Android Open Source License Detection Using Static Code Analysis

Gyoosik Kim*, and Dong-Myung Shin*

1, 2LSare Inc, Republic of Korea
(E-mail: leo@lsware.com, roland@lsware.com)

ABSTRACT Open source refers to software that allows anyone to view and use source code without any particular restrictions. A user of open source can be used for secondary creative and commercial purposes within a given license. OSS is especially prevalent in mobile environments, with an estimated 90 percent of Android apps and half of iOS apps utilizing OSS. We purpose to identify and detect OSS applications for Android platforms. We extract unique information from Android applications. Moreover, use this extracted information to measure similarities among Android apps and others, to determine compliance with open source software licenses.

Keywords: Open Source Software (OSS), Android, OSS License, API Call Frequency

1. INTRODUCTION

Open source software (OSS) refers to software that allows anyone to view and use source code without any particular restrictions. A user of Open Source can be used for secondary creative and commercial purposes within a given license. OSS is especially prevalent in mobile environments, with an estimated 90 percent of Android apps and half of iOS apps utilizing OSS. However, while OSS is widely used, OSS license violation cases are also increasing. We purpose method of OSS, to identify and detect OSS applications for Android platforms. To do this, we modify static analysis tool of AOSP (Android Open Source Projects), dexdump, to extract unique information from Android app. Use this extracted information to measure similarities among Android apps and others, to determine compliance with open source software licenses.

2. PROPOSAL METHOD

2.1. Static Analysis of Android Application

Android application is a packaged file used to distribute Android's software and has an Android Application Package (apk) as an extension. Classes.dex(file) in the apk, a component of the Android application, is a bytecode that runs on the Dalvik Virtual Machine (DVM), which contained in Android OS[3]. The classes.dex can be reverse engineered using reverse engineering tools such as baksmali, Androguard, and dexdump. These tools convert the dalvik bytecode into a smali code, which is the intermediate form of Java source code and dalvik bytecode. We modified the dexdump tool and extracted unique feature from the app by using this premodified dexdump. We use API call frequency as unique
There are two methods of measuring API call frequency: Dynamic and Static. The dynamic method is executing the app and measure how many times the API is invoked. On the other hand, the static method is counting the instruction, which is related to API called, in code area(section). Our technique statically counts how many times API call in code area and uses it as its unique feature information.

### 2.2. Measuring Android API Call Frequency

The opcode of dalvik bytecode has a total of 252 opcodes ranging from 0x00 to 0xff. There are 26 formats according to the combination of opcode and operand. The opcode number from 0x6e to 0x72 are commands to call a function and these opcode's format is 35c. Check the operand of the opcode to measure the function call frequency. To do this, we modify static analysis tool of AOSP(Android Open Source Project), dexdump, to extract unique information from Android applications. We should filter only the API among these measured functions call frequency. Many other studies[1,2], used the java reflection technique to analyze the Android.jar which used to develop Android application, to signify how many java API exist on Android. We also analyzed the android.jar using java reflection technique. There are a total of 133,290 APIs in Android API version 25, which are filtered to measure API call frequency.

### 2.3. Feature Selection

Since Android has lots of APIs, so we use a feature selection, which is one of the data reduction methods widely used in machine learning, to reduce the dimension. The feature selection algorithm uses the filter method algorithm. Filter methods select feature(API) regardless of the model. They are based only on general features like the correlation with the variable to predict. Filter methods suppress the least interesting features(API). The other features(API) will be part of a classification or a regression model used to classify or to predict data. These methods are particularly useful in computation time and robust to overfitting.

### 3. CONCLUSION AND FUTURE WORK

Measure the similarities between Android apps using the features(API) selected in section 2.3. The algorithms for measuring similarity are manhattan distance, TF-IDF cosine similarity, and Euclidean distance. We will use these algorithms to measure and test the similarity of Android apps. This study can help ensure compliance with licenses in the mobile environment where open sources are most widely used.

### ACKNOWLEDGMENT

This research project was supported by the Ministry of Culture, Sports, and Tourism (MCSTT) and by the Korea Copyright Commission in 2018.

### REFERENCES