Volume 6, Issue 3, March 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Fraud Detection & Prevention of Mobile Apps using Optimal Aggregation Method

Vivek Pingale, Laxman Kuhile, Pratik Phapale, Pratik Sapkal, Prof. Swati Jaiswal
SKNSITS, Lonavala, Maharashtra,
India

Abstract: ***The number of mobile Apps has grown at a huge rate over the past few years. Ranking fraud in the mobile App market refers to fraudulent or fake activities which have a purpose of strike up the Apps in the popularity list. It becomes more and more frequent for App developers to posting bogus 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 brief 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 by using mining leading session algorithm. Furthermore, we investigate three types of evidences, i.e., ranking based evidences, rating based evidences and review based evidences, by studying historical records. we used an optimal aggregation method to integrate all the evidences for fraud detection. Finally, we evaluate the proposed system with real-world App data collected from the Google 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.***

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

I. INTRODUCTION

The number of mobile Apps has grown rapidly over the past few years. For example, as of the end of 2014, there are more than 13 million Apps at Google Play. To stimulate the development of mobile Apps, many App stores launched daily App leaderboards, which demonstrate the chart rankings ofmost 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 advertisement 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, some App developers resort to some fraudulent means to deliberately boost their Apps and 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 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. the literature, while there are some related work, such as web ranking spam detection [7] [8] [9], online review spam detection [10][11][12], and mobile App recommendation [13] [14][15][16], 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. In this paper, we provide a brief 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 by using mining leading session algorithm. Such leading sessions can be useful 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 by analyzing its historical records. We propose an optimization based aggregation method to integrate all the evidences for fraud detection.The II part indicates the literature survey of all the papers, III indicates the existing system along with the drawbacks, IV part includes the proposed system with the architecture and android framework and V part represents the conclusion of the paper.

II. LITERATURE SURVEY

D. M. Blei, A. Y. Ng, and M. I. Jordan[1], describes the latent Dirichlet allocation (LDA), a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document. We present efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation. We report results in document modeling, text classification, and collaborative filtering, comparing to a mixture of unigrams model and the probabilistic LSI model.

Y. Ge, H. Xiong, C. Liu, and Z.-H. Zhou [2], explains the advances in GPS tracking technology have enabled us to install GPS tracking devices in city taxis to collect a large amount of GPS traces under operational timeconstraints. These GPS traces provide unparallel opportunities for us to uncover taxi driving fraud activities. In this paper, the author developed a taxi driving fraud detection system, which is able to systematically investigate taxi driving fraud. In this system, we first provide functions to find two aspects of evidences: travel route evidence and driving distance evidence. Furthermore,a third function is designed to combine the two aspects of evidences based on Dempster-Shafer theory. To implement the system, we first identify interesting sites from a large amount of taxi GPS logs. Then, a parameter-free method to mine the travel route evidences. Also the introduction of route mark is used to represent a typical driving path from an interesting site to another one.

D. F. Gleich and L.-h. Lim[3], describes the process of rank aggregation is intimately intertwined with the structure of skew-symmetric matrices. The author applied a recent advances in the theory and algorithms of matrix completion to skew-symmetric matrices. This combination of ideas produces a new method for ranking a set of items. The essence of our idea is that a rank aggregation describes a partially filled skew-symmetric matrix. The authors extended an algorithm for matrix completion to handle skew-symmetric data and use that to extract ranks for each item. Our algorithm applies to both pair-wise comparison and rating data. Because it is based on matrix completion, it is robust to both noise and incomplete data. We show a formal recovery result for the noiseless case and present a detailed study of the algorithm on synthetic data and Netflix ratings.

A. Klementiev, D. Roth, K. Small, and I. Titov[4], provides the information that many applications in information retrieval, natural language processing, data mining, and related fields require a ranking of instances with respect to a specified criteria as opposed to a classification. Furthermore, for many such problems, multiple established ranking models have been well studied and it is desirable to combine their results into a joint ranking, a formalism denoted as rank aggregation. This work presents a novel unsupervised learning algorithm for rank aggregation(ULARA) which returns a linear combination of the individual ranking functions based on the principle of rewarding ordering agreement between the rankers.

A. Klementiev, D. Roth, and K. Small[5], explains the need to meaningfully combine sets of rankings often comes up when one deals with ranked data. Although a number of heuristic and supervised learning approaches to rank aggregation exist, they require domain knowledge or supervised ranked data, both of which are expensive to acquire. In order to address these limitations, they propose a mathematical and algorithmic framework for learning to aggregate (partial) rankings without supervision. The framework for the cases of combining permutations and combining top-k lists, and propose a novel metric for the latter. Experiments in both scenarios demonstrate the effectiveness of the proposed formalism.

III. EXISTING SYSTEM

Millions of mobile Apps has grown at a huge rate over the past few years. Many App stores launched daily App leaderboards, which demo the rankings of most popular Apps. A higher rank on the leaderboard usually leads to a huge number of downloads and million dollars in revenue, instead of relying on traditional marketing solutions. Some App developers resort to some fake means to willfully 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, rating and reviews in a very short time.

Some algorithms are available for detecting harmful mobile apps such as Stateless model checking of Event-Drive application, viceroy algorithm but they are not too effective to solve a problem.

Algorithm for detecting of fraud in android apps is available for example mining leading sessions but they only detect some fraudulent activities.

IV. PROPOSED SYSTEM

In proposed system we overcome the drawbacks of Mining leading session algorithm which is based on ranking, review & rating. 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 download and ratings in a very short time. However, the instant download information of each mob. 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, the focus is 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 evidences based on the download information and App developers' reputation. Second, the proposed approach can detect ranking fraud happened in A,' historical leading sessions.

Ranking fraud detection in mobile apps is actually to detect ranking fraud within leading session of mobile apps. Specifically we identified first leading sessions based on Apps historical ranking records. Then with the analysis of Apps' ranking behaviours we characterized some fraud evidences from historical records.

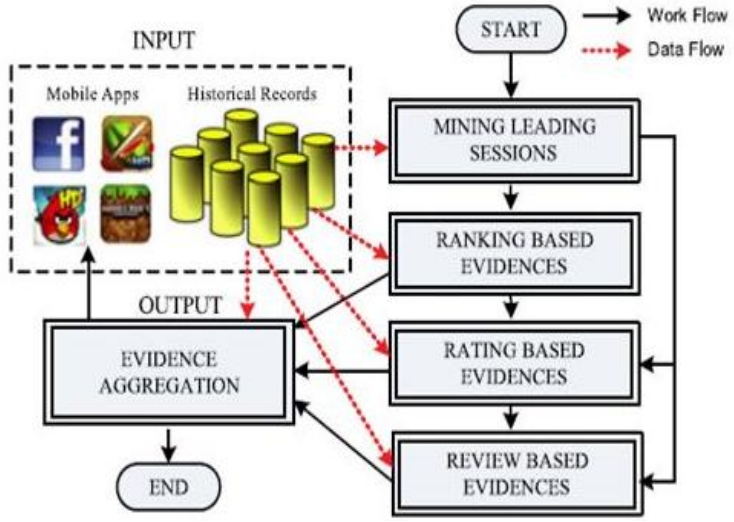

Fig1: System architecture

The ranking based evidences can be affected some Apps' developer reputation and some legitimate marketing campaigns, such as "limited-time discount". This method is not enough to detect fraudulent Apps' so we propose two new methods of fraud evidences based on Apps' historical rating and review records. Additionally, we developed an unsupervised evidence-aggregation method to integrate these types of evidences.

V. MINING LEADING SESSION

There are two main steps for mining leading sessions. First, 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 miningleading sessions for a given App a.

Algorithm1 Mining leading sessions reffered from Discovery of Ranking Fraud for Mobile Apps Hengshu Zhu, HuiXiong, Senior Member, IEEE, Yong Ge, and Enhong Chen, Senior Member, IEEE

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

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

In Algorithm 1, we denote each leading event e and session s as tuples <te start; te end > and <ts start; ts end; Es> respectively, where Es 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 sessions to decide whether they belong to the same leading session based on Definition 2. Particularly, if ðte start _ tsendÞ< f, 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 16). Thus, this algorithm can identify leading events and sessions by scanning a's historical ranking records only once.

The drawbacks of the existing system are as follows:

1. When an App was promoted with the help of ranking manipulation it could be top in leaderboard and more new users could be purchased that product.
2. Affect other App reputation.
3. There is not any method exist to detect and block fraudulent apps.

VI. EXTRACTING EVIDENCE FOR RANKING FRAUD DETECTION

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

VII. EVIDENCE AGGREGATION ALGORITHM

1. Analyze the historical records of mobile apps.
2. Differentiate the evidences as Ranking based, Rating based, Review based.
3. Aggregate these evidences by using optimal aggregation algorithm.
4. Design Android application framework

Step1: Analyzing of historical records is nothing but obtaining the app related information from google play store and apple store. Historical records consist Rank of the applications in leaderboard, Rating given by users to apps, different reviews of different types of users, no of downloads off the apps.

VIII. 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, *risingphase,maintaining phase* and *recession phase.* Specifically, in each leading event, an App's ranking first increases to a peak position in position for a period (i.e., maintaining phase), and finally decreases till the end of the event (i.e., recession phase) the leaderboard (i.e., rising phase).

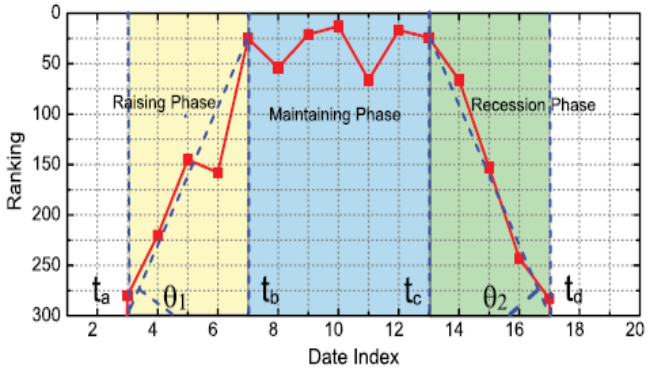

Fig2: Leaderboard (phases)

IX. RATING BASED EVIDENCES

User rating is one of the most important features of App advertisement. An App which has higher rating may attract more users to download and can also be ranked higher in the leaderboard. Thus, rating manipulation is also an important perspective of ranking fraud. 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. an App with rating manipulation might have surprisingly high ratings in the fraudulent leading sessions with respect to its historical ratings.

X. 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 first read its historical reviews to ease their decision making, and a mobile

App contains more positive reviews may attract more users to download. Therefore, imposters often post fake reviews in the leading sessions of a specific App in order to inflate the App download, and thus propel the App's ranking position in the leaderboard.

Fagin's algorithm [17]

In this section, They discuss FA (Fagin's Algorithm) [Fag99]. This algorithm is implemented in Garlic [CHS+95], an experimental IBM middleware system; see [WHRB99] for interesting details about the implementation and performance in practice. Chaudhuri and Gravano [CG96]consider ways to simulate FA by using ''filter conditions'', which might say, for example, that the color score is at least 0.2.6 FA works as follows.

1. Do sorted access in parallel to each of the m sorted lists Li: (By ''in parallel'', we mean that we access the top member of each of the lists under sorted access, then we access the second member of eachof the lists, and so on.)7 Wait until there are at least k ''matches'', that is, wait5QBIC is a trademark of IBM Corporation.6Chaudhuri and Gravano originally saw an early version of the conference paper (in the 1996 ACM Symposium on Principles of Database Systems) that expanded into the journal version [Fag99]. It is not actually important that the lists be accessed ''in lockstep''. In practice, it may be convenient to allow the sorted lists to be accessed at different rates, in batches, etc. Each of the algorithms in this paper where there is ''sorted access in parallel'' remain correct even when sorted access is not in lockstep. Furthermore, all of our instance optimality results continue to hold even when sorted access is not in lockstep, as long as the rates of sorted access of the lists are within constant multiples of each other.

R. Fagin et al. / Journal of Computer and System Sciences 66 (2003) 614–656 619until there is a set of at least k objects such that each of these objects has been seen in each of the m lists.

2. For each object R that has been seen, do random access as needed to each of the lists Li to find the ith field xi of R:

3. Compute the grade tðRÞ ¼ tðx1;y; xmÞ for each object R that has been seen. Let Y be a set containing the k objects that have been seen with the highest grades (ties are broken arbitrarily). The output is then the graded set fðR; tðRÞÞ j RAYg:It is fairly easy to show [Fag99]that this algorithm is correct for monotone aggregation functions t (that is, that the algorithm successfully finds the top k answers). If there are N objects in the database, and if the orderings in the sorted lists are probabilistically independent, then the middleware cost of FA is OðNðm 1Þ=mk1=mÞ; with arbitrarily high probability [Fag99].

An aggregation function t is strict [Fag99]if tðx1;y; xmÞ ¼ 1 holds precisely when xi ¼ 1 for every i: Thus, an aggregation function is strict if it takes on the maximal value of 1 precisely when each argument takes on this maximal value. We would certainly expect an aggregation function representing the conjunction to be strict (see the discussion in [Fag99]). In fact, it is reasonable to think of strictness as being a key characterizing feature of the conjunction Fagin shows that his algorithm is optimal with high probability in the worst case if the aggregation function is strict (so that, intuitively, we are dealing with a notion of conjunction),and if the orderings in the sorted lists are probabilistically independent. In fact, the access pattern of FA is oblivious to the choice of aggregation function, and so for each fixed database, the middleware cost of FA is exactly the same no matter what the aggregation function is. This is true even for a constant aggregation function; in this case, of course, there is a trivial algorithm that gives us the top k answers (any k objects will do) with Oð1Þ middleware cost. So FA is not optimal in any sense for some monotone aggregation functions t: As a more interesting example, when the aggregation function is max (which is not strict), it is shown in [Fag99]that there is a simple algorithm that makes at most mk sorted accesses and no random accesses that finds the top k answers. By contrast, as we shall see, the algorithm TA is instance optimal for every monotone aggregation function, under very weak assumptions.

XI. MATHEMATICAL MODEL

Let S is the Whole System Consist of

S= {I, P, O}

I = Input.

I = {U,Q,MA}

U = User

U = {u1,u2….un}

Q = Query Entered by user.

Q = {q1, q2, q3…qn}

MA = Mobile Apps

MA = {ma1, ma2, ma3…..man}

P = Process.

A. Mining Leading Sessions:

There are two main steps for mining leading sessions

1. We need to discover leading events from the App's historical ranking records.
2. We need to merge adjacent leading events for constructing leading sessions.

B. Ranking Based Evidences

We should first analyze the basic characteristics of leading events for extracting fraud evidences. Therefore, we should first analyze the basic characteristics of leading events for extracting fraud evidences.

1. By analyzing the Apps' historical ranking records, we observe that Apps' ranking behaviors in a leading eventual ways satisfy a specific ranking pattern, which consists of three different ranking phases, namely

- Rising phase:
- Maintaining phase:
- Recession phase:

2. 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.

***C. Rating Based Evidences*:**

The ranking based evidences are useful for ranking fraud detection.

D. Review Based Evidences:

Most of the App stores also allow users to write some textual comments as App reviews. Such review scan reflect the personal perceptions and usage experiences of existing users for particular mobile Apps.

***E. Evidence Aggregation*:**

After extracting three types of fraud evidences, the next challenge is how to combine them for ranking fraud detection.

- We propose an unsupervised approach based on fraud similarity to combine these evidences.
- We define the final evidence score $\psi^*(s)$ as a linear combination of all the existing evidences as Equation.
- 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.

F. Android Framework

Android is a most powerful mobile platform and it powers hundreds of millions of mobile devices in more than 190 countries of the world. Android is a fully power packed operating system that provides strong base to the world supporting lakhs of applications and games for android users as well as an open marketplace supporting Android App Development. It gives you a single and a unique application model which enables you to deploy your apps broadly for Application development and App Development to hundreds of millions of users across a wide range of devices that is from phones to tablets and beyond. Android has undertaken15 powerful,open source and cross platform frameworks. These frameworks enhance Android App Development and Mobile App Development.

XII. CONCLUSION

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. Moreover, we proposed an mining Leading session algorithm for obtain mining leading session and aggregation method. 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.

REFERENCES

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

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

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

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

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

[6] Hengshu Zhu, HuiXiong, Senior Member, IEEE, Yong Ge, and Enhong Chen, Senior Member, IEEE, "*Discovery of Ranking Fraud for Mobile Apps*", vol.13,n0.1,Jan 2015

[7] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly, "*Detecting spam web pages through content analysis,*" in Proc. 15th Int. Conf. World Wide Web, 2006, pp. 83–92.

[8] N. Spirin and J. Han, "*Survey on web spam detection: Principles and algorithms,*" SIGKDD Explor. Newslett., vol. 13, no. 2, pp. 50– 64, May 2012.

[9] B. Zhou, J. Pei, and Z. Tang, "*A spamicity approach to web spam detection,*" in Proc. SIAM Int. Conf. Data Mining, 2008, pp. 277–288.

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

[11] Z. Wu, J. Wu, J. Cao, and D. Tao, "*HySAD: A semi-supervised hybrid shilling attack detector for trustworthy product recommendation,*" in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2012, pp. 985–993.

[12] S. Xie, G. Wang, S. Lin, and P. S. Yu, "*Review spam detection via temporal pattern discovery,*" in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2012, pp. 823–831.

[13] K. Shi and K. Ali, "*Getjar mobile application recommendations with very sparse datasets,*" in Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2012, pp. 204–212.

[14] B. Yan and G. Chen, "*AppJoy: Personalized mobile application discovery,*" in Proc. 9th Int. Conf. Mobile Syst., Appl., Serv., 2011, pp. 113–126.

[15] H. Zhu, H. Cao, E. Chen, H. Xiong, and J. Tian, "*Exploiting enriched contextual information for mobile app classification,*" in Proc. 21stACMInt. Conf. Inform. Knowl. Manage., 2012, pp. 1617–1621.

[16] H. Zhu, E. Chen, K. Yu, H. Cao, H. Xiong, and J. Tian, "*Mining personal context-aware preferences for mobile users,*" in Proc. IEEE 12th Int. Conf. Data Mining, 2012, pp. 1212–1217.

[17] Ronald Fagin,a,_ AmnonLotem,b and Moni Naorc,1, "*Optimal aggregation algorithms for middleware*"Journal of Computer and System Sciences 66 (2003) 614–656 ,1 April 2002

