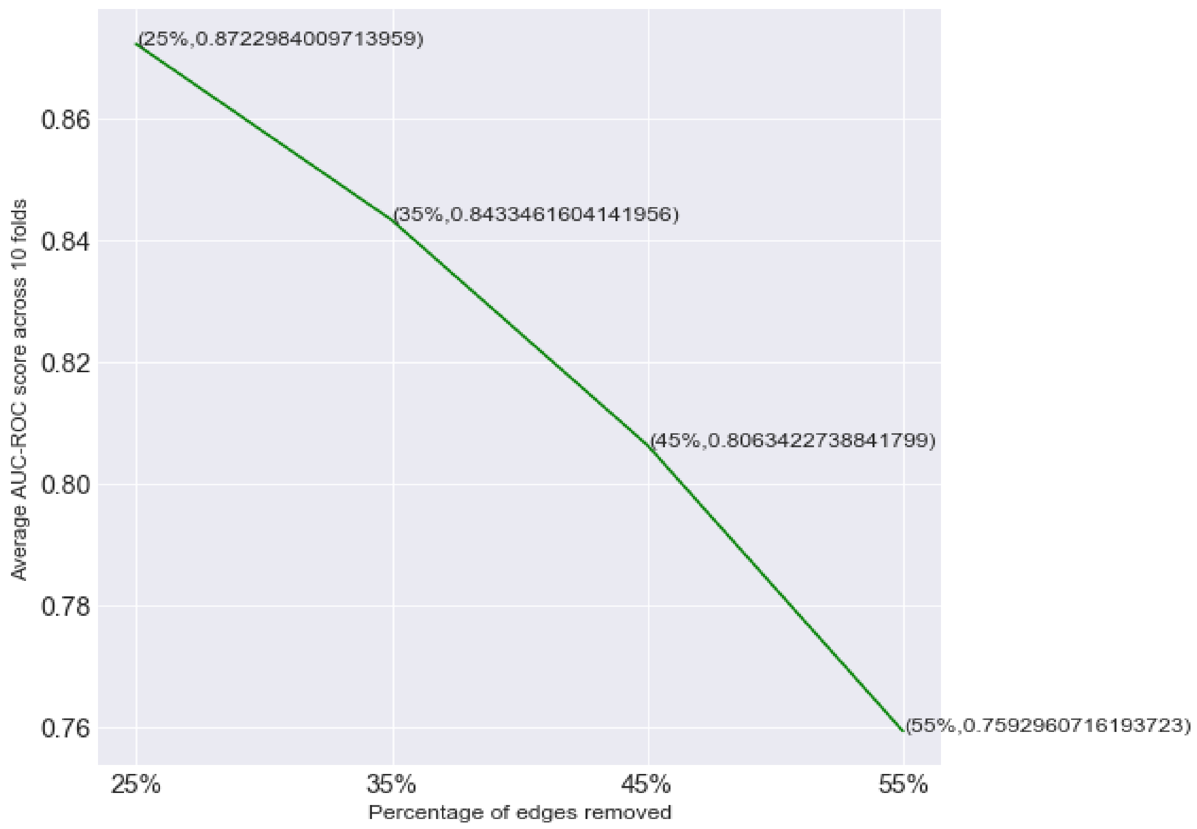
MLN ASSIGNMENT 4 REPORT

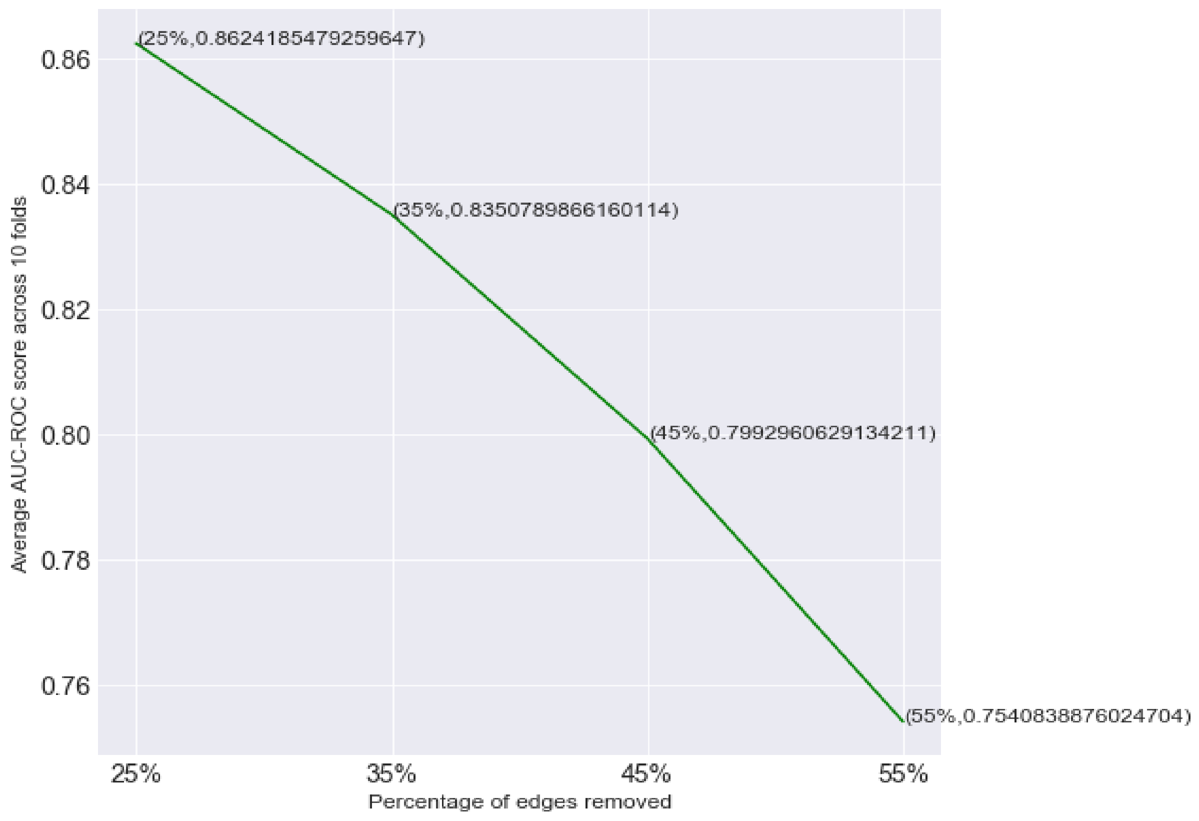
MT19021

Problem 1:

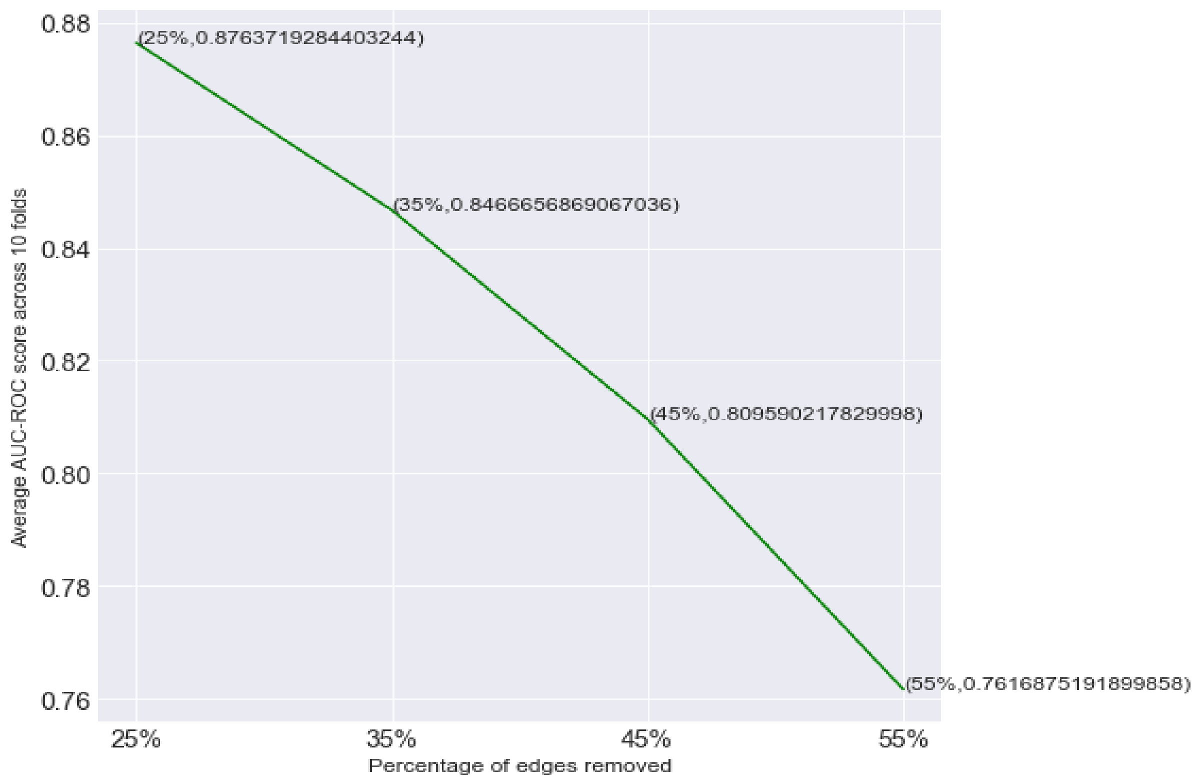
Plot using heuristics as Common neighbours



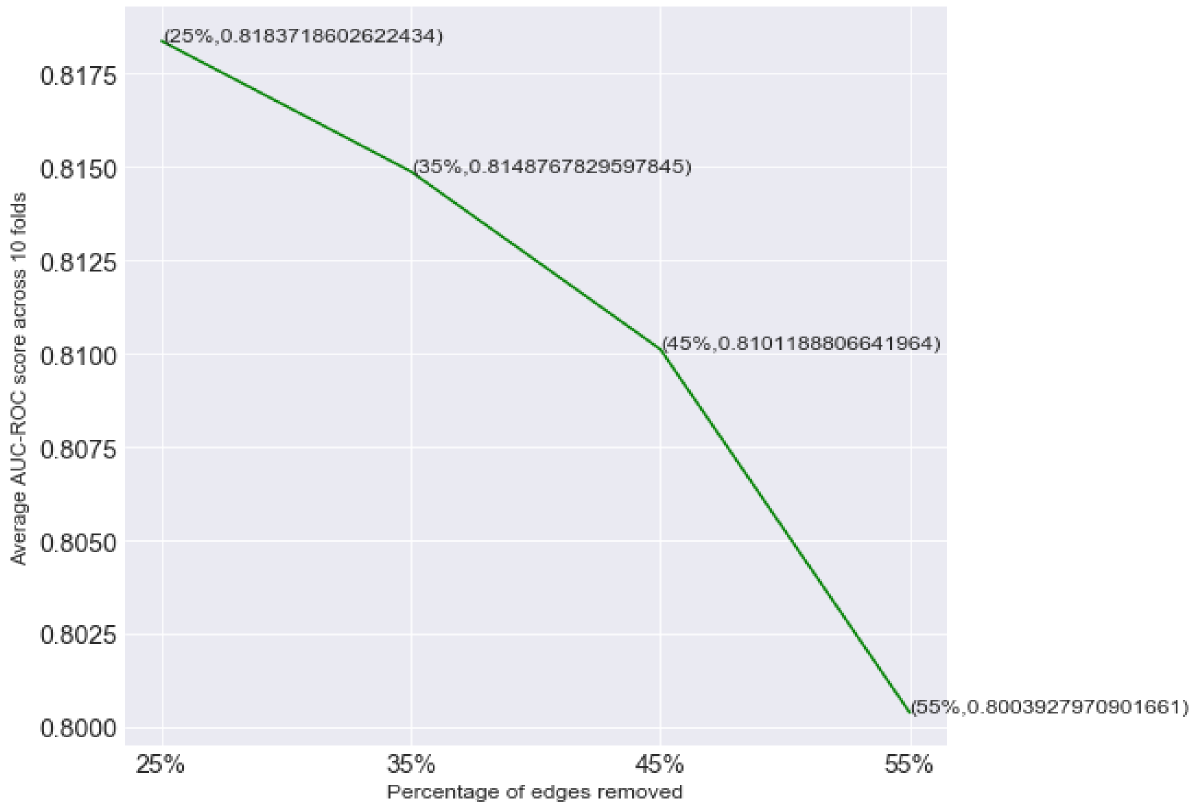
Plot using heuristics as Jaccard Index



Plot using heuristics as Adamic Adar index



Plot using heuristics as Preferential attachment



```
Using heuristic as Common neighbours
10 fold average ROC-AUC score on removing 25% of edges = 0.8722984009713959
10 fold average ROC-AUC score on removing 35% of edges = 0.8433461604141956
10 fold average ROC-AUC score on removing 45% of edges = 0.8063422738841799
10 fold average ROC-AUC score on removing 55% of edges = 0.7592960716193723

Using heuristic as Jaccard Index
10 fold average ROC-AUC score on removing 25% of edges = 0.8624185479259647
10 fold average ROC-AUC score on removing 35% of edges = 0.8350789866160114
10 fold average ROC-AUC score on removing 45% of edges = 0.7992960629134211
10 fold average ROC-AUC score on removing 55% of edges = 0.7540838876024704

Using heuristic as Adamic Adar index
10 fold average ROC-AUC score on removing 25% of edges = 0.8763719284403244
10 fold average ROC-AUC score on removing 35% of edges = 0.8466656869067036
10 fold average ROC-AUC score on removing 45% of edges = 0.809590217829998
10 fold average ROC-AUC score on removing 55% of edges = 0.7616875191899858

Using heuristic as Preferential attachment
10 fold average ROC-AUC score on removing 25% of edges = 0.8183718602622434
10 fold average ROC-AUC score on removing 35% of edges = 0.8148767829597845
10 fold average ROC-AUC score on removing 45% of edges = 0.8101188806641964
10 fold average ROC-AUC score on removing 55% of edges = 0.8003927970901661
```

It is observed that for all 4 Link Prediction heuristics, the average ROC-AUC score drops when the percentage of edges randomly removed from the original graph is increased.

Also, the 3 heuristics common neighbor, Jaccard and Adamic Adar have quite similar average ROC-AUC score for 10 folds while the score for preferential attachment heuristic is lower.

Problem 2a:

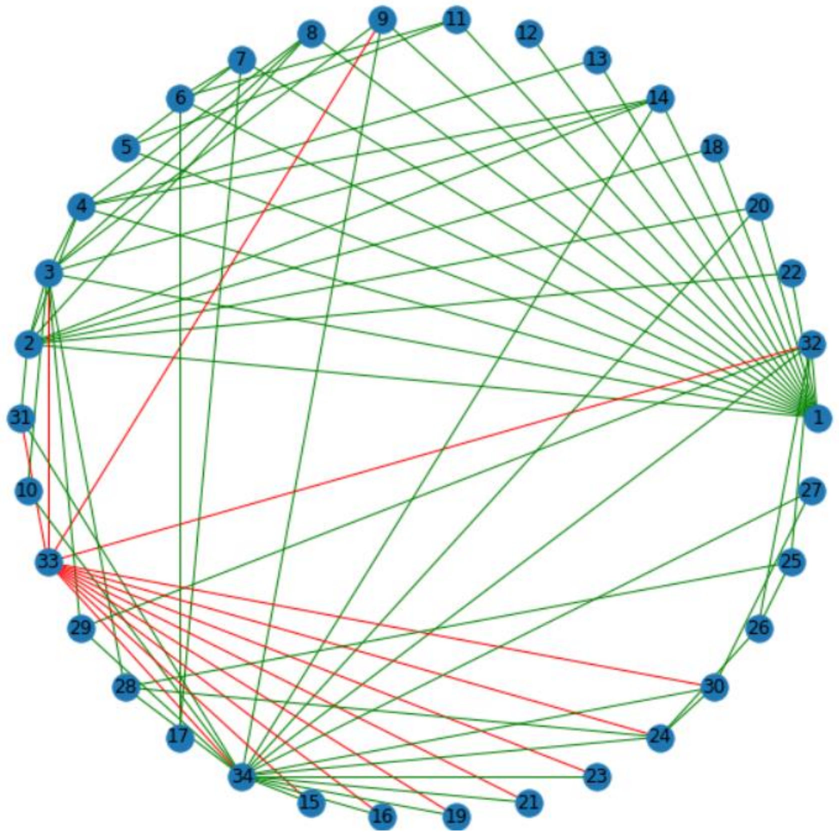


Figure: Visualisation of graph with edges of node 33 coloured red

Degree of node 33 = 12

Node '33' is having edges with nodes that are mostly connected to only one other node i.e. node '34' i.e. those nodes which have degree = 2. Also, '33' is connected to node '9' which is connected to node '1' therefore node '33' is also acting as a bridge between the 2 communities.

Problem2b: DeepWalk is a specialised case of node2vec where we do unbiased random walks instead of the biased random walk. The biasness is introduced in node2vec due to the 2 parameters p(return parameter) and q(in-out parameter). Now we can do DeepWalk by using p=1 and q=1.

In this way the walker has equal probability of exploring both the local structure (i.e. communities that is taken care of by p) and global structures (i.e. structural similarity that is taken care of by q).

Problem 2c:
The list of the five nodes generated with the lowest shortest path distances to node 33 is:

23

34

30

21

16

Here we need to explore local substructure of node ‘33’ more as we need to find cluster to which node ‘33’ belongs as that cluster will only contain the nodes with the shortest path to node ‘33’. For this we need to have a higher value of 1/p as compared to the value of 1/q where p is the return parameter and q is the in-out parameter. Thus we do more BFS kind of walk as compared to DFS kind of walk.

Therefore, p value needs to be set lower than q while doing the random walk in node2vec and also $0 < p < 1$ and $q > 1$. Here I have used p=0.1 and q=1

I tried some other values also of p and q such that $p < q$. But, setting these values gave good results as all the 5 nodes returned are directly connected to node 33.

Problem 2d: Here degree of node 34 = 17

The list of 5 nodes generated along with their degrees is:

node 33 with degree = 12

node 32 with degree = 6

node 3 with degree = 10

node 2 with degree = 9

node 1 with degree = 16

Here I have set p=1 and q=0.5

We need to do more of DFS here as the structural similarity needs to be found. And less of BFS type of walk needs to be done comparatively. Structural similarity is found by making the random walker visit nodes which are far from the target node as they may have structure similar to that of structure node.

i.e. setting 1/q > 1/p

Therefore, p needs to be more than q and also 0 < q <1 and p >=1

Problem 2e:

The Node2Vec representation of node ‘3’ has the closest L2 (Euclidean) distance from node 33.
The degree of node 33 = 12
The degree of node 3 = 10

Yes. This is what is expected. Here the degree of node 3 is quite similar to the degree of node 33 i.e. both of them have high degrees.
Here, we have kept p=100 and q=0.01 i.e. p > q. Thus, during random walk more of global structure will be explored and nodes that are far more from node 33 but have structures (like degree of node) similar to node 33 will be visited.

Problem 3:
On using message function as:

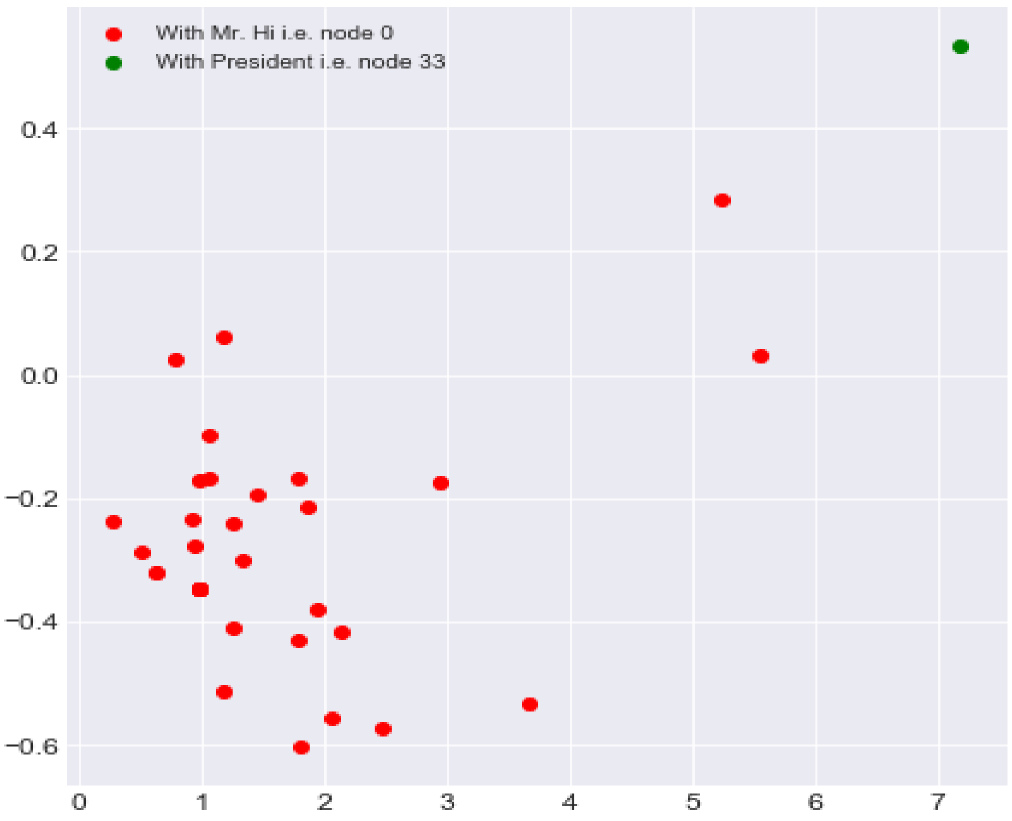
$$m_{j\to i} = h_j$$

And reduce function as:

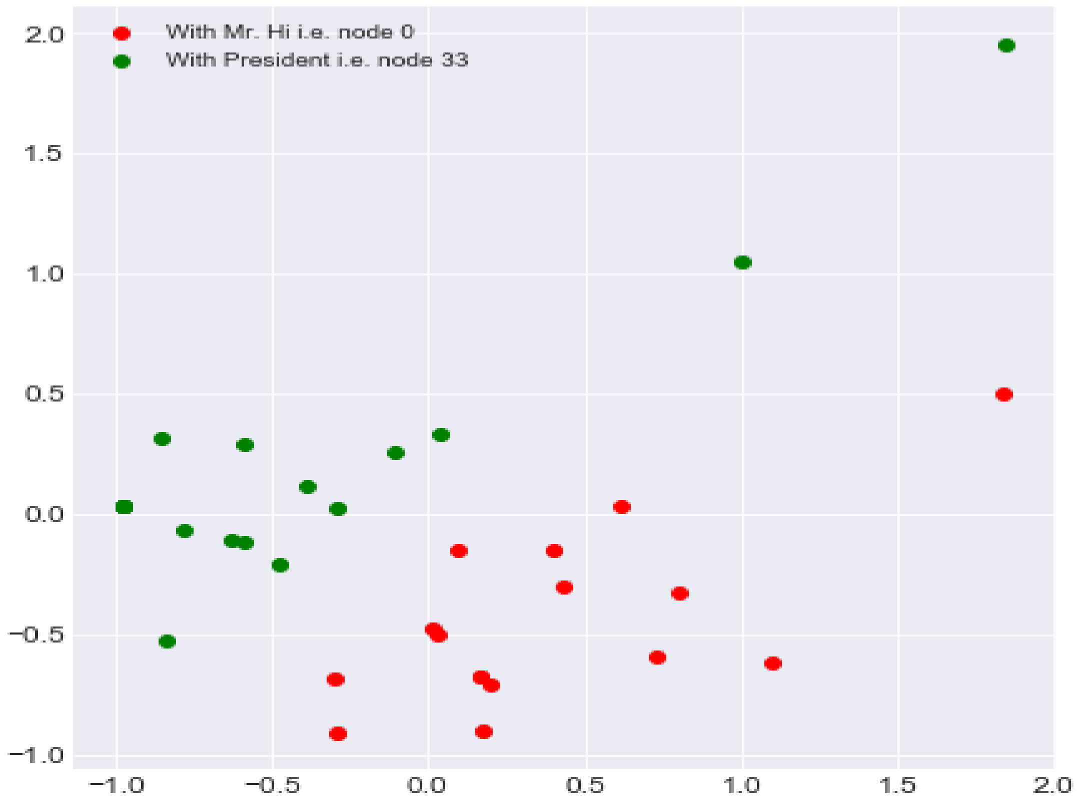
$$\tilde{h_i} = \sum_{j\in\mathcal{N}(i)} m_{j\to i}$$

The training is being done for 300 epochs. The node embeddings at 5 intermediate epochs is are shown below. It can be seen how the clusters are gradually formed and the embeddings move closer to each other and finally 2 communities are formed.

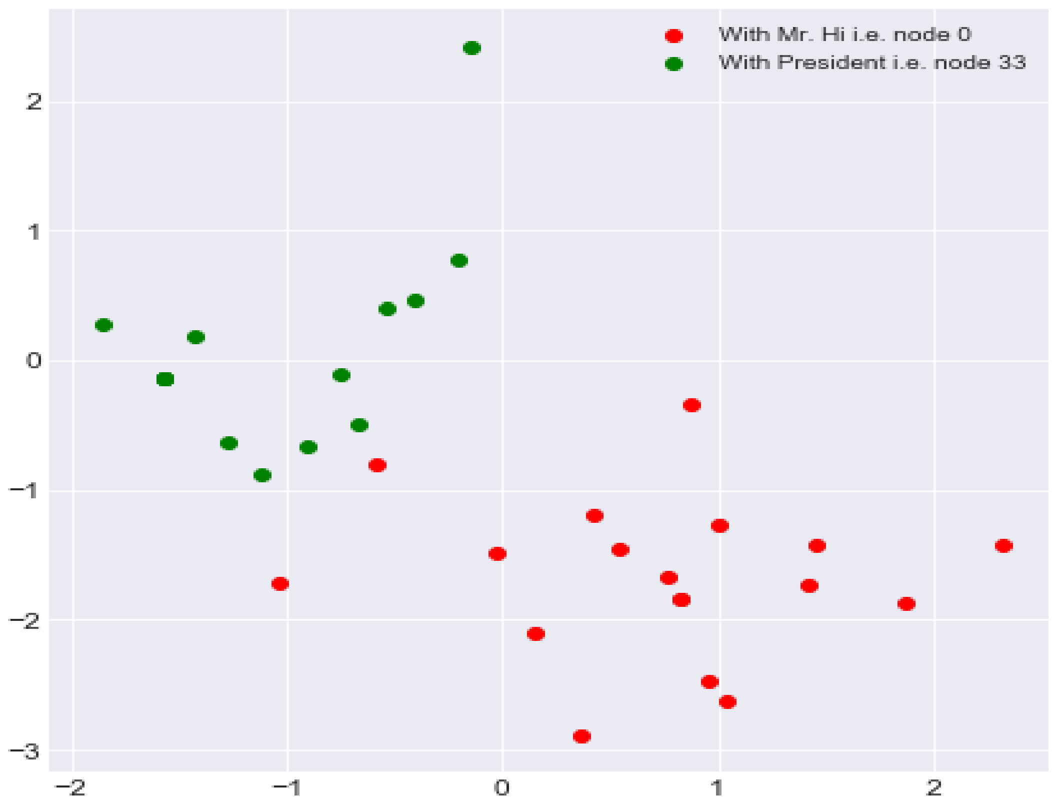
At epoch 0:



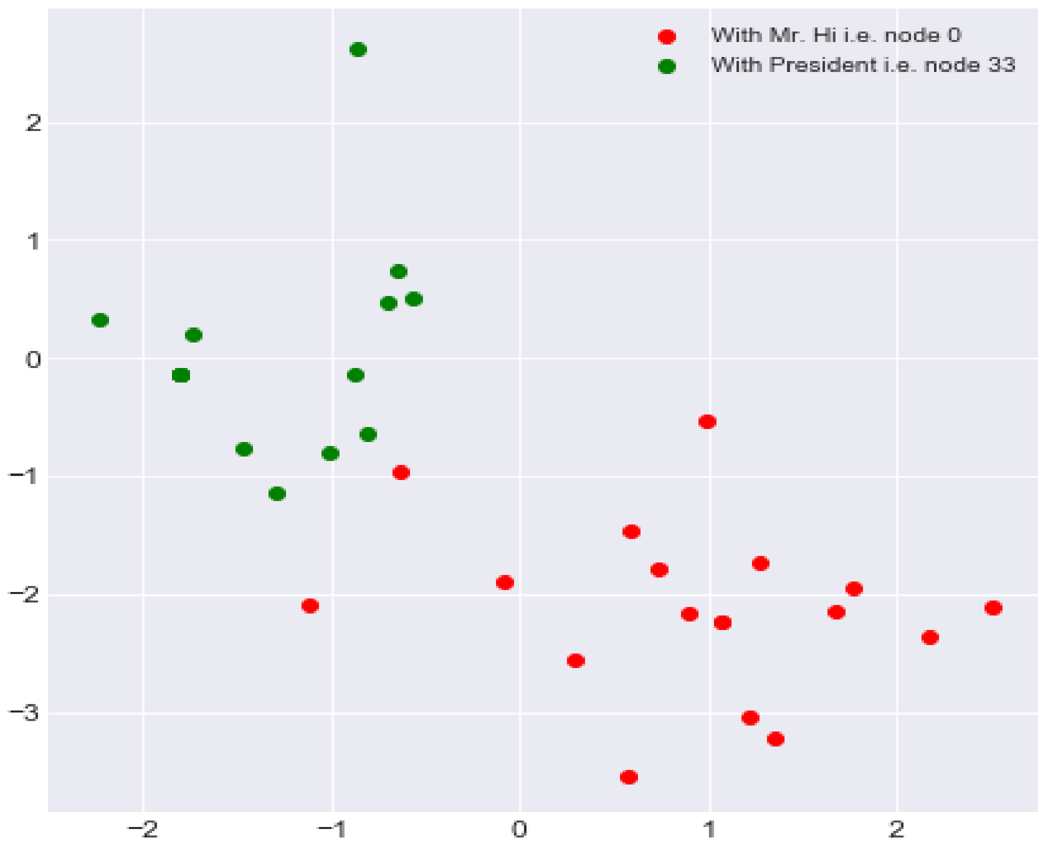
At epoch 69:



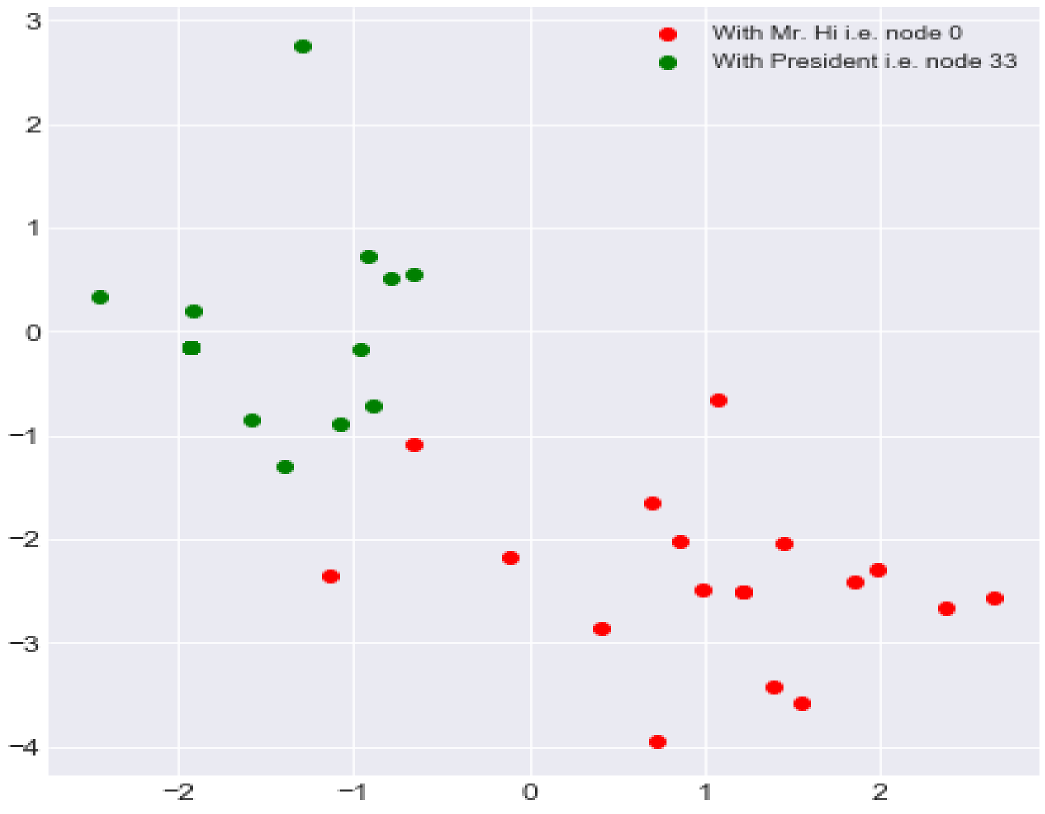
At epoch 138:



At epoch 207:



At epoch 276:



Using message function as:

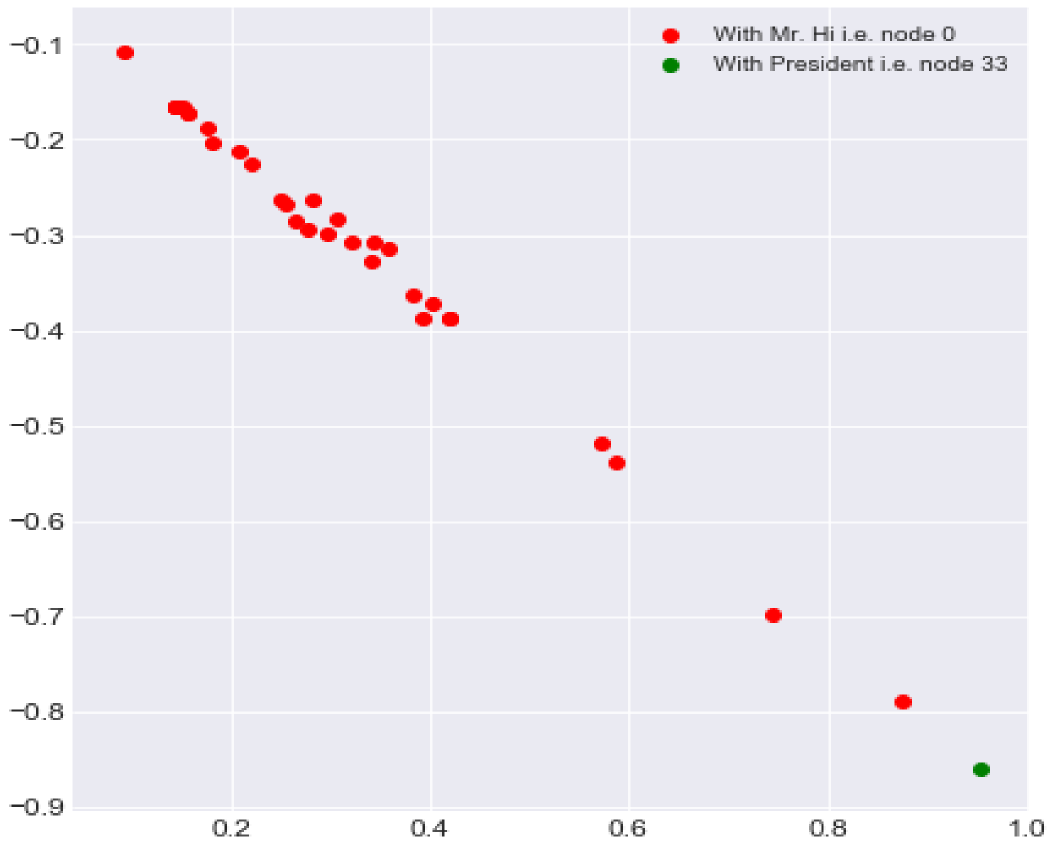
$$m_{j\rightarrow i} = \frac{1}{\sqrt{d_j}} h_j$$

And reduce function as:

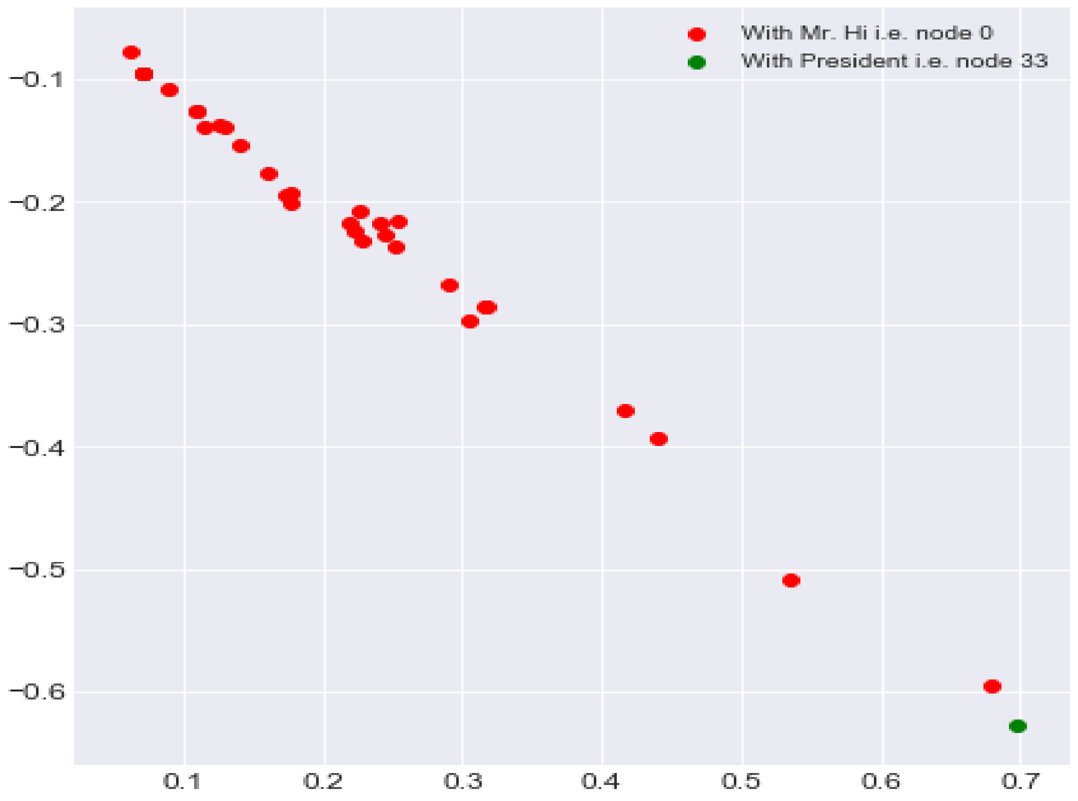
$$\tilde{h_i} = \frac{1}{\sqrt{d_i}} \sum_{j\in\mathcal{N}(i)} m_{j\rightarrow i}$$

The training is being done for 300 epochs. The node embeddings at 5 intermediate epochs is are shown below. It can be seen how the clusters are gradually formed and the embeddings move closer to each other.

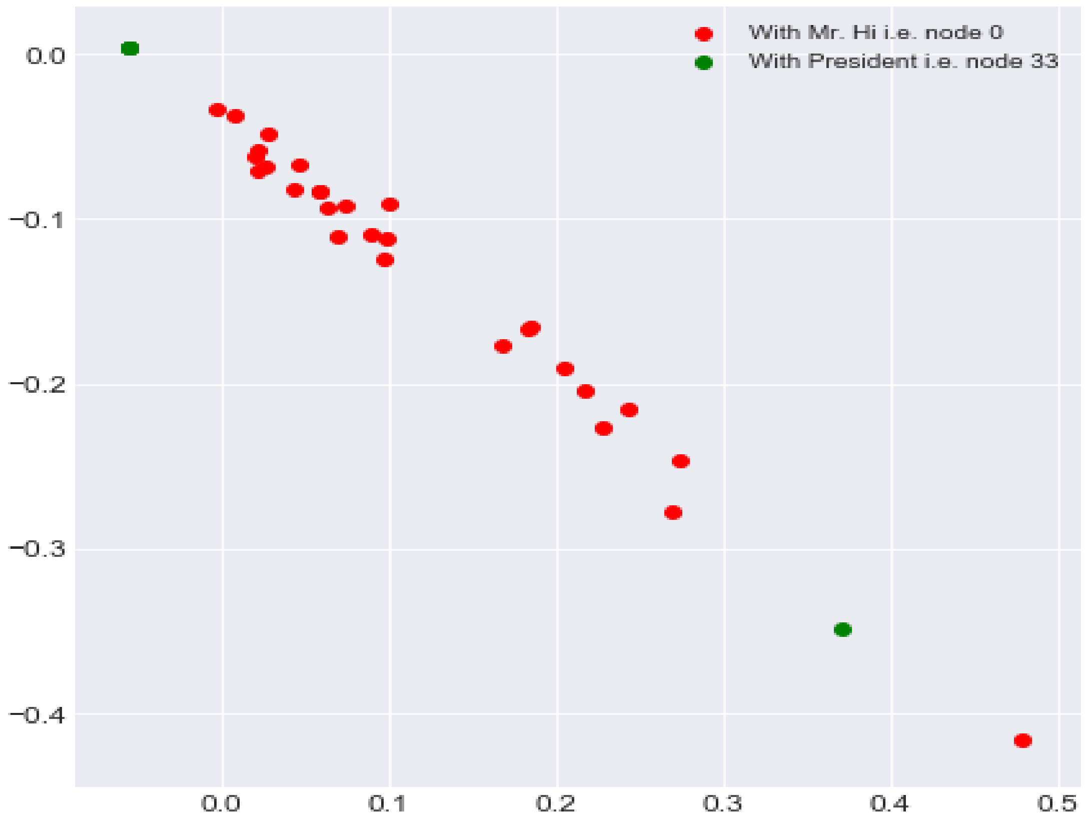
<u>At epoch 0:</u>



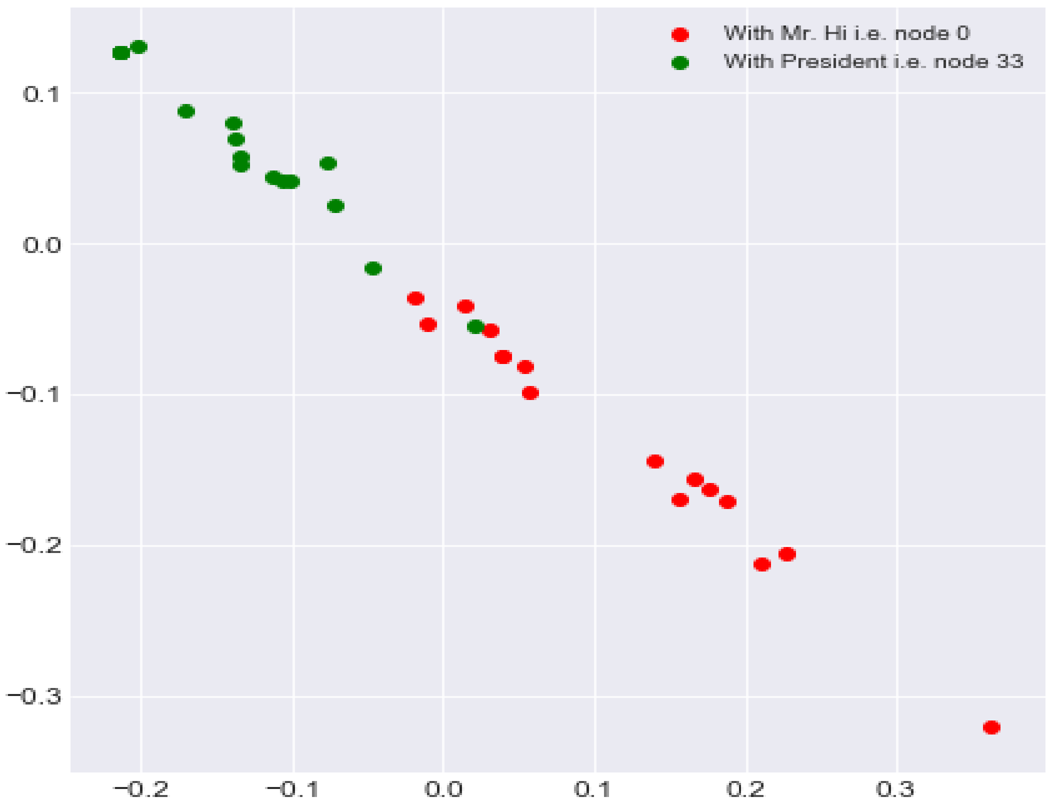
At epoch 69:



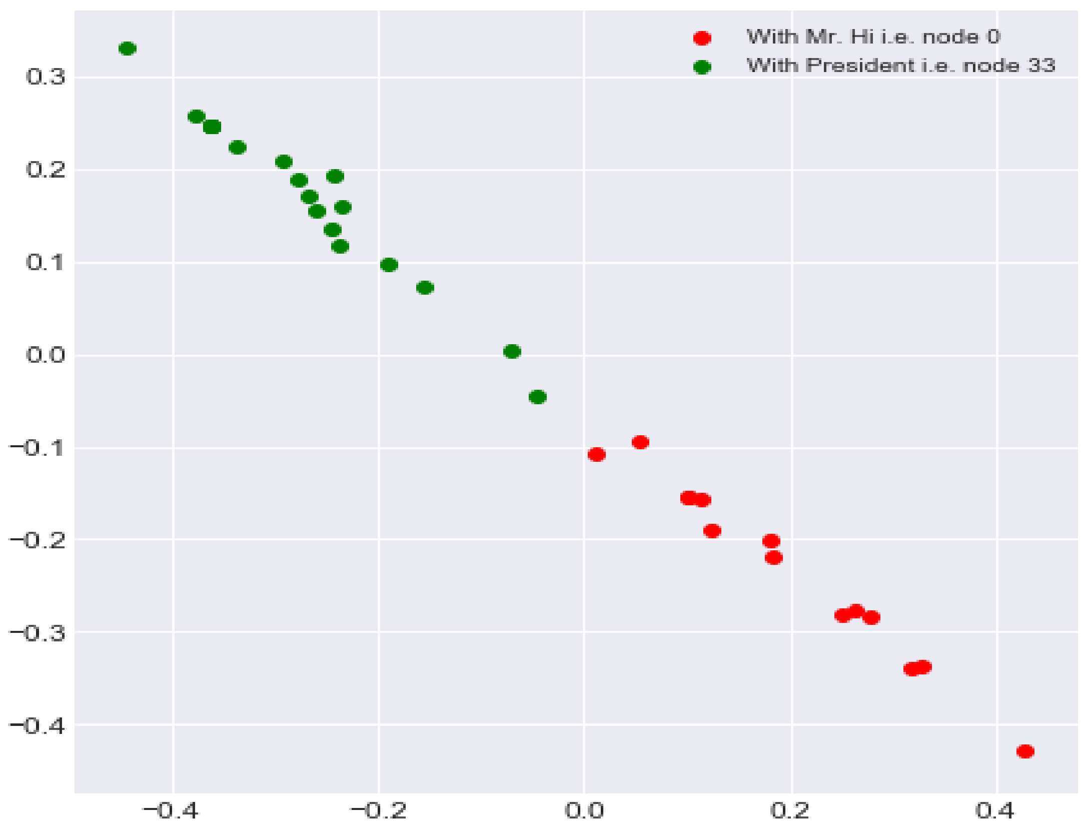
At epoch 138:



At epoch 207:



At epoch 276:


With Mr. Hi i.e. node 0
With President i.e. node 33
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
−0.4
−0.2
0.0
0.2
0.4