence).

Using the application descriptions with these labels in the training phase (Figure 4.4.), an evaluation was performed in order to define the ground truth results. Another evaluation was then performed so as to find the effect of the user reviews. For the training phase of the evaluation, the model was trained using the application descriptions (Figure 4.4.). For the test phase, the classification was performed using only application description at first. If the prediction result for the application description was smaller than 0.5, which means no

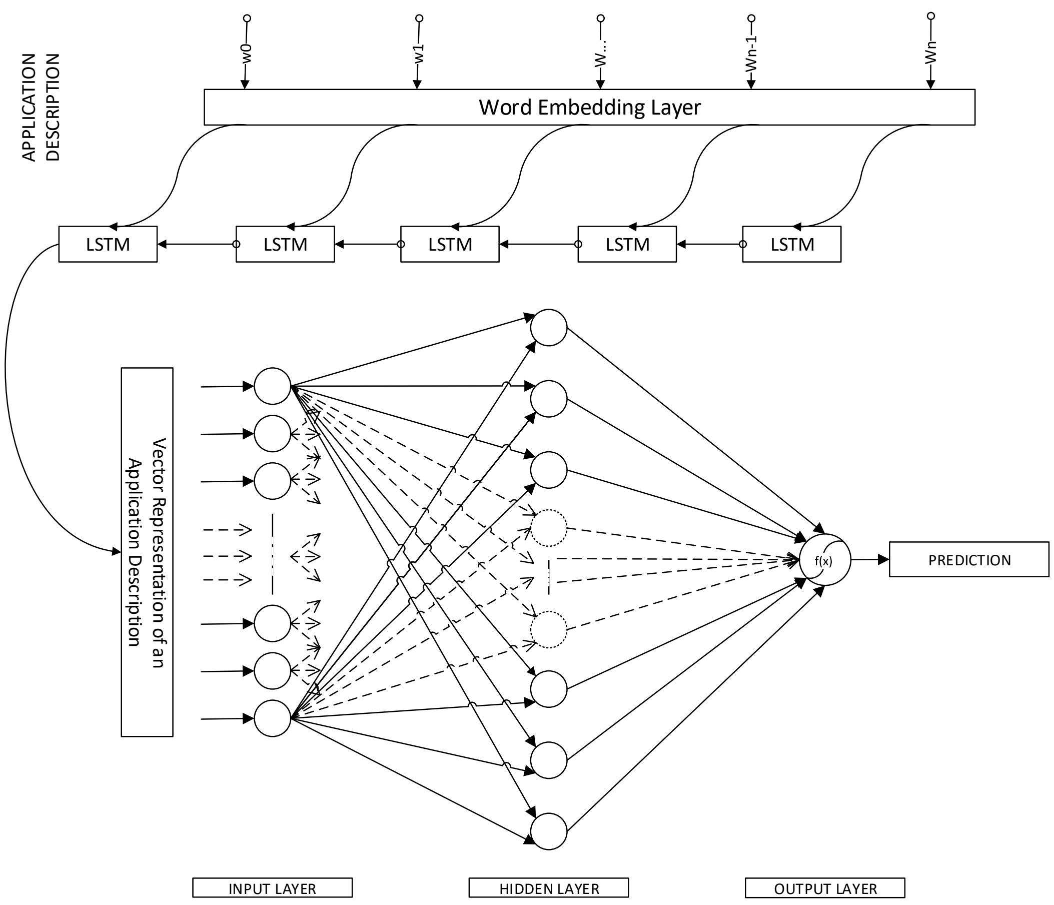


Figure 4.4. Architecture of the description-based LSTM model

expression was found relevant to the permission under test, the user reviews of the application were used as input to the test phase (Figure 4.5.). For this reason, user reviews are represented with a dashed arrow in Figure 4.3.. In such cases, the prediction result of the application was updated with the results achieved using user reviews. Here, the most helpful three user reviews, rated as five stars, were concatenated and then used the as input text.

The architecture of the description-based LSTM model is illustrated in Figure 4.4.. In Equation 4 and Equation 5, application descriptions are used. But, in cases where reviews were used for the test phase, the descriptions were replaced with concatenated user reviews (d_i $->r_i$). For the rest of the section, descriptions are supposed to be thought as reviews where user reviews are used as input. Test phase of the description-based model is illustrated as shown in Figure 4.5..

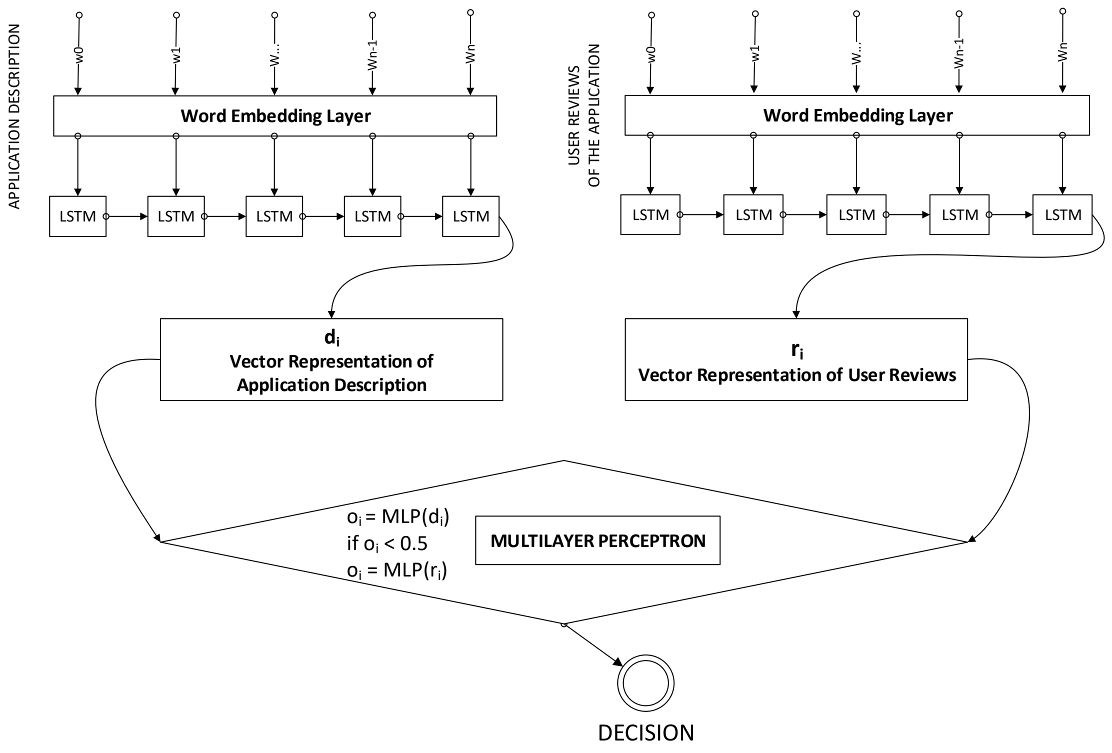


Figure 4.5. Test phase of the description-based LSTM model

For each word within each description, fastText embedding vectors were fetched, starting from the first word, to the last word of the description.

$$d_i = LSTM(x_{1:n}, i) \tag{4}$$

The *i*th application description is $description_i = \{x_1, \cdots, x_n\}$ and the embedding vector for the *j*th word within the description is x_j. The output of LSTM is the compositional representation of the input application description. This vector representation, as in d_i in Equation 4, is then fed into MLP for classification. Prediction result, as in o_i in Equation 5, is the output of the sigmoid activation function as 1 (application description including an indicator of the studied permission), or 0 (application description without permission need).

$$o_i = \text{sigmoid}(MLP(d_i)) \tag{5}$$

4.3. Implementation Details

The inputs for both models are text data and by nature the length of text inputs are variable. Under the circumstances a dynamic framework is needed to implement the neural network and DyNet library [62] is assigned for the task. For representation of words, as explained in Section 2.2., pretrained fastText [30] word embedding vectors are used. The vector dimension size of each word embedding is 300. Therefore the input size of LSTM network is set to 300. The hidden layer size of LSTM network is 128. MLP also has hidden layer size 128. Output of the model is gained from a sigmoid activation function and the prediction result is a value between 0 and 1.

Since the problem is handled as a binary classification problem, binary cross entropy is used as the loss function.

To update network weights in training, Adaptive Moment Estimation (Adam) optimizer [63] is used. Adam is an optimization algorithm introduced in 2015, and it can be used instead of the classical stochastic gradient descent procedure [64]. Some advantages of using Adam are as follows; it is computationally efficient, it has low memory requirement, and it fits to problems with very sparse gradients.

Before training, sentences are shuffled (in description-based model descriptions are shuffled) in order to prevent the model from being effected by the order of sentences (or descriptions) in input data. 10-fold cross-validation is applied for both sentence-based and description-based models.

5. EXPERIMENTS AND RESULTS

In this chapter, the dataset prepared for the study and the datasets used from related studies for comparative purposes are introduced in Section 5.1.. The metrics that are used for the experimental evaluations are defined in Section 5.2.. Finally, Section 5.3. presents the experimental results and a discussion.

5.1. Datasets

Three different datasets were used in order to evaluate the study's proposed models and compare them with the related studies in the literature. Applications in all these datasets were downloaded from the Google Play Store. The DesRe (DEScriptions and REviews of Android applications) dataset was introduced in the current study. Please note that although it is suggested in its name, the dataset does not contain annotated user reviews. However, in the future, user reviews are planned to be included to the data set. In addition to DesRe, other two other datasets from the related studies, namely WHYPER [18] and AC-Net [20] are given in details in this thesis. All of the datasets include mobile application descriptions, and each sentence in the application descriptions were labeled as positive or negative, based on being either a permission sentence or not for each particular permission. In addition, user reviews of applications in the AC-Net dataset were downloaded from the Google Play Store in this thesis. These user reviews were used in order to evaluate the effect of user reviews on assessing the description-to-permission fidelity of applications.

In the remainder of this subsection:

- Datasets from related studies which are used for comparative purposes against the proposed approach are detailed in Section 5.1.1.
- Existing datasets in related works are discussed, and their shortcomings listed in Section 5.1.2.. The section also describes how each of these shortcomings are handled during the preparation of the DesRe dataset.
- The DesRe dataset is introduced and its preparatory phases explained in detail in Section 5.1.3.

5.1.1. Existing Datasets for Assessing Description-to-Permission Fidelity in the Literature

Since the current study uses supervised machine learning techniques in order to find permission indicators among application metadata, existing datasets in this research domain were examined and utilized, and are reviewed in this section. These datasets include description sentences labeled as positive or negative for specific application permissions. The datasets were used for both training and testing purposes in the current study.

The **WHYPER** [18] framework takes the description of an application and evaluates it according to the semantic model formed by the study's researchers, and tries to determine whether or not the description includes a sentence about the application permission. The original study used a total of 581 applications for three permission types, and 9,953 sentences in total. The three permissions in the WHYPER dataset are READ_CONTACTS, READ_CALENDAR, and RECORD_AUDIO.

The 581 mobile applications that form the WHYPER dataset were selected from an application repository that contained 16,000 applications. The repository held the 500 most downloaded free applications from within each application category in January 2012 from the Google Play Store. Of these applications, 200 were randomly selected that included each of the aforementioned three permission types. From these 600 applications, 581 had definitions in the English language and were used to create the WHYPER dataset (Table 5.1.).

Table 5.1. Details of WHYPER dataset

Permission	#Application	#Application+	#Sentence	#Sentence+
READ_CONTACTS	190	107	3,379	235
READ_CALENDER	191	86	2,752	283
RECORD_AUDIO	200	119	3,822	245

#Application Number of applications

#Application+ #Application containing manually labeled permission sentence

#Sentence Number of sentences

#Sentence+ #Sentence manually labeled as permission sentence

Three people (annotators) marked each sentence in the dataset manually by inspecting sentences as to whether or not they contained an expression indicating if one of the three permissions existed. In cases in which at least two of the annotators agreed, the relevant sentences were marked as positive. In cases in which there was only one annotator, the review/marking process was subsequently assessed by all three annotators. The three annotators also marked sentences by performing a keyword-based search according to the keywords listed in Table 5.2..

Table 5.2. Keywords used by WHYPER for the keyword-based search

Permission	**Keywords**
READ_CONTACTS	contact, name, number, email, data
READ_CALENDER	calendar, year, day, event, month, date
RECORD_AUDIO	capture, audio, microphone, voice, record

AC-Net [20] was another study that used labeled application descriptions for the purpose of research experimentation. The study used the description of an application and evaluated it in order to determine whether or not the description included a statement about the permission. The study involved 1,415 popular applications labeled for 11 different permission groups. The application descriptions contained a total of 24,724 sentences. Table 5.3. lists the permission groups with the corresponding number of applications for each group, the number of sentences, and the number of positively labeled sentences. The main difference with the WHYPER dataset is that sentences were labeled for multiple permissions in the AC-Net dataset, rather than only one specific permission.

Table 5.3. Details of AC-Net dataset

Permission Group	**#Application**	**#Sentence**	**#Sentence+**
STORAGE	1,304	23,101	1,338
CONTACT	951	17,353	937
LOCATION	732	12,887	724
CAMERA	406	7,372	522
MICROPHONE	350	6,371	319
SMS	337	6,484	524
CALL_LOG	282	5,457	323
PHONE	280	5,445	199
CALENDAR	197	3,637	289
SETTINGS	369	7,016	560
TASKS	538	10,203	344

#Application Number of applications

#Sentence Number of sentences

#Sentence+ #Sentence manually labeled as permission sentences

Three people (annotators) marked each sentence in the dataset manually by inspecting sentences according to whether or not they contained expressions indicating any of the aforementioned permission groups. In total, 14 annotators took part in the labeling process. In cases in which at least two of the annotators agreed, the relevant sentence were marked as positive. In cases in which there was only one annotation, the review/marking process was subsequently assessed using additional annotators.

5.1.2. Evaluation of Existing Datasets for Assessing Description-to-Permission Fidelity

During the examination of the manually labeled data from the WHYPER and AC-Net datasets, the researcher revealed certain labeling patterns and differences between the labeling approaches of each study.

Similarities that were noted between the sentences labeled as positive for the RECORD_AUDIO permission are listed as follows:

- Expressions meaning any voice interaction between a user and a mobile device (including voice commands, interactive learning, reaction to user voice, karaoke, blowing into the microphone etc.);
- Operations based on a user's voice (speech recognition, using application as a hearing aid etc.);
- Sentences that mean recording every input to the device (e.g., record everything);
- Statements that include creating or saving file types (e.g., MP3, WAV using the user's voice);
- Screen capturing (based on capability to capture audio with the screen);
- Sentences that mean recording calls.

Additionally, on the labeling of the RECORD_AUDIO permission, the following different approaches were noted:

- Expressions that include recording video (record video, video recording etc.) were considered as positive samples for the RECORD_AUDIO permission by the AC-Net dataset. On the other hand, the WHYPER dataset took them into account as negative samples. In such cases, the Android Developer Guide [65] was consulted. The Camera API Guide [66] shows that the RECORD_AUDIO permission is needed in order to implement a video recording application.
- For the RECORD_AUDIO permission, sentences that mean making calls are considered as positive samples by the AC-Net dataset, but as negative samples by the WHYPER dataset. In the Android Developer Guide [65], the development instructions for implementing a calling application [67] do not require the RECORD_AUDIO permission.

Since the performance of the proposed models are each affected by labeling, statements including recording video, recording calls and making calls were labeled separately and differently for possible comparison with the WHYPER or AC-Net datasets within future studies. Expressions that include recording video were labeled as 5, recording call were labeled as 6 and sentences that mean making call were labeled as 7. For evaluation tests, meaning of recording video and recording a call were considered as a requirement for RECORD_AUDIO permission.

The assessment of annotations for READ_CONTACTS permission is presented below:

- Sharing something via e-mail or text message was labeled as statement sentence in AC-Net but in WHYPER those expressions were labeled as permission sentences. In the scope of this study, a sentence about sharing something over e-mail was considered as statement sentence unless there was a specific expression mentioning reading contact data. On the other hand a sentence for sending text messages was considered as permission sentence by this study.
- Expressions about sharing something with friends and family were taken into consideration as statement sentences in AC-Net. On the other hand different labels were encountered for these expressions in WHYPER dataset. These sentences were considered as permission sentences in the scope of this study.
- Sentences about cloud synchronization operations were labeled as permission sentences in AC-Net dataset. In WHYPER dataset, there was not any cloud synchronization expression. The expressions for cloud synchronization of contacts were considered as permission sentences and labeled as 1 in the created dataset.
- Another conflict relates to tagging phrases such as "share ... via social media accounts" as permission sentences for the READ_CONTACTS permission. While both the WHYPER and AC-Net datasets tagged such sentences as permission sentences, these apps were essentially just sending simple data to other apps. As understood from Android's intent mechanism [68, 69], an application which provides its users with a sharing mechanism through the medium of an external application does not require permissions which are already required by the external application. In other words, it is the external application that is responsible for accessing the contacts data of its user. Therefore, these were tagged as statement sentences in the DesRe dataset constructed under the current study. Again, for such cases, the Android Developer Guide [65] was followed.

STORAGE permission group was not included in the WHYPER dataset. Investigating the AC-Net dataset, it was seen that there were 1,304 applications requesting STORAGE permission and there were 23,101 sentences within these applications. 1,338 sentences from

23,101 were labeled as permission sentences, which corresponds to 5.8% of the total number sentences within applications that requested the STORAGE permission.

STORAGE permission group contains two individual permissions which are used in order to access the external storage of a mobile device. These permissions are as follows;

- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE

Instructions residing at the Android Developer Guide [70, 71] illustrates the differences of internal storage and external storage as follows:

- Internal storage is always available, on the other hand external storage can be removed from the device;
- Internal storage is readable only for the application itself, but the files in external storage is available to any application;
- Files in external storage are not removed if the application is uninstalled from the device. On the contrary, files saved in internal storage are removed when the application that saved the files is uninstalled from the device.

To determine whether or not a sentence included a requirement for the STORAGE permission, instructions from Android Developer Guide and the investigation of AC-Net dataset were taken into consideration. In addition to the base rules listed, the following approaches were used:

- Sentences that mean using any external storage device were labeled as positive (SD-card, flash drive etc.).
- Expressions meaning any data synchronization or data backup were considered as a permission requirement.
- Operations based on sharing data via social media (video, photo etc.) were labeled as negative.

- Sentences that mean recording video, audio, and voice were considered as permission sentences.
- Statements that include downloading or saving data to mobile device were considered as permission sentences unless the usage of data was described to be within the application. An expression of saving books for offline reading is a good example of a sentence to be labeled as negative.

Certain mislabeled sentences were also encountered in the existing datasets, with such incorrect markings evaluated as having been applied by human error. Regardless of the labels, similar sentences (which obviously should have been marked as positive) were marked as positive samples in the DesRe dataset (e.g., *"Make it your own deck by adding your voice or your child's!"* or *"record and playback up to 6 tracks, if not sufficient, then drag 'n' drop one track onto another and they will merge into one"*).

There were also some expressions found that were labeled both positive and negative within the AC-Net and WHYPER datasets. Those expressions were also considered as having been marked as such by human error and were therefore excluded (e.g., *"This is a simple voice recorder"* was marked as positive, but *"RecForge is a high quality sound recorder (far better than default sound recorder)"* was marked as negative in the WHYPER dataset, whereas similar sentences were marked as positive in the DesRe dataset of the current study. It should be noted that sentences considered as being incorrectly marked by human error could be readily detected through a simple keyword-based search.

5.1.3. The DesRe Dataset

For the purposes of the current study, for each studied permission, a set of applications with corresponding descriptions were downloaded from the Google Play Store. The selection criteria for the downloaded applications were as follows:

- Applications having been downloaded by users at least 10,000 times;
- Applications with description lengths of at least 500 characters;
- Applications that are "free to download" (no monetary fee).

Since the current study employed supervised machine learning techniques in order to find inconsistencies between the requested permissions and application descriptions, the DesRe dataset was carefully constructed to contain annotated descriptions (description sentences) and permissions. Not only do mobile device users need to understand the functionality of selected permissions through reading app descriptions, but also that these permissions need to be able to gain access to critical resources (dangerous permissions [72]). Therefore, this should be explicitly given within the app description.

In the current study, priority was given to those permissions used in previous studies [18–20, 34] for comparative purposes. Based on these criteria, two permissions in the DesRe dataset were selected among the permissions used by all of the aforementioned previous studies, RECORD_AUDIO and READ_CONTACTS. In total, three permissions were included in the DesRe dataset due to difficulties posed in labeling all dangerous permissions manually. Therefore the last entry in the DesRe dataset was the STORAGE permission group. This group contains two individual permissions, READ_EXTERNAL_STORAGE and WRITE_EXTERNAL_STORAGE, which are used in order to access the external storage of a mobile device. The reason for including STORAGE as a permission group rather than one specific permission was that any application granted WRITE_EXTERNAL_STORAGE permission is also implicitly granted READ_EXTERNAL_STORAGE permission [73]. The STORAGE permission group is one of the most requested permissions in mobile applications [19, 20]. It can also be found among the most comprehended permissions by users [14], and is the most mentioned permission in security-related user reviews [48]; indicating that users are significantly concerned with access to external resources by mobile applications. Therefore, the STORAGE permission group was specifically included in the current study.

For each permission, at least 1,000 applications from various categories were downloaded. After filtering out invalid applications (such as those having non-English sentences in their descriptions), the application descriptions were split into sentences by using the Natural Language Toolkit (NLTK) [74], and were then annotated manually by two people in order to indicate whether or not the requested permission was mentioned in the application description. Table 5.4. presents the details of DesRe dataset and comparison of it with WHYPER and AC-Net datasets.

Table 5.4. Comparison of DesRe dataset with WHYPER and AC-Net datasets

Dataset	**READ_CONTACTS**			**RECORD_AUDIO**			**STORAGE**		
	#App	#S	#S+	#App	#S	#S+	#App	#S	#S+
WHYPER	190	3,379	235	200	3,822	245	-	-	-
AC-Net	951	17,353	937	350	6,371	319	1,304	23,101	1338
DesRe	832	25,011	1,740	1,008	31,989	2,224	801	25,909	764

#App Number of applications

#S Number of sentences

#S+ Number of sentences manually labeled as permission sentences

5.2. Evaluation Metrics

In this section, the metrics used to evaluate the results of the current study, as well as the metrics used to make comparisons with the related studies, are described.

As stated in Chapter 3., WHYPER [18] and AUTOCOG [19] use accuracy, precision, recall and F-Score metrics to evaluate their performance. ACODE [34] uses accuracy and precision values. In order to calculate the values of these metrics, the confusion matrix must be calculated (number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) need to be calculated).

On the other hand, in AC-Net [20] and in this study, area under receiver operating characteristic curve (ROC-AUC) and precision-recall curve (PR-AUC) are used to evaluate the performance of the models.

Accuracy is the ratio of correctly predicted sentences (or descriptions) to the total number of sentences (or descriptions). It is a significantly meaningful metric when the number of false positives and false negatives are found to be close to each other.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Precision is the ratio of the number of correctly predicted positive labels to the total number of positively predicted labels.

$$Precision = \frac{TP}{TP + FP}$$

Recall is the ratio of correctly predicted positive labels to the total number of actual positive labels. This metric can be used as the sensitivity value of the system.

$$Recall = \frac{TP}{TP + FN}$$

F-Score value is the weighted harmonic mean of Precision and Recall.

$$F - Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

Area Under Receiver Operating Characteristic Curve (ROC-AUC) and Precision-Recall Curve (PR-AUC)

There was a significant imbalance found within the classes of both the datasets (AC-Net dataset and also DesRe dataset). For instance, only 1,740 of 25,011 sentences are labeled as permission sentences for the READ_CONTACTS permission in DesRe dataset. The standard accuracy metric is not considered suitable for imbalanced datasets. The cost of errors will not be the same for different classes in such highly imbalanced datasets. Therefore, ROC-AUC and PR-AUC were used as evaluation metrics in the current study, and are known to be useful for domains with imbalanced class distribution. They are also widely used in studies which require evaluation metrics that are insensitive to imbalanced class distribution [75, 76].

To evaluate the success of the model, using scikit-learn library [77] in python, area under precision-recall curve (PR-AUC) and area under receiver operating characteristic curve (ROC-AUC) values are calculated for each experiment. The ROC curve shows the performance of a binary classifier system as its threshold is varied. The curve information is summarized into one number by calculating the area under the ROC curve (Equation 6).

$$ROC - AUC \ = \ \frac{\sum_{i\epsilon positive\_class} rank_i - \frac{N_p*(N_p+1)}{2}}{N_p * N_n} \tag{6}$$

Here N_p and N_n denote the number of positive and negative samples.

The precision-recall curve illustrates the trade-off for different threshold values between precision and recall. When input data is imbalanced, precision-recall metric is a useful metric to determine the success of data classification. In the current study, the average precision score metric is used to summarize precision-recall curve as the weighted mean of precisions achieved at each threshold, by using the increase in recall from the previous threshold as the weight (Equation 7).

$$PR - AUC \ = \ \sum_n (R_n - R_{n-1})P_n \tag{7}$$

where P_n and R_n are the precision and recall values .

5.3. Experiments and Results

Experiments were performed separately for the sentence-based model and for the description-based model.

5.3.1. Performance of the Sentence-Based Model

Experiments for the sentence-based model were performed first using the AC-Net dataset, and then using the DesRe dataset. The results of the two experimentations were then compared. Also, for each studied permission, the sentence-based model was trained and then tested separately.

For each experiment, k-Fold Cross-Validation was applied to the AC-Net and DesRe datasets, taking the value of k = 10. First, the sentences were shuffled. Then, for the 10-Fold Cross-Validation, the dataset was divided into 10 equal groups. Nine groups are used together for the training phase, and the remaining group was used for the testing phase, and the experimental score was calculated. This process was then repeated nine times (10 in total). In each repetition, the data used for the test phase was selected from data not previously used for testing. The overall score of the model was gained by calculating the arithmetic mean of the 10 repetition scores.

AC-Net dataset includes 24,724 description sentences. By feeding 22,251 labeled description sentences into the model, LSTM learns to create the compositional vector representation of a description sentence, and MLP learns to classify it as a permission sentence or a statement sentence. Then, the remainder of the description sentences (2,473 sentences) were used as test sentences and fed into the trained model one by one. With respect to the features to be learned, LSTM creates a vector representation of each test description sentence, and then MLP classifies them as permission sentences or statement sentences. Test results were then calculated for each iteration of the k-Fold Cross-Validation.

The results of the sentence-based model's experiments using the AC-Net dataset are presented in 5.5. and results of related studies are presented for comparative purposes in Table 5.6..

Table 5.5. Performance of sentence-based model on AC-Net dataset

Permission Group	**fastText Embeddings**	
	ROC-AUC	**PR-AUC**
CONTACTS	0.97	0.71
CALENDAR	0.98	0.72
MICROPHONE	0.94	0.39

Table 5.6. Performance of related studies on AC-Net dataset [1]

Permission Group	**ROC-AUC**					**PR-AUC**				
	AC-Net	**Key-Based**	**WHYPER**	**AUTOCOG**	**ACODE**	**AC-Net**	**Key-Based**	**WHYPER**	**AUTOCOG**	**ACODE**
CONTACTS	0.97	0.72	0.6	0.68	0.76	0.74	0.41	0.28	0.41	0.43
CALENDAR	0.99	0.98	0.65	0.89	0.95	0.84	0.71	0.28	0.5	0.71
MICROPHONE	0.96	0.8	0.59	0.79	0.79	0.49	0.44	0.24	0.4	0.51
STORAGE	0.94	0.67	0.56	0.63	-	0.66	0.4	0.33	0.35	-

[1] These results are taken from [20]

When the test results were examined excluding the AC-Net, it was clear that the proposed model was more successful than the previous studies with respect to ROC-AUC metric for all three permissions. It was also observed that the proposed model were better than the previous studies with respect to PR-AUC metric except for MICROPHONE permission. When results of all studies presented in the table and results of the current study were examined, it is seen that two keyword-based studies gave better results for the MICROPHONE permission. The reason behind these results is the keywords used for MICROPHONE permission. The keywords presented in Table 5.2. were considered as powerful discriminators for detecting necessity of the MICROPHONE permission.

When the results were compared with AC-Net, it was seen that ROC-AUC scores were very close, but the results of this study were slightly lower in the PR-AUC scores. In order to investigate the reason behind this difference, the test results for READ_CONTACTS permission were examined in detail. At this stage, READ_CONTACTS permission was selected because the READ_CALENDAR permission was not used in the DesRe dataset and the performance of AC-Net was also lower than the keyword-based studies for the MICROPHONE permission.

In order to evaluate results, sentences are extracted with manually-applied labels and prediction results calculated by the system. After that, False Positives (sentences manually labeled as statement sentences but classified as permission sentences by the model) and False Negatives (sentences manually labeled as permission sentences but classified as statement sentences by the model) were analyzed for the READ_CONTACTS permission.

On the AC-Net dataset, for the READ_CONTACTS permission, there were 23,780 sentences manually marked as statement sentences. The prediction result of the proposed model was < 0.50 for 23,545 of the 23,780 sentences. When the threshold was raised to 0.75, the number of sentences rose to 23,720. The remaining 60 sentences were then examined, and the false positive results with high prediction results are detailed as follows, with sample sentences presented in Table 5.7. and in Table 5.8.:

Table 5.7. Example results for READ_CONTACTS permission with sentence-based model

S#	Sentence	PR	ML
1.	Contact your group to the new boss	0.76	0
2.	Go to the account area and use the Booking Number and PIN from your confirmation	0.84	0
3.	Please contact us at team loves to hear from its users	0.0003	0
4.	If you have questions about your premium account please email us at and we'll respond as quickly as we can	0.001	0
5.	Find apps that can access to your personal information (GPS location read contact data	0.81	0
6.	SMS feature is required to share special photo moments in the kitchen with your contact list	0.91	0
7.	When the synchronization is turned off your contact data is not read or modified	0.87	0
8.	Most of the permissions you see are to deliver content to your device like contact details or dialing a phone call if you choose to	0.85	0
9.	Block numbers from those you dont want to be able to contact you	0.91	0
10.	Blacklist Plus is an easy to use call blocker	0.85	0
11.	Powerful call blocker and blacklist (text SMS blacklist too)	0.78	1
12.	Blocking of anonymous (private) numbers Blocking of unknown numbers	0.8	1
13.	Transform your everyday photos and videos into works of art and share them with your family and friends via social media	0.08	1
14.	The app also allows you to share the scripture with friends via email text message and popular social media sites	0.36	0
15	Register with your Facebook account or your email and join the Poker madness today	0.63	1

[S#] Sentence number.

[CS] Prediction result of the proposed model

[ML] Manual label applied by annotators

Table 5.8. Example results for READ_CONTACTS permission with sentence-based model (continued)

S#	Sentence	PR	ML
16	After installing and registering verify your cell phone number (for security reasons) and your Facebook account	0.73	0
17	On more than million devices around the world people are reading listening to watching and sharing the Bible using the #1 rated Bible Appcompletely free	0.0003	1
18	Clash of Spartan is absolutely free but some games may need to be paid for	0.0007	1
19	This application will deeply analyze your phonebook and bring it close to perfection	0.004	1
20	This component provides core functionality like authentication to your Google services synchronized contacts access to all the latest user privacy settings and higher quality lower-powered location based services	0.08	1

$^{S\#}$ Sentence number.

^{CS} Prediction result of the proposed model

^{ML} Manual label applied by annotators

- There were sentences including the word "contact" or "account" that were defined as permission sentences with a high prediction result (Sentences 1 to 4). Even though it seems words like "contact" and "account" have confounding effects on the results at first, the findings show that they do not have such an effect. The proposed method successfully classifies sentences with prediction results < 0.01 for several sentences, such as those containing expressions of "contact us," "contact of this app," "contact support," "account management," "official account," and "premium account." The reason for this is that the proposed method is able to extract meaning from these expressions based on context since there were several examples in AC-Net dataset to learn from, but insufficient to learn the use of the word "contact" and "account" in the contexts of those which were incorrectly classified.

- Sentences such as Sentence 5 were clearly missed by the proposed method, but which was not an easy sentence to define. However, as far as the researcher is aware, it would not be handled any better by any other previously introduced system.

- When Sentences 6, 7, and 8 were examined (there are also additional examples), even though they were marked manually as statement sentences, the researcher believes that these sentences explicitly specify a requirement for the READ_CONTACTS permission, and should thereby be marked as permission sentences. Similar sentences were also found which had already been labeled as permission sentences in the same dataset. Under such circumstances, these sentences do not appear to be false positives, but further instances of incorrect human annotation. The researcher, therefore, aimed to avoid similar errors in annotation when preparing the DesRe dataset.

- In the evaluation of both false positive and false negative results, the researcher encountered description sentences for call-blocking applications that had been annotated as statement sentences (Sentences 9 and 10). When the AC-Net database was searched, sentences with similar meanings and words were found, but with different labels (e.g., Sentence 11). There were also sentences describing an application to block anonymous numbers, but these had been labeled as permission sentences (e.g., Sentence 12). It is therefore suggested that a certain number of false positives and false negatives may stem from this same scenario.

- Similar situations were encountered in sentences related to sharing on social media. There were sentences labeled differently, even though they stated similar actions and needs. The results of the proposed method seemed to be consistent, yet because of the aforementioned scenario, for the given two example sentences (Sentences 13 and 14), under the proposed method one sentence was under the true negative category and the other under the false negative category.

  The researcher believes that the last two circumstances had confounding effects on the training of the proposed system.

The number of sentences annotated as permission sentence in the AC-Net dataset was 943 for the READ_CONTACTS permission. Using the proposed method, the prediction result was found to be > 0.50 for 560 of these 943 sentences. When the sentences were investigated along with the results, the researcher found that the proposed method predicted result as

< 0.25 for 239 sentences. These 239 sentences can be spilt within four different categories. Two are sentences based on call-blocking applications or for sharing on social media, the effect of which was been previously discussed. Another category found was for sentences connected to account-related applications. In the AC-Net dataset, sentences were annotated for the CONTACTS permission group, rather than only for the READ_CONTACTS permission. The CONTACTS permission group contains the READ_CONTACTS, WRITE_CONTACTS, and also the GET_ACCOUNTS permissions. Several of these sentences were found to be labeled as permission sentences (e.g., Sentence 15), and several as statement sentences (e.g., like Sentence 16). In the current study, the researcher created a dataset that included the READ_CONTACTS permission, and thereby attempted to avoid the confusion between application account data and user contact data during the annotation process. The final category found was also connected to human errors of annotation and inaccurate markings, which are of course possible with any such human-effort-based exercise (e.g., Sentences 17 and 18). However, two example sentences which indicate requiring permission (Sentences 19 and 20) failed to be detected in the experimentation of the proposed method.

The results of the presented analysis on READ_CONTACTS permission showed that the consistency of the dataset has a great effect on the success of the proposed supervised methods. Taking these into consideration, DesRe dataset of annotated descriptions (to be more precise, description sentences) is created with careful attention. To create the dataset, the development instructions from the Android Developer Guide [65] were followed for annotation.

Finally, the proposed method was trained and evaluated on the DesRe dataset that was created and introduced in the current study. The results are presented in Table 5.9..

Table 5.9. Performance of sentence-based model on DesRe dataset

Permission Group	**Fasttext Embeddings**	
	ROC-AUC	**PR-AUC**
READ_CONTACTS	0.98	0.81
RECORD_AUDIO	0.96	0.80
STORAGE	0.97	0.72

In evaluation of binary classifiers that are used for classifying imbalanced datasets, PR curves are more descriptive on reliability of classification performance [78]. The PR curves for

3 permissions, and for each fold within 10-fold Cross Validation were illustrated in Appendix D.

The test results showed a considerable increase in PR-AUC scores. At the same time, it was seen that the ROC-AUC scores increased and got closer to 1. When these results were considered, it was seen that the proposed supervised learning-based model was successful in learning the expressions about permissions and can be used successfully in evaluating the fidelity of an application. The increase in results also illustrated us the importance of consistency in preparing datasets for supervised machine learning models. Sample sentences from the sentence-based model test results are presented in the Appendix E together with the prediction scores for a detailed examination.

5.3.2. Performance of the Description-Based Model

In order to evaluate the effects of mobile device users' reviews, a description-based model was trained and tested on the AC-Net dataset. The description-based LSTM model was trained using application descriptions as input to the LSTM, rather than feeding in the sentences within the description. Therefore, the model was trained using each description as a single document. In order to define the label for each description, sentence labels were investigated. If an application description contained at least one sentence labeled as a permission sentence, then the label of the application description was defined as "1," else it was defined as "0." Using the application descriptions with these labels for the training phase, an evaluation test was performed in order to define ground truth results.

Another evaluation test was performed in order to discover the effects of user reviews on assessing description-to-permission fidelity. In this test, each description was tested as to whether or not the permission under test was stated in the description. If the prediction score for the application description was found to be <0.5, which means no expression was found relevant to the permission under test, the user review data of the application was evaluated. In such a case, the prediction score for the application was updated in accordance with the result from the user reviews assessment. The most helpful three user reviews, which were each rated with five stars, were concatenated and used as input text. The results are presented in Table 5.10..

Table 5.10. Performance of description-based model on AC-Net dataset

Permission Group	Without Reviews		With Reviews	
	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
READ_CONTACTS	0.57	0.41	0.69	0.53
RECORD_AUDIO	0.48	0.15	0.61	0.21
STORAGE	0.65	0.56	0.51	0.47

The results show the positive effects of user reviews on the detection of permission expressions, especially for the READ_CONTACTS and RECORD_AUDIO permissions. However, the performance of the description-based model was found to be considerably lower than for the sentence-based model. This was deemed to be due mostly to the number of training samples, with the number of descriptions considerably less than the total number of sentences within the dataset. This effect was also observed for the STORAGE permission.

Even though using user reviews increased the detection capability of the description-based model for READ_CONTACTS and RECORD_AUDIO permissions, user reviews did not seem to be helpful for STORAGE permission group. The STORAGE permission group was selected as the third permission in this study because it was one of the most requested permissions in applications [19, 20] and also was found to be among the most comprehended permissions by users [14]. Moreover, it was the most mentioned permission in security related user reviews [48]. One reason for the unexpected experiment results for STORAGE permission group can be defined as the small number of permission expression samples within the dataset. The lack of positive samples adversely affects the learning rate of the description-based model. Another important reason is related to the way the user reviews are used. In the proposed model, most helpful 5-starred three user reviews are chosen to be used as input for description-based model. If these reviews are not security or privacy related reviews, than they are not useful for assessing the description-to-permission fidelity. One way to solve this problem is to filter each downloaded user review using the trained model. Then in the case that application description prediction result is < 0.5, only the privacy or security related user reviews will be used, if there is any. Another improvement can be made by using descriptions and user reviews together at the test phase by giving weights to descriptions and user reviews, in cases where the prediction result of description is < 0.5.

6. CONCLUSION

6.1. Conclusion

In this thesis, a novel approach for the description-to-permission fidelity problem was proposed in order to identify inconsistencies between requested mobile application permissions and application metadata by using NLP techniques and RNNs. To this end, a new annotated description dataset called DesRe was created. Therefore, the contribution of this study to the literature is twofold.

Since the proposed model was a supervised machine learning model, the existing datasets and their preparation approaches were investigated. Taking these approaches and Android application implementation instructions into consideration, a new annotated description dataset with 2641 applications for three types of permissions, namely RECORD_AUDIO (1008 apps), READ_CONTACTS (832 apps) and STORAGE (801 apps), was introduced. The newly created dataset called DesRe was shared with the community [1].

The prominent studies assessing the description-to-permission fidelity were analyzed in detail, an extensive evaluation of the proposed model with other state-of-the-art models was given and the results were discussed in the thesis.

In the scope of the study, two different models were introduced; sentence-based model and description-based model. The sentence-based model is able to detect if a given sentence is a permission sentence or a statement sentence. On the other hand, the description-based model was introduced in order to see the effects of user reviews on assessing the description-to-permission fidelity of an application.

The proposed sentence-based approach outperforms the previous studies in the literature, namely WHYPER [18], AUTOCOG [19] and ACODE [34]. On the hand hand, the proposed method shows comparable with the most recent study in the literature, AC-Net [20]. Since both study is based on RNN, these results are expected. However, the proposed approach shows better results (higher ROC-AUC and PR-AUC) on the DesRe dataset than the AC-Net dataset. Because it is believed that the DesRe dataset is more correctly annotated based on the Android Developer Guide than the other datasets in the literature as shown in the thesis.

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

The proposed method is similar to the recent neural model called AC-Net [20], since both utilize RNNs. However, the AC-Net model is based on GRUs, whereas the current study uses LSTMs. Another difference between the AC-Net model and the current study is the inclusion of mobile device users' reviews. Whilst it is possible that some descriptions do not adequately mention the required permissions, it has been shown in this study that it is possible to interpret application descriptions by way of examining user reviews. The results of the current study's experimentations showed that using user reviews could improve the classification capability of models designed to assess the description-to-permission fidelity. To the best of our knowledge, it is the first study that explores the effects of users reviews on this problem. In order to explore the effects of users reviews, user reviews of applications in the AC-Net dataset were downloaded and used in experiments.

The models introduced within this thesis can be used in real life as a supporting tool for permission mechanism. The approach and tool proposed in this thesis can be used in order to warn the users if the application description does not explain the necessity of a requested permission, or to encourage application developers to better express their application functionality for requested permissions and also to prevent the upload of an application which has insufficient description for a requested permission.

6.2. Limitations of the Study

In the scope of the current study, experimentation showed that the results for the description-based model were considerably lower than for the sentence-based model. The main reason for this outcome is deemed to be the number of samples in the dataset that was used. Since the number of description sentences far exceeded the application descriptions, the description-based model could not learn as much in training as the sentence-based model.

Extending the dataset would likely have a positive effect on the description-based model results and even the sentence-based results. Another limitation of the proposed approach relates to the way that the user reviews were used. User reviews were added to the description-based model tests, but only where the classification score of the application description is lower than 0.5. The most helpful three user reviews that were rated with five stars were concatenated and then used as input text. If these user reviews were not privacy- or security-related reviews and more importantly if they do not include any explanation of the usage of permission under the test, then using these reviews could have been adversely affected the results. It was not considered efficient to manually label users' reviews because there were too many, and manually annotating them would require an excessive time. However, user reviews could be annotated automatically using the trained model in the future. Then those annotations could be verified manually in a more efficient way than annotating all reviews manually.

6.3. Future Work

Since the description-based model suffers from inadequate sample numbers from which to learn, extending the dataset to train models could improve the experiment results. In order to improve the model, in future studies, user review data could be annotated using the trained sentence-based model. If a review includes a permission sentence, it could be annotated as a permission review as well. Hence, when the prediction score of an application description is less than a predefined threshold, a set of such user reviews of the application under test that is annotated as permission reviews, could be used for classification.

In the scope of the current study, in order to evaluate the effects of user reviews on detecting indicators of permission needs and to compare these results with AC-Net [20], user reviews of applications in the AC-Net [20] dataset were downloaded. In the future, the DesRe dataset is planned to be extended with annotated user reviews regarding permission usage in applications. Hence, the effects of useful user reviews (permission reviews) could be better evaluated.

A APPENDIX : GLOBAL MARKET SHARE HELD by LEADING MOBILE OSs

In this appendix, Mobile Operating System Market Shares held by the leading mobile OSs from 2009 to 2019 [7] are presented (Table 1.1. and Table 1.2.). A visual representation of this data is in Figure 1.1..

Table 1.1. Mobile operating system market share from 2009 to 2013

Quarter	Android	iOS	Series 40	Symbian	BlackBerry	Samsung	Others
2009-1	1,69	37,45	0	36,94	4,37	0	19,55
2009-2	1,27	38,11	0	38,23	7,13	0	15,26
2009-3	2,27	33,01	0	31,8	8,79	1,43	22,7
2009-4	3,36	31,41	0	36,37	9,54	1,84	17,47
2010-1	5,43	31,92	0	33,91	11,71	1,69	15,34
2010-2	4,44	28,26	0	33,28	14,43	2,78	16,81
2010-3	9,01	24,81	0	32,1	17,13	2,91	14,04
2010-4	12	22,66	0	31,43	18,49	3,48	11,93
2011-1	15,22	24,64	0	30,51	14,52	4,53	10,57
2011-2	17,27	21,72	0	32,23	12,83	5,26	10,69
2011-3	20,13	20,17	0	32,14	11,71	5,81	10,04
2011-4	21,94	23,44	0	31,51	8,53	5,43	9,15
2012-1	23,83	24,47	0	31,22	6,7	5,68	8,1
2012-2	24,23	23,79	9,54	21,89	5,67	6,73	8,14
2012-3	28,01	24,5	14,96	12,75	4,71	6,66	8,4
2012-4	31,75	23,56	14,97	11,1	3,9	6,6	8,12
2013-1	37	26,71	12,69	8,35	3,33	4,83	7,09
2013-2	38,17	25,79	12,9	7,71	3,4	4,62	7,41
2013-3	39,26	23,61	14,24	6,34	3,72	4,68	8,13
2013-4	41,29	21,45	13,73	5,61	3,75	4,76	9,41

Table 1.2. Mobile operating system market share from 2014 to 2019

Quarter	Android	iOS	Series 40	Symbian	BlackBerry	Samsung	Others
2014-1	46,84	23,38	10,91	4,01	2,7	3,89	8,27
2014-2	51,79	23,6	8,69	3,06	1,96	3,1	7,8
2014-3	54,72	24,4	7,17	2,39	1,61	2,43	7,28
2014-4	58,77	24,23	5,72	1,8	1,3	1,77	6,41
2015-1	60,85	22,84	5,15	1,54	1,23	1,44	6,93
2015-2	63,73	20,4	4,43	1,31	1,21	1,12	7,79
2015-3	65,49	19,18	3,39	1,06	1,27	0,91	8,7
2015-4	66,29	18,7	2,95	0,87	1,06	0,84	9,29
2016-1	66,91	19,29	2,47	0,72	0,96	0,74	8,91
2016-2	68,51	19,21	1,99	0,57	0,91	0,68	8,12
2016-3	69,02	19,77	1,59	0,46	0,66	0,64	7,86
2016-4	71,61	18,95	1,06	0,32	0,51	0,52	7,03
2017-1	71,71	19,56	0,85	0,26	0,41	0,45	6,74
2017-2	72,46	19,52	0,67	0,21	0,3	0,35	6,47
2017-3	73,21	19,48	0,53	0,18	0,26	0,33	6
2017-4	73,24	20,08	0,51	0,17	0,2	0,34	5,45
2018-1	74,46	20,18	0,41	0,15	0,15	0,3	4,34
2018-2	76,38	19,04	0,32	0,12	0,11	0,26	3,77
2018-3	76,92	20,17	0,22	0,08	0,08	0,26	2,26
2018-4	74,15	22,85	0,2	0,07	0,06	0,3	2,37
2019-1	74,66	22,83	0,17	0,05	0,05	0,28	1,96
2019-2	75,24	22,75	0,12	0,04	0,05	0,23	1,56

Operating Systems labeled as others : Windows, Nokia, Sony Ericsson, Linux, bada, LG, Tizen, Playstation, Firefox OS, MeeGo, Nintendo 3DS, Nintendo, Brew, webOS and rest of them.

B APPENDIX : DANGEROUS PERMISSIONS and PERMISSION GROUPS

Table 2.1. Dangerous permissions and permission groups [1]

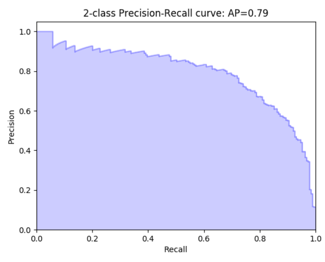
Permission Group	**Permissions**
CALENDAR	READ_CALENDAR, WRITE_CALENDAR
CALL_LOG	READ_CALL_LOG, WRITE_CALL_LOG, PROCESS_OUTGOING_CALLS
CAMERA	CAMERA
CONTACTS	READ_CONTACTS, WRITE_CONTACTS, GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, ACCESS_BACKGROUND_LOCATION, ACCESS_MEDIA_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE, READ_PHONE_NUMBERS, CALL_PHONE, ANSWER_PHONE_CALLS, ACCEPT_HANDOVER, ADD_VOICEMAIL, USE_SIP
SENSORS	BODY_SENSORS, ACTIVITY_RECOGNITION
SMS	SEND_SMS, RECEIVE_SMS, READ_SMS, RECEIVE_WAP_PUSH, RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE, WRITE_EXTERNAL_STORAGE

C APPENDIX : ANDROID VERSIONS and API LEVELS

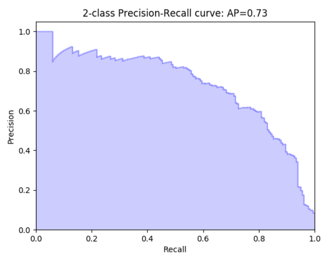
Table 3.1. Android versions and API levels

Name	Version	API Level	Released	Build Version Code
Android 10	10.0	29	Aug 2019	.Q
Pie	9.0	28	Aug 2018	.P
Oreo	8.1	27	Dec 2017	.OMr1
Oreo	8.0	26	Aug 2017	.O
Nougat	7.1	25	Dec 2016	.NMr1
Nougat	7.0	24	Aug 2016	.N
Marshmallow	6.0	23	Aug 2015	.M
Lollipop	5.1	22	Mar 2015	.LollipopMr1
Lollipop	5.0	21	Nov 2014	.Lollipop
Kitkat Watch	4.4W	20	Jun 2014	.KitKatWatch
Kitkat	4.4	19	Oct 2013	.KitKat
Jelly Bean	4.3	18	Jul 2013	.JellyBeanMr2
Jelly Bean	4.2-4.2.2	17	Nov 2012	.JellyBeanMr1
Jelly Bean	4.1-4.1.1	16	Jun 2012	.JellyBean
Ice Cream Sandwich	4.0.3-4.0.4	15	Dec 2011	.IceCreamSandwichMr1
Ice Cream Sandwich	4.0-4.0.2	14	Oct 2011	.IceCreamSandwich
Honeycomb	3.2	13	Jun 2011	.HoneyCombMr2
Honeycomb	3.1.x	12	May 2011	.HoneyCombMr1
Honeycomb	3.0.x	11	Feb 2011	.HoneyComb
Gingerbread	2.3.3-2.3.4	10	Feb 2011	.GingerBreadMr1
Gingerbread	2.3-2.3.2	9	Nov 2010	.GingerBread
Froyo	2.2.x	8	Jun 2010	.Froyo
Eclair	2.1.x	7	Jan 2010	.EclairMr1
Eclair	2.0.1	6	Dec 2009	.Eclair01
Eclair	2.0	5	Nov 2009	.Eclair
Donut	1.6	4	Sep 2009	.Donut
Cupcake	1.5	3	May 2009	.Cupcake
Base	1.1	2	Feb 2009	.Base11
Base	1.0	1	Oct 2008	.Base

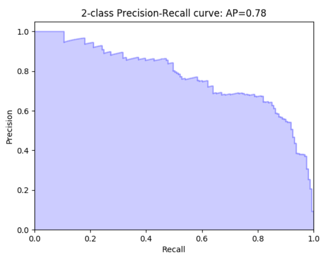
D APPENDIX : PR CURVES of the SENTENCE-BASED MODEL EXPERIMENTS



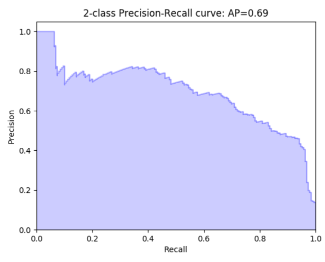
READ_CONTACTS fold 1



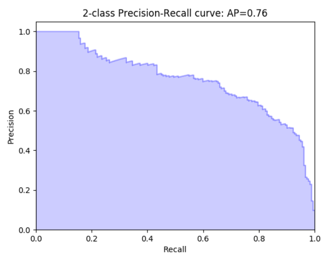
READ_CONTACTS fold 2



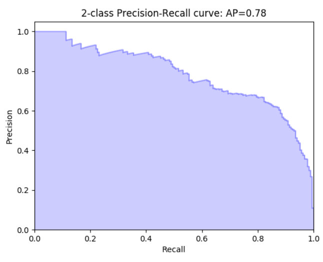
READ_CONTACTS fold 3



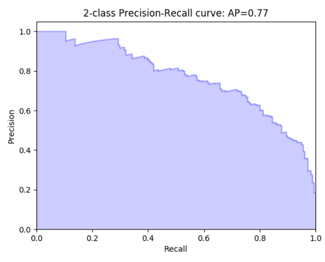
READ_CONTACTS fold 4



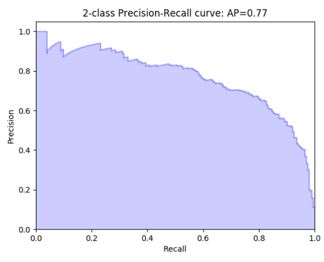
READ_CONTACTS fold 5



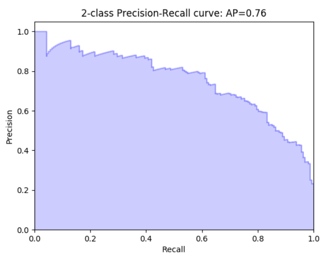
READ_CONTACTS fold 6



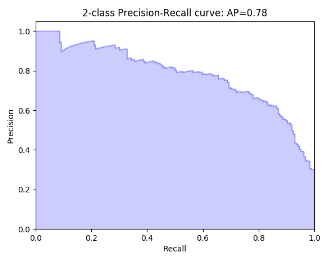
READ_CONTACTS fold 7



READ_CONTACTS fold 8

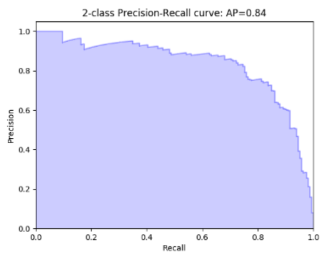


READ_CONTACTS fold 9

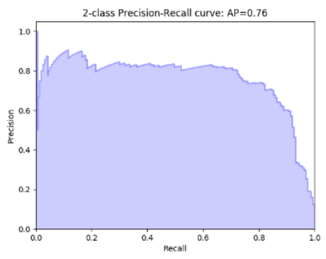


READ_CONTACTS fold 10

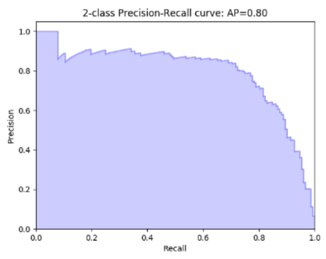
Figure 4.1. PR curves of experiments for READ_CONTACTS permission (DesRe)



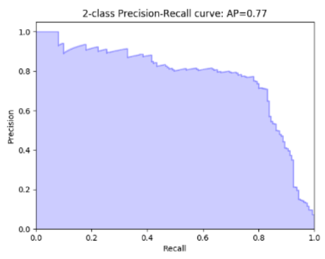
RECORD_AUDIO fold 1



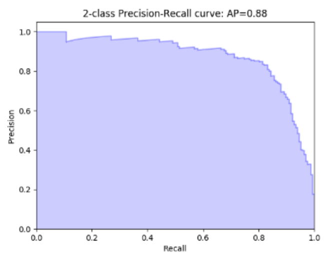
RECORD_AUDIO fold 2



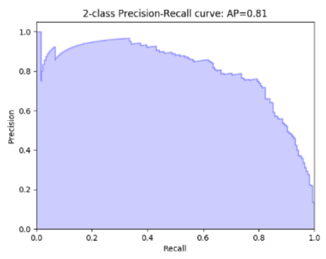
RECORD_AUDIO fold 3



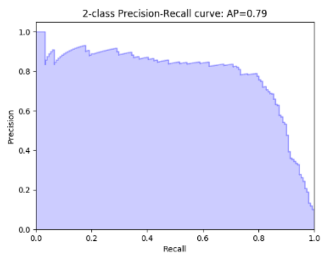
RECORD_AUDIO fold 4



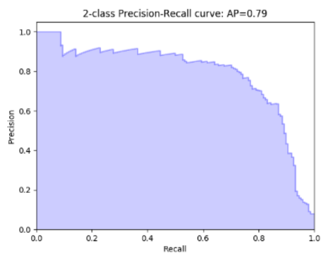
RECORD_AUDIO fold 5



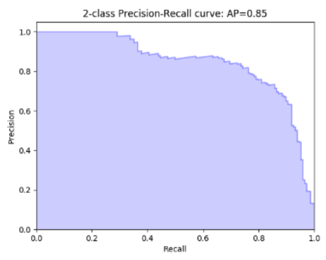
RECORD_AUDIO fold 6



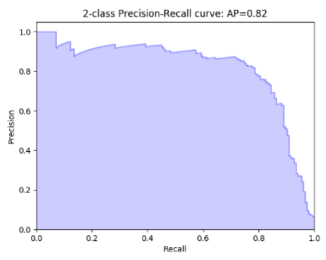
RECORD_AUDIO fold 7



RECORD_AUDIO fold 8

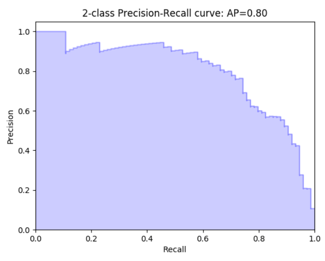


RECORD_AUDIO fold 9

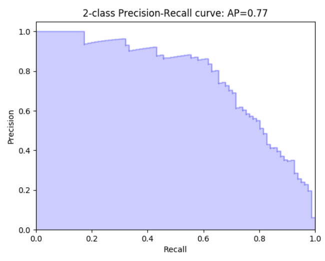


RECORD_AUDIO fold 10

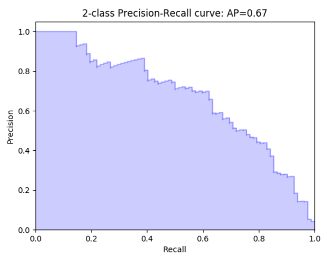
Figure 4.2. PR curves of experiments for RECORD_AUDIO permission (DesRe)



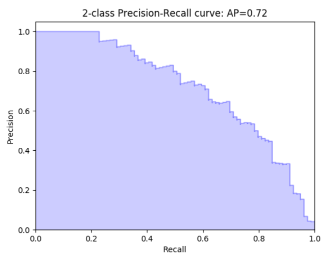
STORAGE fold 1



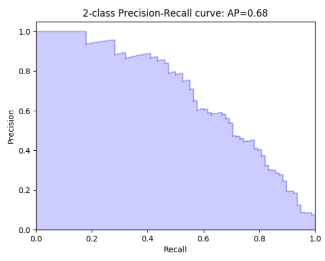
STORAGE fold 2



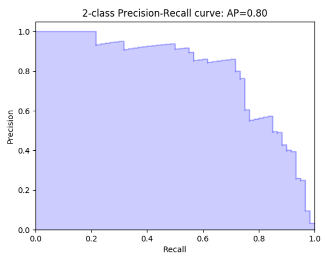
STORAGE fold 3



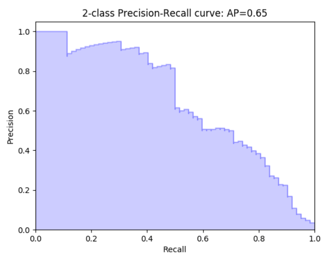
STORAGE fold 4



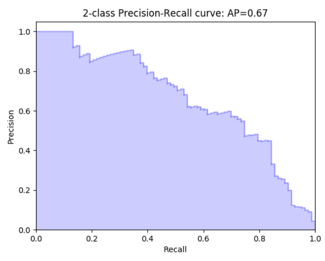
STORAGE fold 5



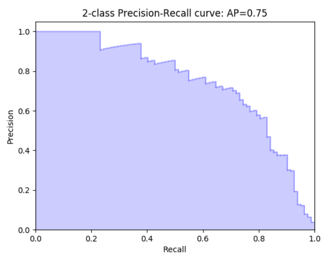
STORAGE fold 6



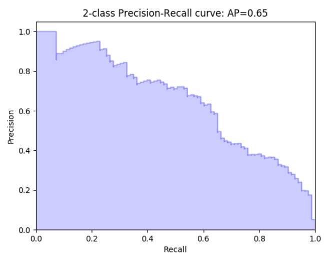
STORAGE fold 7



STORAGE fold 8



STORAGE fold 9



STORAGE fold 10

Figure 4.3. PR curves of experiments for STORAGE permission (DesRe)

E APPENDIX : EXAMPLE RESULTS of the SENTENCE-BASED MODEL

Table 5.1. Top scored true positive and lowest scored true negative results for RECORD_AUDIO permission with sentence-based model

S#	Sentence	PR	ML
1.	Audio Recorder - Voice Recorder app has beautiful and friendly interface as a tape recorder.	0.98	1
2.	The Audio Recorder - Voice Recorder app is easy to use as a tape recorder.	0.98	1
3.	Other names which you can call it: voice recorder, audio recorder, sound recorder.	0.98	1
4.	Record Audio: The app supports recording your own samples.	0.98	1
5.	Screen recorder - Recorder and Video Editor is a free, stable and easy-to-use screen recorder application, the highest quality video for Android devices, allowing you to record video on the phone screen while playing games smoothly and clearly.	0.98	1
6.	Play with him and explore all the fun-filled corners of his house.	0.0000002	0
7.	2nd, multiple number: burner number, disposable number, anonymous free call, anonymous texting	0.0000002	0
8.	Here, you can explore temples in the desert with a chance of finding E-Cores to unlock new skills and weapons.	0.0000003	0
9.	Subscribe to your favorite celebrities, brands, websites, artists, or sports teams to follow their News Feeds from the convenience of your Facebook Lite app!	0.0000003	0
10.	We provide you with the best typing experience and it is for FREE.	0.0000003	0

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.2. Top scored false positive results for RECORD_AUDIO permission with sentence-based model

S#	Sentence	PR	ML
1.	Pro Guitar Tuner is a chromatic tuner that works like an ordinary guitar tuner but right on your Android device.	0.97	0
2.	Screen Recorder is a free screen recorder no root app to record and capture your mobile screen in video formats with or without front camera.	0.96	0
3.	can not record phone calls.	0.95	0
4.	Record –Automatically record test results with a digital logbook that eliminates manual entry.	0.95	0
5.	Record the shooting location information	0.95	0
6.	In Tune by Ear mode - Chromatic Guitar Tuner will play out each note for you so you can tune your instrument.	0.94	0
7.	Pause/resume screen recording	0.93	0
8.	[Voice Talkback] Voice Talkback anywhere, as if you are at home.	0.93	0
9.	2 recordings can be analyzed in comparison	0.93	0
10.	Chromatic tuner	0.93	0
11.	It’s a stable screen video recorder for you to record the tutorials of any app	0.93	0
12.	VideoShow Recorder allows you to record game while playing, capture screen with one touch and edit video with filters, effects, music.	0.92	0

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.3. Lowest scored false negative results for RECORD_AUDIO permission with sentence-based model

S#	Sentence	PR	ML
1.	Add polish to your vocals with studio effects.	0.00004	1
2.	A: Spectroid uses dBFS (Full Scale) where 0 dB is the maximum power that the microphone can measure, so the decibel values are negative because the measured power is less than the maximum power.	0.00006	1
3.	USE YOUR OWN SOUND SAMPLES ===	0.00006	1
4.	the ability to determine the deviation from the base frequency in cents.	0.00007	1
5.	Check your headset and microphone to make sure they are working properly.	0.00011	1
6.	Flipgrid is the leading video discussion platform used by millions of PreK to PhD students, educators, and families around the world.	0.00012	1
7.	Check in on your pets while at the office, keep tabs on the nanny while on vacation, or find out who's been stealing those Amazon packages from your front door – with Foscam App, you can.	0.00013	1
8.	Adopt panda Kiki as your very own virtual pet, talk to him, pet and poke him, feed him, dress him up, garden and even mix the magic color with him!	0.00013	1
9.	FreeConferenceCall.com is an easy-to-use collaboration tool that provides the freedom and flexibility to hold online meetings with HD audio, video conferencing and screen sharing.	0.00013	1
10.	Sound Meter is in the 4th set of the Smart Tools collection.	0.00014	1
11.	USB Microphone support (More info: https://sbaud.io/wavstudio-usb-microphone-support/)	0.00016	1
12.	The recognition speed depends on the speed of internet.	0.00018	1

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.4. Top scored true positive and lowest scored true negative results for READ_CONTACTS permission with sentence-based model

S#	Sentence	PR	ML
1.	In some android devices, contacts deleted from SIM card reappears after phone restart, we are working on to resolve this issue.	0.99	1
2.	Contact Sync - Save your friends ' contacts to your phone.	0.99	1
3.	Duplicate Email (Same email id in all contacts) : Here app merges contacts and make a single contact.	0.99	1
4.	Access to contacts (phone book): App uses contacts/addresses in the phone book for the location search service (e.g.	0.99	1
5.	Duplicate Contacts Manager- Number Based : (Free Feature) Same Contact Number stored in different Names ?	0.99	1
6.	We match you with 2 million drivers that we've screened and trained so that you can relax in knowing that your goods are in safe hands.	0.0000076	0
7.	Write to us- management@goibibo.com	0.0000077	0
8.	If you have any feedback, questions, concerns, leave us a review on Google Play Store or email us at: care@hike.in	0.0000079	0
9.	If you have any suggestions to us or there are any ringtones you want us to provide for you, you can mail to us.	0.0000078	0
10.	Any unused portion of a free trial period, if offered, will be forfeited when the user purchases a subscription to that publication, where applicable	0.0000083	0

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.5. Top scored false positive results for READ_CONTACTS permission with sentence-based model

S#	Sentence	PR	ML
1.	Securely wipe all contacts or a single contact, clipboard, etc.	0.99	0
2.	Search through local and server contacts (Gmail, MSN Hotmail, Outlook and Live) with search suggestions as you type	0.98	0
3.	Manage contact ringtone	0.98	0
4.	For any enquiries, please contact: 24/7 call-center (1240) and in-app live chat, or True iService.	0.97	0
5.	VoIP to native dialer integration (optional) and integration with the native contact list	0.97	0
6.	You get access to the address, contact number, pictures and menu of the restaurant, all at one single place.	0.97	0
7.	Known loved by over 10 million users worldwide, Contacts+ is brought to you by the Contacts Plus Team - awarded 'top developer' on Google Play!	0.97	0
8.	have a chat with Contact centre agent	0.97	0
9.	Get together with 1 or 24 of your friends and family on a HD video call.	0.96	0
10.	And, of course, because they are away from their parents in many of these cases, they need a mobile app like family locator to stay safe and in constant contact with their parents.	0.96	0
11.	Visit ecu.org or contact a member service representative at 800.999.2328 to gain instant access.	0.95	0
12.	Note: when you send any message (even to your Saved messages) your status will be "online" for a short time (ghost always works this way), so it is recommended to set Last activity in privacy settings to "Nobody" or "My contacts".	0.95	0

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.6. Lowest scored false negative results for READ_CONTACTS permission with sentence-based model

S#	Sentence	PR	ML
1.	Personal spam filter will protect you from annoying mailings.	0.00021	1
2.	Supports 6 SIP accounts, up to 6-way audio conference, and 24 virtual BLF keys	0.00025	1
3.	android.permission.READ_CONTACTS	0.00046	1
4.	This application supports integration of up to 6 SIP accounts, 6-way voice conferencing, and allows users to monitor their IP PBX (such as Grandstream's UCM6100 series IP PBX and UCM6510 IP PBX) while utilizing speed dial with up to 24 virtual BLF keys.	0.00055	1
5.	First, users have to create account for which they would like to make credit or debit entries.Accounts can be created using contacts.Users can also create and define category for each account.	0.00069	1
6.	Easily invite players to upcoming games and practices, communicate with your players and coaches, follow your stats : whatever your sport and your practice level.	0.00114	1
7.	Lets track your loved ones and automobiles like cars, bike and van with this amazing letstrack app.	0.00134	1
8.	Instantly check the location of your children/kids, giving you the peace of mind that they are safe and well free of charge!	0.00138	1
9.	Schedule appointments, lunches, gatherings (w/ reminders)	0.00148	1
10.	You can also change the way your mail list appears—simply hit the 'Compact Mail List' tab to get a simplified view.	0.00171	1
11.	Control who sees your posts with easy-to-use privacy options	0.00215	1
12.	Automatic reminders are sent to your customers on their pending payments to help you close unpaid invoices / bills and receipts and ensure payments are made as quickly as possible without you having to spend much time on them.	0.00302	1

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.7. Top scored true positive and lowest scored true negative results for STORAGE permission group with sentence-based model

S#	Sentence	PR	ML
1.	- Connect with Facebook account and sync the saved progress across multiple devices	0,99	1
2.	• SD card storage	0,99	1
3.	• READ_EXTERNAL_STORAGE: Editing photos from SD card	0,99	1
4.	That means you can choose to store your recording files on SD card.	0,99	1
5.	- Send and receive faxes, by accessing photos, email attachments, and cloud storage such as Dropbox and Box.	0,99	1
6.	- You can color with 9 unique customizable painting tools.	0,000029	0
7.	Please contact us at voamobileapps@gmail.com.	0,000031	0
8.	"Beautiful and easy to use."	0,000033	0
9.	weight tracker for controling your weight.	0.000034	0
10.	• Kids alphabet games - Preschool English learning app(Basics) game, for kids, children of age group 2-4 years	0.000049	0

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.8. Top scored false positive results for STORAGE permission group with sentence-based model

S#	Sentence	PR	ML
1.	Export/Import contacts to/from a VCF file.	0,96	0
2.	- Each record file can be renamed, shared, set as ringtone, deleted.	0,96	0
3.	- Save the recording file.	0,96	0
4.	VR 360 Photo Gallery : Store and access your 360 photos	0,95	0
5.	Furthermore, when logged in, user-generated filters will be backed up to the Cloud so they will never be lost.	0,94	0
6.	With voice recorder - audio recorder, it is really helpful to help you to record lectures, interviews, conference agendas, evidences, etc or used to record, practice and edit for the best recording	0,94	0
7.	Save audio recordings to internal memory and share easily if required	0,94	0
8.	You don't need to share your contacts with Yahoo or other websites in order to have a backup.	0,93	0
9.	Screen share photos, web and Google Drive, Dropbox or Box files	0,93	0
10.	Audio recorder has great edit tools for cutting and joining recording files.	0,93	0
11.	Super Call Recorder can automatically record your phone calls in real-time.	0,93	0
12.	• Customizable recordings folder	0,92	0

S# Sentence number

PR Prediction result of the proposed model

ML Manual label applied by annotators

Table 5.9. Lowest scored false negative results for STORAGE permission group with sentence-based model

S#	Sentence	PR	ML
1.	Have fun taking selfies with your friends!	0,00054	1
2.	You can now upload your own songs to play	0,00072	1
3.	Car Dashdroid has built-in music features, allowing you to play your music with a special widget for controls and track details, all without exiting the app (an updated music player will be available soon).	0,00079	1
4.	If you like the program and want to get rid of the ads, get wireless synchronisation and in general do a good thing, please buy the full version.	0,00095	1
5.	Whether you workout, yoga or meditation exercises, or even learning: Set up several stopwatches on your smartphone, store them and use them again and again.	0,00104	1
6.	Share real data to drive real change: Join a growing network of thousands of Contributors around the world who are completing tasks every day to help make policies, products or services better for everyone.	0,00108	1
7.	Powerful: Enjoy group chats up to 100,000 members, share large videos and documents of all type, send hundreds of free emotional stickers!	0,00119	1
8.	Log in to Navitel.Friends/Cloud service using your social network profile (Facebook, Twitter, VKontakte)	0,0016	1
9.	Even if you switch devices, you'll never lose your place.	0,00189	1
10.	- Play on your device or send to your home entertainment system via Chromecast or with Android TV.	0,00195	1
11.	* Pick up where you left off on Android, iOS, or on your web browser	0,00221	1
12.	Easy photo-sharing	0,00239	1

$^{S\#}$ Sentence number

^{PR} Prediction result of the proposed model

^{ML} Manual label applied by annotators

REFERENCES

[1] Dangerous permissions. https://developer.android.com/reference/android/Manifest.permission.html, **(Access date: September, 2019)**.

[2] Simon Kemp. Digital 2019: Global internet use accelerates. https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates, **(Access date: May, 2019)**.

[3] Halil Bayrak. 2019 türkiye İnternet kullanım ve sosyal medya İstatistikleri. https://dijilopedi.com/2019-turkiye-internet-kullanim-ve-sosyal-medya-istatistikleri, **(Access date: May, 2019)**.

[4] Statista. Number of apps available in leading app stores as of 2nd quarter 2019. https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/, **(Access date: August, 2019)**.

[5] StatCounter. Mobile operating system market share worldwide. http://gs.statcounter.com/os-market-share/mobile/worldwide, **(Access date: June, 2019)**.

[6] StatCounter. Mobile operating system market share turkey. http://gs.statcounter.com/os-market-share/mobile/turkey, **(Access date: June, 2019)**.

[7] StatCounter. Global market share held by the leading mobile operating systems. http://gs.statcounter.com/os-market-share/mobile/worldwide/#quarterly-200901-201902, **(Access date: June, 2019)**.

[8] McAfee. Mcafee mobile threat report. https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2019.pdf, **(Access date: August, 2019)**.

[9] JR Raphael. The big secret behind google play protect on android. https://www.computerworld.com/article/3210587/google-play-protect-android.html, **(Access date: May, 2019)**.

[10] JR Raphael. Android market security: An interview with android's vp of engineering. https://www.computerworld.com/article/2472262/

android-market-security--an-interview-with-android-s-vp-of-engineering.html, **(Access date: May, 2019)**.

[11] JR Raphael. Exclusive: Inside android 4.2's powerful new security system. https://www.computerworld.com/article/2473570/exclusive--inside-android-4-2-s-powerful-new-security-system.html, **(Access date: May, 2019)**.

[12] JR Raphael. How google just quietly made your android phone more secure. https://www.computerworld.com/article/2474247/how-google-just-quietly-made-your-android-phone-more-secure.html, **(Access date: May, 2019)**.

[13] Sarah Perez. App submissions on google play now reviewed by staff, will include age-based ratings. https://techcrunch.com/2015/03/17/app-submissions-on-google-play-now-reviewed-by-staff-will-include-age-based-ratings/, **(Access date: May, 2019)**.

[14] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 3:1–3:14. ACM, New York, NY, USA, **2012**.

[15] A. Martín, A. Calleja, H. D. Menéndez, J. Tapiador, and D. Camacho. Adroit: Android malware detection using meta-information. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8. **2016**.

[16] Tao Ban, Takeshi Takahashi, Shanqing Guo, Daisuke Inoue, and Koji Nakao. Integration of multi-modal features for android malware detection using linear svm. In *2016 11th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 141–146. IEEE, **2016**.

[17] Alessandra Gorla, Ilaria Tavecchia, Florian Gross, and Andreas Zeller. Checking app behavior against app descriptions. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 1025–1035. ACM, New York, NY, USA, **2014**.

[18] Rahul Pandita, Xusheng Xiao, Wei Yang, William Enck, and Tao Xie. Whyper: Towards automating risk assessment of mobile applications. In *Proceedings of the 22Nd USENIX Conference on Security*, SEC'13, pages 527–542. USENIX Association, Berkeley, CA, USA, **2013**.

[19] Zhengyang Qu, Vaibhav Rastogi, Xinyi Zhang, Yan Chen, Tiantian Zhu, and Zhong Chen. Autocog: Measuring the description-to-permission fidelity in android applications. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 1354–1365. ACM, New York, NY, USA, **2014**.

[20] Y. Feng, L. Chen, A. Zheng, C. Gao, and Z. Zheng. Ac-net: Assessing the consistency of description and permission in android apps. *IEEE Access*, 7:57829–57842, **2019**.

[21] Build your first android app in java. `https://codelabs.developers.google.com/codelabs/build-your-first-android-app/#0`, **(Access date: Oct, 2019)**.

[22] Build your first android app in kotlin. `https://codelabs.developers.google.com/codelabs/build-your-first-android-app-kotlin/#0`, **(Access date: Oct, 2019)**.

[23] Application fundamentals. `https://developer.android.com/guide/components/fundamentals`, **(Access date: Oct, 2019)**.

[24] Dalvik executable format. `https://source.android.com/devices/tech/dalvik/dex-format`, **(Access date: Oct, 2019)**.

[25] Permissions overview. `https://developer.android.com/guide/topics/permissions/overview`, **(Access date: September, 2019)**.

[26] Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, **1954**. doi:10.1080/00437956.1954.11659520.

[27] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, **1972**.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, **2013**.

[29] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. **2014**.

[30] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 5:135–146, **2017**.

[31] Common crawl. `https://commoncrawl.org/`, **(Access date: September, 2019)**.

[32] Wikipedia. `https://www.wikipedia.org/`, **(Access date: September, 2019)**.

[33] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, **1998**.

[34] Takuya Watanabe, Mitsuaki Akiyama, Tetsuya Sakai, and Tatsuya Mori. 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)*, pages 241–255. USENIX Association, Ottawa, **2015**.

[35] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, **1995**. ISSN 0001-0782. doi:10.1145/219717.219748.

[36] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. European Language Resources Association (ELRA), Genoa, Italy, **2006**.

[37] Marie-Catherine de Marneffe and Christopher D. Manning. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, CrossParser '08, pages 1–8. Association for Computational Linguistics, Stroudsburg, PA, USA, **2008**.

[38] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1606–1611. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **2007**.

[39] Evgeniy Gabrilovich and Shaul Markovitch. Wikipedia-based semantic interpretation for natural language processing. *J. Artif. Int. Res.*, 34(1):443–498, **2009**.

[40] Android asset packaging tool. `https://developer.android.com/studio/command-line/aapt2`, **(Access date: September, 2019)**.

[41] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. Pscout: Analyzing the android permission specification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 217–228. ACM, New York, NY, USA, **2012**.

[42] A tool for reverse engineering android apk files. `https://ibotpeaches.github.io/Apktool/`, **(Access date: September, 2019)**.

[43] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, **2014**.

[44] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. **2018**.

[45] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, **2003**.

[46] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, **1967**.

[47] Deguang Kong, Lei Cen, and Hongxia Jin. Autoreb: Automatically understanding the review-to-behavior fidelity in android applications. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 530–541. ACM, New York, NY, USA, **2015**.

[48] Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. Short text, large effect: Measuring the impact of user reviews on android app security privacy. **2019**.

[49] J. Wu, M. Yang, and T. Luo. Pacs: Pemission abuse checking system for android applictions based on review mining. In *2017 IEEE Conference on Dependable and Secure Computing*, pages 251–258. **2017**.

[50] Marti A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, **1998**.

[51] Hannu Toivonen. *Apriori Algorithm*, pages 60–60. Springer US, Boston, MA, **2017**.

[52] L. Yu, T. Zhang, X. Luo, L. Xue, and H. Chang. Toward automatically generating privacy policy for android apps. *IEEE Transactions on Information Forensics and Security*, 12(4):865–880, **2017**.

[53] Mu Zhang, Yue Duan, Qian Feng, and Heng Yin. Towards automatic generation of security-centric descriptions for android apps. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 518–529. ACM, New York, NY, USA, **2015**.

[54] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of android malware in your pocket. **2014**.

[55] Tingmin Wu, Lihong Tang, Zhiyu Xu, Sheng Wen, Cécile Paris, Surya Nepal, Marthie Grobler, and Yang Xiang. Catering to your concerns: Automatic generation of personalised security-centric descriptions for android apps. *CoRR*, abs/1805.07070, **2018**.

[56] Oliver P. John and Sanjay Srivastava. The big-five trait taxonomy: History, measurement, and theoretical perspectives. **1999**.

[57] Xinli Yang, David Lo, Li Li, Xin Xia, Tegawendé F. Bissyandé, and Jacques Klein. Characterizing malicious android apps by mining topic-specific data flow signatures. *Information and Software Technology*, 90:27 – 39, **2017**.

[58] JL Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, **1990**.

[59] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, **1997**.

[60] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, **1997**.

[61] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. **2018**.

[62] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Manish Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *ArXiv*, abs/1701.03980, **2017**.

[63] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, **2014**.

[64] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, **1991**.

[65] Android developer guide. `https://developer.android.com/`, **(Access date: May, 2019)**.

[66] Camera api. `https://developer.android.com/guide/topics/media/camera.html`, **(Access date: May, 2019)**.

[67] Build calling application. `https://developer.android.com/guide/topics/connectivity/telecom/selfManaged`, **(Access date: May, 2019)**.

[68] Intent — android developers. `https://developer.android.com/reference/android/content/Intent.html`, **(Access date: May, 2019)**.

[69] Sending simple data to other apps — android developers. `https://developer.android.com/training/sharing/send.html`, **(Access date: May, 2019)**.

[70] Data and file storage overview. `https://developer.android.com/guide/topics/data/data-storage#filesExternal`, **(Access date: Oct, 2019)**.

[71] Save files on device storage. `https://developer.android.com/training/data-storage/files.html#WriteExternalStorage`, **(Access date: Oct, 2019)**.

[72] Dangerous permissions. `https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions`, **(Access date: September, 2019)**.

[73] Read external storage permission. `https://developer.android.com/reference/android/Manifest.permission#READ_EXTERNAL_STORAGE`, **(Access date: September, 2019)**.

[74] Natural language toolkit. `https://www.nltk.org/`, **(Access date: September, 2019)**.

[75] Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. 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, **2019**.

[76] Nitesh V. Chawla. *Data Mining for Imbalanced Datasets: An Overview*, pages 853–867. Springer US, Boston, MA, **2005**.

[77] Scikit-learn machine learning in python. `https://scikit-learn.org/stable/`, **(Access date: October, 2019)**.

[78] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):1–21, **2015**.