Android Malware Classification through Authorship Attribution Using FastText

Arman Kabiri, Samin Fakharian

OUTLINE

I

BACKGROUND

BACKGROUND

- Android
  - A mobile operating system based on a modified version of the Linux kernel
  - Designed primarily for touchscreen mobile devices such as smartphones and tablets.
  - One of the most popular operating systems used by billions of users.
- Malware
  - The popularity of Android has attracted attackers and hackers attention to itself as a good target for developing malicious applications.This unprecedented increase is often associated with the inherent openness of the Android platform, and the corresponding lack of security measures to prevent potential abuse.

BACKGROUND (CONT.)

Android Malware Detection

- Generic methods
  - RiskRanker
    - scalably analyze whether a particular app exhibits dangerous behavior
  - DroidRanger
    - retrieve, associate and reveal malicious logics at the “opcode level”
- Multi-feature analysis
  - Providing insight into a malware sample
- Large-scale binary classification
  - Identifying malicious and benign samples during the triage stage

BACKGROUND (CONT.)

Android Malware Detection

- Textual analysis using natural language processing (NLP) techniques
  - Statistical machine learning techniques
    - Classifying android applications using count-based features
    - Works on the signature of the widely recorded malware.
  - Forming signature for the application's writer
    - Base on their style of writing
      - Detection of previously seen, as well as new malware generated by particular authors.

BACKGROUND (CONT.)

- Word embeddings
  - Continuous representations of words in a high-dimensional vector space
  - Trained on a huge corpus of text using machine learning methods
  - Capturing semantic similarities between words

Word: function

0.13	-0.42	0.02	-0.21	0.31	-0.72	-0.22	…	0.38	0.01	0.18	-0.53	0.19	0.57
1	2	3	4	5	6	7		295	296	297	298	299	300

- Popular word embeddings:
  - Word2Vec
  - Glove
  - FastText

2

RELATED WORKS

RELATED WORKS

- Portable malware detection using supervised machine learning techniques (Karbab & Debbabi, 2019)
    - Model the behavioral reports into sequence of words to detect and attribute malware using bag-of-words and also build ML ensembles on top of bag-of-words features.
- Android malware analysis and detection technology based on Attention-CNN-LSTM (Killam et al., 2016)
    - Using open source malware datasets to extract texture fingerprint information of Android malware to show how similar the binary file blocks are in the AndroidManifest.xml which is treated as a text document.

RELATED WORKS (CONT.)

- Novel detection method based on deep learning (Xiao, Zhang, Mercaldo, Hu, and Sangaiah, 2019)
  - Considering there is some semantic information in system call sequences as the natural language and treating one system call sequence as a sentence in the language and using Long Short-Term Memory they construct a classifier based on similarity score between classes.
- Android malware detection using author attribution (Kalgutkar et al., 2018)
  - Extracting string literals from APK files of Android application and utilizing bag-of-words to represent strings, they fit a classifier using SVM algorithm to classify android applications to their corresponding developers.

3

METHODOLOGY

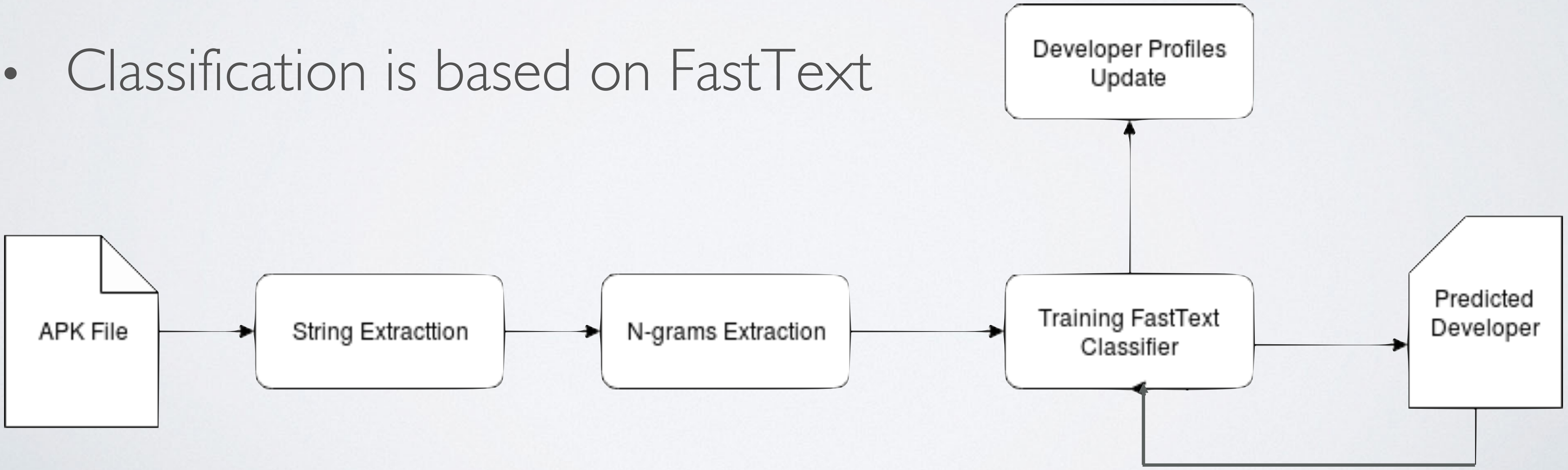
HYPOTHESIS

The recognition of the application developers could lead to the detection of the malicious applications.

HOW?

METHODOLOGY

- Approach: Natural Language Processing Techniques
- Exploiting the coding style of the developers
- Classifying android applications to the developers
- Classification is based on FastText



STRING EXTRACTION

- Android application package (APK)
- Strings representing developers' coding style:

  1. DEX file

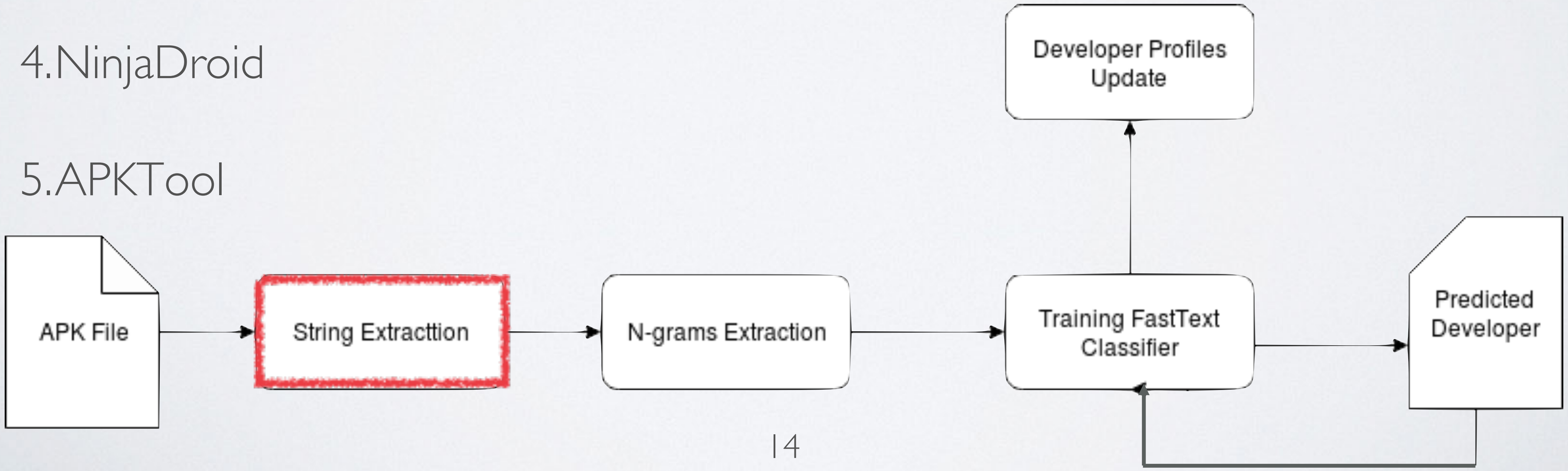
  2. Layout files

  3. Application String (String.xml file)

- Tools used for string extraction:

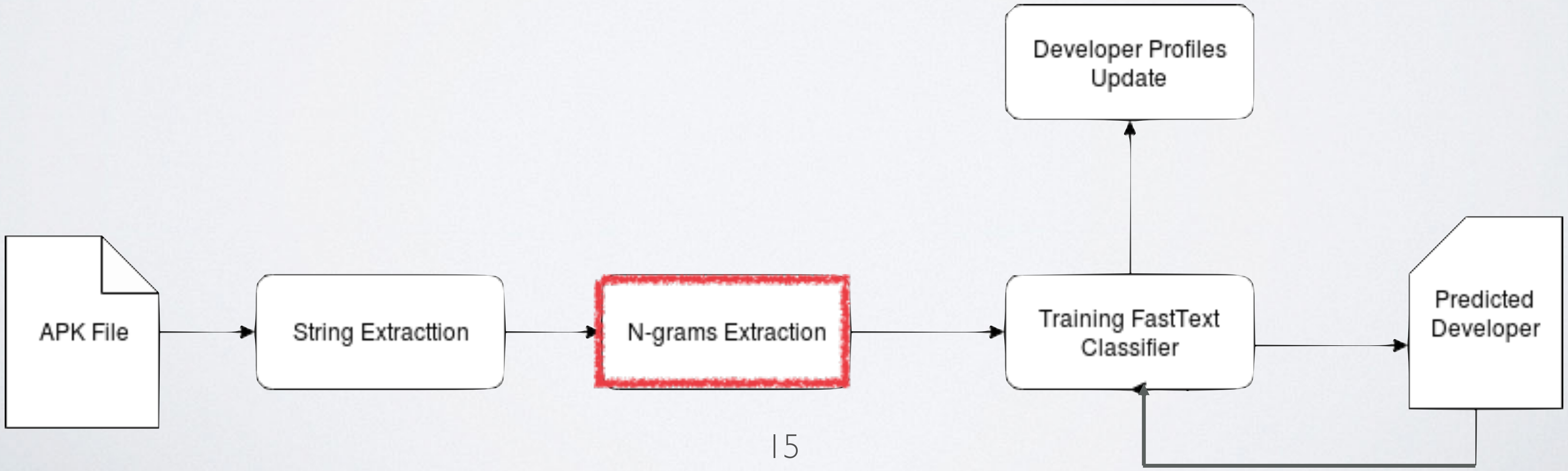
  4.NinjaDroid

  5.APKTool



MODEL ARCHITECTURE

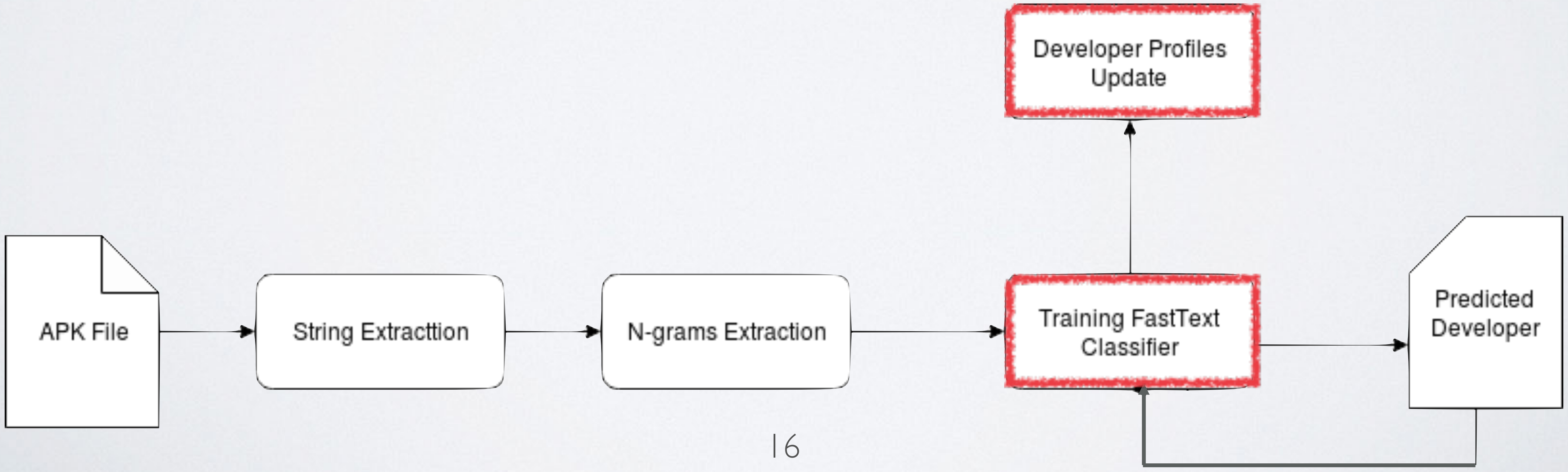
- Strings used in each APK comes from the previous component
- Producing unique N-grams (1, 2, 3, ..) features from strings
- Example:
    - n_list_of_users {n, list, of, users, list_of_users, …}



TRAINING CLASSIFIER

- FastText:

1. Each feature (String used in an APK) is given a random embedding.
2. The embeddings are averaged to represent the APK
3. The computed embedding is fed into a linear classifier
4. Softmax function is applied to on the output to predict the class
5. Embeddings and the weights are updated w.r.t. Cross-entropy loss



4

EXPERIMENTAL SETUP

DATASETS

Dataset	Developers	Applications	Description
Malicious	10	262	Malicious applications collected from Koodous system
Benign	40	1559	Benign applications collected from 8 different Android markets

EVALUATION CRITERIA

- Accuracy : $\dfrac{\text{number of correct predictions}}{\text{number of all instances}}$

- F-score $= 2 \cdot \dfrac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

BASELINE

- The model proposed by Kalgutkar et al. in
  - “Android authorship attribution through string analysis”
  - Count-based features
  - SVM Algorithm

IMPLEMENTATION

- Language: Python
- Hyper-Parameters:
    - Epoch: 10
    - Initial learning-rate: .001
    - embedding-size=50

5

RESULTS

RESULTS

Model	Accuracy	F-Score
BENIGN DATASET (Developer Classification)		
Proposed Model	**96.1%**	**94.8%**
Kalgutkar et. all.	94.4%	93.1%
MALWARE DATASET (Developer Classification)		
Proposed Model	**86.7%**	**85.9%**
Kalgutkar et. all.	81.63%	82.12%
MIXED DATASET (Application Classification)		
Proposed Model	**95.7%**	**95.1%**
Kalgutkar et. all.	-	-

6

CONCLUSION

CONCLUSION

- Designing an Android authorship attribution system
- With goal of malware detection
- Leveraging string data within APKs by using natural language processing approaches (FastText)
- 3 % Improvement in terms of accuracy and F-score

REFERENCES

- Connor, T., & Ryszard, W. (2019). Apktool. Retrieved Accessed: 2019-10-30, from https://ibotpeaches.github.io/Apktool/
- Crussell, J., Gibler, C., & Chen, H. (2013). Scalable semantics-based detection of similar android applications. In Proc. of esorics (Vol. 13).
- Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., . . . Sheth, A. N. (2014). Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. ACM Transactions on Computer Systems (TOCS), 32(2), 5.
- Gonzalez, H., Stakhanova, N., & Ghorbani, A. A. (2014). Droidkin: Lightweight detection of android apps similarity. In International conference on security and privacy in communication networks (pp. 436-453).
- Gordon, M. I., Kim, D., Perkins, J. H., Gilham, L., Nguyen, N., & Rinard, M. C. (2015). Information flow analysis of android applications in droidsafe. In Ndss (Vol. 15, p. 110).
- Rovelli, P. (2019). Ninjadroid. Retrieved Accessed: 2019-10-30, from https://github.com/rovellipaolo/NinjaDroid

REFERENCES (COUNT.)

- Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X. (2012). Riskranker: scalable and accurate zero-day android malware detection. In Proceedings of the 10th international conference on mobile systems, applications, and services (pp. 281-294).
- Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.
- Kalgutkar, V., Stakhanova, N., Cook, P., & Matyukhina, A. (2018). Android authorship attribution through string analysis. In Proceedings of the 13th international conference on availability, reliability and security (p. 4).
- Karbab, E. B., & Debbabi, M. (2019). Maldy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis reports. Digital Investigation, 28, S77-S87.
- Killam, R., Cook, P., & Stakhanova, N. (2016). Android malware classification through analysis of string literals. Text Analytics for Cybersecurity and Online Safety (TACOS).
- Luo, S., Liu, Z., Ni, B., Wang, H., Sun, H., & Yuan, Y. (2019). Android malware analysis and detection based on attention-cnn-lstm. Journal of Computers, 14 (1), 31-44.

THANK YOU!

Any Questions?