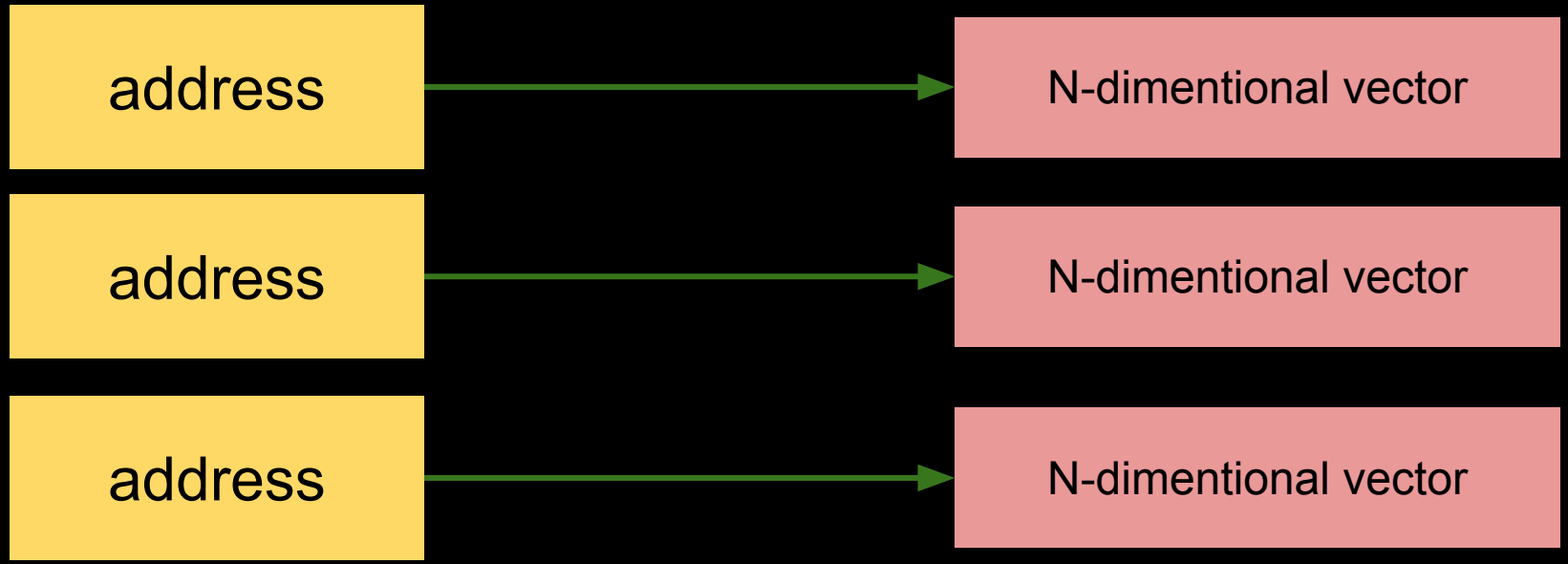
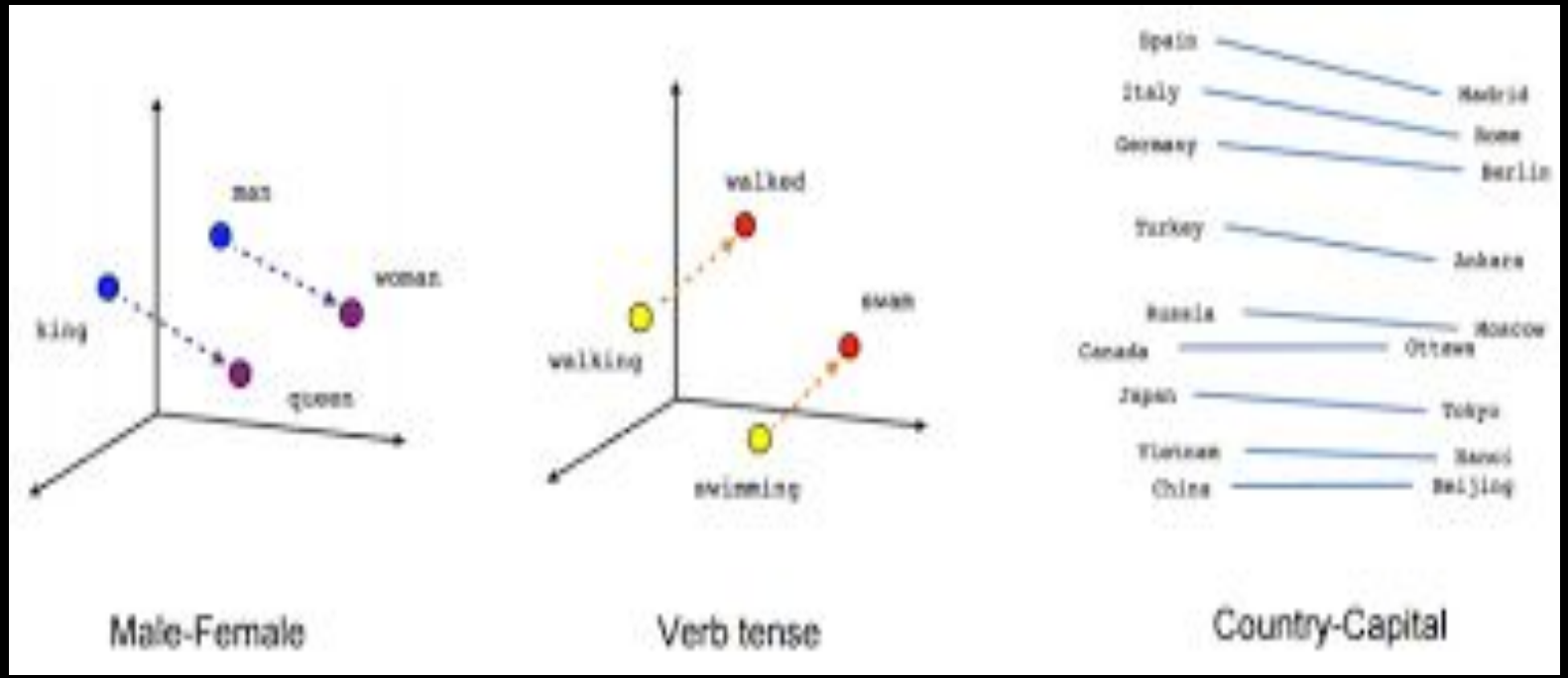
Bitcoin-address clustering using transaction graphs

The Problem statement

- The main objective is to map each address to a n-dimensional vector space and then train these vectors to reveal useful information.
- The training is based on transactions between addresses.
- The obtained vectors can be used for clustering address into trading hubs.
- The vectors can also be used to trace multiple addresses to a single user(user clustering)


address
address
address
N-dimentional vector
N-dimentional vector
N-dimentional vector

This is similar to word-embeddings

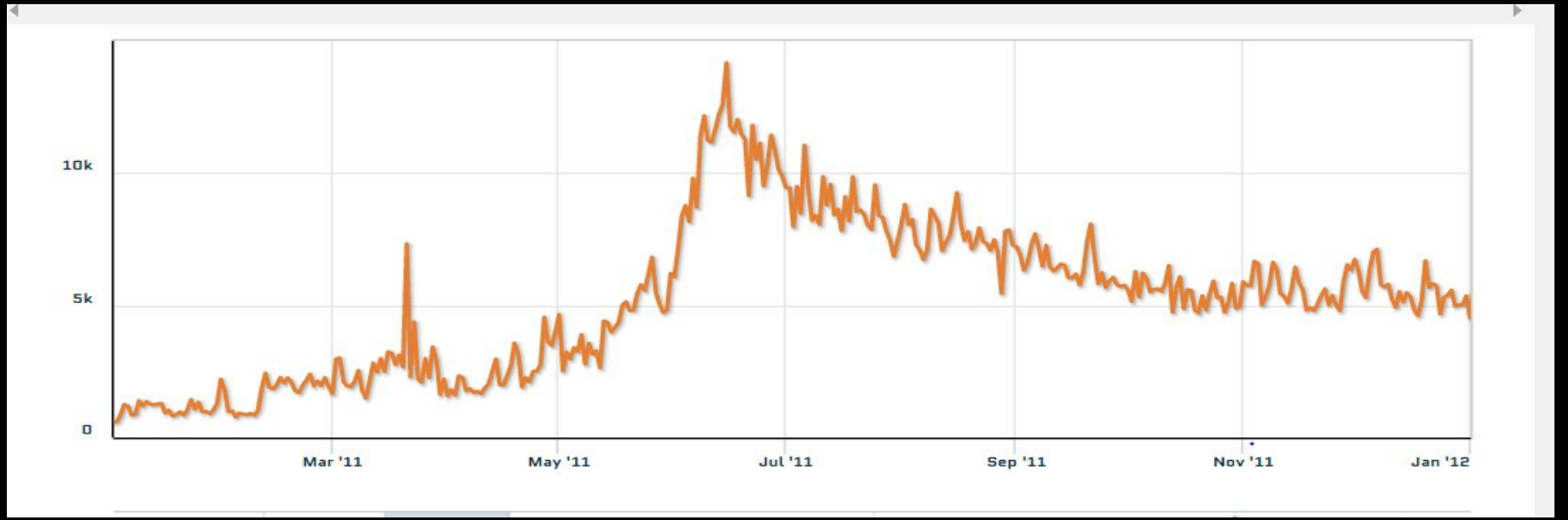


Data-extraction

- I use google Bigdata to query all my datasets.
- I have extracted transaction data for 2010 and 2011.
- The data extracted contains
- Sender address
- Receiver address
- Amount transferred
- (The data is grouped by sender and receiver so in case of multiple transactions the amount just adds up)

That’s a lot of transactions!!!!

- This graph show number of transactions per day.
- As you can see there have been even 15k transactions per day in 2011!!

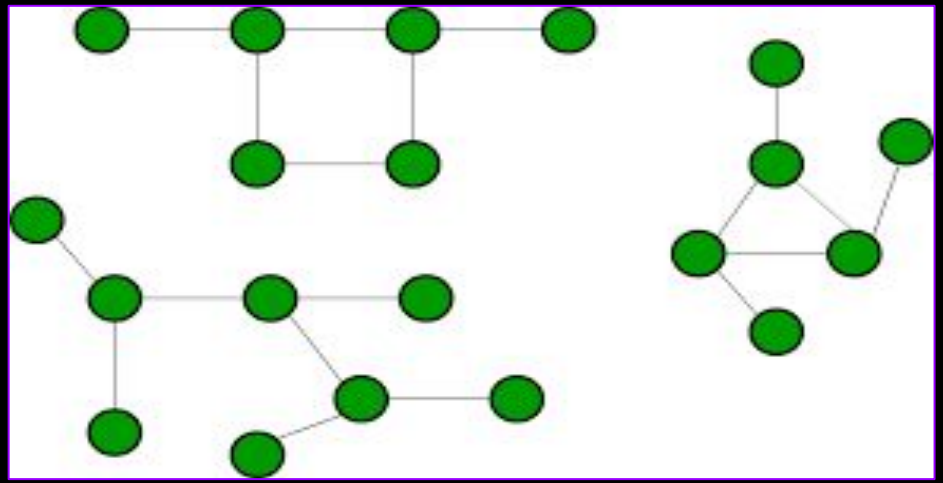


Pre-processing

- The data is now converted into graphs.(both directed and undirected)
- All the bitcoin addresses are used as nodes.
- Edges are weighted (the amount of bitcoins in the transaction)
- Two nodes are connected if either of them made a bitcoin transfer to the another (undirected)
- Two nodes “a” and “b” are connected if “a” made a bitcoin transfer to “b” (directed)

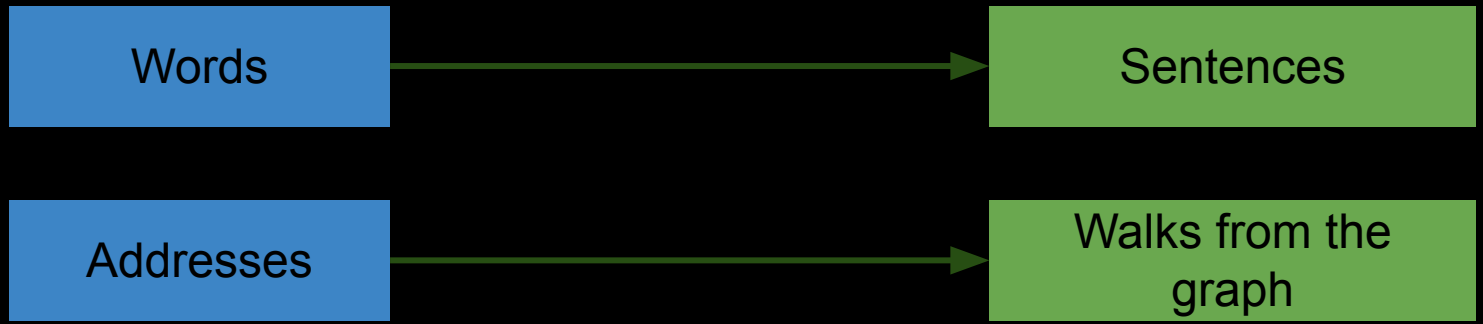
Finding connected components

- This step is done in order to optimize the machine learning algorithm.
- I separate all the strongly connected components of the graph.
- Now as there are no transactions between 2 components these components are fed to the machine learning algorithm separately.



Walking the graph!!!

- As mentioned before this task of training node vectors is similar to training word vectors
- Word vectors are trained on the basis of the sentence structure using either Skip-grams or CBOW model
- So my task boils down to creating sequences of nodes(addresses) that have some useful meaning (analogous to sentences for words)



Walks … DFS or BFS

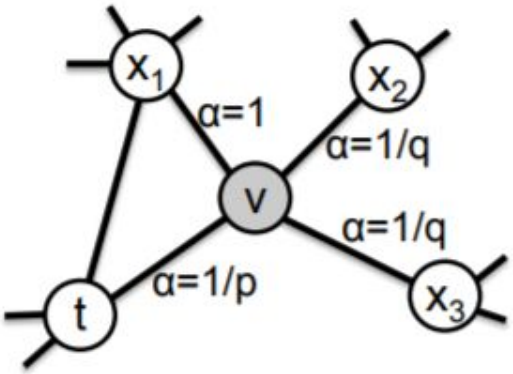
- As mentioned in this paper
- https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf
- “The breadth-first and depth-first sampling represent extreme scenarios in terms of the search space they explore leading to interesting implications on the learned representations”
- I use the solution mentioned in the same paper…..
- Random walks

Random walks

- The core idea is to assign a probability to the next possible nodes in the walk.
- This is a second order random walk as explained in the paper.
- Let's say we have traversed the edge “t-v”.
- Now from “v” we assign probabilities to all v-x.
- This is based on distances of all x to the

  t (previous node). Hence the second order.

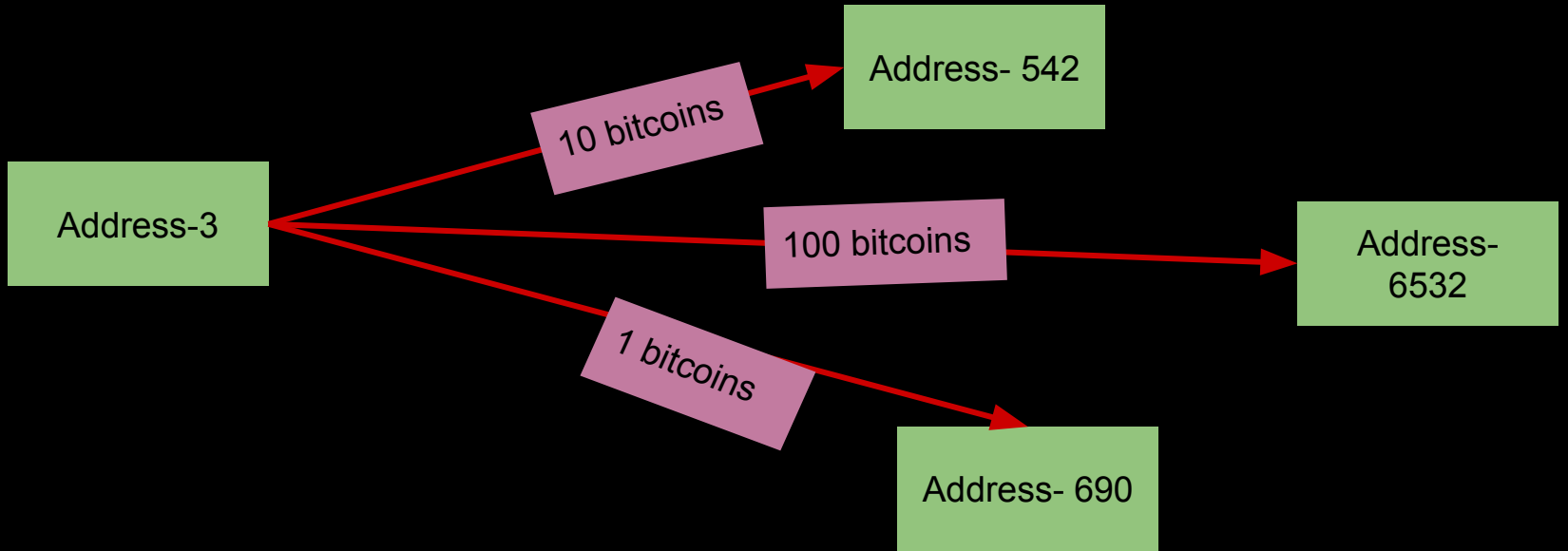
$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



- For each address
- Start a random walk of max length “n”. The sequence of addresses generated for each walk is stored.
- Each of these “walks” generated is analogous to a sentence in english language.
- The only difference….
- Sentence - Sequence of words that makes sense.
- Walk - sequence of addresses that makes sense(as the walks are based on the graph)

Follow the money!!!

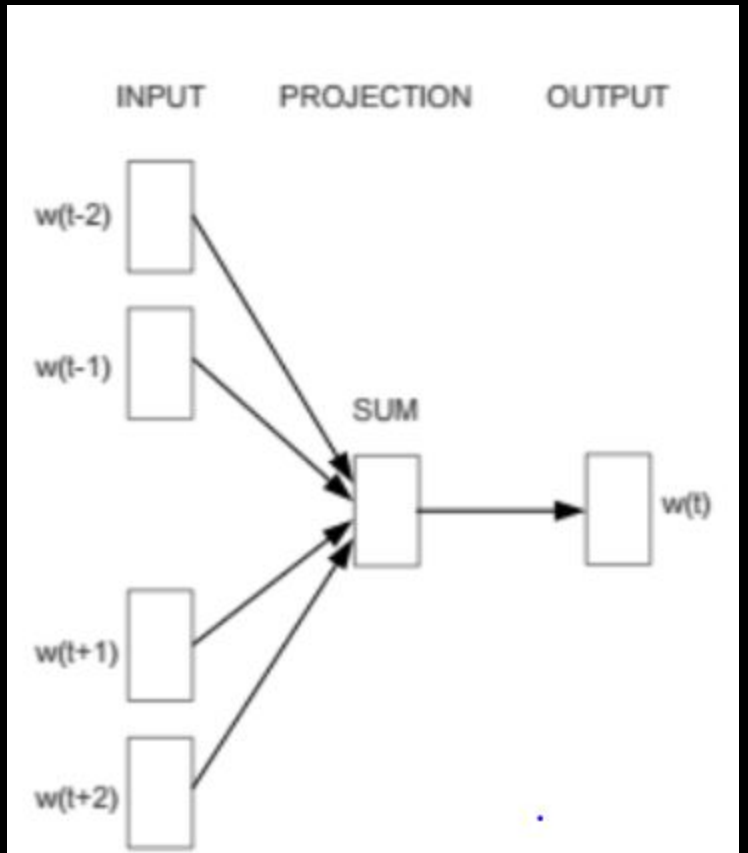
- In addition to the random walk discussed , I have made the probabilities biased towards larger transactions.



- In the above example assume that we are currently at “Address-3”
- Then the chances for the walk to choose “Address-6532” is more than the alternatives

Training

- Once we have the sequence of addresses all there is left is to put there through any word to vector model
- I use google previded “Gensim” for training.
- Each unique address as assigned a vector of 100 dimension(node-embedding).
- I use CBOW model to train these node-embeddings with a window size of 8



CBOW

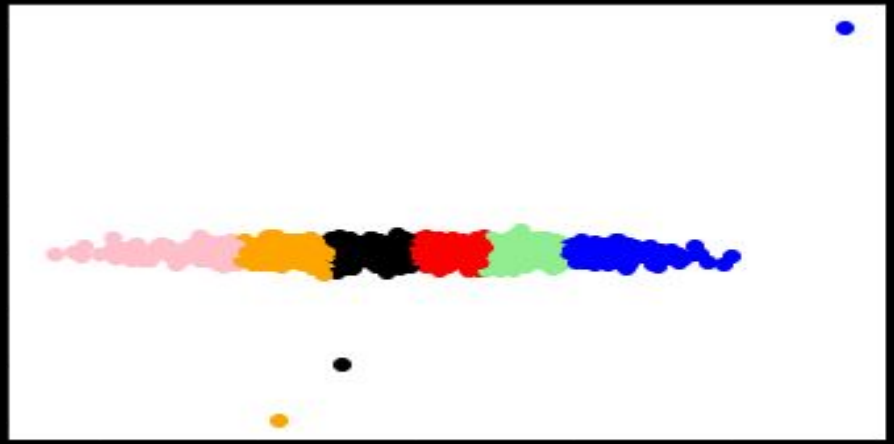
(Continuous Bag Of Words)

Results

- Much like word to vector i have trained node embeddings , these could be used for further applications like word-embedding are used in NLP tasks.
- As this project was unsupervised i don't have any score to report.
- But as the training method is the same as CBOW implemented by google gensim the results should be appropriately good.

Interaction graph clustering

- This is 2D representation of 100 dimensional space.
- This is clustering of 1970 nodes belonging to the same connected component.



- The above figure clusters all the nodes(addresses that have done a lot of bitcoin trading with each other)
- The cluster appears to be close due to following reasons…
- All the nodes belong to the same strongly connected component
- Its compressed using PCA from 100 dimension to 2.
- Scaling is off as few nodes in the component have very little trading to do with others

Citation

- I have used part of the code from
- https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf