IEEE Transactions on Knowledge and Data Engineering,Volume:27,Issue:1,Issue Date :Jan.1.2015

Discovery of Ranking Fraud for Mobile Apps

Hengshu Zhu[1,2] Hui Xiong[2]* Yong Ge[3] Enhong Chen[1]
[1]University of Science and Technology of China, {zhs, cheneh}@ustc.edu.cn
[2]Rutgers University, hxiong@rutgers.edu
[3]UNC Charlotte, yong.ge@uncc.edu

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 develops 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 investigate two types of evidences, ranking based evidences and rating based evidences, by modeling Apps' ranking and rating behaviors through statistical hypotheses tests. In addition, we propose an optimization based aggregation method to integrate all the evidences for fraud detection. Finally, we evaluate the proposed system with real-world App data collected from the Apple's App Store for a long time period. In the experiments, we validate the effectiveness of the proposed system, and show the scalability of the detection algorithm as well as some regularity of ranking fraud activities.

Categories and Subject Descriptors

H.2.8.d [**Information Technology and Systems**]: Database Applications - Data Mining

Keywords

Ranking Fraud Detection, Mobile Apps

1. INTRODUCTION

The number of mobile Apps has grown at a breathtaking rate over the past few years. For example, as of the end of April 2013, there are more than 1.6 million Apps at Apple's App store and Google Play. To stimulate the development of mobile Apps, many App stores launched daily App leaderboards, which demonstrate the chart rankings of most popular Apps. Indeed, the App leaderboard is one of the most important ways for promoting mobile Apps. A higher rank on the leaderboard usually leads to a huge number of downloads and million dollars in revenue. Therefore, App developers tend to explore various ways such as advertising campaigns to promote their Apps in order to have their Apps ranked as high as possible in such App leaderboards.

*Contact Author.

However, as a recent trend, instead of relying on traditional marketing solutions, shady App developers resort to some fraudulent means to deliberately boost their Apps and eventually manipulate the chart rankings on an App store. This is usually implemented by using so-called "*bot farms*" or "*human water armies*" to inflate the App downloads and ratings in a very short time. For example, an article from VentureBeat [3] reported that, when an App was promoted with the help of ranking manipulation, it could be propelled from number 1,800 to the top 25 in Apple's top free leaderboard and more than 50,000-100,000 new users could be acquired within a couple of days. In fact, such ranking fraud raises great concerns to the mobile App industry. For example, Apple has warned of cracking down on App developers who commit ranking fraud [2] in the Apple's App store.

In the literature, while there are some related work, such as web ranking spam detection [11, 13, 14], online review spam detection [10, 15, 16], and mobile App recommendation [12, 17, 18, 19], the problem of detecting ranking fraud for mobile Apps is still under-explored. To fill this crucial void, in this paper, we propose to develop a ranking fraud detection system for mobile Apps. Along this line, we identify several important challenges. First, ranking fraud does not always happen in the whole life cycle of an App, so we need to detect the time when fraud happens. Second, due to the huge number of mobile Apps, it is difficult to manually label ranking fraud for each App, so it is important to have a way to automatically detect ranking fraud without using any benchmark information. Finally, due to the dynamic nature of chart rankings, it is not easy to identify and confirm the evidences linked to ranking fraud.

Indeed, our careful observation reveals that fraudulent Apps do not always be ranked high in the leaderboard, but only in some *leading events*, which form different *leading sessions*. Note that we will introduce both leading events and leading sessions in detail later. In other words, ranking fraud usually happens in these leading sessions. Therefore, detecting ranking fraud of mobile Apps is actually to detect ranking fraud within leading sessions of mobile Apps. Specifically, we first propose a simple yet effective algorithm

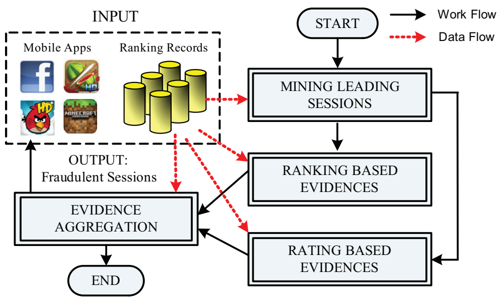


Figure 1: The framework of the ranking fraud detection system for mobile Apps.

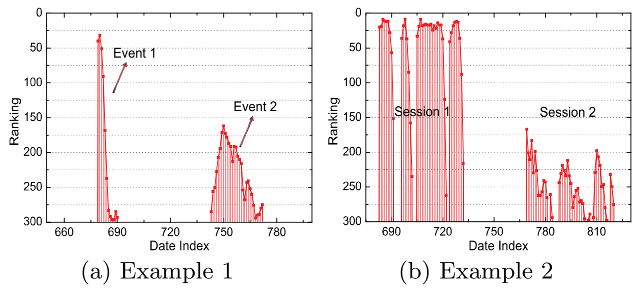


Figure 2: (a) Examples of leading events; (b) Examples of leading sessions of mobile Apps.

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. Nonetheless, the ranking based evidences can be affected by some legitimate marketing campaigns, such as "limited-time discount". As a result, it is not sufficient to only use ranking based evidences. Therefore, we further propose two functions to discover rating based evidences, which reflect some anomaly patterns from Apps' historical rating records. In addition, we develop an unsupervised evidence-aggregation method to integrate these two types of evidences for evaluating the credibility of leading sessions from mobile Apps. Figure 1 shows the framework of our ranking fraud detection system for mobile Apps.

It is worth noting that all the evidences are extracted by modeling Apps' ranking and rating behaviors through statistical hypotheses tests. The proposed framework is scalable and can be extended with other domain-generated evidences for ranking fraud detection. Finally, we evaluate the proposed system with real-world App data collected from the Apple's App store for a long time period. Experimental results show the effectiveness of the proposed system, the scalability of the detection algorithm as well as some regularity of ranking fraud activities.

Overview. The remainder of this paper is organized as follows. In Section 2, we introduce some preliminaries and how to mine leading sessions for mobile Apps. Section 3 presents how to extract ranking and rating based evidences and combine them for ranking fraud detection. In Section 4 we make some further discussion about the proposed approach. In Section 5, we report the experimental results on two long-term real-world data sets. Section 6 provides a brief review of related works. Finally, in Section 7, we conclude the paper.

2. IDENTIFYING LEADING SESSIONS FOR MOBILE APPS

In this section, we first introduce some preliminaries, and then show how to mine leading sessions for mobile Apps from their historical ranking records.

2.1 Preliminaries

The App leaderboard demonstrates top K popular Apps with respect to different categories, such as "*Top Free Apps*" and "*Top Paid Apps*". Moreover, the leaderboard is usually updated periodically (e.g., daily). Therefore, each mobile App a has many historical ranking records which can be denoted as a time series, $R_a = \{r_1^a, \cdots, r_i^a, \cdots, r_n^a\}$, where $r_i^a \in \{1, ..., K, +\infty\}$ is the ranking of a at time stamp t_i; $+\infty$ means a is not ranked in the top K list; n denotes the number of all ranking records. Note that, the smaller value r_i^a has, the higher ranking the App obtains.

By analyzing the historical ranking records of mobile Apps, we observe that Apps are not always ranked high in the leaderboard, but only in some *leading events*. For example, Figure 2 (a) shows an example of leading events of a mobile App. Formally, we define a leading event as follows.

DEFINITION 1 (LEADING EVENT). *Given a ranking threshold $K^* \in [1, K]$, a leading event **e** of App **a** contains a time range $T_e = [t_{start}^e, t_{end}^e]$ and corresponding rankings of **a**, which satisfies $r_{start}^a \le K^* < r_{start-1}^a$, and $r_{end}^a \le K^* < r_{end+1}^a$. Moreover, $\forall t_k \in (t_{start}^e, t_{end}^e)$, we have $r_k^a \le K^*$.*

Note that we apply a ranking threshold K^* which is usually smaller than K here because K may be very big (e.g., 1000), and the ranking records beyond K^* (e.g., 100) are not very useful for detecting the ranking manipulations.

Furthermore, we also find that some Apps have several adjacent leading events which are close to each other and form a *leading session*. For example, Figure 2(b) shows an example of adjacent leading events of a given mobile App, which form two leading sessions. Particularly, a leading event which does not have other nearby neighbors can also be treated as a special leading session. The formal definition of leading session is as follows.

DEFINITION 2 (LEADING SESSION). *A leading session **s** of App **a** contains a time range $T_s = [t_{start}^s, t_{end}^s]$ and n adjacent leading events $\{e_1, ..., e_n\}$, which satisfies $t_{start}^s = t_{start}^{e_1}$, $t_{end}^s = t_{end}^{e_n}$ and there is no other leading session s^* makes $T_s \subseteq T_{s^*}$. Moreover, $\forall i \in [1, n)$, we have $(t_{start}^{e_{i+1}} - t_{end}^{e_i}) < \phi$, where ϕ is a predefined threshold.*

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.

2.2 Mining Leading Sessions

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

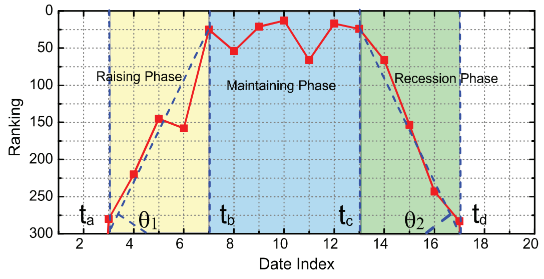
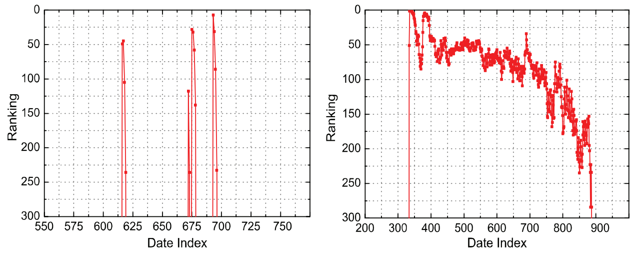


Figure 3: An example of different ranking phases of a leading event.



(a) Example 1 (b) Example 2

Figure 4: Two real-world examples of leading events.

leading events for constructing leading sessions. Specifically, Algorithm 1 demonstrates the pseudo code of mining leading session for a given App a.

Algorithm 1 Mining Leading Sessions

Input 1: a's historical ranking records R_a;
Input 2: the ranking threshold K^*;
Input 3: the merging threshold ϕ;
Output: the set of a's leading sessions S_a;
Initialization: $S_a = \emptyset$;

```
1: $E_a = \emptyset$; $e = \emptyset$; $s = \emptyset$; $t^e_{start} = 0$;
2: for each $i \in [1, |R_a|]$ do
3:     if $r^a_i \le K^*$ and $t^e_{start} == 0$ then
4:         $t_{start} = t_i$;
5:     else if $r^a_i > K^*$ and $t^e_{start} \ne 0$ then
6:         //found one event;
7:         $t^e_{end} = t_{i-1}$; $e =< t^e_{start}, t^e_{end} >$;
8:         if $|E_a| == \emptyset$ then
9:             $E_a \cup = e$; $t^s_{start} = t^e_{start}$; $t^s_{end} = t^e_{end}$;
10:        else if $|E_a| > 1$ and $(t^e_{start} - t^{e^*}_{end}) < \phi$ then
11:            //$e^*$ is the last leading event before $e$ in $E_a$;
12:            $E_a \cup = e$; $t^s_{end} = t^e_{end}$;
13:        else then
14:            //found one session;
15:            $s =< t^s_{start}, t^s_{end}, E_a >$;
16:            $S_a \cup = s$; $E_a = \emptyset$; $s = \emptyset$ is a new session;
17:            go to Step 7;
18:        $t^e_{start} = 0$; $e = \emptyset$ is a new leading event;
19: return $S_a$
```

In Algorithm 1, we denote each leading event e and session s as tuples $< t^e_{start}, t^e_{end} >$ and $< t^s_{start}, t^s_{end}, E_s >$ respectively, where E_s is the set of leading events in session s. Specifically, we first extract individual leading event e for the given App a (i.e., Step 2 to 5) from the beginning time. For each extracted individual leading event e, we check the time span between e and the previous leading event e^* to decide whether they belong to the same leading session based on Definition 2. Particularly, if $(t^e_{start} - t^{e^*}_{end}) \ge \phi$, e will be considered as a new leading session (i.e., Step 6 to 13). Thus, this algorithm can identify leading events and sessions by scanning a's historical ranking records only once.

3. EXTRACTING EVIDENCES FOR RANKING FRAUD DETECTION

In this section, we study how to extract and combine fraud evidences for ranking fraud detection.

3.1 Ranking based Evidences

According to the definitions introduced in Section 2, a leading session is composed of several leading events. Therefore, we should first analyze the basic characterizes of leading events for extracting fraud 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*. Specifically, in each leading event, an App's ranking first increases to a peak position in the leaderboard (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). Figure 3 shows an example of different ranking phases of a leading event. Indeed, such a ranking pattern shows an important understanding of leading event. In the following, we formally define the three ranking phases of a leading event.

DEFINITION 3 (RANKING PHASES OF A LEADING EVENT). *Give a leading event **e** of App **a** with time range $[t^e_{start}, t^e_{end}]$, where the highest ranking position of **a** is r^a_{peak}, which belongs to ΔR. The **rising phase** of **e** is a time range $[t^e_a, t^e_b]$, where $t^e_a = t^e_{start}$, $r^a_b \in \Delta R$ and $\forall t_i \in [t^e_a, t^e_b)$ satisfies $r^a_i \notin \Delta R$. The **maintaining phase** of **e** is a time range $[t^e_b, t^e_c]$, where $r^a_c \in \Delta R$ and $\forall t_i \in (t^e_c, t^e_{end}]$ satisfies $r^a_i \notin \Delta R$. The **recession phase** is a time range $[t^e_c, t^e_d]$, where $t^e_d = t^e_{end}$.*

Note that, in Definition 3, ΔR is a ranking range to decide the beginning time and the end time of the maintaining phase. t^e_b and t^e_c are the first and last time when the App is ranked into ΔR. It is because an App, even with ranking manipulation, cannot always maintain the same peak position (e.g., rank 1) in the leaderboard but only in a ranking range (e.g., top 25). If a leading session s of App a has ranking fraud, a's ranking behaviors in these three ranking phases of leading events in s should be different from those in a normal leading session. Actually, we observe that each App with ranking manipulation always has an expected ranking target, such as top 25 in leaderboard for one week, and the hired marketing firms also charge money according to such ranking expectation (e.g., \$1000/day in top 25). Therefore, for both App developers and marketing firms, the earlier the ranking expectation meets, the more money can be earned. Moreover, after reaching and maintaining the expected ranking for a required period, the ranking manipulation will be stopped and the ranking of the malicious App will decrease dramatically. As a result, the suspicious leading events may contain very short rising and recession phase. Meanwhile, the cost of ranking manipulation with high ranking expectations is quite expensive. Therefore, the fraudulent Apps often have a short maintaining phase with high ranking positions in each leading event.

Figure 4 (a) shows an example of ranking records from one of the reported suspicious Apps [4]. In this figure, we can see that this App has several impulsive leading events with high ranking positions. In contrast, the ranking behaviors of a normal App's leading event may be completely different. For example, Figure 4 (b) shows an example of ranking records from a popular App "*Angry Birds: Space*", which contains a leading event with a long time range (i.e., more than one year), especially for the recession phase. In fact, once a normal App is ranked high in the leaderboard, it often owns a lot of honest fans and may attract more and more users to download. As a result, this App will be ranked high in the leaderboard for a long time. Based on the above discussion, we propose some ranking based signatures of leading sessions to construct fraud evidences for ranking fraud detection.

• **EVIDENCE 1.** As shown in Figure 3, we use two shape parameters θ_1 and θ_2 to quantify the ranking patterns of the rising phase and the recession phase of App a's leading event e, which can be computed by

$$\theta_1^e = \arctan(\frac{K^* - r_b^a}{t_b^e - t_a^e}), \;\; \theta_2^e = \arctan(\frac{K^* - r_c^a}{t_d^e - t_c^e}). \tag{1}$$

where K^* is the ranking threshold in Definition 1. Intuitively, a large θ_1 may indicate that the App has been bumped to a high rank within a short time, and a large θ_2 may indicate that the App has dropped from a high rank to the bottom within a short time. Therefore, a leading session, which has more leading events with large θ_1 and θ_2 values, has higher probability of having ranking fraud. Here, we define a fraud signature $\overline{\theta}_s$ for a leading session as follows.

$$\overline{\theta}_s = \frac{1}{|E_s|}\sum_{e\in s}(\theta_1^e + \theta_2^e), \tag{2}$$

where $|E_s|$ is the number of leading events in session s. Intuitively, if a leading session s contains significantly higher $\overline{\theta}_s$ compared with other leading sessions of Apps in the leaderboard, it has high probability of having ranking fraud. To capture this, we propose to apply statistical hypothesis test for computing the significance of $\overline{\theta}_s$ for each leading session. Specifically, we define two statistical hypotheses as follows and compute the p-value of each leading session.

▷ HYPOTHESIS 0*: The signature $\overline{\theta}_s$ of leading session s is not useful for detecting ranking fraud.*

▷ HYPOTHESIS 1*: The signature $\overline{\theta}_s$ of leading session s is significantly greater than expectation.*

Here, we propose to use the popular Gaussian approximation to compute the p-value with the above hypotheses. Specifically, we assume $\overline{\theta}_s$ follows the Gaussian distribution, $\overline{\theta}_s \sim \mathcal{N}(\mu_{\overline{\theta}}, \sigma_{\overline{\theta}})$, where $\mu_{\overline{\theta}}$ and $\sigma_{\overline{\theta}}$ can be learnt by the popular maximum-likelihood estimation (MLE) method from the observations of $\overline{\theta}_s$ in all Apps' historical leading sessions. Then, we can calculate the p-value by

$$\mathbb{P}\big(\mathcal{N}(\mu_{\overline{\theta}}, \sigma_{\overline{\theta}}) \geq \overline{\theta}_s\big) = 1 - \frac{1}{2}\left(1 + \text{erf}\Big(\frac{\overline{\theta}_s - \mu_{\overline{\theta}}}{\sigma_{\overline{\theta}}\sqrt{2}}\Big)\right), \tag{3}$$

where erf(x) is the Gaussian Error Function as follows,

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt. \tag{4}$$

Intuitively, a leading session with a smaller p-value $\mathbb{P}$ has more chance to reject HYPOTHESIS 0 and accept HYPOTHESIS 1. This means it has more chance of committing ranking fraud. Thus, we define the evidence as

$$\Psi_1(s) = 1 - \mathbb{P}\big(\mathcal{N}(\mu_{\overline{\theta}}, \sigma_{\overline{\theta}}) \geq \overline{\theta}_s\big). \tag{5}$$

EVIDENCE 2. As discussed above, the Apps with ranking fraud often have a short maintaining phase with high ranking positions in each leading event. Thus, if we denote the maintaining phase of a leading event e as $\Delta t_m^e = (t_c^e - t_b^e + 1)$, and the average rank in this maintaining phase as $\overline{r}_m^e$, we can define a fraud signature χ_s for each leading session as follows,

$$\chi_s = \frac{1}{|E_s|}\sum_{e\in s}\frac{K^* - \overline{r}_m^e}{\Delta t_m^e}, \tag{6}$$

where K^* is the ranking threshold in Definition 1. If a leading session contains significantly higher χ_s compared with other leading sessions of Apps in the leaderboard, it has high chance of having ranking fraud. To capture such signatures, we define two statistical hypotheses as follows to compute the significance of χ_s for each leading session.

▷ HYPOTHESIS 0*: The signature χ_s of leading session s is not useful for detecting ranking fraud.*

▷ HYPOTHESIS 1*: The signature χ_s of leading session s is significantly higher than expectation.*

Here, we also propose to use the Gaussian approximation to calculate the p-value with the above hypotheses. Specifically, we assume χ_s follows the Gaussian distribution, $\chi_s \sim \mathcal{N}(\mu_\chi, \sigma_\chi)$, where μ_χ and σ_χ can be learnt by the MLE method from the observations of χ_s in all Apps' historical leading sessions. Then, we can calculate the evidence by

$$\Psi_2(s) = 1 - \mathbb{P}\big(\mathcal{N}(\mu_\chi, \sigma_\chi) \geq \chi_s\big). \tag{7}$$

EVIDENCE 3. The number of leading events in a leading session, i.e., $|E_s|$, is also a strong signature of ranking fraud. For a normal App, the recession phase indicates the fading of popularity. Therefore, after the end of a leading event, it is unlikely to appear another leading event in a short time unless the App updates its version or carries out some sales promotion. Therefore, if a leading session contains much more leading events compared with other leading sessions of Apps in the leaderboard, it has high probability of having ranking fraud. To capture this, we define two statistical hypotheses to compute the significance of $|E_s|$ for each leading session as follows.

▷ HYPOTHESIS 0*: The signature $|E_s|$ of leading session s is not useful for detecting ranking fraud.*

▷ HYPOTHESIS 1*: The signature $|E_s|$ of leading session s is significantly lager than expectation.*

Since $|E_s|$ always has discrete values, we propose to leverage the Poisson approximation to calculate the p-value with the above hypotheses. Specifically, we assume $|E_s|$ follows the Poisson distribution, $|E_s| \sim \mathcal{P}(\lambda_s)$, where the parameter λ_s can be learnt by the MLE method from the observations of $|E_s|$ in all Apps' historical leading sessions. Then, we can calculate the p-value as follows,

$$\mathbb{P}\big(\mathcal{P}(\lambda_s) \geq |E_s|\big) = 1 - e^{-\lambda_s}\sum_{i=0}^{|E_s|}\frac{(\lambda_s)^i}{i!}. \tag{8}$$

Therefore, we can compute the evidence by

$$\Psi_3(s) = 1 - \mathbb{P}\big(\mathcal{P}(\lambda_s) \geq |E_s|\big). \tag{9}$$

The values of the above three evidences $\Psi_1(s)$, $\Psi_2(s)$ and $\Psi_3(s)$ are all within the range of $[0, 1]$. Meanwhile, the higher evidence value a leading session has, the higher probability this session has ranking fraud activities.

3.2 Rating based Evidences

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 θ_1 due to the developers' credibility and the "words of mouth" advertising effect. Moreover, some of the legal marketing services, such as "limited-time discount", may also result in significant ranking based evidences. To solve this issue, we also study how to extract fraud evidences from Apps' historical rating records.

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 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. Intuitively, if an App has ranking fraud in a leading session s, the ratings during the time period of s may have anomaly patterns compared with its historical ratings, which can be used for constructing rating based evidences. Thus, we define two rating fraud evidences based on user rating behaviors as follows.

EVIDENCE 4. For a normal App, the average rating in a specific leading session should be consistent with the average value of all historical ratings. In contrast, an App with rating manipulation might have surprisingly high ratings in the fraudulent leading sessions with respect to its historical ratings. Here, we define a fraud signature $\Delta\mathcal{R}_s$ for each leading session as follows,

$$\Delta\mathcal{R}_s = \frac{\overline{\mathcal{R}}_s - \overline{\mathcal{R}}_a}{\overline{\mathcal{R}}_a}, \quad (s \in a) \tag{10}$$

where $\overline{\mathcal{R}}_s$ is the average rating in leading session s, and $\overline{\mathcal{R}}_a$ is the average historical rating of App a. Therefore, if a leading session has significantly high value of $\Delta\mathcal{R}_s$ compared with other leading sessions of Apps in the leaderboard, it has high probability of having ranking fraud. To capture this, we define statistical hypotheses to compute the significance of $\Delta\mathcal{R}_s$ for each leading session as follows.

- $\triangleright$ HYPOTHESIS 0: *The signature $\Delta\mathcal{R}_s$ of leading session s is not useful for detecting ranking fraud.*
- $\triangleright$ HYPOTHESIS 1: *The signature $\Delta\mathcal{R}_s$ of leading session s is significantly higher than expectation.*

Here, we use the Gaussian approximation to calculate the p-value with the above hypotheses. Specifically, we assume $\Delta\mathcal{R}_s$ follows the Gaussian distribution, $\Delta\mathcal{R}_s \sim \mathcal{N}(\mu_\mathcal{R}, \sigma_\mathcal{R})$, where $\mu_\mathcal{R}$ and $\sigma_\mathcal{R}$ can be learnt by the MLE method from the observations of $\Delta\mathcal{R}_s$ in all Apps' historical leading sessions. Then, we compute the evidence by

$$\Psi_4(s) = 1 - \mathbb{P}\big(\mathcal{N}(\mu_\mathcal{R}, \sigma_\mathcal{R}) \geq \Delta\mathcal{R}_s\big). \tag{11}$$

EVIDENCE 5. In the App rating records, each rating can be categorized into $|L|$ discrete rating levels, e.g., 1 to 5, which represent the user preferences of an App. The rating distribution with respect to the rating level l_i in a normal App a's leading session s, $p(l_i|\mathcal{R}_{s,a})$, should be consistent with the distribution in a's historical rating records, $p(l_i|\mathcal{R}_a)$, and vice versa. Specifically, we can compute the distribution by $p(l_i|\mathcal{R}_{s,a}) = \left(\frac{N^s_{l_i}}{N^s_{(.)}}\right)$, where $N^s_{l_i}$ is the number of ratings in s and the rating is at level l_i, $N^s_{(.)}$ is the total number of ratings in s. Meanwhile, we can compute $p(l_i|\mathcal{R}_a)$ in a similar way. Then, we use the Cosine similarity between $p(l_i|\mathcal{R}_{s,a})$ and $p(l_i|\mathcal{R}_a)$ to estimate the difference as follows.

$$\mathcal{D}(s) = \frac{\sum_{i=1}^{|L|} p(l_i|\mathcal{R}_{s,a}) \times p(l_i|\mathcal{R}_a)}{\sqrt{\sum_{i=1}^{|L|} p(l_i|\mathcal{R}_{s,a})^2} \times \sqrt{\sum_{i=1}^{|L|} p(l_i|\mathcal{R}_a)^2}}. \tag{12}$$

Therefore, if a leading session has significantly lower value of $\mathcal{D}(s)$ compared with other leading sessions of Apps in the leaderboard, it has high probability of having ranking fraud. To capture this, we define statistical hypotheses to compute the significance of $\mathcal{D}(s)$ for each leading session as follows.

- $\triangleright$ HYPOTHESIS 0: *The signature $\mathcal{D}(s)$ of leading session s is not useful for detecting ranking fraud.*
- $\triangleright$ HYPOTHESIS 1: *The signature $\mathcal{D}(s)$ of leading session s is significantly lower than expectation.*

Here, we use the Gaussian approximation to compute the p-value with the above hypotheses. Specifically, we assume $\mathcal{D}(s)$ follows the Gaussian distribution, $\mathcal{D}(s) \sim \mathcal{N}(\mu_\mathcal{D}, \sigma_\mathcal{D})$, where $\mu_\mathcal{D}$ and $\sigma_\mathcal{D}$ can be learnt by the MLE method from the observations of $\mathcal{D}(s)$ in all Apps' historical leading sessions. Then, we can compute the evidence by

$$\Psi_5(s) = 1 - \mathbb{P}\big(\mathcal{N}(\mu_\mathcal{D}, \sigma_\mathcal{D}) \leq \mathcal{D}(s)\big). \tag{13}$$

The values of two evidences $\Psi_4(s)$ and $\Psi_5(s)$ are in the range of $[0, 1]$. Meanwhile, the higher evidence value a leading session has, the more chance this session has ranking fraud activities.

3.3 Evidence Aggregation

After extracting ranking and rating based 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 [9], score based models [6] and Dempster-Shafer rules [5]. 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 ranking similarity to combine these evidences.

Specifically, we define the final evidence score $\Psi^*(s)$ as a linear combination of all the existing evidences as follows.

$$\Psi^*(s) = \sum_{i=1}^{N_\Psi} w_i \times \Psi_i(s), \tag{14}$$

where $N_\Psi = 5$ is the number of evidences, and weight $w_i \in [0, 1]$ is the aggregation parameter of evidence $\Psi_i(s)$, which satisfies $\sum_{i=1}^{N_\Psi} w_i = 1$. Thus, the problem of evidence aggregation becomes how to learn the proper parameters $\{w_i\}$ from the historical ranking records.

A basic assumption in our evidence aggregation approach is that effective evidences should rank leading sessions from a similar conditional distribution, while poor evidences will lead to a more uniformly random ranking distribution [8]. In other words, evidences that tend to be consistent with the plurality of evidences will be given higher weights and evidences which tend to disagree will be given smaller weights. Specifically, given a set of leading sessions, we first rank them by each evidence score and obtain N_Ψ ranked lists.

Let us denote $\pi_i(s)$ as the ranking of session s returned by $\Psi_i(s)$, then we can calculate the average ranking for leading session s by,

$$\overline{\pi}(s) = \frac{1}{N_\Psi} \sum_{i=1}^{N_\Psi} \pi_i(s). \tag{15}$$

Then, for each evidence score $\Psi_i(s)$, we can measure its consistence using the variance-like measure,

$$\sigma_i(s) = \big(\pi_i(s) - \overline{\pi}(s)\big)^2. \tag{16}$$

If $\sigma_i(s)$ is small, the corresponding $\Psi_i(s)$ should be given a bigger weight and vice versa. Therefore, given an App set $A = \{a_i\}$ with their leading sessions $\{s_j\}$, we can define the evidence aggregation problem as an optimization problem that minimizes weighted variances of the evidences over all leading sessions, that is

$$\arg\min_{\mathbf{w}} \sum_{a\in A} \sum_{s\in a} \sum_{i=1}^{N_\Psi} w_i \cdot \sigma_i(s), \tag{17}$$

$$s.t. \sum_{i=1}^{N_\Psi} w_i = 1;\ \forall w_i \geq 0. \tag{18}$$

In this paper, we exploit the gradient based approach with exponentiated updating [7, 8] to solve this problem. To be specific, we first assign $w_i = \frac{1}{N_\Psi}$ as the initial value, then for each s, we can compute the gradient by,

$$\nabla_i = \frac{\partial w_i \cdot \sigma_i(s)}{\partial w_i} = \sigma_i(s). \tag{19}$$

Thus, we can update the weight w_i by

$$w_i = \frac{w_i^* \times \exp(-\lambda \nabla_i)}{\sum_{j=1}^{N_\Psi} w_j^* \times \exp(-\lambda \nabla_j)}, \tag{20}$$

where w_i^* is the last updated w_i, and λ is the learning rate, which is empirically set $\lambda = 0.01$ in our experiments.

Finally, we can exploit Equation (14) to estimate the final evidence score of each leading session. Moreover, given a leading session s with a predefined threshold τ, we can determine that s has ranking fraud if $\Psi^*(s) > \tau$.

4. DISCUSSION

In this section, we provide some discussion about the proposed ranking fraud detection system for mobile Apps.

First, the download information is an important signature for detecting ranking fraud, since ranking manipulation is to use so-called "bot farms" or "human water armies" to inflate the App downloads and ratings in a very short time. However, the instant download information of each mobile App is often not available for analysis. In fact, Apple and Google do not provide accurate download information on any App. Furthermore, the App developers themselves are also reluctant to release their download information for various reasons. Therefore, in this paper, we mainly focus on extracting evidences from Apps' historical ranking records and rating records for ranking fraud detection. However, our approach is scalable for integrating other evidences if available, such as the evidences based on the download information.

Second, the proposed approach can detect ranking fraud happened in Apps' historical leading sessions. However, sometime, we need to detect such ranking fraud from Apps' current ranking observations. Actually, given the current ranking r^a_{now} of an App a, we can detect ranking fraud for it in two different cases. First, if $r^a_{now} > K^*$, where K^* is the ranking threshold introduced in Definition 1, we believe a does not involve in ranking fraud, since it is not in a leading event. Second, if $r^a_{now} < K^*$, which means a is in a new leading event e, we treat this case as a special case that $t^e_{end} = t^e_{now}$ and $\theta_2 = 0$. Therefore, it also can be detected by the proposed approach.

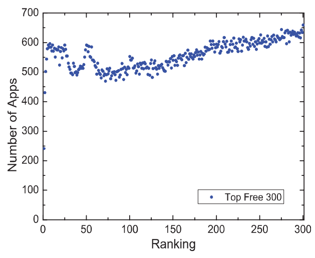
Finally, after detecting ranking fraud for each leading session of a mobile App, the remainder problem is how to estimate the credibility of this App. Indeed, we can also use Equation (14) to evaluate each App. To be specific, we define an App fraud score $\mathcal{F}(a)$ for each App a according to how many leading sessions of a contain ranking fraud.

$$\mathcal{F}(a) = \sum_{s\in a} [\![\Psi^*(s) > \tau]\!] \times \Psi^*(s) \times \Delta t^s, \tag{21}$$

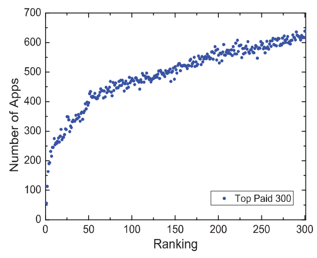
where $s \in a$ denotes that s is a leading session of App a; $[\![x]\!] = 1$ if $x = True$, and 0 otherwise; and $\Delta t^s = (t^s_{end} - t^s_{start} + 1)$ is the time range of s. Intuitively, an App contains more leading sessions, which have high ranking fraud scores and long time ranges, will have higher App fraud scores.

Table 1: Statistics of the experimental data.

	Top Free 300	Top Paid 300
App Num.	9,784	5,261
Ranking Num.	285,900	285,900
Avg. Ranking Num.	29.22	54.34
Rating Num.	14,912,459	4,561,943
Avg. Rating Num.	1,524.17	867.12



(a) Top Free 300 data set



(b) Top Paid 300 data set

Figure 5: The distribution of the number of Apps w.r.t different rankings.

5. EXPERIMENTAL RESULTS

In this section, we evaluate the performances of ranking fraud detection using real-world App data.

5.1 The Experimental Data

The experimental data sets were collected from the "Top Free 300" and "Top Paid 300" leaderboards of Apple's App Store (U.S.) from February 2, 2010 to September 17, 2012. The data sets contain the daily chart rankings [1] of top 300 free Apps and top 300 paid Apps, respectively. Moreover, each data set also contains the user ratings and comment information. Table 1 shows the detailed data characteristics.

Figures 5 (a) and 5 (b) show the distributions of the number of Apps with respect to different rankings in these data sets. In the figures, we can see that the number of Apps with

[1]The information was collected at 11:00PM (PST) each day.

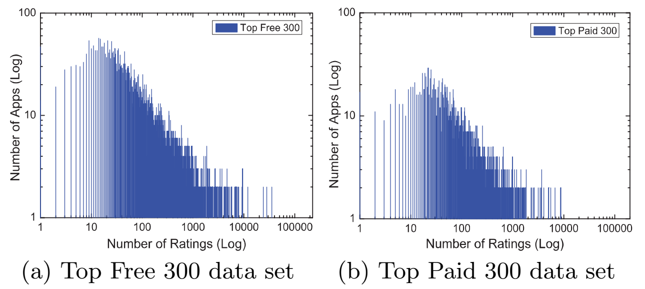


Figure 6: The distribution of the number of Apps w.r.t different numbers of ratings.

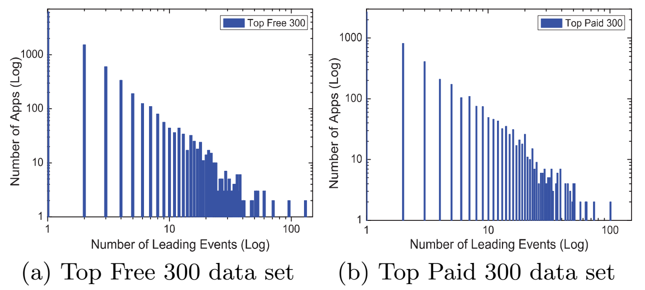


Figure 7: The distribution of the number of Apps w.r.t different numbers of leading events.

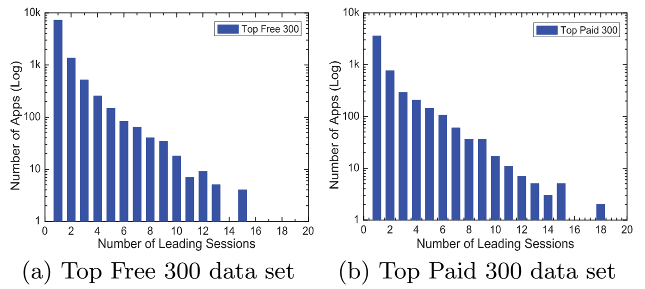


Figure 8: The distribution of the number of Apps w.r.t different number of leading sessions.

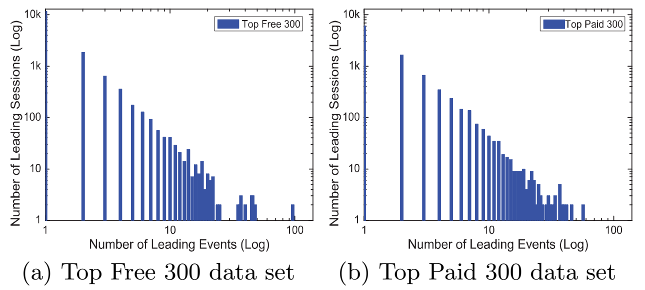


Figure 9: The distribution of the number of leading sessions w.r.t different number of leading events.

low rankings is more than that of Apps with high rankings. Moreover, the competition between free Apps is more than that between paid Apps, especially in high rankings (e.g., top 25). Figures 6 (a) and 6 (b) show the distribution of the number of Apps with respect to different number of ratings in these data sets. In the figures, we can see that the distribution of App ratings is not even, which indicates that only a small percentage of Apps are very popular.

5.2 Mining Leading Sessions

Here, we demonstrate the results of mining leading sessions in both data sets. Specifically, in Algorithm 1, we set the ranking threshold $K^* = 300$ and threshold $\phi = 7$. This denotes two adjacent leading events can be segmented into the same leading session if they occur within one week of each other. Figure 7 and Figure 8 show the distribution of the number of Apps with respect to different numbers of contained leading events and leading sessions in both data sets. In these figures, we can see that only a few Apps have many leading events and leading sessions. The average numbers of leading events and leading sessions are 2.69 and 1.57 for free Apps, and 4.20 and 1.86 for paid Apps. Moreover, Figures 9 (a) and 9 (b) show the distribution of the number of leading sessions with respect to different numbers of contained leading events in both data sets. In these figures, we can find only a few leading sessions contain many leading events. This also validates the evidence Ψ_3. Indeed, the average number of leading events in each leading session is 1.70 for free Apps and 2.26 for paid Apps.

5.3 Human Judgement based Evaluation

To the best of our knowledge, there is no existing benchmark to decide which leading sessions or Apps really contain ranking fraud. Thus, we develop three intuitive baselines and invite five human evaluators to validate the effectiveness of our approach **EA-RFD** (**E**vidence **A**ggregation based **R**anking **F**raud **D**etection).

5.3.1 *Baselines*

The first baseline **Ranking-RFD** stands for Ranking evidence based Ranking Fraud Detection, which estimates ranking fraud for each leading session by only using ranking based evidences (i.e., Ψ_1 to Ψ_3). These three evidences are integrated by the method introduced in Section 3.3.

The second baseline **Rating-RFD** stands for Rating evidence based Ranking Fraud Detection, which estimates the ranking fraud for each leading session by only using rating based evidences (i.e., Ψ_4 and Ψ_5). These two evidences are integrated by the method introduced in Section 3.3. Above two baselines are used for evaluating the effectiveness of different kinds of evidences.

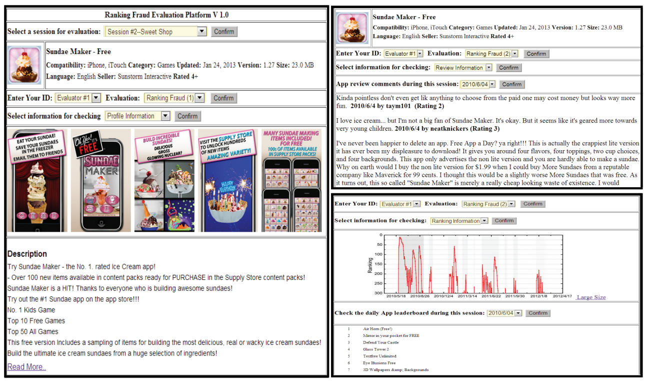
The third baseline **E-RFD** stands for Evidence based Ranking Fraud Detection, which estimates the ranking fraud for each leading session by both ranking and rating based evidences without evidence aggregation. Specifically, it ranks leading sessions by Equation 14, where each w_i is set to be 1/5 equally. This baseline is used for evaluating the effectiveness of our ranking aggregation method.

Note that, according to Definition 3, we need to define some ranking ranges before extracting ranking based evidences for EA-RFD, Rank-RFD and E-RFD. In our experiments, we segment the rankings into 5 different ranges, i.e., $[1, 10]$, $[11, 25]$, $[26, 50]$, $[51, 100]$, $[101, 300]$, which are commonly used in App leaderboards.

5.3.2 *The Experimental Setup*

To study the performance of ranking fraud detection by each approach, we set up the evaluation as follows.

First, for each approach, we selected 20 top ranked leading sessions (i.e., most suspicious sessions), and 20 bottom



Ranking Fraud Evaluation Platform V 1.0

Select a session for evaluation: Session #2--Sweet Shop Confirm

Sundae Maker - Free

Compatibility: iPhone, iTouch **Category:** Games **Updated:** Jan 24, 2013 **Version:** 1.27 **Size:** 23.0 MB
Language: English **Seller:** Sunstorm Interactive **Rated 4+**

Enter Your ID: Evaluator #1 Evaluation: Ranking Fraud (1) Confirm

Select information for checking Profile Information Confirm

EAT YOUR SUNDAE! SAVE YOUR SUNDAES IN THE FREEZER EMAIL THEM TO FRIENDS
Oh Yes! IT'S FREE
SUNDAE MAKER
BUILD INCREDIBLE SUNDAES! DELICIOUS GROSS GLOWING NUCLEAR?
VISIT THE SUPPLY STORE TO UNLOCK HUNDREDS OF NEW ITEMS AMAZING VARIETY!
MANY SUNDAE MAKING ITEMS INCLUDED! FOR FREE 100s OF ITEMS AVAILABLE IN SUPPLY STORE PACKS!

Description
Try Sundae Maker - the No. 1. rated Ice Cream app!
- Over 100 new items available in content packs ready for PURCHASE in the Supply Store content packs!
Sundae Maker is a HIT! Thanks to everyone who is building awesome sundaes!
Try out the #1 Sundae app on the app store!!!
No. 1 Kids Game
Top 10 Free Games
Top 50 All Games
This free version Includes a sampling of items for building the most delicious, real or wacky ice cream sundaes!
Build the ultimate ice cream sundaes from a huge selection of ingredients!
Read More..

Sundae Maker - Free

Compatibility: iPhone, iTouch **Category:** Games **Updated:** Jan 24, 2013 **Version:** 1.27 **Size:** 23.0 MB
Language: English **Seller:** Sunstorm Interactive **Rated 4+**

Enter Your ID: Evaluator #1 Evaluation: Ranking Fraud (2) Confirm

Select information for checking: Review Information Confirm

App review comments during this session: 2010/6/04 Confirm

Kinda pointless don't even get lik anything to choose from the paid one may cost money but looks way more fun. **2010/6/4 by taym101 (Rating 2)**

I love ice cream... but I'm not a big fan of Sundae Maker. It's okay. But it seems like it's geared more towards very young children. **2010/6/4 by neatknickers (Rating 3)**

I've never been happier to delete an app. Free App a Day? ya right!!! This is actually the crappiest lite version it has ever been my displeasure to download! It gives you around four flavors, four toppings, two cup choices, and four backgrounds. This app only advertises the non lite version and you are hardly able to make a sundae. Why on earth would I buy the non lite version for $1.99 when I could buy More Sundaes from a reputable company like Maverick for 99 cents. I thought this would be a slightly worse More Sundaes that was free. As it turns out, this so called "Sundae Maker" is merely a really cheap looking waste of existence. I would

Enter Your ID: Evaluator #1 Evaluation: Ranking Fraud (2) Confirm

Select information for checking: Ranking Information Confirm

Check the daily App leaderboard during this session: 2010/6/04 Confirm

1	Air Horn (Free!)
2	Mirror in your pocket for FREE
3	Defend Your Castle
4	Glass Tower 2
5	Textfree Unlimited
6	Eye Illusions Free
7	3D Wallpapers &amp; Backgrounds

Figure 10: The screenshots of our ranking fraud evaluation platform.

ranked leading sessions (i.e., most normal sessions) from each data set. Then, we merged all the selected sessions into a pool which consists 114 unique sessions from 84 unique Apps in "Top Free 300" data set, and 128 unique sessions from 71 unique Apps in "Top Paid 300" data set.

Second, we invited five human evaluators who are familiar with Apple's App store and mobile Apps to manually label the selected leading sessions with score 1 (i.e., **Fraud**) and 0 (i.e., **Non-fraud**). Specifically, for each selected leading session, each evaluator gave a proper score by comprehensively considering the profile information of the App (e.g., descriptions, screenshots), the trend of rankings during this session, the App leaderboard information during this session, the trend of ratings during this session, and the user comments during this session. Moreover, they can also download and try the corresponding Apps for obtaining user experiences. Particularly, to facilitate their evaluation, we develop a *Ranking Fraud Evaluation Platform*, which ensures that the evaluators can easily browse all the information. Also, the platform demonstrates each leading session in random orders, which guarantees there is no relationship between leading sessions' order and their fraud scores. Figure 10 shows the screenshot of the platform. The left panel shows the main manu, the right upper panel shows the user comments for the given session, and the right lower panel shows the ranking related information for the given session.

Third, after human evaluation, each leading session s is assigned a fraud score $f(s) \in [0, 5]$. Thus, we can exploit the popular metric Normalized Discounted Cumulative Gain (NDCG) for determining the performance of ranking fraud detection by each approach. Specifically, the discounted cumulative gain given a cut-off rank K can be calculated by

$$DCG@K = \sum_{i=1}^{K} \frac{2^{f(s_i)} - 1}{\log_2(1+i)},$$

where $f(s_i)$ is the human labeled fraud score. The $NDCG@K$ is the $DCG@K$ normalized by the $IDCG@K$, which is the $DCG@K$ value of the ideal ranking list of the returned results, i.e., we have

$$NDCG@K = \frac{DCG@K}{IDCG@K}.$$

$NDCG@K$ indicates how well the rank order of given sessions returned by an approach with a cut-off rank K. The

Table 2: The consistence of human evaluation on Top Free 300 data set.

	Eval. 1	Eval. 2	Eval. 3	Eval. 4	Eval. 5
Eval. 1	43 (71)	37 (62)	39 (57)	40 (62)	39 (59)
Eval. 2	-	48 (66)	43 (56)	44 (64)	44 (59)
Eval. 3	-	-	52 (62)	43 (56)	41 (54)
Eval. 4	-	-	-	47 (67)	44 (55)
Eval. 5	-	-	-	-	50 (64)

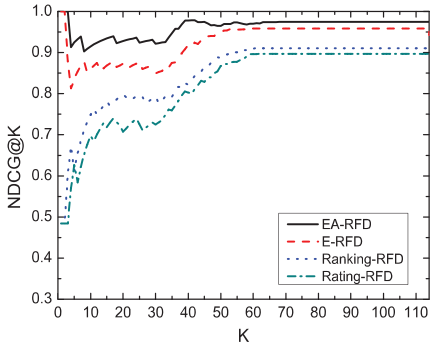
Table 3: The consistence of human evaluation on Top Paid 300 data set.

	Eval. 1	Eval. 2	Eval. 3	Eval. 4	Eval. 5
Eval. 1	53 (75)	45 (66)	48 (61)	44 (64)	46 (64)
Eval. 2	-	57 (71)	53 (58)	51 (66)	53 (63)
Eval. 3	-	-	62 (66)	49 (58)	52 (59)
Eval. 4	-	-	-	55 (73)	46 (62)
Eval. 5	-	-	-	-	59 (69)

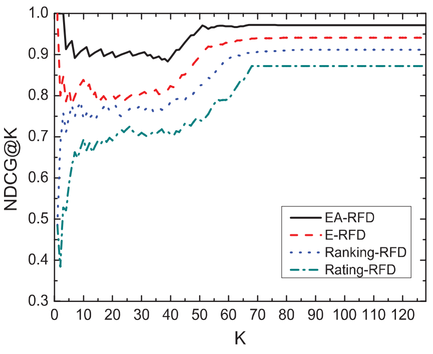
larger $NDCG@K$ value, the better performance of ranking fraud detection.

5.3.3 Overall Performances

Here, we first show the consistence of evaluation results by five human evaluators. Table 2 shows the numbers of fraud and non-fraud (i.e., numbers in brackets) labeled by five evaluators in the diagonal cells, and the number of overlapping fraud and non-fraud (i.e., numbers in brackets) between each pair of evaluators in Top Free 300 data set. Table 3 shows the same results of Top Paid 300 data set. As



(a) Top Free 300 data set



(b) Top Paid 300 data set

Figure 11: The $NDCG@K$ results of each approach.

shown in the tables, these five evaluators are consistent in their judgements of both fraud and non-fraud labels. Specifically, all the five evaluators agree on 32 fraud sessions and 49 non-fraud sessions in Top Free 300 data set, which con-

Table 4: The reported suspicious mobile Apps.

	EA-RFD	E-RFD	Rank.-RFD	Rat.-RFD
Tiny Pets	**2.89%**	3.91%	4.09%	6.88%
Social Girl	**4.41%**	7.42%	6.68%	8.53%
Fluff Friends	**1.17%**	2.67%	3.75%	5.31%
Top Girl	**1.64%**	1.76%	2.08%	6.81%
VIP Poker	**3.25%**	5.73%	5.23%	4.10%
Sweet Shop	**4.23%**	6.82%	8.23%	6.32%
Crime City	**3.12%**	3.67%	5.31%	8.62%

stitute 71.1% of 114 evaluated sessions. Note that, 9 labeled fraud sessions among them are from the external reported suspicious Apps [3, 4], which validates the effectiveness of our human judgement. Similarly, they agree on 39 fraud sessions and 51 non-fraud sessions in Top Paid 300 data set, which constitute 70.3% of 128 evaluated sessions. Moreover, we compute the Cohen's kappa coefficient [1] between each pair of evaluators to estimate the inter-evaluator agreement. Specifically, the values of Cohen's kappa coefficient are between 0.67 to 0.73 in Top Free 300 data set and between 0.70 to 0.73 in Top Paid 300 data set, which indicate the substantial agreement [10].

Figures 11 (a) and 11 (b) show the $NDCG@K$ results of each detection approach in two data sets. First, in these figures, we can see that EA-RFD consistently outperforms other baselines and the improvement is more significant for smaller K. Second, E-RFD outperforms Ranking-RFD and Rating-RFD slightly. This indicates that leveraging two kinds of evidences is more effective than only using one type of evidences, even if without evidence aggregation. Third, by comparing Ranking-RFD and Rating-RFD, we can observe that the ranking based evidences are more effective than rating based evidences. This is because rating manipulation is only a supplementary to ranking manipulation. Finally, EA-RFD outperforms E-RFD, which validates the effectiveness of our evidence aggregation approach.

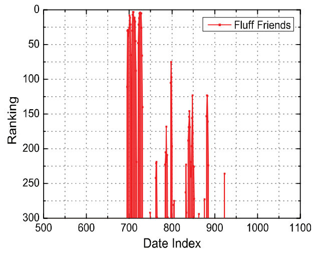
5.4 Case Study: Evaluating App Credibility

As introduced in Section 4, our approach can be used for evaluating the credibility of Apps by Equation 21. Here, we study the performance of evaluating App credibility based on the prior knowledge from existing reports. Specifically, as reported by IBTimes [4], there are eight free Apps which might involve in ranking fraud. In this paper, we use seven of them in our data set (*Tiny Pets*, *Social Girl*, *Fluff Friends*, *Crime City*, *VIP Poker*, *Sweet Shop*, *Top Girl*) for evaluation. Indeed, we try to study whether each approach can find these suspicious Apps with high rankings, since a good ranking fraud detection system should have the capability of capturing these suspicious Apps. Particularly, instead of setting a fixed fraud threshold τ in Equation 21, we treat top 10% ranked leading sessions as suspicious sessions to compute the credibility of each App. Table 4 shows the top percentage position of each App in the ranked list returned by each approach. We can see that our approach EA-RFD can rank those suspicious Apps into higher positions than other baseline methods.

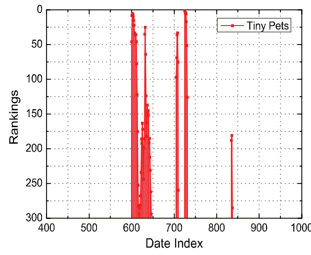
Figure 12 shows the ranking records of the above Apps (Limited by space, we only show two of them). In this figure, we find all these Apps have clear ranking based fraud evidences, which validate the effectiveness of our approach.

5.5 Robustness of Evidence Aggregation

A learning process is required for evidence aggregation. After learning the aggregation model on a historical data set,

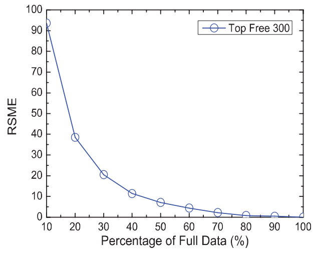


(a) Fluff Friends

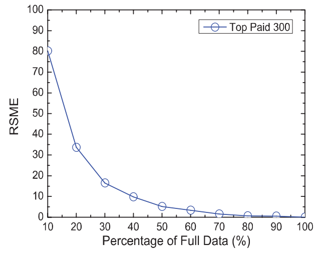


(b) Tiny Pets

Figure 12: The demonstration of the ranking records of two reported suspicious Apps.



(a) Top Free 300 data set



(b) Top Paid 300 data set

Figure 13: Robustness of the aggregation model.

each new test App can reuse this model for detecting ranking fraud. Specifically, the learnt weight parameters (i.e., w_i) in our approach EA-RFD are 0.24 (0.22), 0.30 (0.28), 0.19 (0.18), 0.15 (0.17), and 0.12 (0.15) for each evidence in Top Free 300 (Top Paid 300) data set, respectively. It indicates that ranking based evidences are more effective than rating based evidences. However, it is not clear how many learning data are required? To study this problem and validate the robustness of our approach, we first rank all leading sessions by modeling with weight parameters learnt from the entire data set. Then we also rank all leading sessions by modeling with weight parameters learnt from different segmentation of the entire data set (i.e., 10%,...,100%). Finally, we test the root mean squared error (RMSE) of the ranking of leading sessions between different results. Figure 13 shows the results of robust test on two data sets. We can find that the aggregation model does not need a lot of learning data, thus the robustness of our approach is reasonable.

6. RELATED WORK

Generally speaking, the related works of this study can be grouped into three categories.

The first category is about Web ranking spam detection. Specifically, the Web ranking spam refers to any deliberate actions which bring to selected Web pages an unjustifiable favorable relevance or importance [14]. For example, Ntoulas *et al.* [11] have studied various aspects of content-based spam on the Web and presented a number of heuristic methods for detecting content based spam. Zhou *et al* [14] have studied the problem of unsupervised Web ranking spam detection. Specifically, they proposed an efficient online link spam and term spam detection methods using spamicity. Recently, Spirin *et al.* [13] have reported a survey on Web spam detection, which comprehensively introduces the principles and algorithms in the literature. Indeed, the work of Web ranking spam detection is mainly based on the analysis of ranking principles of search engines, such as PageRank and query term frequency. This is different from ranking fraud detection for mobile Apps.

The second category is focused on detecting online review spam. For example, Lim *et al.* [10] have identified several representative behaviors of review spammers and model these behaviors to detect the spammers. Wu *et al.* [15] have studied the problem of detecting hybrid shilling attacks on rating data. The proposed approach is based on the semi-supervised learning and can be used for trustworthy product recommendation. Xie *et al.* [16] have studied the problem of singleton review spam detection. Specifically, they solved this problem by detecting the co-anomaly patterns in multiple review based time series. Although some of above approaches can be used for anomaly detection from historical rating records, they are not able to extract fraud evidences for a given time period (i.e., leading session).

Finally, the third category includes the studies on mobile App recommendation. For example, Yan *et al.* [17] developed a mobile App recommender system, named Appjoy, which is based on user's App usage records to build a preference matrix instead of using explicit user ratings. Also, to solve the sparsity problem of App usage records, Shi *et al.* [12] studied several recommendation models and proposed a content based collaborative filtering model, named Eigenapp, for recommending Apps in their Web site Getjar. In addition, some researchers studied the problem of exploiting enriched contextual information for mobile App recommendation. For example, Zhu *et al.* [19] proposed a uniform framework for personalized context-aware recommendation, which can integrate both context independency and dependency assumptions. However, to the best of our knowledge, none of previous works has studied the problem of ranking fraud detection for mobile Apps.

7. CONCLUDING REMARKS

In this paper, 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 and rating 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. Experimental results showed the effectiveness of the proposed approach.

Acknowledgement. This work was supported in part by grants from Natural Science Foundation of China (NSFC, Grant No. 61073110 and 71028002), Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20113402110024), the Key Program of National Natural Science Foundation of China (Grant No. 60933013), and National Key Technology Research and Development Program of the Ministry of Science and Technology of China (Grant No. 2012BAH17B03). The work was also partially supported by grants from National Science Foundation (NSF) via grant numbers CCF-1018151 and IIS-1256016.

8. REFERENCES

[1] http://en.wikipedia.org/wiki/cohen's_kappa.

[2] https://developer.apple.com/news/index.php?id=0-2062012a.

[3] http://venturebeat.com/2012/07/03/apples-crackdown-on-app-ranking-manipulation/.

[4] http://www.ibtimes.com/apple-threatens-crackdown-biggest-app-store-ranking-fraud-406764.

[5] Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou. A taxi driving fraud detection system. In *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ICDM '11, pages 181–190, 2011.

[6] D. F. Gleich and L.-h. Lim. Rank aggregation via nuclear norm minimization. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 60–68, 2011.

[7] J. Kivinen and M. K. Warmuth. Additive versus exponentiated gradient updates for linear prediction. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC '95, pages 209–218, 1995.

[8] A. Klementiev, D. Roth, and K. Small. An unsupervised learning algorithm for rank aggregation. In *Proceedings of the 18th European conference on Machine Learning*, ECML '07, pages 616–623, 2007.

[9] A. Klementiev, D. Roth, and K. Small. Unsupervised rank aggregation with distance-based models. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 472–479, 2008.

[10] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 939–948, 2010.

[11] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 83–92, 2006.

[12] K. Shi and K. Ali. Getjar mobile application recommendations with very sparse datasets. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 204–212, 2012.

[13] N. Spirin and J. Han. Survey on web spam detection: principles and algorithms. *SIGKDD Explor. Newsl.*, 13(2):50–64, May 2012.

[14] N. Spirin and J. Han. Survey on web spam detection: principles and algorithms. *SIGKDD Explor. Newsl.*, 13(2):50–64, May 2012.

[15] Z. Wu, J. Wu, J. Cao, and D. Tao. Hysad: a semi-supervised hybrid shilling attack detector for trustworthy product recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 985–993, 2012.

[16] S. Xie, G. Wang, S. Lin, and P. S. Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 823–831, 2012.

[17] B. Yan and G. Chen. Appjoy: personalized mobile application discovery. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, MobiSys '11, pages 113–126, 2011.

[18] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian. Exploiting enriched contextual information for mobile app classification. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 1617–1621, 2012.

[19] H. Zhu, E. Chen, K. Yu, H. Cao, H. Xiong, and J. Tian. Mining personal context-aware preferences for mobile users. In *Proceedings of the IEEE 12th International Conference on Data Mining*, ICDM'12, pages 1212–1217, 2012.