isrjournals

International Journal of Advanced Research in Computer Science Engineering and Information Technology

Volume: 6, Issue: 3,Special Issue: 1 ,Mar,2021 ,ISSN_NO: 2321-3337

Boosting app clone detection efficiency through robust approach neural network application

Mrs. A. Sunitha M.E., Professor of Computer Science Department,
Miss. R. Aishwarya, Student of Computer Science Department,
Miss. A. Catherine, Student of Computer Science Department,
St. Joseph College of Engineering, Sriperumbudur, Chennai

Abstract

The scope of this project is to detect cloned Android app pairs but not to identify which is the original one and which is the cloned one. We only focus on the apps whose user interface features are defined in layout XML files. Apps with no or very few layout files, such as web apps, games based on third-party engines, and background apps with only services, are out of the scope of our paper. Just like previous static detection systems, apps whose user interface is dynamically defined by programs are also out of the scope of our paper. All of the apps used in our evaluation are not paid apps, so we can crawl enough apps to simulate the real-world scenario.

The description of an app on the third-party market may be missing or not match the apps implementation. Although the consistent description is not essential in our detection system, it can greatly improve our detection speed. Hence, apps with inconsistent description will increase the time complexity of our method. Our paper does not concern these apps, because (a) this kind of app arouses the vigilance of users very easily and (b) the inspection by the app market is becoming more and more strict.

Key Terms: ANN – Artificial Neural Network, CFG – Control Flow Graph, SDK – Software Development Kit.

1. Introduction

It can detect full and partial level similarity between applications. We evaluate DroidCC clone detection approach on real time data-set and count the Recall and Precision, which is quite significant. Furthermore, our results show that our approach is very efficient and effective in detecting different types of clones to check the similarity level in Android applications. Mobile apps have seen widespread adoption in recent year, with over 2.8 million apps in Google Play and billions of downloads.

2. Literature Survey

Guoai xu and Chengpeng zhang proposed the developing-related characteristics of developers have dropped much after compiling from source code (e.g., number of spaces, tabs, use of upper or lower case for variable names, which were widely used in source code level authorship attribution). For the array length related features, we count the average, median and standard deviation of the length of the arrays that used in each app. In addition, we also count the ratio of arrays with constant length used in each app.

Hui liu and yanjie jiang proposed that for each of the methods from subject applications (noted as m), we select a small set of JDK methods that are both syntactically matching and lexically similar to m. The selection of candidate JDK methods is critical for the performance (time complexity) of the proposed approach. JDK is composed of thousands of methods. Consequently, it would be time and resource consuming (if not impossible) to compare m against all of these methods. We extract methods from the subject applications that may be replaced with some API methods. In this paper, we focus on the detection of methods that are independent of instance variables.

Seong-je cho, and minkyu park proposed that a malicious attacker might apply code obfuscation to avoid app clone detection. Therefore, it is necessary to consider the effects of code obfuscation when detecting cloned apps. In this paper, we design and implement a tool called RomaDroid, which can detect efficiently cloned apps based on features inherent in each app's AndroidManifest.xml file.

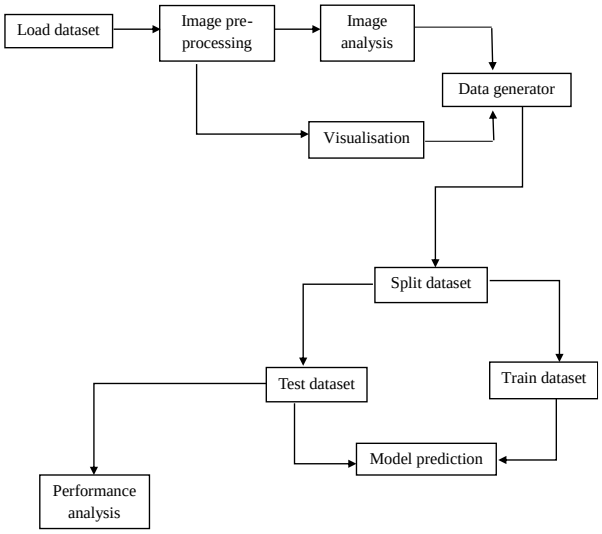
Jonathan and Crussell Clint proposed a full app similarity detection, A Darwin represents each app as a set of features. In the simplest case, two very similar apps will have mostly or completely overlapping feature sets. Dissimilar apps' feature sets, on the other hand, should have little to no overlap. This is captured in the Jaccard Similarity of their two feature sets FA and FB, which reduces the problem of finding similar app to that of finding similar sets.

Volume: 6, Issue: 3,Special Issue: 1 ,Mar,2021 ,ISSN_NO: 2321-3337

3.System Design

The detection process can be divided into four stages. Firstly, we modify the declarations of activities in AndroidMani fest.xml generated by decompiling the APK file, thus each activity of the modified app can be started by "AM" command. Subsequently we start all of the activities and extract corresponding UI information.

. ARCHITECTURE



Exploratory data analysis is generally cross-classified in two ways. First, each method is either non-graphical or graphical. And second, each method is either univariate or multivariate (usually just bivariate). Non-graphical methods generally involve calculation of summary statistics, while graphical methods obviously summarize the data in a diagrammatic or pictorial way.

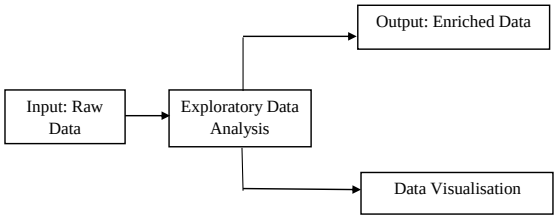
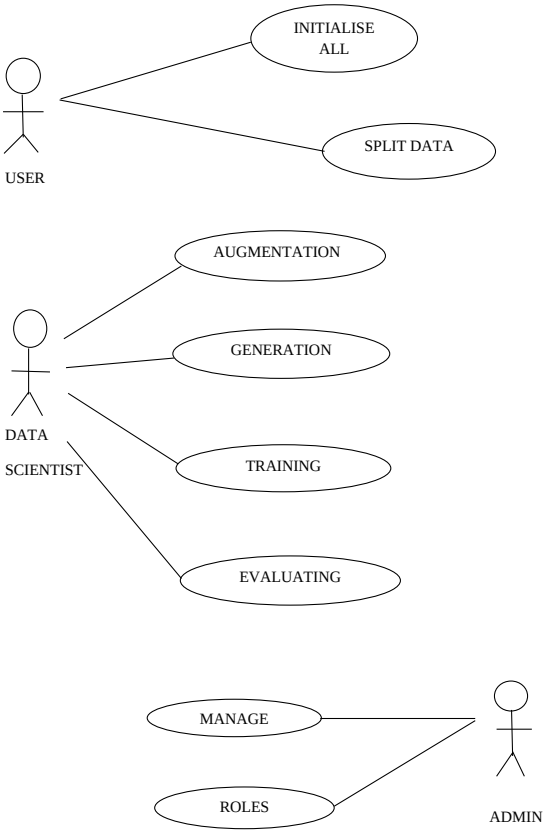
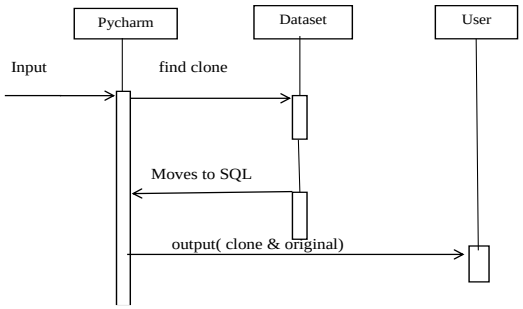


Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.



Volume: 6, Issue: 3,Special Issue: 1 ,Mar,2021 ,ISSN_NO: 2321-3337

To minimize indexing info and use less storage, two very important parameters was considered. The fixed size of hash value and the size of window.



The image can be scaled outward or inward. While scaling outward, the final image size will be larger than the original image size. Most image frameworks cut out a section from the new image, with size equal to the original image.

We extract all method invocation features from each component class that declares intent filters and construct a birthmark. provides a high embedding capacity, good imperceptibility, and strong robustness against all geometric attacks (i.e., rotation, scaling, cropping, and cutting), non-geometric attacks (i.e., Gaussian filtering, sharpness, salt and pepper noise, median filtering, JPEG compression).

3. Implementation

It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations. In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

self.input_shape = input_shape

self.output_size = output_size

def predict_greedy (self, model, input_img, require_sparse_label = True, sequence_length = 1)

current_context=Utils.Sparsify(current_context, self. output_size)

probas=model.predict(input_img,np.array([current_context]))

prediction = np. Argmax(probas)

trained_weights_path="static/model"format for payload:

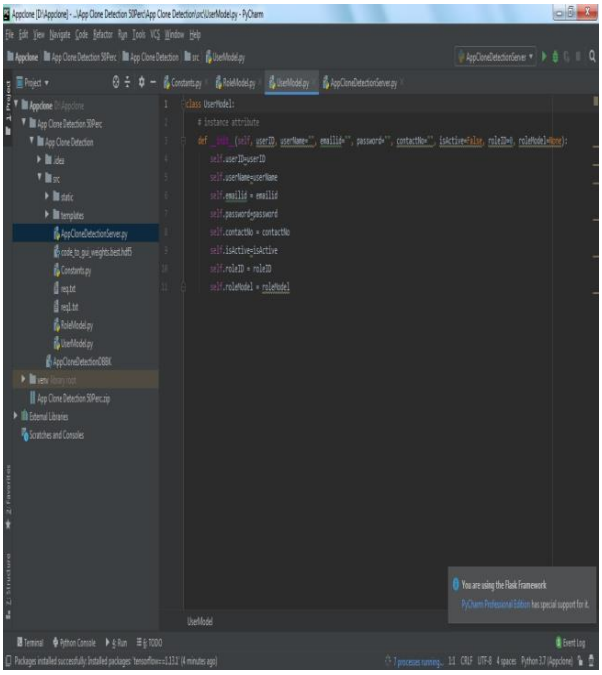
A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as

isrjournals

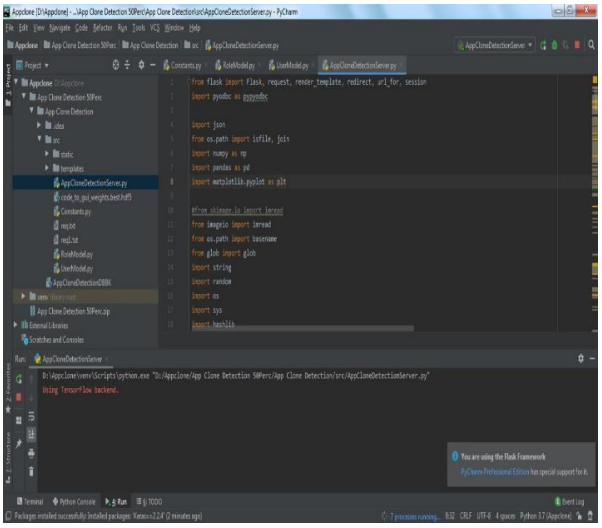
International Journal of Advanced Research in Computer Science Engineering and Information Technology

Volume: 6, Issue: 3,Special Issue: 1 ,Mar,2021 ,ISSN_NO: 2321-3337

C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line.



To minimize indexing info and use less storage, two very important parameters was considered. The fixed size of hash value and the size of window.



One key thing to note about this operation is that image dimensions may not be preserved after rotation. If your image is a square, rotating it at right angles will preserve the image size. If it's a rectangle, rotating it by 180 degrees would preserve the size. Rotating the image by finer angles will also change the final image size. Below are examples of square images rotated at right angles. Feature extraction techniques are helpful in various image processing applications e.g., character recognition. As features define the behaviour of an image, they show its place in terms of storage taken, efficiency in classification and obviously in time consumption also.

Conclusion and Future Enhancement

In this paper, we propose a novel app cloning/repackaging detection tool. For similarity measurement, it explores the method signature of an app like the existing detection tools. The unique feature of our proposal is that it considers only the component classes that receive implicit intents and extracts the method signature related to the classes at a small code. The evaluation results show that this application can effectively detect app clones among the packed apps, without compromising the legitimate app hardening service.

We plan to trace the clones back to the original diagrams and visualize them in the model. We believe that in future, our approach can relatively easily be extended to other kinds of UML and behavioural model representations.

References

1. M. Sun, M. Li, and J. C.S. Lui, “DroidEagle: Seamless detection of visually similar Android apps,” in Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, 2015, pp. 9

2. L. Li, T. F. Bissyandé, and J. Klein, “SimiDroid: Identifying and explaining similarities in Android apps,” in IEEE Trustcom/BigDataSE/ICESS, 2017, pp. 136-143.

3. H. Wang, Y. Guo, Z. Ma, and X. Chen, "WuKong: a scalable and accurate two-phase approach to Android app clone detection," in Proceedings of the 2015 International Symposium on Software Testing and Analysis, ACM, 2015, pp. 71-82

4. Yang W, Zhang Y, Li J, et al. AppSpear: Bytecode Decrypting and DEX Reassembling for Packed Android Malware[M]//Research in Attacks, Intrusions, and Defences. Springer International Publishing, 2015: 359- 381.