Trustrank using Pregel Framework

Anirudh Srinivasan
CS20BTECH11059

Haritha R
AI20BTECH11010

Nelakuditi Rahul Naga
AI20BTECH11029

Rahul V
AI20BTECH11030

Saketh Vaddamani
CS20BTECH11054

1. Problem Statement

We need to find the Trust scores of various dealers in the given dataset by implementing the Trustrank algorithm using Pregel framework. We first discuss in brief about the Trustrank algorithm (in general, not only fraud detection) and Pregel framework.

1.1. TrustRank Algorithm

Search engines utilise the TrustRank algorithm to assess a website's credibility. Researchers at Stanford University and Yahoo! created it in the beginning as a solution to the growing issue of internet spam.

The fundamental premise of TrustRank is to locate other trustworthy sites in the network by first identifying a group of reliable seed sites. The system evaluates the connections between the sites and rates each one according to how close it is to the trusted seed sites. The greater a site's TrustRank rating, the nearer it is to a reliable seed site.

The TrustRank algorithm also considers additional elements that may have an impact on a site's credibility, including the site's age, the calibre of its material, and the standing of its publisher or owner. TrustRank is able to recognise websites that are probably trustworthy and ranks them higher in search results by fusing these elements with link analysis.

The ability of the TrustRank algorithm to identify and penalise websites that attempt to influence search results by adopting spammy techniques like link farms or keyword stuffing is one of its main advantages. Even though these sites may have a lot of links pointing at them, the TrustRank algorithm will detect this and lower the site's overall score if the links come from low-quality or unreliable websites.

In conclusion, the TrustRank algorithm uses an analysis of linkages between websites and other variables that may have an impact on a site's trustworthiness to determine which websites are trustworthy. Utilising TrustRank, search engines are able to fight spam and other forms of online manipulation while simultaneously giving users more precise and pertinent search results.

1.2. Pregel Framework

Google created the Pregel framework as a computational approach for handling massive graph data. In 2010, Grzegorz Malewicz, Matthew H. Austern, Aart J.C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski published a paper in which they introduced it.

The Pregel framework is made to solve graph processing issues that are too big to fit in a single machine's memory. The Bulk Synchronous Parallel (BSP) computing model, which entails breaking the computation into supersteps and synchronising the data between them, forms the basis of this system. Pregel divides a graph into smaller subgraphs that are each processed by a different worker thread.

The Pregel framework's fault tolerance mechanism is one of its essential components. Without losing any information or progress, the framework is built to handle failures of either a single worker thread or of a whole computer. When a failure occurs, the framework automatically replays any lost messages and repeats the calculation on a different machine.

Numerous graph processing applications, such as social network analysis, recommendation engines, and online link analysis, have made use of the Pregel architecture. Processing graphs with billions of vertices and edges, which are typical in many contemporary applications, is where it excels.

In summary, the Pregel framework is a powerful tool for processing large-scale graph data using a simple and intuitive programming model. It provides fault tolerance and scalability features that make it well-suited for modern graph processing applications.

2. Description of the dataset

The dataset given consists of Iron dealers data in csv format. It has a Seller ID, representing a seller, Buyer ID representing a Buyer, and the Value of transaction represented by Value. An example of the format is shown below:

S.No.	Seller ID	Buyer ID	Value
1	1309	1011	1225513
2	1259	1011	1710778
3	1259	1011	1694993
4	1090	1079	538375

Table 1. Basic Format of the Data

We were also given bad ids, in a separate csv file with only bad ids present. So, we need to use these bad ids to implement Trust-rank algorithm and find trust scores of various other dealers.

	Seller ID	Buyer ID	Value
count	130535.000000	130535.000000	1.305350e+05
mean	1309.358287	1182.851258	6.930965e+05
std	294.435026	169.809657	5.696676e+05
min	1001.000000	1001.000000	1.000600e+04
25%	1078.000000	1060.000000	2.361085e+05
50%	1214.000000	1112.000000	5.571960e+05
75%	1488.000000	1276.000000	1.074405e+06
max	2190.000000	1887.000000	2.124000e+07

Table 2. Basic Statistics of the Data

The basic statistics of the data are as shown above 2. Other details about the dataset have been provided in the code that we have submitted.

3. Algorithms used

We are interested in implementing Trustrank algorithm on the given dataset using Pregel Framework. Pregel is a powerful framework oriented towards graph-based algorithms for distributed computing. Pregel is designed to scale very easily on large-scale computer clusters.

The input to our Pregel program are the Iron dealers dataset and the bad nodes dataset which have been described above. Note that the Iron dealers dataset can be thought of as a edge-weighted multi-directed graph with edges from Sellers to Buyers and transaction amount as corresponding edge weights.

We first create arrays of the Bad Nodes,Not-Bad Nodes and Nodes using the given datasets. Then we construct a weight matrix named **outgoing_weights** that stores the total amount of transaction between a Seller and a Buyer for all possible Seller and Buyer combinations.

For implementing the Trustrank algorithm, we first have to initialize the trust score for each node. We do the following initialization :

1. Initialize the trust score of each Bad node as 1.
2. Initialize the trust score of each Not-Bad node as 0.
3. Finally, normalize the trust scores array so that the trust scores sum up to one.

Along with the initialization of Trust-scores for each node (or) vertex, we also append the list of neighbouring (or) adjacent vertices (the vertices connected through an outgoing edge from the current vertex) for each vertex. As some nodes (or) vertices might not have any outgoing edges, for such nodes we make every node (including itself) as neighbours. This helps in avoiding the problem of accumulation of trust scores at some nodes. Each vertex also has an boolean attribute which determines whether it is active or not. Vertices start active, but change to inactive after the completion of it's assigned number of supersteps. The computation halts when every vertex is inactive.

Once the input graph is initialized in the above mentioned manner, the Pregel computation of Trust scores proceeds through a series of **supersteps**. During each superstep each vertex does two things :

1. The trust score of each node is updated. The updated trust score of the node depends upon on the trust score of the node at the end of the previous superstep, as well as the messages sent to the vertex during the previous superstep. More concretely, we update the trust score as we do in the Pagerank algorithm with damping factor d as 0.85. Note that the damping factor d denotes the probability that at any superstep, the messages will continue following the outgoing links/edges.
2. It sends messages to each of it's neighbouring (or) adjacent vertices. The messages a vertex sends depends on it's current trust score and the messages it has sent during the previous superstep. More concretely, if a vertex has outgoing edges, then it distributes it's trust score among it's neighbours proportional to the transaction amount between them. This was calculated and stored in the **outgoing_weights** matrix as mentioned earlier. On the contrary, if a vertex does not have any outgoing edges, then the trust score of the vertex is equally distributed between all the nodes (including itself) since for such nodes we made every node (including itself) as neighbours.

The above process has been repeated for 50 supersteps for each node to arrive at the final trust scores for all the nodes.

4. Final Results and Conclusions

Even though the we have used the terminology trust-scores repeatedly in the previous section, essentially the scores we finally get for the nodes represent the degree of badness of a node. This is because of the following reasons:

1. We have initialized the scores of bad nodes as one (before normalization) initially.
2. In a fraud network graph, the bad nodes predominantly tend to have transactions with bad nodes only. Hence, the outgoing (or) forward links in the given dataset are between bad nodes.

The final trust scores for all the nodes have been output in an ascending order in a csv file.

References

[1] M. Nielsen Pregel Github - Link

[2] M. Nielsen Pregel Blog - Link

[3] PageRank Wikipedia - Link