भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad

Assignment 2 - Fraud Analytics (CS6890) - Spring 2024

Trust Rank

submitted by

Simanta Das : CS23MTECH11018
Ashish B Emmanuel : CS23MTECH11004
Akshat Gupta : CS23MTECH11001
Ashutosh Rajput : CS23MTECH11005
Patnala Sai Kumar : CS23MTECH14020

CONTENTS

1. Abstract

The report presents an application of trust rank algorithm to identify potential bad senders in the network. The assignment includes a list of transactions with each transactions containing a sender, receiver, and transaction amount along with a list of known bad senders. The trust rank algorithm which was originally designed to identify web spam is repurposed to detect bad senders in the transaction network. The algorithm operates by assigning a 'fraud score' to each known bad sender and their proximity to other bad senders in the graph.

The report details the methodology of applying trust rank algorithm to transactional data which includes the details about constructing a graph, assigning fraud scores and iterative propagation of fraud scores. The results of this assignment showcases the effectiveness of using trust rank algorithm to find out potential bad senders that were originally not part of bad senders list.

2. Introduction

Transactions have moved from physical exchanges to online platforms in the present digital age which creates a complex network of senders and receivers. This shift has brought about numerous advantages which includes speed, convenience, and global reach. However, this has also opened up new avenues for fraudulent activities which makes the security of transaction systems a growing concern.

One of the key challenges in maintaining the integrity of transaction systems is the identification of malicious transaction senders. In this assignment we use a modified form of trust rank algorithm to proactively detect the potential bad actors.

2.1 Problem statement

1. Two CSV files are provided
   a) Payments.csv: where each row is a transaction with first column being sender, second column being receiver and third column being amount of money sent.
   b) bad_senders.csv: which contains a list of bad senders.
2. Find out all the potential bad senders using Trust Rank algorithm.

2.2 Description of the dataset

1) Payments.csv:
   a. Has a list of 1,30,535 transactions, these transactions can be between same senders and receivers.
   b. Contains 799 unique senders or receivers
2) bad_senders.csv has a list of 20 unique bad senders

2.3 Understanding the Trust Rank Algorithm

Trust Rank algorithm was originally developed by researchers at Stanford University to combat web spam. The algorithm operates on the notion that good pages link to good pages and only rarely ever link to spam pages. It assigns a trust score to each page in a web graph measured on the page's likelihood of being a good page.

The Trust Rank algorithm begins with manually identifying a set of pages that are verified to be good and these will be known as seed pages. These seed pages are assigned an initial trust score. The algorithm then iteratively propagates these allocated trust scores from the seed pages to their respective linked pages. The propagation is done in such a way that a page receives a higher trust score if it is linked from pages with high trust scores. Trust Rank's reliability diminishes on a web page as it's distance increases from the seed set.

2.4 Adaptation of Trust Rank algorithm to find bad senders

In the case of payments network, Trust rank algorithm can be adapted to assign 'fraud scores' to senders. Here the list of bad senders provided by the 'bad_senders.csv' can be considered as the seed set and transaction from sender to receiver be considered as a link between them.

The algorithm then iteratively propagates these allocated fraud scores from the seed set to their respective linked actors same as in Trust Rank algorithm. Here we assume that bad senders are more likely to send money to bad senders and hence senders with higher 'fraud score' can be considered as potential bad senders.

3. Methodology

3.1 Reading from 'Payments.csv' and building a graph

a) We use networkx module to build the graph, we chose directional multi-graph as there can be multiple transactions between the same sender and receiver and also the direction of the transaction is important.

b) In the graph, the senders and receivers will be nodes of the graph and the transaction will be edge connecting them.

c) For each row we read from the CSV file, we add an edge with first column being the node from which the edge originates and second column being the node to which edge is directed to. We will set the amount of transaction as the weight of that particular edge.

3.2 Reading the list of bad senders (seed set) and setting the fraud score for each node

a) We read the set of bad senders from 'bad_senders.csv' and store it in a list.

b) In the graph, for each node which is in the bad senders list we assign a value of $\frac{1}{numberofbadnodes}$ and for rest of the nodes we assign a value of 0

3.3 Normalizing edge weights

a) We iterate through all the nodes and for each node we calculate the sum of edge weights of all the outgoing edges of that node. For node i, let's call the cumulative outgoing edge weight CW_i.

b) For each node i, we will iterate through their outgoing edges and normalize the weights by dividing the edge weight by CW_i.
For outgoing edge j, normalized weight = $\frac{edgeweight_j}{CW_i}$

3.4 Iterative propagation of fraud scores and updating bad senders list

a) We iterate through the nodes of the graph till one of the below two conditions is met.

i) Maximum number of iterations reached.

ii) The change in node values between previous iteration and the current iteration is negligible meaning scores have 'converged'

b) In each iteration, we propagate scores in the following way

i) For each node we store all the outgoing edges for that node in a list

ii) We iterate through the list of outgoing edges and update the score of target node
$target\_node(score) = target\_node(score) + edge\_weight * origin\_node(score)$

iii Next we have to smoothen the propagated score by re-distributing a part of score across all nodes, we do this by defining a constant $'\alpha'$ and
$node(score) = \alpha * node(score) + (1 - \alpha) * (\frac{1}{total\ number\ of\ nodes})$

iv Normalize fraud scores
node(score) = $\frac{node(score)}{sum\ of\ scores\ of\ all\ nodes}$

3.5 Updating bad senders list

a) We assume that all the bad senders in the seed set (bad_senders.csv) will still remain bad senders even after completion of iterative propagation.

b) Hence we set the least score among the initial bad senders as a threshold and any senders with score below this threshold will be considered as honest senders and any senders with score above this threshold will be considered as a potential bad sender.

c) We update the list of our bad senders based on this notion.

3.6 Pseudo-code

Algorithm 1 Trust Rank Algorithm

READ 'bad_senders.csv' and STORE in bad_senders[]
Create a directed multi-graph (GRAPH)

READ the 'Payments.csv' file and add edges to the GRAPH
for $row = first\_row$ to $last\_row$ **do**
  GRAPH.add_edge(origin_node(row[0]), target_node(row[1]), edge_weight(row[2]))
end for

for *node* in $GRAPH$ from $first\_NODE$ to $last\_NODE$ **do**
  if *node* in $bad\_senders[]$ **then**
    $node(value) = 1/\text{size}(bad\_senders[])$
  else
    $node(value) = 0$
  end if
end for

for *node* in $GRAPH$ from $first\_NODE$ to $last\_NODE$ **do**
  for *edge* in *node.outedges* **do**
    edge.weight = edge.weight/(sum of all outgoing edge weight for that node)
  end for
end for

for $i = 0$ to $MAX\_ITERATION$ **do**
  for *node* in $GRAPH$ from $first\_NODE$ to $last\_NODE$ **do**
    for *edge* in *node.outedges* **do**
      $edge.target\_node(value) += edge.origin\_node(value) * edge.weight$
    end for
  end for
  for *node* in $GRAPH$ from $first\_NODE$ to $last\_row$ **do**
    $node(value) = \alpha * node(value) + (1 - \alpha)/(totalnumberofnodes)$
  end for
  if MAX change in *node* value $< threshold$ **then**
    BREAK
  end if
end for

min_score = lowest score among bad senders
for *node* in $GRAPH$ from $first\_NODE$ to $last\_NODE$ **do**
  if $node(value) < min\_score$ **then**
    ADD node to bad_senders[]
  end if
end for

bad_senders[] is updated with all newly detected bad senders.

4. Results and observation

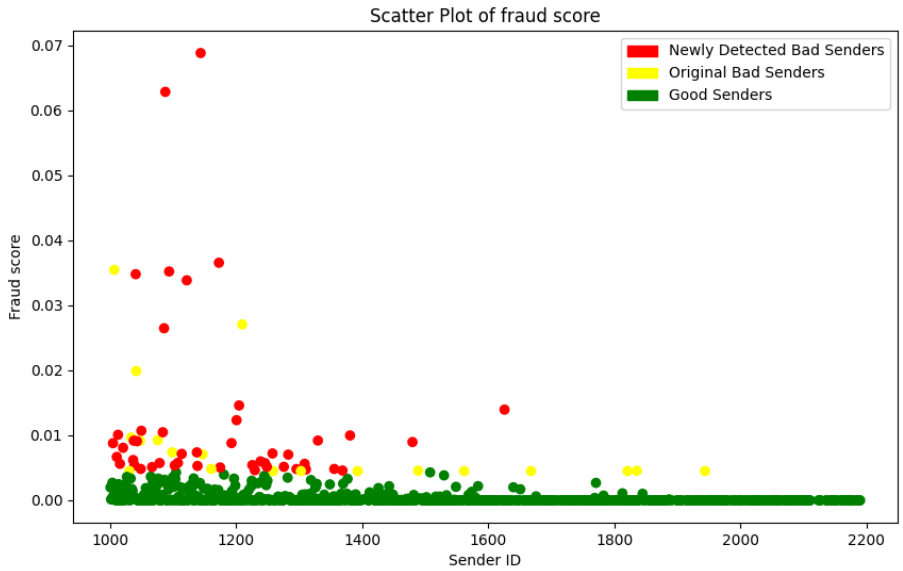

Figure 1: Plot showing the distribution of fraud scores with sender ID on X-axis and fraud score on Y-axis

- In the above Figure 1, the X axis represents sender ID, Y axis represents fraud scores of senders after convergence.
- Red dots indicate all the new senders who were deemed to be bad senders, yellow dots indicate the original bad senders (seed set) and green dots indicate all the good senders.

```
List of bad senders: [1303, 1259, 1562, 1147, 1393, 1031, 1210, 1042, 1048, 1256, 1668, 1161, 1007, 1034, 1836, 1099, 1489, 1821, 1076, 1944]
Bad nodes and their associated value
Node: 1259, Value: 0.05
Node: 1147, Value: 0.05
Node: 1393, Value: 0.05
Node: 1210, Value: 0.05
Node: 1042, Value: 0.05
Node: 1256, Value: 0.05
Node: 1668, Value: 0.05
Node: 1007, Value: 0.05
Node: 1034, Value: 0.05
Node: 1099, Value: 0.05
Node: 1076, Value: 0.05
Node: 1944, Value: 0.05
Node: 1048, Value: 0.05
Node: 1031, Value: 0.05
Node: 1562, Value: 0.05
Node: 1821, Value: 0.05
Node: 1161, Value: 0.05
Node: 1489, Value: 0.05
Node: 1303, Value: 0.05
Node: 1836, Value: 0.05
Number of nodes:  799
Number of edges:  130535
**************************************************
Distributing fraud scores. Please wait.....
Converged in iteration number 83
The bad node with the least value is 1303 with a value of 0.004506639445303334.

List of updated bad nodes
[1309, 1011, 1259, 1147, 1393, 1039, 1210, 1005, 1042, 1256, 1668, 1007, 1034, 1099, 1076, 1944, 1048, 1205, 1276, 1138, 1079, 1037, 1049, 1013, 1084, 1480, 1016, 1356, 1108, 1246, 1031, 1114, 1562, 1144,
1283, 1103, 1021, 1201, 1050, 1330, 1088, 1038, 1193, 1139, 1067, 1175, 1296, 1239, 1230, 1311, 1086, 1094, 1122, 1258, 1626, 1173, 1041, 1043, 1381, 1226, 1249, 1821, 1369, 1161, 1489, 1303, 1836]
Total number of bad nodes 67
```

Figure 2: Screenshot of trust rank code output

4.1 Inference

1) From the Figure 2 we can observe that the scores converged in iteration number 83.
2) At the end of convergence, we had 67 bad senders in total including the original 20 bad senders, which means that we were able to identify 47 new potential bad senders.
3) We can infer the same thing from Figure 1 plot too.

5. Conclusion

This assignment has been helpful in demonstrating the potential of Trust Rank algorithm as a proactive measure in identifying malicious transaction senders part of a transactional network. By assigning Trust scores (fraud scores in our case) to a list of known bad senders, we were able to identify additional potential bad senders, thereby leveraging the already tried and tested algorithm for a new purpose with slight modifications.

Future research in this field could also explore integration of the Trust Rank algorithm with other machine learning techniques for improved identification of bad senders. Additionally, applicability of the Trust Rank algorithm in other domains could also be explored along with it. In conclusion, the report reaffirms the significance of proactive measures like identifying bad senders beforehand in combating fraudulent activities in digital transaction systems.

6. References

- Gyongyi, Z., Garcia-Molina, H., & Pedersen, J. (2004). Combating Web Spam with TrustRank. In Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30 (pp. 576–587).
- https://en.wikipedia.org/wiki/TrustRank
- https://pdfs.semanticscholar.org/244a/9230d09530df3db206f543fb5ec57676d2b6.pdf