Report on

Trust Rank Algorithm

Arnab Ghosh, Kritik Agarwal, Raghavendra Kulkarni, Shagun Sharma, and Trishita Saha

Instructor: Dr. Sobhan Babu Chintapalli
Fraud Analytics using Predictive and Social Network Techniques (CS6890)
Dept. of Computer Science And Engineering
Indian Institute of Technology, Hyderabad

ABSTRACT

With the beginning of the digital era, the number of financial transactions has increased exponentially on a daily basis, contributing to the enormous amount of data generated. Although digitization has enabled genuine users to opt for safe and cashless transactions, it has also widened the possibilities of anonymous fraudulent activities. The rising number of financial fraudulent cases necessitates the research of novel methods for detecting them. Machine Learning techniques on Graph Neural Networks have shown promising results in this domain. We implement a modified version of one such technique, the TrustRank algorithm, on a Payment Network Dataset to detect Fraudulent money senders. The results show that the modified TrustRank algorithm can detect new fraudulent money senders, other than the previously known ones, through trust propagation.

Keywords: Propagation, Receivers, Senders, TrustRank

PROBLEM STATEMENT

Given a Payment Network as a Directional Multi-graph with senders and receivers as nodes, payments as edges, and some senders known as fraudulent or bad senders prior, find all the fraudulent or bad senders across the network using TrustRank Algorithm.

DATASET DESCRIPTION

The dataset consists of two .csv files:

- Payments.csv
- bad_sender.csv

Payments.csv

The Payments.csv contains 130,535 entries with each entry in the form of three comma separated integers. These integers denote the sender ID, receiver ID and the amount transferred in the payment respectively. Table 1 below shows the first 10 entries of the Payments.csv file.

Sender	Receiver	Amount
1309	1011	123051
1309	1011	118406
1309	1011	112456
1309	1011	120593
1309	1011	166396
1309	1011	177817
1309	1011	169351
1309	1011	181038
1309	1011	133216
1309	1011	133254

Table 1. The first 10 entries of Payments.csv

bad_sender.csv

The bad_sender.csv contains 20 entries with each entry denoting the ID of the known fraudulent sender. These 20 fraudulent senders IDs are:
1303, 1259, 1562, 1147, 1393, 1031, 1210, 1042, 1048, 1256, 1668, 1161, 1007, 1034, 1836, 1099, 1489, 1821, 1076, 1944.

THE TRUSTRANK ALGORITHM

Algorithm 1 The TrustRank Algorithm

```
Require: Graph G denoting the Payment Network
         List $L$ containing the known Bad sender IDs
         Damping factor $\beta$
         Maximum number of iterations $max\_iter$
         Tolerance in error $tol$
Ensure:  Mapping $score$ containing all the users mapped to their scores
  $\alpha \leftarrow 1/G.nodes.length$
  $score \leftarrow$ Empty Mapping
  for each $node$ in $G.nodes$:
    if $node$ in $L$
      $score[node] \leftarrow 1/L.length$
    else
      $score[node] \leftarrow 0$
  for $iteration \leftarrow 1$ to $max\_iter$ do
    for each $node$ in $G.nodes$:
      if $node$ in $L$
        $new\_score[node] \leftarrow 1/L.length$
      else
        $new\_score[node] \leftarrow 0$
    for each $node$ in $G.nodes$:
      for each $neighbor, edge$ in $G[node]$:
        $new\_score[node] \leftarrow new\_score[node] + score[node] \times edge.weight$
    for each $node$ in $G.nodes$:
      $new\_score[node] \leftarrow new\_score[node] \times \beta + \alpha \times (1-\beta)$
    $stop \leftarrow$ true
    for each $node$ in $G.nodes$:
      if $abs(new\_score[node] - score[node]) > tol$
        $stop \leftarrow$ false
        break
    if $stop$
      break
    else
      $new\_score \leftarrow score$
  return $score$
```

Explanation

- The user *scores* in the above algorithm always lie in the range $[0,1]$. Unlike the standard *PageRank* or *TrustRank* algorithm, this modified version assigns a non-zero score to the known fraudulent senders and 0 to all the others.
- In every iteration, all the nodes distribute their scores to their neighbors through the outgoing links. The amount of score a neighbor gets from the node is directly proportional to the weight of the edge present between them.
- After the distribution of the scores, every node adds a smoothing factor α which is equal to inverse of the number of nodes in the graph, to avoid dominance of any one particular set of nodes.

Classification
We need a threshold to classify the nodes as fraudulent senders and genuine senders. To set this threshold, we find the minimum among all the final converged scores of the known fraudulent senders. A node with a score higher than the threshold is a fraudulent sender, and a node with a score lower than the threshold is a genuine sender.

Algorithm 2 The Classification

```
Require: Mapping $score$ denoting the node scores
         List $L$ containing the known fraudulent sender IDs
Ensure:  List $L_1$ containing the all the fraudulent sender IDs
  $threshold \leftarrow$ None
  $L_1 \leftarrow [\,]$
  for each $node$ in $L$:
    if $threshold$ is None or $score[node] \leq threshold$
      $threshold \leftarrow score[node]$
  for each $node$ in $score$.keys:
    if $score[node] \geq threshold$
      Append $node$ to $L_1$
  return $L_1$
```

IMPLEMENTATION
We make use of the ***NetworkX*** API in Python for creating and managing the graph. ***NetworkX*** is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Preprocessing
We normalize the outgoing edge weights of every node to sum to 1, so that at the end of every score distribution iteration, the updated scores of the nodes lie in the range [0, 1]. Also, we set the personalization scores for known fraudulent senders in such a way that the sum of scores of all the nodes is 1.

NetworkX
The main functionalities of NetworkX used for implementing the algorithm are:

- **networkx.MultiDigraph():** We use this function to initialize a Directional Multigraph where we store the senders and receivers as nodes and the payments as edges. The amount of payment done is taken as the edge weight. This function returns an empty Directional Multigraph.

- **graph.add_edge():** We use this function to construct the graph. It takes two node IDs as arguments and adds an edge between them. Optionally, it also takes the weight of the edge (if the graph is weighted) as the third argument. If the nodes are not created before adding the edge, then this function first creates the node and then adds the edge between them.

- **graph.edges():** We use this function to iterate through every edge of the graph and normalize its edge weight to make all the outgoing edge weights of every node sum to 1. This function returns an iterable dictionary containing all the edges of the graph.

- **graph.nodes():** We use this function to iterate through every node of the graph during the assignment of the personalization scores to those nodes which correspond to known Fraudulent senders and also during each iteration of the algorithm for score distribution. This function returns an iterable list containing the IDs of all the nodes in the graph.

- **graph[node].items():** We use this function to iterate through all the neighbors of a particular node during the score distribution process. This function returns an iterable dictionary containing the neighbors mapped to the set of incident edges and edge weights.

RESULTS

By running the above algorithm, we found 17 other fraudulent senders apart from the known ones. The newly detected fraudulent sender IDs are 1088, 1122, 1094, 1480, 1038, 1144, 1041, 1201, 1138, 1011, 1173, 1205, 1626, 1013, 1050, 1084, 1086. The scores of top 10 fraudulent bad senders are

Sender ID	Score
1007	0.1860087378457204
1088	0.17730705598268548
1144	0.17410612265169015
1210	0.12414467292186009
1042	0.09687094123706333
1086	0.09643505791988981
1034	0.07704341088085703
1076	0.07295612362538079
1201	0.06249344189009486
1094	0.06243371355838374

Table 2. The scores of top 10 Fraudulent senders

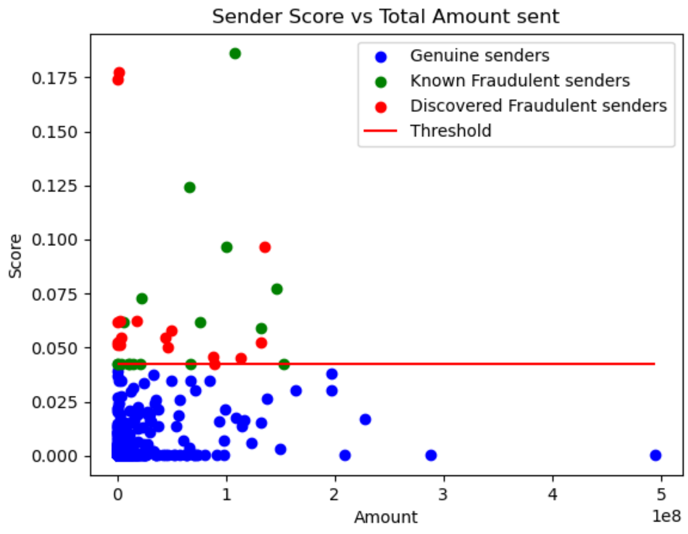

Figure 1. Sender Score vs Total Amount sent

Inference

Figure 1 above shows the scattered distribution of the Score of a Sender against the Total Amount sent.

- The red horizontal line is the boundary of threshold to classify the senders into genuine senders and fraudulent senders.
- The blue dots below the line of threshold represent the genuine senders. 762 out of 799 are found to be genuine senders.
- The green dots above the line of threshold represent the known fraudulent senders. There were 20 senders known to be fraudulent initially.
- The red dots above the line of threshold represent the newly discovered fraudulent senders. The algorithm is able to detect 17 other fraudulent senders.
- The presence of red dots near the Y-axis indicate that the algorithm is able to detect new fraudulent senders that were less actively sending money but were acting as receivers from other fraudulent senders.

CONCLUSION

- We use a modified version of the TrustRank algorithm to find new fraudulent senders from known ones in a Payment Network. We exploit the property that a fraudulent sender will send money to only a fraudulent sender.
- So, we use the fraud score distribution phenomena to propagate the fraud score to other fraudulent senders.

REFERENCE

[1] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, “Combating Web Spam with TrustRank,” VLDB Endowment, 2004. Available: https://www.vldb.org/conf/2004/RS15P3.PDF