ADVANCED TECHNIQUE FOR DETECTION OF RANKING FRAUDS IN MOBILE APPLICATIONS

Monika Hanumantrao Patil

Abstract— Nowadays ranking fraud of mobile Apps had increases in wide range, which have purpose of striking up to Apps in the popularity list. The developers are supposed to work on software but there are also many developers which makes fraud ranking and puffed their App’s sales. On the leaderboard the ranking of App’s can be manipulated, so this is main problem. Here a holistic view is determined for ranking fraud and new system is used for detection of ranking fraud for mobile App’s. The given system first works on locating the ranking fraud by mining the historical data i.e. mining the document collection with active period. Furthermore, explore three types of evidences like ranking based, rating based and reviews based evidences. Here, Bisecting K-means and Hybrid clustering algorithms are used to integrate all evidences for fraud detection. Finally the system is evaluated with real world App’s data collected from Apples App store. The experiments shows virtue of given system and shows scalability and regularity of ranking fraud activities and algorithms.

Keywords: Clustering, stemming, weighing, document vector, bisecting k-means clustering, hybrid clustering

I. INTRODUCTION

In the recent years quantity of mobile App’s has been increased at incredible rate. Many App stores launched daily App leaderboards, to promote development of Mobile Apps which reveal the chart rankings of most popular Apps.

Certainly, the App leaderboard is one of the most significant way for recommendation of mobile Apps. A higher rank on the leaderboard usually leads to a huge number of downloads and million dollars in interest. Therefore, App developers tend to scout various ways such as advertising strategy to encourage their Apps in order to have their Apps ranked as high as possible in such App leaderboards. However, as a recent trend, instead of depending on traditional marketing solutions, suspicious App developers spot to some fraudulent means to knowingly boost their Apps. After period of time developers manipulate the chart rankings on an App store. This is usually implemented by using so called “bot farms” or “human water armies” to increase the App downloads, ratings and reviews in a very short time [1]

It is important to identify several important challenges. First, fraud is happened any time during the whole life cycle of app, so the identification of the exact time of fraud is needed. Second, due to the huge number of mobile Apps, so it is important to automatically detect fraud without using any basic information, it is difficult to manually label ranking fraud for each App.

Mobile Apps aren’t always ranked high in the leaderboard. Only in some leading events ranking i.e. fraud usually happens in leading sessions. Therefore, main aim is to detect ranking fraud of mobile Apps within leading sessions.

Figure 1 shows the framework of ranking fraud detection system for mobile App’s. In which all the leading sessions can be mined from historical records of mobile App’s. After analyzing App’s behavior, it is find that the fraudulent Apps often have different ranking patterns in each leading session compared with normal Apps. Hence, by distinguishing some fraud evidences from Apps’ historical ranking records develop three functions to extract such ranking based fraud evidences.
In spite of, the ranking based evidences can be affected by App developers’ status and some legal marketing strategy. As a result, it is not enough to only use ranking based evidences. Therefore, further enhanced two types of fraud evidences based on Apps’ rating and review history, which reflect some inconsistent patterns from Apps’ historical rating and review records. An unsupervised evidence-aggregation method to combine these three types of evidences for evaluating the credibility of leading sessions from mobile Apps.

II. LITERATURE SURVEY

Specifically there are three type in this study which are reference for current system of fraud detection of mobile App’s. web ranking spam detection, online review spam, mobile App recommendation.

Web ranking spam detection: Most of web spam detection methods are uses supervised classification method. In unsupervised spam detection concept of spamcity introduced which is measure how likely the page is spam. The spamicity is more flexible and user-controllable measure with compare to supervised classification methods. Link spam and online link spam are the methods using spamicity which are efficient and does not need training [1].

Online review spam: The next category is based on detecting online review spam. A model which consist of behaviors of online activities. First, Spammers targets the specific product group or product for increase their impact. Second, spammers diverts from other reviewers in their ratings of products. Scoring method is useful to measure degree of spam for each reviewer. This method detected spammers have more significant impact on rating [2].

Mobile App recommendation: A mobile App recommender system is concern with privacy and security awareness. Which is design for equip the recommender system with the functionality. It is allows to detect automatically detect and evaluate the mobile Apps security. Then it can provide App recommendations by considering popularity and the user’s security preferences. Specifically, Mobile App have higher security risk because App’s are insecure and data access permissions are easily available to implement. App hash tree is another approach which is based on modern portfolio theory for recommending App’s by striking a balance between the App’s popularity and the users security and concerns. [3] [4].

Table 1. Literature review

<table>
<thead>
<tr>
<th>year</th>
<th>Author</th>
<th>Paper</th>
<th>Method</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>D. M. Blei, A. Y. Ng, and M. I. Jordan</td>
<td>“Lantentdirichlet allocation”</td>
<td>EM algorithm for empirical Bayes parameter estimation</td>
<td>-</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
<td>Title</td>
<td>Methodology</td>
<td>Applications</td>
</tr>
<tr>
<td>------</td>
<td>---------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2007</td>
<td>Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li</td>
<td>“Supervised rank aggregation,”</td>
<td>The method is mainly designed for meta-search (a) it takes order information from base rankers, (b) it makes use of labelled training data, and (c) it trains the final ranking function within a single optimization framework.</td>
<td>Does not apply on other applications such as similarity search and genome informatics.</td>
</tr>
<tr>
<td>2008</td>
<td>A. Klementiev, D. Roth, and K. Small</td>
<td>“Unsupervised rank aggregation with distance-based models”</td>
<td>Aggregate (partial) rankings without supervision which uses distance based model.</td>
<td>Heuristic and supervised learning approaches are expensive to acquire.</td>
</tr>
<tr>
<td>2009</td>
<td>Alexandr eKlementiev, Dan Roth, Kevin Small, and Ivan Titov</td>
<td>“Unsupervised rank aggregation with domain-specific expertise,”</td>
<td>They propose a framework for learning to aggregate votes of constituent rankers with domain specific expertise without supervision.</td>
<td>It is very difficult to obtain for ranking problems, especially for multiple domains.</td>
</tr>
<tr>
<td>2012</td>
<td>Jeevanandam Jotheeswaran Loganathan R. and Madhu Sudhanan B.</td>
<td>“Feature Reduction using Principal Component Analysis for Opinion Mining”</td>
<td>Inverse document frequency is used to extract features from review document.</td>
<td>It is not efficient to manually handle the large amount of opinions which generated during online.</td>
</tr>
<tr>
<td>2012</td>
<td>N. Spirin and J. Han,</td>
<td>“Survey on web spam detection: Principles and algorithms”</td>
<td>Sub-categorization of link based category in many groups.</td>
<td>Based on web spam.</td>
</tr>
<tr>
<td>2014</td>
<td>E. Siegel</td>
<td>“Fake reviews in Google Play and Apple App Store”</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2015</td>
<td>E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw.</td>
<td>“Detecting product review spammers using rating behaviours”</td>
<td>They propose scoring methods to measure the degree of spam for each reviewer and apply them on an Amazon review dataset.</td>
<td>Does not learn behaviour patterns related to spamming so as to improve the accuracy of the current regression model</td>
</tr>
</tbody>
</table>

### III. SYSTEM ARCHITECTURE

The system shows two stages. The first stage has to preprocess the documents, i.e. converting the documents into appropriate needs of data schemes. The second stage has to analyze the available data from first stage and divide it into clusters. This process is carried out by clustering algorithm.
Preprocessing of the input documents: The preprocessing consists of steps shown in figure 2:

- **1. The PoS tagger**: It relies on the text structure and morphological differences to determine the appropriate part-of-speech. A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other tokens), such as noun, verb, adjective, etc. [12].

- **2. Stop words Removal**: To save space and to speed up searching process, the words which are considered as less important should be removed. Any group of words can be chosen as stop words such as "the", "at", "on", "which", etc. [13]

- **3. Stemming**: Words with the same meaning appear in various morphological forms. Stemming algorithm is used to reduce the word to its root or stem. The key terms used in document are expressed by stem rather than original words. For example, consider the words "playing", "played", "play", and "player" can be reduced to the root word, "play".

- **4. Collecting synonyms & hypernyms**: The stemmed words are looked up in the wordnet and their corresponding synonyms and hypernyms are added. Infrequently occurring synsets are discarded, and those that remain form the feature set [14].

- **5. Weighing**: Weights are assigned to give an indication of the importance of a word on each document. This figure above shows how the Document Collection is converted into Documents Vectors using WordNet as the source of information. Once this stage is completed, Document Vectors are given as input to the clustering Algorithm.

Performing Document Clustering using obtained Document Vectors:

The given Figure shows the document clustering process. In this stage the clustering algorithm takes Document Vectors that are generated during the first stage, are considered as input to the clustering Algorithm. The approach uses both agglomerative and divisive hierarchical clustering algorithms [15] to generate a set of good clusters. The algorithm in paper [16] uses the top-down (Bisect K-means) and bottom-up (UPGMA) agglomerative hierarchical clustering algorithms. Then pass the cluster information (centroids) computed from the bisect K-means algorithm to the UPGMA algorithm to correct the inconsistencies occurred due to the wrong decision made while merging or splitting a cluster.
Figure 3. Document clustering Process

Hybrid bisect K-means algorithm uses the Euclidean distance and Cosine similarity measure between the documents/clusters to find their relationship. Euclidean distance measure is used by the K-means algorithm in divisive algorithm to split the document clusters and cosine similarity is used in UPGMA to merge the centroid clusters. Once clusters are generated they can be labeled. Once the clusters are formed, labeling the cluster is important. Because once then label the cluster if will give a brief idea about the type of documents that are contained in that particular cluster.

The most important and challenging characteristics of the vector space models that arise from the text data are high dimensionality and sparsity. Typically, \( w \) is in the thousands and a sparsity of 99% is common. For purposes of efficiency, it is important that the clustering algorithm exploit the sparsity of the data while giving meaningful results at the same time. The spherical k-means algorithm satisfies both these properties and hence algorithm is of choice. Briefly formalize this algorithm highlighting its salient features. Any text clustering algorithm needs an objective notion of similarity between documents. A widely used measure of similarity is the cosine of the angle between two document vectors. Cosine similarity is easy to interpret and simple to compute for sparse vectors and has been used in other information retrieval applications. Here also define the “goodness” or “coherence”, of cluster \( n_j \) as

\[
\sum_{x_i \in \pi j} x_i T c_j
\]

where each \( x_i \) is assumed to be normalized such that \( ||x_i||=1 \) and \( c_j \) is the normalized centroid of cluster \( n_j \),

\[
c_j = \frac{\sum_{x_i \in \pi j} x_i}{||x_i \in \pi j X_i||}
\]

By the Cauchy-Schwarz inequality,

\[
\sum_{x_i \in \pi j} x_i T x_i \leq \sum_{x_i \in \pi j} x_i T c_j \quad \text{for all } z \in R_w
\]

And thus the normalized centroid is the vector that is closest in cosine similarity (in an average sense) to all the document vectors in the cluster \( n_j \), as a result, we also call the vector \( c_j \)s as concept vectors.

Aggregating (3.1) over all clusters, we can measure the goodness of any given partitioning \( \{\pi j\} k_j = 1 \) using the following objective junction:

\[
Q(\{\pi j\} k_j = 1) = \sum k_j = 1 \sum_{x_i \in \pi j} x_i T c_j
\]

Intuitively, the objective function measures the combined coherence of all the k clusters. Having posed the objective function, this algorithm that attempts to maximize its value. This algorithm resembles to a Hybrid clustering algorithm: in fact, Hybrid clustering algorithms have the advantage of not requiring a priori the number of clusters, since the clusters are bisected at each step. In these
algorithms however, the problem is in defining a stopping rule, i.e., deciding if and which clusters have to be still bisected. To this aim, two main approaches are used: the first one applies the simple strategy of bisecting the greatest cluster and the second one is to split the cluster with greatest variance with respect to the centroid of the cluster. [15], [17].

IV. ALGORITHMS

Bisecting K-Means Algorithm

Input: Document Vectors DV Number of Clusters k Number of iterations of k-means ITER

Output: K’Clusters

1. Pick a cluster to split (split the largest)
2. Find 2 sub-clusters using the basic K-means algorithm
3. Repeat step 2, the bisecting step is doing for ITER times and takes the split process that results in the clustering with the highest overall similarity
4. Repeat steps 1, 2 and 3 until the desired number of clusters „k„are reached.

In the above procedure ITER must be sufficiently large so that the change in the cluster centroid from its previous iteration is almost negligible.

Hybrid clustering algorithms:

Hybrid clustering algorithms build a hierarchy of quality clusters. One of the main problems with the Hybrid clustering is that the documents put together in the early stage of the algorithm will never be changed. In other words, Hybrid clustering tries to preserve the local optimization criterion but not the global optimization criterion. If somehow correct these misplaced documents in the generated clusters, so try to preserve the global optimization criterion.

The algorithm uses both the top-down (Bisect K-means) and bottom-up (UPGMA) agglomerative Hybrid clustering algorithms to address this problem. Then pass the K’ cluster information (centroids) computed from the bisect K-means algorithm to the UPGMA algorithm to correct the inconsistencies occurred due to the wrong decision made while merging or splitting a cluster.

First, ran the bisect K-means algorithm on the document collection for a particular value of the K0 until K0 number of document clusters were generated. One cluster with more number of documents or highest intra-cluster similarity value is chosen at each step to split. The generated document clusters should not be empty. Then, calculated the centroids for each of the resulting clusters. Each of these centroids represents a document cluster and all of its documents.

1. Pick a cluster to split. (Initially the whole document collection is used as a single cluster)
2. Find 2 sub clusters using k-means algorithm.
3. Repeat Steps 1 (Initialization step) and 2 (bisecting step) until the K’ > K number of clusters are generated.
4. Compute the centroids (cluster prototypes) for each of the K’ clusters such that each document in a collection be longs to one of these centroids.
5. Construct a K’ X K’ similarity matrix between these centroid clusters.
6. Merge two similar centroid clusters (i.e., place these centroids in the same cluster).
7. Update the centroid clusters similarity matrix.
8. Repeat Steps 6 (Merging step) and 7 (Updating step) until the K clusters of centroids are generated.
9. If two centroids belong to same centroid clusters, then the document clusters of these centroids will go together as a final cluster (Merging step).

V. EXPERIMENTAL SETUP AND RESULTS

The dataset contains “free 300 Apps”, and “Paid 300 App” [18]. Each file contains rating, review and ranking information. The above graphs are compared with existing system [19]. For Performance
evaluation of the approach system is measured. It is based on two parameters i.e. precision and recall. Precision and Recall are defined in terms of a set of retrieved output (e.g. the list of apps produced for a query) and a set of relevant output (e.g. the list of all apps that are relevant for a certain topic).

Here, considered different values of \( k \) for experiment result analysis i.e. 5, 10, 15, 20. All these experiments are evaluated on dataset of size around 2, 27,000 records as per results displayed in above table, the actual ranking of 300 free apps and 300 paid apps. The results provided in actual results is the results count provided by the dataset with ranking and name of apps.

Retrieved results are the results that are retrieved during execution. Out of all apps, the apps that are retrieved are mentioned in this column.

<table>
<thead>
<tr>
<th>Table 2. Results of precision and recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
</tr>
<tr>
<td>---------</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>Avg.</td>
</tr>
</tbody>
</table>

**Precision:** Precision is the fraction of retrieved documents that are relevant to the find. The table (2) shows values for precision which is explained below,

\[
\text{Precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}
\]  

(5)

For e.g. out of 600 apps for 5 cluster, current system is able to retrieve only 564 records And 528 records which are relevant. From above retrieved apps it must have to check relevant apps that are retrieved.

\[
\text{Precision} = \left(\frac{528}{564}\right) \times 100
\]

\[
= 93.61
\]

(6)

The above graph (a) shows results of current and existing system for Precision. Values of current precision are higher in some queries than existing system. It is changed as per queries are changed.

**Recall:** Recall in information retrieval is the fraction of the documents that are relevant to the query that are successfully retrieved. The table (2) shows values of recall which is explained below,

\[
\text{Recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}
\]

(7)

For e.g. out of 600 apps for 5 cluster, current system is using relevant 528 records and 564 records which are relevant. From above retrieved apps it must have to check relevant apps that are retrieved.

\[
\text{Recall} = \left(\frac{528}{600}\right) \times 100
\]

\[
= 88
\]

(8)
The above graph (b) shows results of current and existing system for Recall. Values of current Recall are higher in some queries than existing system. It is changed as per queries are changed.

![Graph (b)](image)

F-Measure is calculated from Precision and recall as,

$$F - Measure = \frac{Precision \times Recall}{Precision + Recall}$$  \hspace{1cm} (9)

For e.g

$$F - Measure = \frac{(2 \times 93.61 \times 88)}{93.61 + 88} = 90.71$$ \hspace{1cm} (10)

The above fig(c) shows results for F-Measure with respect to precision and recall values.

**NDCG (Normalised Discounted cumulative Gain):** For determining the ranking performance of each approach. Specifically, the discounted cumulative gain given a cut-off rank K can be calculated by

$$DCG@K = \sum_{i=1}^{K} \frac{f(s_i) - 1}{\log(1 + i)}$$ \hspace{1cm} (11)

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,

$$NDCG = \frac{DCG@K}{IDCG@K}$$ \hspace{1cm} (12)

The nDCG values for all queries can be averaged to obtain a measure of the average performance of a search engine's ranking algorithm.

The above fig(d) shows the results for NDCG. The performance of current NDCG is better than existing NDCG, but it is changes as per queries variation.

![Graph (d)](image)
The above fig (e) shows Average time with respect to no of queries. It shows the current system working is much better than existing system.

VI. CONCLUSIONS

A ranking fraud detection system for mobile Apps can be performed with new approach. Initially the system shows that ranking fraud happened in leading sessions and provided a way for mining leading sessions for every App from its historical ranking records. Hear an improvement based aggregation methodology is used to integrate all the evidences for evaluating the quality of leading sessions from mobile Apps. A distinctive perspective of this approach is that each one evidence will be modelled by applied by math hypothesis tests, so it's simple to be extended with different evidences from domain data to notice ranking fraud. Finally, for validate the projected system, the system can be experimented with intensive experiments on real-world App information which are collected from the Apple’s App store. Experimental results showed the effectiveness of the current approach.

In the future, the number of instances in the training and test sets should be increased. The number of the attributes can be enhanced by adding information about used services, separation of the day, evening, night call, minutes and charges etc.

REFERENCES


