# Detection of Fraud & Malware Apps in Google Play

Madhana Kalidas. V
Computer Science & Engineering
K.S.R.College of Engineering
Thiruchencode-637215
Namakkal.

Ramar. G
Computer Science & Engineering
K.S.R.College of Engineering
Thiruchencode-637215
Namakkal

Pradeep. K
Computer Science & Engineering
K.S.R.College of Engineering
Thiruchencode-637215
Namakkal.

Navith. N
Computer Science & Engineering
K.S.R.College of Engineering
Thiruchencode-637215
Namakkal

***Abstract*— Ranking fraud in the mobile App market refers to fraudulent or deceptive activities which have a purpose of bumping up the Apps in the popularity list. Indeed, it becomes more and more frequent for App developers to use shady means, such as inflating their Apps' sales or posting phony App ratings, to commit ranking fraud. While the importance of preventing ranking fraud has been widely recognized, there is limited understanding and research in this area. To this end, in this paper, we provide a holistic view of ranking fraud and propose a ranking fraud detection system for mobile Apps. Specifically, we first propose to accurately locate the ranking fraud by mining the active periods, namely leading sessions, of mobile Apps. Such leading sessions can be leveraged for detecting the local anomaly instead of global anomaly of App rankings.**

## INTRODUCTION

The project entitled **"Detection of Fraud & Malware Apps in Google play"** is a mobile app based application developed in dot net using Microsoft visual studio. We planned to develop this application for storing mobile app in app store, and also for retrieving and manipulating the apps.

## FUNCTIONAL REQUIREMENTS:

The mechanism of typicality-based CF recommendation is as follows: First, we cluster all items into several item groups. For example, we can cluster all movies into "war movies," "romance movies," and so on. Second, we form a user group corresponding to each item group (i.e., a set of users who like items of a particular item group), with all users having different typicality degrees in each of the user groups. Third, we build a user-typicality matrix and measure users' similarities based on users' typicality degrees in all user groups so as to select a set of "neighbours" of each user. Then, we predict the unknown rating of a user on an item based on the ratings of the "neighbours" of at user on the item.

## PROBLEM DESCRIPTION

*EXISTING SYSTEM:*

- Fraudulent behaviors in Google Play, the most popular Android app market, fuel search rank abuse and malware proliferation. To identify malware, previous work has focused on app executable and permission analysis. In this paper, we introduce Fairplay, a novel system that discovers and leverages traces left behind by fraudsters, to detect both malware and apps subjected to search rank fraud.

- Fairplay correlates review activities and uniquely combines detected review relations with linguistic and behavioral signals gleaned from Google Play app data (87K apps, 2.9M reviews, and 2.4M reviewers, collected over half a year), in order to identify suspicious apps.

- Fairplay achieves over 95% accuracy in classifying gold standard datasets of malware, fraudulent and legitimate apps. We show that 75% of the identified malware apps engage in search rank fraud. Fairplay discovers hundreds of fraudulent apps that currently evade Google Bouncer's detection technology. Fairplay also helped the discovery of more than 1,000reviews, reported for 193 apps, that reveal a new type of "coercive" review campaign: users are harassed into writing positive reviews, and install and review other apps.

## DRAWBACKS OF EXISTING SYSTEM:

- Although some of the existing approaches can be used for Fairplay detection ,they are not able to extract fraud evidences for a given time period (i.e., leading session).
- Cannot able to detect ranking fraud happened in Apps' historical leading sessions

- There is no existing benchmark to decide which leading sessions or Apps really contain ranking fraud.

PROPOSED SYSTEM:

- We first propose a simple yet effective algorithm to identify the leading sessions of each App based on its historical ranking records. Then, with the analysis of Apps' ranking behaviors, we find that the fraudulent Apps often have different ranking patterns in each leading session compared with normal Apps. Thus, we characterize some fraud evidences from Apps' historical ranking records, and develop three functions to extract such ranking based fraud evidences.
- We further propose two types of fraud evidences based on Apps' rating and review history, which reflect some anomaly patterns from Apps' historical rating and review records.
- In Ranking Based Evidences, by analyzing the Apps' historical ranking records, we observe that Apps' ranking behaviors in a leading event always satisfy a specific ranking pattern, which consists of three different ranking phases, namely, rising phase, maintaining phase and recession phase.
- In Rating Based Evidences, specifically, after an App has been published, it can be rated by any user who downloaded it. Indeed, user rating is one of the most important features of App advertisement. An App which has higher rating may attract more users to download and can also be ranked higher in the leaderboard. Thus, rating manipulation is also an important perspective of ranking fraud.
- In Review Based Evidences, besides ratings, most of the App stores also allow users to write some textual comments as App reviews. Such reviews can reflect the personal perceptions and usage experiences of existing users for particular mobile Apps. Indeed, review manipulation is one of the most important perspective of App ranking fraud.

ADVANTAGES OF PROPOSED SYSTEM:

- The proposed framework is scalable and can be extended with other domain generated evidences for ranking fraud detection.
- Experimental results show the effectiveness of the proposed system, the scalability of the detection algorithm as well as some regularity of ranking fraud activities.
- To the best of our knowledge, there is no existing benchmark to decide which leading sessions or Apps really contain ranking fraud.

SYSTEM ANALYSIS

*REQUIREMENT ANALYSIS*

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, and outputs. The proposed system is achieved by detecting the fraud ranking in mobile apps.

*SYSTEM ANALYSIS*

Besides ratings, most of the App stores also allow users to write some textual comments as App reviews. Such reviews can reflect the personal perceptions and usage experiences of existing users for particular mobile Apps. Indeed, review manipulation is one of the most important perspective of App ranking fraud. Specifically, before downloading or purchasing a new mobile App, users often first read its historical reviews to ease their decision making, and a mobile App contains more positive reviews may attract more users to download. Therefore, imposters often post fake reviews in the leading sessions of a specific App in order to inflate the App downloads, and thus propel the App's ranking position in the leader board. Although some previous works on review spam detection have been reported in recent years [14], [19], [21], the problem of detecting the local anomaly of reviews in the leading sessions and capturing them as evidences for ranking fraud detection are still under-explored.

MODULE DESCRIPTION

*MINING LEADING SESSIONS*

In the first module, we develop our system environment with the details of App like an app store. Intuitively, the leading sessions of a mobile App represent its periods of popularity, so the ranking manipulation will only take place in these leading sessions. Therefore, the problem of detecting ranking fraud is to detect fraudulent leading sessions. Along this line, the first task is how to mine the leading sessions of a mobile App from its historical ranking records. There are two main steps for mining leading sessions. First, we need to discover leading events from the App's historical ranking records. Second, we need to merge adjacent leading events for constructing leading sessions.

RANKING BASED EVIDENCES

In this module, we develop Ranking based Evidences system. By analyzing the Apps' historical ranking records, web serve that Apps' ranking behaviors in a leading event always satisfy a specific ranking pattern, which consists of three different ranking phases, namely, rising phase, maintaining phase and recession phase. Specifically, in each leading event, an App's ranking first increases to a peak position in the leader board (i.e., rising phase), then keeps such peak position for a period (i.e., maintaining phase), and finally decreases till the end of the event (i.e., recession phase).

*RATING BASED EVIDENCES*

In the third module, we enhance the system with Rating based evidences module. The ranking based evidences are useful for ranking fraud detection. However, sometimes, it is not sufficient to only use ranking based evidences. For example, some Apps created by the famous developers, such

as Gameloft, may have some leading events with large values of u1 due to the developers' credibility and the "word-of-mouth" advertising effect. Moreover, some of the legal marketing services, such as "limited-time discount", may also result in significant ranking based evidences.

## REVIEW BASED EVIDENCES

In this module we add the Review based Evidences module in our system. Besides ratings, most of the App stores also allow users to write some textual comments as App reviews. Such reviews can reflect the personal perceptions and usage experiences of existing users for particular mobile Apps. Indeed, review manipulation is one of the most important perspective of App ranking fraud. Specifically, before downloading or purchasing a new mobile App, users often first read its historical reviews to ease their decision making, and a mobile App contains more positive reviews may attract more users to download. Therefore, imposters often post fake reviews in the leading sessions of a specific App in order to inflate the App downloads, and thus propel the App's ranking position in the leader board.

## EVIDENCE AGGREGATION

In this module we develop the Evidence Aggregation module to our system. After extracting three types of fraud evidences, the next challenge is how to combine them for ranking fraud detection. Indeed, there are many ranking and evidence aggregation methods in the literature, such as permutation based models score based models and Dempster-Shafer rules . However, some of these methods focus on learning a global ranking for all candidates. This is not proper for detecting ranking fraud for new Apps. Other methods are based on supervised learning techniques, which depend on the labeled training data and are hard to be exploited. Instead, we propose an unsupervised approach based on fraud similarity to combine these evidences.

## TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. There are various types of test. Each test type addresses a specific testing requirement.

*UNIT TESTING*

In unit testing, we have to test the programs making up the system. For this reason, Unit testing sometimes called as Program testing. Unit testing first on the modules independently of one another, to locate errors. This enables to detect errors in coding and logic in the module. The testing was carried out during programming stage itself.

*INTEGRATION TESTING*

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

*SYSTEM TESTING*

*a)* System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points. In this testing it is based on the coding to assign or performs the function by using the methods and data for the program to be run.

I. WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner coding, structure and language of the software

II. BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box must be written from a definitive source document, such as specification or requirements document. The test provides inputs and responds to outputs without considering how the software works.

*b) SYSTEM IMPLEMENTATION*

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on. The aim of the system illustration was to identify any malfunction of the system. After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly.

## CONCLUSION

Here we developed a ranking fraud detection system for mobile Apps. Specifically, we first showed that ranking fraud happened in leading sessions and provided a method for mining leading sessions for each App from its historical ranking records. Then, we identified ranking based evidences, rating based evidences and review based evidences for detecting ranking fraud. Moreover, we proposed an optimization based aggregation method to integrate all the evidences for evaluating the credibility of leading sessions from mobile Apps. An unique perspective of this approach is that all the evidences can be modeled by statistical hypothesis tests, thus it is easy to be extended with other evidences from domain knowledge to detect ranking fraud. Finally, we validate the proposed system with extensive experiments on real-world App data collected from the Apple's App store.. In the future, we plan to study more effective fraud evidences and analyze the latent relationship among rating, review and rankings. Moreover, we will extend our ranking fraud detection approach with other mobile App related services, such as mobile Apps recommendation, for enhancing user experience.

## FUTURE ENHANCEMENTS

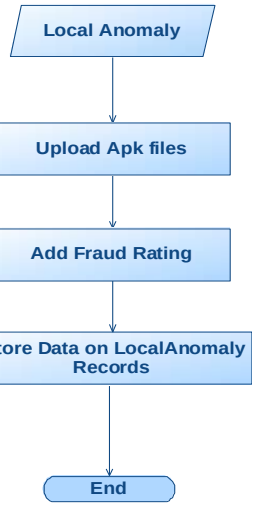
There are several possible future extensions to our work. In TyCo, we do not specify how to cluster resources so

as to find out item groups and the corresponding user groups. One possible future work is to try different clustering methods and see how the recommendation results are affected. How to using parallel computing methods (e.g., MapReduce) to handle the large scale applications is also one of the possible future works.

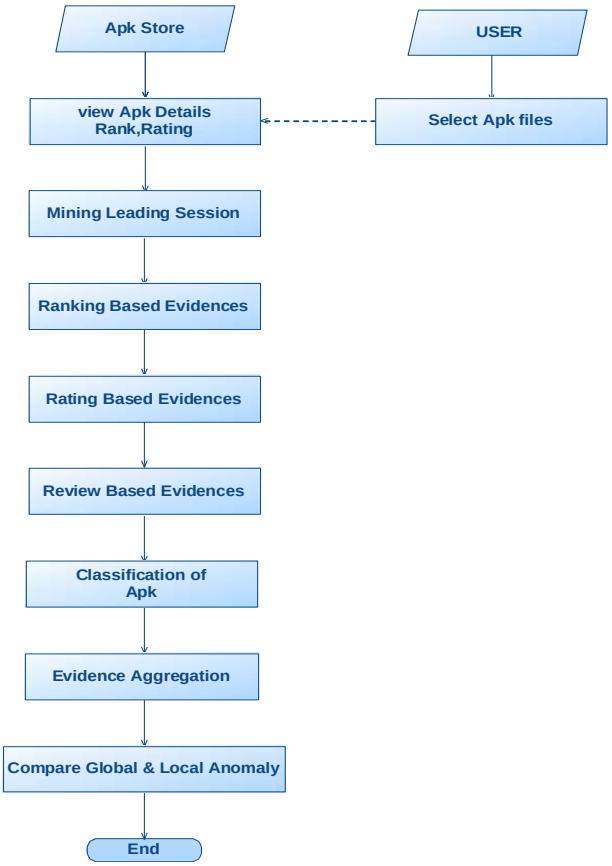
USECASE DIAGRAM

DATAFLOW DIAGRAM

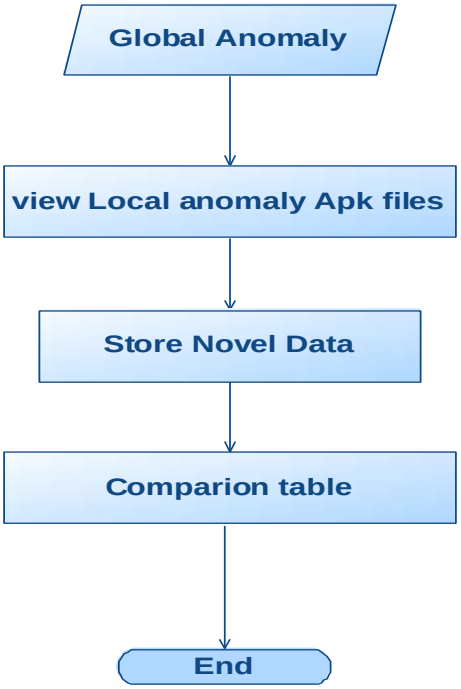
ADMIN




USER




GLOBAL ANOMALY




REFERENCES

[1] L. Azzopardi, M. Girolami, and K. V. Risjbergen, "Investigating the relationship between language model perplexity and ir precision-recall measures," in Proc. 26th Int. Conf. Res. Develop. Inform. Retrieval, 2003, pp. 369–370.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," J. Mach. Learn. Res., pp. 993–1022, 2003.

[3] Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou, "A taxi driving fraud detection system," in Proc. IEEE 11th Int. Conf. Data Mining, 2011, pp. 181–190.

[4] D. F. Gleich and L.-h. Lim, "Rank aggregation via nuclear norm minimization," in Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2011, pp. 60–68.

[5] T. L. Griffiths and M. Steyvers, "Finding scientific topics," Proc. Nat. Acad. Sci. USA, vol. 101, pp. 5228–5235, 2004.

[6] G. Heinrich, Parameter estimation for text analysis, " Univ. Leipzig, Leipzig, Germany, Tech. Rep., http://faculty.cs.byu.edu/~ringger/CS601R/papers/Heinrich-GibbsLDA.pdf, 2008.

[7] [N. Jindal and B. Liu, "Opinion spam and analysis," in Proc. Int. Conf. Web Search Data Mining, 2008, pp. 219–230.

[8] J. Kivinen and M. K. Warmuth, "Additive versus exponentiated gradient updates for linear prediction," in Proc. 27th Annu. ACM Symp. Theory Comput., 1995, pp. 209–218.

[9] A. Klementiev, D. Roth, and K. Small, "An unsupervised learning algorithm for rank aggregation," in Proc. 18th Eur. Conf. Mach. Learn., 2007, pp. 616–623.

[10] A. Klementiev, D. Roth, and K. Small, "Unsupervised rank aggregation with distance-based models," in Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 472–479.

[11] A. Klementiev, D. Roth, K. Small, and I. Titov, "Unsupervised rank aggregation with domain-specific expertise," in Proc. 21st Int. Joint Conf. Artif. Intell., 2009, pp. 1101–1106.

[12] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw, "Detecting product review spammers using rating behaviors," in Proc. 19thACMInt. Conf. Inform. Knowl. Manage., 2010, pp. 939–948.

[13] Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li, "Supervised rank aggregation," in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 481–490.

[14] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh, "Spotting opinion spammers using behavioral footprints," in Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2013, pp. 632–640.

[15] . Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "Detecting spam web pages through content analysis," in Proc. 15th Int. Conf. World Wide Web, 2006, pp. 83–92.
[16] G. Shafer, A Mathematical Theory of Evidence. Princeton, NJ, USA: Princeton Univ. Press, 1976.
[17] K. Shi and K. Ali, "Getjar mobile application recommendations with very sparse datasets," in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2012, pp. 204–212.
[18] [8] L. Azzopardi, M. Girolami, and K. V. Risjbergen, "Investigating the relationship between language model perplexity and ir precision-recall measures," in Proc. 26th Int. Conf. Res. Develop. Inform. Retrieval, 2003, pp. 369–370.
[19] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," J. Mach. Learn. Res., pp. 993–1022, 2003.
Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou, "A taxi driving fraud detection system," in Proc. IEEE 11th Int. Conf. Data Mining, 2011, pp. 181–190