Implementation of TrustRank using Pregel Framework

Rahul Vigneswaran K
CS23MTECH02002

SarveshKumar Purohit
AI22MTECH14006

Rithik Agarwal
CS22MTECH11004

Vinayak Nambiar
AI22MTECH13005

Roshin Roy
AI22MTECH13006

Parth Nitesh Thakkar
CS22MTECH14005

Done as part of the coursework on Fraud Analytics Using Predictive and Social Network Techniques (CS6890). Code available at this link.

1. Problem Statement

The objective is to **identify bad nodes** in a network of nodes with edge weights which indicate the strength of the transaction between a set of nodes. This is to be achieved by developing a method that can effectively identify the malicious or otherwise "bad" nodes **based on patterns in the connections between the nodes**. The resulting methodology should be **scalable to large networks** and as a result, should be implemented using a **distributed** computing framework.

In this work, we leverage TrustRank to find bad nodes by using a set of oracle-provided bad nodes as a seed set. As a result, the resulting rank will be **higher for bad nodes and lower for good nodes**. Intuitively one can think of this as **DistrustRank**. We make use of the Pregel framework for handling the scalability of the method.

2. Description of the dataset

The dataset at hand consists of two files - 'Iron_dealers_data.csv' and 'bad.csv'. The former has values that can be used to generate a directed graph with nodes representing the iron dealers and edges indicating the transactions between them. Each of the edges represents the importance/strength of the transaction held between the seller and the buyer. It should be noted that for the sake of convenience, the IDs of the nodes are mapped to 0-799 for the sake of convenience without any loss of generality. In some cases, where there are multiple edges between the same two nodes, the edges are consolidated by summing up their values. In case of nodes with no outlinks (which can lead to leakage), synthetic links are created back to all the bad nodes with equal weightage.

The latter file consists of IDs of 20 bad nodes, which are used as a seed set in the study.

2.1. Statistics of the dataset

- 'Iron_dealers_data.csv' : Consists of a directed graph where nodes represent Iron dealers and edges represent transactions between them.
  - 'Seller ID' : Node of the seller.
  - 'Buyer ID' : Node of the buyer.
    - The ID are mapped in such a way that it starts from 0, for the sake of convenience without loss of generality.
  - 'Value' : Importance of transaction between the seller and the buyer.
    - There are cases where there exists multiple links between the same two nodes. In those case, the links are consolidated buy summing up the values.
  - Counts :
    - Unique nodes : 799
    - Nodes with no outgoing link : 96
- 'bad.csv' : List of bad nodes for using as a seed set.
  - Bad nodes : 20

3. Algorithm Used

3.1. TrustRank

TrustRank is a **topic-specific variation of PageRank** that incorporates the notion of a trusted seed set of pages. The goal of TrustRank is to make use of these oracle-provided trusted pages to identify other trustworthy pages. The idea is, to **begin with, this set of trusted seed nodes and iteratively propagate the trust score along the graph**

edges. Equation 1 shows the TrustRank equation after taking into account both seed set and edge weights. We show a short pseudo code in Algorithm 1 for the reader's reference. For a much detailed code check link.

Algorithm 1 TrustRank Algorithm with Pregel Framework

Require: Seed set for init (d), Link Weightage (ρ), Iterations (N), Damping factor (β)

Ensure: Nodes with no outlinks are connected back to the seed nodes with equal weightage (to prevent leakage)

while $n \leq N$ **do**

▷ The following is done parallelly for all nodes u

$R(u) = \sum_{v \in B_u} \rho_v * R(v) * \beta + (1 - \beta) * \frac{1}{d_u}$

▷ Main TrustRank Formula

end while

In this work we leverage TrustRank to find bad nodes by using a set of oracle provided bad nodes as seed set. As a result, the resulting rank will be higher for bad nodes and lower for good nodes. Intuitively one can think of this as **DistrustRank**.

$$R(u) = \sum_{v \in B_u} \rho_v * R(v) * \beta + (1 - \beta) * \frac{1}{d_u} \quad (1)$$

where:

u = a node

B_u = the set of u's backlinks

ρ_v = weightage of $R(v)$ based on the number of forward links of page v and its corresponding edge value between node v and u

β = damping factor

d_u = node u's initialized value at start based on the provided seed set

3.1.1 Handling Leakage

Nodes with no outlinks tend to accumulate the incoming values and cause leakage. This is solved in this work by **connecting those nodes back to all the bad nodes with equal weightage**. That way, the accumulated values are propagated back to the bad nodes equally.

3.2. Pregel

To make the method more scalable, we make use of a toy version of a distributed-computing framework developed by Google called **Pregel**. The intended use of the framework is for it to **handle extremely large directed graphs across a cluster of computers**. The crux of the framework is that **each node can operate independently by maintaining its state, receiving messages from its neighbours and updating its state accordingly**. While the current dataset doesn't call for such a scalable framework, this is a demonstration of Pregel's efficacy. It must be noted that in this code each thread acts simulated the presence of multiple clusters of computers and each thread handles the communication of a subset of nodes

3.3. Hyperparameters

- 'num_threads' : 4
- 'damping_factor' : 0.85
- 'iterations' : 100

4. Results

The code takes a directed graph as input and computes the scores for each node in the graph. The output of the algorithm is a list of **DistrustRank** scores, one for each node in the graph.

While there is no ground-truth data available for us to compare our results against, we can check the ranks of the oracle-provided bad node which must be high if the algorithm worked perfectly.

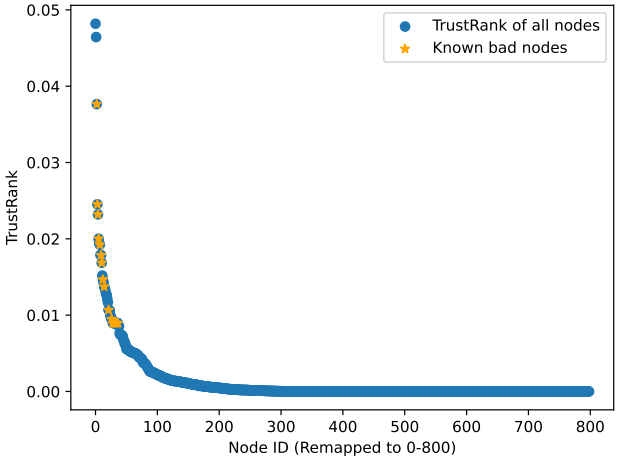

Figure 1. All nodes are arranged in the descending order of their scores and bad nodes are highlighted.

From figure 1 we can see that the majority of the ground-truth bad nodes have a higher score. This **shows that the algorithm has worked as intended**.

Figure 2 shows that most of the nodes have ranks between 0.00 and 0.01 while a few nodes have higher ranks indicating that the majority of the nodes are good nodes excluding a specific few.

Figure 3 shows the iteration-wise TrustRank of each node and also highlights the bad nodes. We can again observe that most of the bad nodes are of a high rank similar to what we saw in the Figure 1 earlier. But **we can also notice that there are some nodes which have a much**

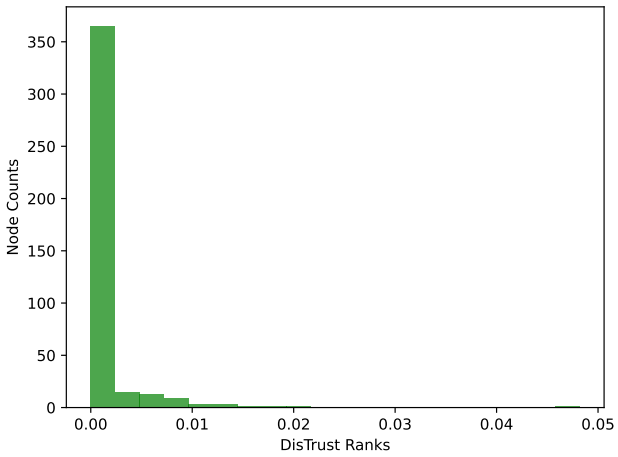

Figure 2. Histogram of the ranks of all the nodes.

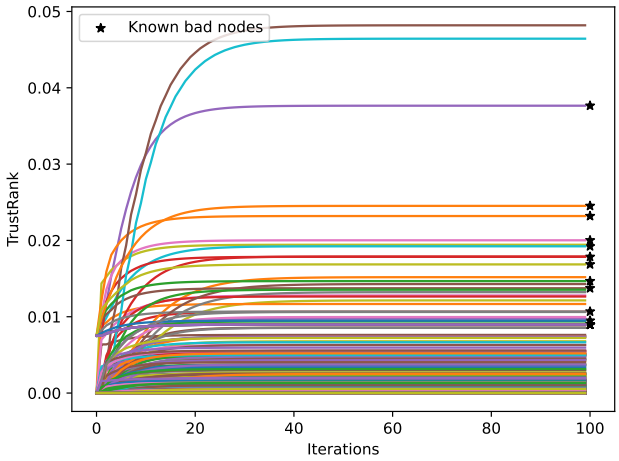

Figure 3. Iteration-wise scores of all the nodes and bad nodes are highlighted.

higher score than the bad nodes. These are the predicted bad nodes and the authorities should start with probing these iron dealers for fraud.