Discovery of Ranking Fraud for Mobile Apps

Hengshu Zhu, Hui Xiong, Senior Member, IEEE, Yong Ge, and Enhong Chen, Senior Member, IEEE

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. Furthermore, we investigate three types of evidences, i.e., ranking based evidences, rating based evidences and review based evidences, by modeling Apps’ ranking, rating and review 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 iOS 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.

Index Terms—Mobile Apps, Ranking Fraud Detection, Evidence Aggregation, Historical Ranking Records, Rating and Review.

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.

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, ratings and reviews in a very short time. For example, an article from VentureBeat [4] 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 [3] in the Apple’s App store.

In the literature, while there are some related work, such as web ranking spam detection [22], [25], [30], online review spam detection [19], [27], [28], and mobile App recommendation [24], [29], [31], [32], 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. Such challenge can be regarded as detecting the local anomaly instead of global anomaly of mobile Apps. 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 scalable 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, which motivates us to discover some implicit fraud patterns of mobile Apps as evidences.

Indeed, our careful observation reveals that mobile Apps are not always 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 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 App developers’ reputation and 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 types of fraud evidences based on Apps’ historical ranking records, and develop three functions to extract such ranking based fraud evidences. Moreover, the ranking based evidences can be affected by App developers’ reputation and 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 types of fraud evidences based on Apps’ historical ranking records, and develop three functions to extract such ranking based fraud evidences.

It is worth noting that all the evidences are extracted by modeling Apps’ ranking, rating and review 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 mobile App data collected from the Apple’s App store for a long time period, i.e., more than two years. 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, rating and review 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 and propose some future research directions.

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, \( R_a = \{r_{a}^{1}, \ldots, r_{a}^{n}\} \), where \( r_{a}^{i} \in \{1, \ldots, K, +\infty\} \) 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 \( r_{a}^{i} \) has, the higher ranking position 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 \( r_{start}^{e} \leq K^{*} < r_{start-1}^{e} \) and \( r_{end}^{e} < K^{*} < r_{end+1}^{e} \). Moreover, \( \forall t_{k} \in (t_{start}^{e}, t_{end}^{e}) \), we have \( r_{a}^{k} \leq K^{*} \).

Note that we apply a ranking threshold \( K^{*} \) which is usually smaller than \( K \) here because \( K^{*} \) may be very big (e.g., more than 1000), and the ranking records beyond \( K^{*} \) (e.g., 300) 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 \( \{e_{1}, \ldots, e_{n}\} \), which satisfies \( t_{start}^{s} = t_{start}^{e_{1}} \leq t_{end}^{e_{1}} = t_{end}^{e_{2}} \leq t_{end}^{e_{2}} \leq t_{end}^{s} \) and there is no other leading session \( s^{*} \) that makes \( T_{s} \subseteq T_{s^{*}} \). Meanwhile, \( \forall i \in [1, n] \), we have \( (t_{start}^{e_{i+1}} - t_{end}^{e_{i}}) \leq \phi \), where \( \phi \) is a predefined time threshold for merging leading events.

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 leading events for constructing leading sessions. Specifically, Algorithm 1 demonstrates the pseudo code of mining leading sessions 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 2: the merging threshold \( \phi \);
Output: the set of \( a \)'s leading sessions \( S_a \);

1. \( E_s = \emptyset \); \( c = \emptyset \); \( s = \emptyset \); \( t_s = 0 \);
2. for each \( i \in [1, |R_a|] \) do
3. if \( t_s^i \leq K' \) and \( t_s^i = 0 \) then
4. \( t_s^i = t_e^i \);
5. else if \( t_s^i > K' \) and \( t_e^i \neq 0 \) then
6. if found one event;
7. \( t_e^i = t_e^i - t_s^i \);
8. if \( E_s = \emptyset \) then
9. \( E_s = e \); \( t_s^i = t_s^i \); \( t_e^i = t_e^i \);
10. else if \( t_s^i - t_e^i < \phi \) then
11. \( E_s = e \); \( t_e^i = t_e^i \);
12. else then
13. if found one session;
14. \( s = c \); \( t_s^i = t_e^i \); \( E_s = \emptyset \);
15. \( S_a \cup s; s = \emptyset \) is a new session;
16. \( E_s = \{ e \}; t_s^i = t_s^i; t_e^i = t_e^i \);
17. begin leading session;
18. return \( S_a \)

In Algorithm 1, we denote each leading event \( e \) and session \( s \) as tuples \( \langle t_s^i, t_e^i \rangle \) and \( \langle t_s^i, t_e^i, E_s \rangle \) 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 7) from the beginning time. For each extracted individual leading event \( e \), we check the time span between \( e \) and the current leading session \( s \) to decide whether they belong to the same leading session based on Definition 2. Particularly, if \( t_s^i - t_e^i < \phi \), \( e \) will be considered as a new leading session (i.e., Step 8 to 16). 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 characteristics 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):

Given a leading event \( e \) of App \( a \) with time range \( [t_s^e, t_e^e] \), where the highest ranking position of \( a \) is \( r_{peak} \), which belongs to \( \Delta R \). The rising phase of \( e \) is a time range \( [t_s^e, t_b^e] \), where \( t_s^e = t_s^e ; t_b^e \in \Delta R \) and \( \forall t_i \in [t_s^e, t_b^e] \) satisfies \( r_i^a \neq \Delta R \). The maintaining phase of \( e \) is a time range \( [t_b^e, t_c^e] \), where \( r_c^e \in \Delta R \) and \( \forall t_i \in [t_b^e, t_c^e] \) satisfies \( r_i^a \neq \Delta R \). The recession phase is a time range \( [t_c^e, t_d^e] \), where \( t_d^e = t_e^e \).

Note that, in Definition 3, \( \Delta R \) is a ranking range to decide the beginning time and the end time of the maintaining phase. \( t_b^e \) and \( t_c^e \) are the first and last time when the App is ranked into \( \Delta R \). 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 find that each App with ranking manipulation always has an expected ranking target (e.g., 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...
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 manipulation will be stopped and the ranking of the malicious App will decrease dramatically. As a result, the manipulation will be stopped and the ranking of the suspicious leading events may contain very short rising phases. Meanwhile, the cost of rising and recession phases. Therefore, the leading event of fraudulent events may contain very short rising phases. Meanwhile, the cost of App stores and the fierce competition between App developers. Therefore, the leading event of fraudulent events often has very short maintaining phase with high ranking positions.

Figure 4 (a) shows an example of ranking records from one of the reported suspicious Apps [5]. 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 lots of honest fans and may attract more and more users to download. Therefore, 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 $\theta_1$ and $\theta_2$ 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

$$
\theta_1 = \arctan \left( \frac{K^*-r_b^e}{\Delta t_b^e - t_a^e} \right), \quad \theta_2 = \arctan \left( \frac{K^*-r_c^e}{\Delta t_c^e - t_d^e} \right),
$$

where $K^*$ is the ranking threshold in Definition 1. Intuitively, a large $\theta_1$ may indicate that the App has been bumped to a high rank within a short time, and a large $\theta_2$ 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 $\theta_1$ and $\theta_2$ values, has higher probability of having ranking fraud. Here, we define a fraud signature $\bar{\theta}_s$ for a leading session as follows.

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

where $|E_s|$ is the number of leading events in session $s$. Intuitively, if a leading session $s$ contains significantly higher $\bar{\theta}_s$ 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 $\bar{\theta}_s$ 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 $\bar{\theta}_s$ of leading session $s$ is not useful for detecting ranking fraud.
- **HYPOTHESIS 1:** The signature $\bar{\theta}_s$ of leading session $s$ is significantly greater than expectation.

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

$$
P(\mathcal{N}(\mu_\theta, \sigma_\theta) \geq \bar{\theta}_s) = 1 - \frac{1}{2} \text{erf} \left( \frac{\bar{\theta}_s - \mu_\theta}{\sigma_\theta \sqrt{2}} \right),
$$

where $\text{erf}(x)$ is the Gaussian Error Function as follows,

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

Intuitively, a leading session with a smaller p-value $P$ 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

$$
\Psi_1(s) = 1 - P(\mathcal{N}(\mu_\theta, \sigma_\theta) \geq \bar{\theta}_s).
$$

**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 $\Delta t_m^e = (t_b^e - t_a^e + 1)$, and the average rank in this maintaining phase as $\tau_m^e$, we can define a fraud signature $\chi_s$ for each leading session as follows

$$
\chi_s = \frac{1}{|E_s|} \sum_{e \in E_s} \frac{K^* - \tau_m^e}{\Delta t_m^e},
$$

where $K^*$ is the ranking threshold in Definition 1. If a leading session contains significantly higher $\chi_s$ 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 $\chi_s$ for each leading session.

- **HYPOTHESIS 0:** The signature $\chi_s$ of leading session $s$ is not useful for detecting ranking fraud.
- **HYPOTHESIS 1:** The signature $\chi_s$ 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 \( \chi_s \) follows the Gaussian distribution, \( \chi_s \sim \mathcal{N}(\mu_\chi, \sigma_\chi) \), where \( \mu_\chi \) and \( \sigma_\chi \) can be learnt by the MLE method from the observations of \( \chi_s \) in all Apps’ historical leading sessions. Then, we can calculate the evidence by

\[
\Psi_2(s) = 1 - \Phi(\mathcal{N}(\mu_\chi, \sigma_\chi) \geq \chi_s).
\]  

**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 larger 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 \( \lambda \) 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,

\[
\mathbb{P}(\mathcal{P}(\lambda_s) \geq |E_s|) = 1 - e^{-\lambda_s} \sum_{i=0}^{\lfloor |E_s| \rfloor} \frac{(\lambda_s)^i}{i!}.
\]  

Therefore, we can compute the evidence by

\[
\Psi_3(s) = 1 - \Phi(\mathcal{P}(\lambda_s) \geq |E_s|).
\]  

Intuitively, 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 contains 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 \( \theta_1 \) 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. To solve this issue, we also study how to extract fraud evidences from Apps’ historical rating records.

**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 \( \Delta R_s \) for each leading session as follows,

\[
\Delta R_s = \frac{\overline{R}_s - \overline{R}_a}{\overline{R}_a}, \quad (s \in a)
\]  

where \( \overline{R}_s \) is the average rating in leading session \( s \), and \( \overline{R}_a \) is the average historical rating of App \( a \). Therefore, if a leading session has significantly higher value of \( \Delta R_s \) 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 \( \Delta R_s \) for each leading session as follows.

- **HYPOTHESIS 0:** The signature \( \Delta R_s \) of leading session \( s \) is not useful for detecting ranking fraud.
- **HYPOTHESIS 1:** The signature \( \Delta R_s \) 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 \( \Delta R_s \) follows the Gaussian distribution, \( \Delta R_s \sim \mathcal{N}(\mu_\Delta, \sigma_\Delta) \), where \( \mu_\Delta \) and \( \sigma_\Delta \) can be learnt by the MLE method from the observations of \( \Delta R_s \) in all Apps’.
historical leading sessions. Then, we can compute the evidence by
\[ \Psi_4(s) = 1 - \mathbb{P}(N(\mu_R, \sigma_R) \geq \Delta R_s). \] (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’s leading session \(s_i\), \(p(l_i|R_s, a)\), should be consistent with the distribution in a’s historical rating records, \(p(l_i|R_a)\), and vice versa. Specifically, we can compute the distribution by \(p(l_i|R_s, a) = \left( \frac{N^s_i}{N_s} \right)\), where \(N^s_i\) 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 \(p(l_i|R_a)\) in a similar way. Then, we use the Cosine similarity between \(p(l_i|R_s, a)\) and \(p(l_i|R_a)\) to estimate the difference as follows.
\[ D(s) = \frac{\sum_{i=1}^{L} p(l_i|R_s, a) \times p(l_i|R_a)}{\sqrt{\sum_{i=1}^{L} p(l_i|R_s, a)^2 \times \sum_{i=1}^{L} p(l_i|R_a)^2}}. \] (12)

Therefore, if a leading session has significantly lower value of \(D(s)\) 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 \(D(s)\) for each leading session as follows.

\(\triangleright\) **HYPOTHESIS 0:** The signature \(D(s)\) of leading session \(s\) is not useful for detecting ranking fraud.

\(\triangleright\) **HYPOTHESIS 1:** The signature \(D(s)\) 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 \(D(s)\) follows the Gaussian distribution, \(D(s) \sim N(\mu_D, \sigma_D)\), where \(\mu_D\) and \(\sigma_D\) can be learnt by the MLE method from the observations of \(D(s)\) in all Apps’ historical leading sessions. Then, we can compute the evidence by
\[ \Psi_5(s) = 1 - \mathbb{P}(N(\mu_D, \sigma_D) \leq D(s)). \] (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 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. Specifically, before downloading or purchasing a new mobile App, users often firstly 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 leaderboard. 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. To this end, here we propose two fraud evidences based on Apps’ review behaviors in leading sessions for detecting ranking fraud.

**EVIDENCE 6.** Indeed, most of the the review manipulations are implemented by bot farms due to the high cost of human resource. Therefore, review spammers often post multiple duplicate or near-duplicate reviews on the same App to inflate downloads [19], [21]. In contrast, the normal App always have diversified reviews since users have different personal perceptions and usage experiences. Based on the above observations, here we define a fraud signature \(Sim(s)\), which denotes the average mutual similarity between the reviews within leading session \(s\). Specifically, this fraud signature can be computed by following steps.

First, for each review \(c_i\) in leading session \(s\), we remove all stop words (e.g., “of”, “the”) and normalize verbs and adjectives (e.g., “plays” → “play”, “better” → “good”).

Second, we build a normalized words vector \(\overline{w_{c_i}}\) for each review \(c_i\), where \(n\) indicates the number of all unique normalized words in all reviews of \(s\). To be specific, here we have \(dim[n] = \sum_{i=1}^{freq_c} 1 \leq i \leq n\), where \(freq_c\) is the frequency of the \(i\)-th word in \(c\).

Finally, we can calculate the similarity between two reviews \(c_i\) and \(c_j\) by the Cosine similarity \(\text{Cos}(\overline{w_{c_i}}, \overline{w_{c_j}})\). Thus, the fraud signature \(Sim(s)\) can be computed by
\[ Sim(s) = \frac{2 \times \sum_{1 \leq i < j \leq N_s} \text{Cos}(\overline{w_{c_i}}, \overline{w_{c_j}})}{N_s \times (N_s - 1)}. \] (14)

where \(N_s\) is the number of reviews during leading session \(s\). Intuitively, the higher value of \(Sim(s)\) indicates more duplicate/near-duplicate reviews in \(s\). Thus, if a leading session has significantly higher value of \(Sim(s)\) 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 \(Sim(s)\) for each leading session as follows.

\(\triangleright\) **HYPOTHESIS 0:** The signature \(Sim(s)\) of leading session \(s\) is not useful for detecting ranking fraud.

\(\triangleright\) **HYPOTHESIS 1:** The signature \(Sim(s)\) of leading session \(s\) is significantly higher than expectation.

Here, we use the Gaussian approximation to compute the \(p\)-value with the above hypotheses. Specifically, we assume \(Sim(s)\) follows the Gaussian distribution, \(Sim(s) \sim N(\mu_{Sim}, \sigma_{Sim})\), where \(\mu_{Sim}\) and \(\sigma_{Sim}\) can be learnt by the MLE method from the observations of \(Sim(s)\) in all Apps’ historical leading sessions. Then, we can compute the evidence by
\[ \Psi_6(s) = 1 - \mathbb{P}(N(\mu_{Sim}, \sigma_{Sim}) \geq Sim(s)). \] (15)
EVIDENCE 7. From the real-world observations, we find that each review $c$ is always associated with a specific latent topic $z$. For example, some reviews may be related to the latent topic “worth to play” while some may be related to the latent topic “very boring”. Meanwhile, since different users have different personal preferences of mobile Apps, each App $a$ may have different topic distributions in their historical review records. Intuitively, the topic distribution of reviews in a normal leading session $s$ of App $a$, i.e., $p(z|s)$, should be consistent with the topic distribution in all historical review records of $a$, i.e., $p(z|a)$. It is because that the review topics are based on the users’ personal usage experiences but not the popularity of mobile Apps. In contrast, if the reviews of $s$ have been manipulated, the two topic distributions will be markedly different. For example, there may contain more positive topics, such as “worth to play” and “popular”, in the leading session.

In this paper, we propose to leverage topic modeling to extract the latent topics of reviews. Specifically, here we adopt the widely used Latent Dirichlet Allocation (LDA) model [9] for learning latent semantic topics. To be more specific, the historical reviews of a mobile App $a$ is assumed to be generated as follows. First, before generating $C_a$, $K$ prior conditional distributions of words given latent topics $\phi_z$ are generated from a prior Dirichlet distribution $\alpha$. Second, a prior latent topic distribution $\theta_a$ is generated from a prior Dirichlet distribution $\alpha$ for each mobile App $a$. Then, for generating the $j$-th word in $C_a$ denoted as $w_{a,j}$, the model firstly generates a latent topic $z$ from $\theta_a$ and then generates $w_{a,j}$ from $\phi_z$. The training process of LDA model is to learn proper latent variables $\theta = \{P(z|C_a)\}$ and $\phi = \{P(w|z)\}$ for maximizing the posterior distribution of review observations, i.e., $P(C_a|\alpha, \beta, \theta, \phi)$. In this paper, we use a Markov chain Monte Carlo method named Gibbs sampling [12] for training LDA model. If we denote the reviews in leading session $s$ of $a$ as $C_{s,a}$, we can use the KL-divergence to estimate the difference of topic distributions between $C_a$ and $C_{s,a}$.

$$D_{KL}(s||a) = \sum_k P(z_k|C_s,a) \ln \frac{P(z_k|C_s,a)}{P(z_k|C_a)},$$  \hspace{1cm} (16)

where $P(z_k|C_a)$ and $P(z_k|C_s,a) \propto P(z_k) \prod_{w \in C_s,a} P(w|z_k)$ can be obtained through the LDA training process. The higher value of $D_{KL}(s||a)$ indicates the higher difference of topic distributions between $C_a$ and $C_{s,a}$. Therefore, if a leading session has significantly higher value of $D_{KL}(s||a)$ 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 $D_{KL}(s||a)$ for each leading session as follows.

- **HYPOTHESIS 0:** The signature $D_{KL}(s||a)$ of leading session $s$ is not useful for detecting ranking fraud.
- **HYPOTHESIS 1:** The signature $D_{KL}(s||a)$ of leading session $s$ is significantly higher than expectation.

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

$$\Psi_1(s) = 1 - P(N(\mu_{DL}, \sigma_{DL}) \geq D_{KL}(s||a)).$$  \hspace{1cm} (17)

The values of two evidences $\Psi_0(s)$ and $\Psi_1(s)$ 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.

3.4 Evidence Aggregation

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 [17], [18], score based models [11], [26] and Dempster-Shafer rules [10], [23]. 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.

Specifically, we define the final evidence score $\Psi^*(s)$ as a linear combination of all the existing evidences as Equation 18. Note that, here we propose to use the linear combination because it has been proven to be effective and is widely used in relevant domains, such as ranking aggregation [16], [20].

$$\Psi^*(s) = \sum_{i=1}^{N_s} w_i \times \Psi_i(s), \hspace{0.5cm} s.t. \sum_{i=1}^{N_s} w_i = 1,$$  \hspace{1cm} (18)

where $N_s = 7$ is the number of evidences, and weight $w_i \in [0, 1]$ is the aggregation parameter of evidence $\Psi_i(s)$. Thus, the problem of evidence aggregation becomes how to learn the proper parameters $\{w_i\}$ from the training leading sessions.

We first propose an intuitive assumption as **Principle 1** for our evidence aggregation approach. Specifically, we assume that effective evidences should have similar evidence scores for each leading session, while poor evidences will generate different scores from others. 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. To this end, for each evidence score $\Psi_i(s)$, we can measure its consistency using the variance-like measure

$$\sigma_i(s) = (\Psi_i(s) - \overline{\Psi}(s))^2,$$  \hspace{1cm} (19)

where $\overline{\Psi}(s)$ is the average evidence score of leading session $s$ obtained from all $N_s$ evidences. 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

\[
\arg\min_w \sum_{a \in A} \sum_{s \in a} N_q \cdot w_i \cdot \sigma_i(s), \tag{20}
\]

\[
\text{s.t.} \sum_{i=1}^{N_q} w_i = 1; \forall w_i \geq 0. \tag{21}
\]

In this paper, we exploit the gradient based approach with exponentiated updating \([15], [16]\) to solve this problem. To be specific, we first assign \( w_i = \frac{1}{N_q} \) as the initial value, then for each \( s \), we can compute the gradient by,

\[
\nabla_i = \frac{\partial w_i \cdot \sigma_i(s)}{\partial w_i} = \sigma_i(s). \tag{22}
\]

Thus, we can update the weight \( w_i \) by

\[
w_i = \frac{w_i^* \times \exp(-\lambda \nabla_i)}{\sum_{j=1}^{N_q} w_j^* \times \exp(-\lambda \nabla_j)}, \tag{23}
\]

where \( w_i^* \) is the last updated weight value \( w_i \), and \( \lambda \) is the learning rate, which is empirically set \( \lambda = 10^{-2} \) in our experiments.

Finally, we can exploit Equation (18) to estimate the final evidence score of each leading session. Moreover, given a leading session \( s \) with a predefined threshold \( \tau \), we can determine that \( s \) has ranking fraud if \( \Psi^*(s) > \tau \).

However, sometimes only using evidence scores for evidence aggregation is not appropriate. It is because different evidences may have different score range to evaluate leading sessions. For example, some evidences may always generate higher scores for leading sessions than the average evidence score, although they can detect fraudulent leading sessions and rank them in accurate positions.

Therefore, here we propose another assumption as Principle 2 for our evidence aggregation approach. Specifically, we assume that effective evidences should rank leading sessions from a similar conditional distribution, while poor evidences will lead to a more uniformly random ranking distribution \([16]\). To this end, given a set of leading sessions, we first rank them by each evidence score and obtain \( N_q \) 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

\[
\bar{\pi}(s) = \frac{1}{N_q} \sum_{i=1}^{N_q} \pi_i(s). \tag{24}
\]

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

\[
\sigma_i^2(s) = (\pi_i(s) - \bar{\pi}(s))^2. \tag{25}
\]

If \( \sigma_i^2(s) \) is small, the corresponding \( \Psi_i(s) \) should be given a bigger weight and vice versa. Then we can replace \( \sigma_i(s) \) by \( \sigma_i^*(s) \) in Equation 20, and exploit similar gradient based approach that is introduced above for learning the weights of evidences.

4 Discussion

Here, 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, rating and review 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 and App developers’ reputation.

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 \( r_{\text{now}}^n \) of an App \( a \), we can detect ranking fraud for it in two different cases. First, if \( r_{\text{now}}^n > K^* \), 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 \( r_{\text{now}}^n < K^* \), which means \( a \) is in a new leading event \( e \), we treat this case as a special case that \( r_{\text{end}}^e = r_{\text{now}}^n \) and \( \theta_o = 0 \). Therefore, such real-time ranking frauds 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, our approach can discover the local anomaly instead of the global anomaly of mobile Apps. Thus, we should take consideration of such kind of local characteristics when estimating the credibility of Apps. To be specific, we define an App fraud score \( F(a) \) for each App \( a \) according to how many leading sessions of \( a \) contain ranking fraud.

\[
F(a) = \sum_{s \in a} [\Psi^*(s) > \tau] \times \Psi^*(s) \times \Delta t^*, \tag{26}
\]

where \( s \in a \) denotes that \( s \) is a leading session of App \( a \), and \( \Psi^*(s) \) is the final evidence score of leading session \( s \) that can be calculated by Equation 18. In particular, we define a signal function \([x] \) (i.e., [x] = 1 if \( x = \text{True} \), and 0 otherwise) and a fraud threshold \( \tau \) to decide the top \( k \) fraudulent leading sessions. Moreover, \( \Delta t^* = (t_{\text{end}}^e - t_{\text{start}}^e + 1) \) is the time range of \( s \), which indicates the duration of ranking fraud. Intuitively, an App contains more leading sessions, which have high fraud evidence scores and long time duration, will have higher App fraud scores.
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 of top 300 free Apps and top 300 paid Apps, respectively. Moreover, each data set also contains the user ratings and review information. Table 1 shows the detailed data characteristics of our data sets.

<table>
<thead>
<tr>
<th></th>
<th>Top Free 300</th>
<th>Top Paid 300</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Num.</td>
<td>9,784</td>
<td>5,261</td>
</tr>
<tr>
<td>Ranking Num.</td>
<td>285,900</td>
<td>285,900</td>
</tr>
<tr>
<td>Avg. Ranking Num.</td>
<td>29.22</td>
<td>54.34</td>
</tr>
<tr>
<td>Rating Num.</td>
<td>14,912,459</td>
<td>4,561,943</td>
</tr>
<tr>
<td>Avg. Rating Num.</td>
<td>1,524.17</td>
<td>867.12</td>
</tr>
</tbody>
</table>

Figures 6 (a) and 6 (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 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 7 (a) and 7 (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.

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

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 8 and Figure 9 show the distributions 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 10 (a) and 10 (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 $\Psi_3$. 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 four intuitive baselines and invite five human evaluators to validate the effectiveness of our approach EA-RFD (Evidence Aggregation based Ranking Fraud Detection). Particularly, we denote our approach with score based aggregation (i.e., Principle 1) as EA-RFD-1, and our approach with rank based aggregation (i.e., Principle 2) as EA-RFD-2, respectively.
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., $\Psi_1$ to $\Psi_3$). These three evidences are integrated by our aggregation approach.

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., $\Psi_4$ and $\Psi_5$). These two evidences are integrated by our aggregation approach.

The third baseline Review-RFD stands for Review evidence based Ranking Fraud Detection, which estimates the ranking fraud for each leading session by only using review based evidences (i.e., $\Psi_6$ and $\Psi_7$). These two evidences are integrated by our aggregation approach.

Particularly, here we only use the rank based aggregation approach (i.e., Principle 2) for integrating evidences in above baselines. It is because that these baselines are mainly used for evaluating the effectiveness of different kinds of evidences, and our preliminary experiments validated that baselines with Principle 2 always outperform baselines with Principle 1.

The last baseline E-RFD stands for Evidence based Ranking Fraud Detection, which estimates the ranking fraud for each leading session by ranking, rating and review based evidences without evidence aggregation. Specifically, it ranks leading sessions by Equation 18, where each $w_i$ is set to be 1/7 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-1, EA-RFD-2, 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. Furthermore, we use the LDA model to extract review topics as introduced in Section 3.3. Particularly, we first normalize each review by the Stop-Words Remover [6] and the Porter Stemmer [7]. Then, the number of latent topic $K_z$ is set to 20 according to the perplexity based estimation approach [8], [31]. Two parameters $\alpha$ and $\beta$ for training LDA model are set to be $50/K$ and 0.1 according to [13].
the inter-evaluator agreement. The values of Cohen’s kappa coefficient are between 0.66 to 0.72 in the user evaluation. This indicates the substantial agreement [19].

Finally, we further ranked the leading sessions by each approach with respect to their fraudulent scores, and obtained six ranked lists of leading sessions. In particular, if we treat the commonly agreed fraud sessions (i.e., 89 sessions in Top Free 300 data set, 94 sessions in Top Paid 300 data set) as the ground truth, we can evaluate each approach with three widely-used metrics, namely Precision@K, Recall@K, F@K [2]. Also, we can exploit the metric Normalized Discounted Cumulative Gain (NDCG) for determining the ranking performance of each approach. Specifically, the discounted cumulative gain given a cut-off rank $K$ can be calculated by $DCG@K = \sum_{i=1}^{K} \frac{f(s_i)}{\log_2(1+i)}$, 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 larger $NDCG@K$ value, the better performance of ranking fraud detection.

### 5.3.3 Overall Performances

In this subsection, we present the overall performances of each ranking fraud detection approach with respect to different evaluation metrics, i.e., Precision@K, Recall@K, F@K, and NDCG@K. Particularly, here we set the maximum $K$ to be 200, and all experiments are conducted on a 2.8GHz × 2 quad-core CPU, 4G main memory PC.

Figures 12 and Figures 13 show the evaluation performance of each detection approach in two data sets. From these figures we can observe that the evaluation results in two data sets are consistent. Indeed, by analyzing the evaluation results, we can obtain several insightful observations. Specifically, first, we find that our approach, i.e., EA-RFD-2/EA-RFD-2, consistently outperforms other baselines and the improvements are more significant for smaller $K$ (e.g., $K < 100$). This result clearly validates the effectiveness of our evidence aggregation based framework for detecting ranking fraud.

Second, EA-RFD-2 outperforms EA-RFD-1 sightly in terms of all evaluation metrics, which indicates that rank based aggregation (i.e., Principle 2) is more effective than score based aggregation (i.e., Principle 1) for integrating fraud evidences. Third, our approach consistently outperforms E-RFD, which validates the effectiveness of evidence aggregation for detecting ranking fraud. Fourth, E-RFD have better detection performance than Ranking-RFD, Rating-RFD and Review-RFD. This indicates that leveraging three kinds of evidences is more effective than only using one type of evidences, even if without evidence aggregation.

Finally, by comparing Ranking-RFD, Rating-RFD and Review-RFD, we can observe that the ranking based evidences are more effective than rating and review based evidences. It is because rating and review manipulations are only supplementary to ranking manipulation. Particularly, we observe that Review-RFD may not be able to lead to the good performance in terms of all evaluation metrics on the two data sets. A possible reason behind this phenomenon is that review manipulation (i.e., fake-positive reviews) does not directly affect the chart ranking of Apps, but may increase the possibility of inflating App downloads and ratings. Therefore, the review manipulation does not necessarily result in ranking fraud due to the unknown ranking principles in the App Store. However, the proposed review based evidences can be helpful as supplementary for ranking fraud detection. Actually, in our preliminary experiments, we found that the review based evidences could always improve the detection performances while being used together with other evidences. This clearly validates the effectiveness of the review based evidences.
Fig. 14. Case study of reported suspicious mobile Apps.

To further validate the experimental results, we also conduct a series of paired T-test of 0.95 confidence level which show that the improvements of our approach, i.e., EA-RFD-2/EA-RFD-1, on all evaluation metrics with different $K$ compared to other baselines are all statistically significant.

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 26. Here, we study the performance of evaluating App credibility based on the prior knowledge from existing reports. Specifically, as reported by IBTimes [5], 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 $\tau$ in Equation 26, we treat top 10% ranked leading sessions as suspicious sessions to compute the credibility of each App.

Figure 14 shows the top percentage position of each App in the ranked list returned by each approach. We can see that our approach, i.e., EA-RFD-2 and EA-RFD-1, can rank those suspicious Apps into higher positions than other baseline methods. Similarly as the results in Section 5.3.3, only leveraging single kind of evidences for fraud detection cannot obtain good performance, i.e., finding such suspicious Apps in high positions.

Figure 15 shows the ranking records of the above Apps (limited by space, we only show four of them). In this figure, we find all these Apps have clear ranking based fraud evidences. For example, some Apps have very short leading sessions with high rankings (i.e., Evidence 1 and 2), and some Apps have leading session with many leading events (i.e., Evidence 3). These observations clearly validate the effectiveness of our approach.

5.5 Efficiency and Robustness of our Approach

The computational cost of our approach majorly comes from the task of extracting three kinds of fraud evidences for the given leading sessions. Indeed, the main processes of this task can be calculated offline in advance. For example, the LDA model can be trained offline and the fraud signatures of the existing leading sessions can also be mined in advance and stored in the server. In this case, the process of extracting evidences for each leading session will be very fast (less than 100 millisecond on average in our experiments).

Meanwhile, a learning process is required for evidence aggregation. After learning the aggregation model on a historical data set, each new test App can reuse this model for detecting ranking fraud. However, it is still 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 16 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 [30]. For example, Ntoulas et al. [22] 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. [30] have studied the problem of supervised Web ranking spam detection. Specifically, they proposed an efficient online link spam and term spam detection methods using spamicity. Recently, Spirin et al. [25] 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. [19] have identified several representative behaviors of review spammers and model these behaviors to detect the spammers. Wu et al. [27] 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. [28] 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 and review 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. [29] developed a mobile App recommender system, named Appjoi, 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. [24] 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. [32] 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.

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.

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.

Acknowledgement. This work was supported in part by grants from National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010), Natural Science Foundation of China (NSFC, Grant No.71329201), National High Technology Research and Development Program of China (Grant No. 85214AA012303), Science and Technology Development of Anhui Province (Grants No. 1301022064), the International Science and Technology Cooperation Plan of Anhui Province (Grant No. 1303063008). This work was also partially supported by grants from National Science Foundation (NSF) via grant numbers CCF-1018151 and IIS-1256016.

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, rating based evidences and review based evidences for detecting ranking fraud.

REFERENCES

Hengshu Zhu is currently a Ph.D. student in the School of Computer Science and Technology at University of Science and Technology of China (USTC). He was supported by the China Scholarship Council (CSC) as a visiting research student at Rutgers, the State University of New Jersey, USA, for more than one year. He received his B.E. degree in Computer Science from USTC, China, in 2009. His main research interests include mobile data mining, recommender systems, and social networks. During his Ph.D. study, he received the KSEM-2011 and WAIM-2013 Best Student Paper Award. He has published a number of papers in refereed journals and conference proceedings, such as IEEE TMC, ACM TIST, WWW Journal, KAIS, ACM CIKM, and IEEE ICDM. He also has served as a reviewer for numerous journals, such as IEEE TSE, KDD, ICDM, ACM RecSys, and ACM Transactions on Data and Knowledge Engineering. He is a member of the IEEE.

Yong Ge received his Ph.D. in Information Technology from Rutgers, The State University of New Jersey in 2013, the M.S. degree in Signal and Information Processing from the University of Science and Technology of China (USTC) in 2008, and the B.E. degree in Information Engineering from Xi’an Jiao Tong University in 2005. He is currently an Assistant Professor at the University of North Carolina at Charlotte. His research interests include data mining and business analytics. He received the ICDM-2011 Best Research Paper Award, Excellence in Academic Research (one per school) at Rutgers Business School in 2013, and the Dissertation Fellowship at Rutgers University in 2012. He has published prolifically in refereed journals and conference proceedings, such as IEEE TKDE, ACM TOIS, ACM TKDD, ACM TIST, ACM CIKM, SIAM SDM, IEEE ICDM, and ACM RecSys. He has served as Program Committee members at the ACM CIKM 2013, the International Conference on Web-Age Information Management 2013, and IEEE ICDM 2013. Also he has served as a reviewer for numerous journals, including IEEE TKDE, ACM TIST, KAIS, Information Science, and TSMC-B.

Enhong Chen is currently a Professor and Vice Dean of School of Computer Science, Vice Director of the National Engineering Laboratory for Speech and Language Information Processing, and Dean of School of Information and Knowledge Engineering at University of Science and Technology of China (USTC), winner of the National Science Fund for Distinguished Young Scholars of China. He received the B.S. degree form Anhui University, Master degree from Hefei University of Technology and Ph.D degree in computer science from USTC.

His research interests include data mining and machine learning, social network analysis and recommender systems. He has published lots of papers on refereed journals and conferences, including TKDE, TMC, KDD, ICDM, NIPS, and CIKM. He has served on program committees of numerous conferences including KDD, ICDM, SDM. He received the Best Application Paper Award on KDD-2008 and Best Research Paper Award on ICDM-2011. He is a senior member of the IEEE.