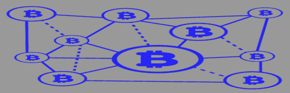
Forecasting Bitcoin Transactions

Raghav Pemmireddy, Maximo Moyer, Grant Sterman, Jacob Good

Team Name: Save Brown Squash

Introduction

Paper:
Bitcoin Transaction Forecasting with Deep Network Representation Learning, Wenqi Wei et al. 2020



Description:
We tried to predict transactions between bitcoin accounts using supervised deep learning on structured (network) data.

Implementation:

- Random walk and skipgram models to form embedding representations of nodes in a transaction network
- A prediction neural net to determine likelihood of a transaction between pairs of nodes given embeddings
- Test predictions using F1 accuracy

Usefulness:
Predictions of transactions between accounts can assist in fraud detection or price prediction for unregulated currencies such as Bitcoin

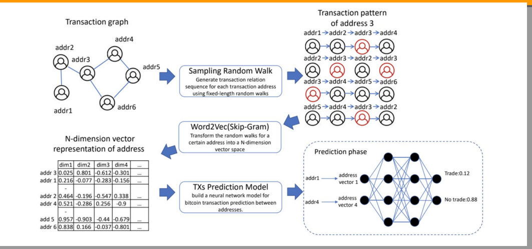
Data

Dataset:
We used the same bitcoin blockchain dataset as the paper, consisting of 280 million transactions. Due to time and memory limitations we ran the model on the first 100k sender-receiver pairs.

# blocks	508,241
# accounts	297,816,881
# transactions	298,325,122
# sender-receiver pairs	2,536,261,805

Figure 1: From implemented paper

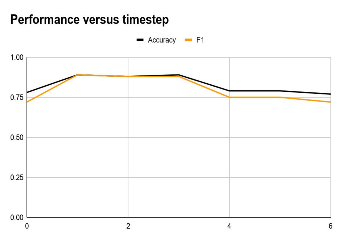
Architecture



Overview At Each Timestep:

1) PreProcess
   a) Create 10 graphs, one for each timestep, defining each timestep as the creation of 10k unique sender-receiver pairs.
2) Node2Vec:
   a) Create 10 random walks for each node in the graph, then decay, delete, and append appropriate random walks from previous timestep
   b) Create pairs of nodes using skip grams with window size 1.
   c) Train node embeddings using the skipgrams.
3) Predict:
   a) For a given time t, take in graphs t and t+1, embedding matrix from time t and the node ID to embedding ID dictionary from time t.
   b) Find nodes which have transactions, concatenate node embeddings, and label 1 (indicating a transaction).
   c) Negative sample transactions with label 0 to balance data.
   d) Train the prediction model on transactions in graph t, using 3 dense layers, first 2 with relu, last with softmax activation.
   e) Test our trained model on transactions in graph t+1.
4) Evaluation
   a) Accuracy and F1 score

Results



- We were only able to get embeddings for 8/10 timesteps and train on 7/10 timesteps due to training time constraints
- Mean Accuracy: .83
- Mean F1: .79
- Outperformed paper implementation
- Decreasing accuracy observed at later timesteps, curious if this would continue with more data

Issues

Problems

- Training time of random walks was extremely slow
- We were not able to implement the static and transaction amounts graphs due to time constraints
- Prediction Test Data was small given that there was not much carry over in nodes transactions between given pairs from one timestep to the next

Solution Attempts for Problems

- Optimized code and vectorized operations wherever possible
- Tried to work fast!
- We believe this also existed in the original paper, used negative sampling to double sample size

Future Work/Lessons

Models

- It would be great to implement the transaction amount and static graphs to see how the three models compare to one another

Data

- We would like to run our model on the last 100k sender receiver pairs to see if similarly effective when the character of the graph changes
- Train embeddings for more graphs at different timesteps so that overall there is more test data
- Run similar types of models on different kinds of financial data i.e. stocks, bonds, derivatives etc.
  - Could likely use a static graph model as accounts remain consistent across time

Learned

- Preprocessing data can be a more complex task than the actual model
- Quantitative prediction models require sound qualitative understanding of the topic at hand (in our case, Blockchain).