Analyzing Citation Networks

Ameya Rathod

February 18, 2024

All tasks including the bonus task have been completed. This report includes complete details about the findings, conclusions and methodologies for the same.

1 Introduction

This report is based on the citation network created based on the Arxiv HEP-PH(high-energy physics phenomenology) papers. The dataset is in form of a directed graph consisting of 34,546 papers with 421,578 edges. If a paper i cites j, then we have a directed edge from i to j. The dataset consists of papers from 1992 to 2002.

Two tasks have been completed that include a graph exploration task, a community detection or clustering task and finally a bonus link prediction task. Each section deals with a particular task in order.

2 Graph Exploration

2.1 Building a graph

We use a built-in python library ”networkx” to create a directed graph from the dataset provided. The dataset is very large to visualize as a graph. However, figure 1 shows a network having about 25% of the total edges of the dataset. The graph is rendered using Gephi after creating a .gexf file using networkx.

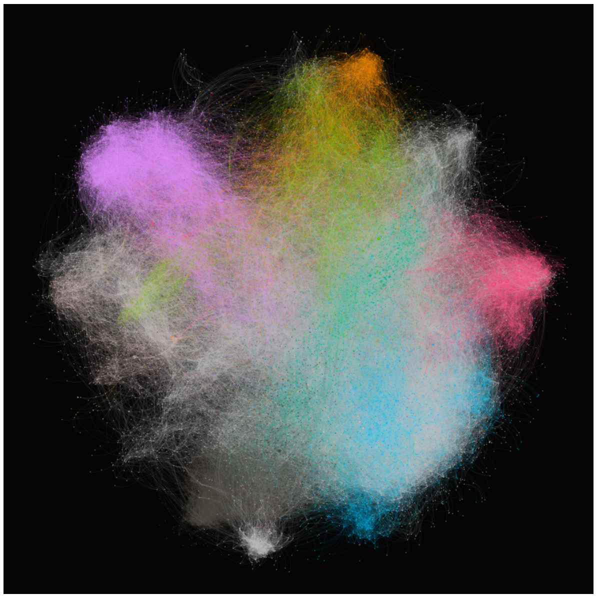


Figure 1: Network having 25% edges

2.2 Graph Analysis

Graph analysis can be done using a lot of factors: diameter, density, centrality, strongly connected components, in-degrees and out-degrees. But before going into that we try to find out some trends that can be seen as each year progresses.

2.2.1 Trend of increase/decrease in number of papers over years

As expected, the number of papers submitted in a year grew with time from 1992 to 2001. For doing this part, we code up a program that went through the cit-HepPh-dates.txt(downloaded from this site and available in the Datasets folder on the GitHub repository) and counted the frequency of papers submitted in each year from 1992 to 2002. For plotting, we use the matplotlib.pyplot library. Figure 2 represents the plot we get from the dataset.

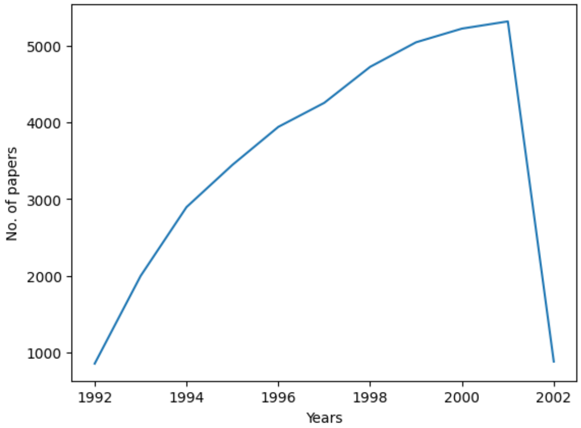


Figure 2: Number of papers submitted v/s the year of submission

The sudden drop in 2002 is seen because the dataset has papers submitted until March only for the year 2002. However in order to predict number of papers submitted in 2002, it is possible to train ML models to predict the same. However they won't be accurate for predicting for many years after 2002 due to less data points to train the model. Prediction for year 2002 is expected to be highly accurate due to data point availability near year 2002. An example for this is provided in PaperPredictor.ipynb in the GitHub repository. The prediction for the year 2002 is 5999 papers which is very close to the actual number as seen here.

2.2.2 Trend of increase/decrease in number of citations over years

The number of citations in a year is basically the sum of number of citations in all the papers submitted that year. The metric tells us about the extent of previous research work used for research done that year. This is done using the dataset cit-HepPh.txt (downloaded from this site) which has all the edges present in the network. Each paper maps to a year in a dictionary and we get all the citations yearwise in a list. The plot for the same is shown in Figure 3.

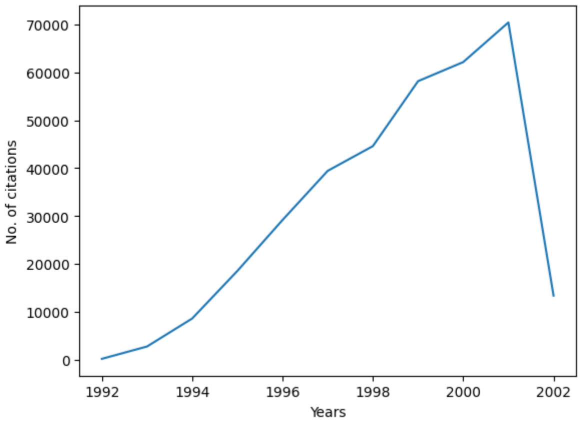


Figure 3: Number of citations in the year v/s the year

The decrease in year 2002 is again because of limitation of papers in dataset until March 2002 only. However the trend is different from that of number of papers, that trend was almost quadratic ($ax^2 + bx + c$ with negative a, positive b and negative c with a, b being very very large in magnitude). This trend on the other hand is almost linear as we can clearly see. This implies that increase in number of citations is not that much as compared to increase in number of papers which implies that new research work which isn't much related to previous work is being carried out over years.

2.2.3 Density of the network

We perform temporal analysis on the density of the network using the density function provided by networkx module. This involved building cumulative graphs over the years from 1992 to 2002.
The density of a network is defined as:

$$Density = \frac{Total Edges in the Graph}{Number of Possible Edges}$$

Let m be the number of edges in the graph and n be the number of nodes in the graph. Then,

$$Density For Directed = \frac{m}{n(n-1)} \quad (1)$$

$$Density For Undirected = \frac{m}{n(n-1)/2} \quad (2)$$

The trend over years for this citation network is as shown in Figure 4.

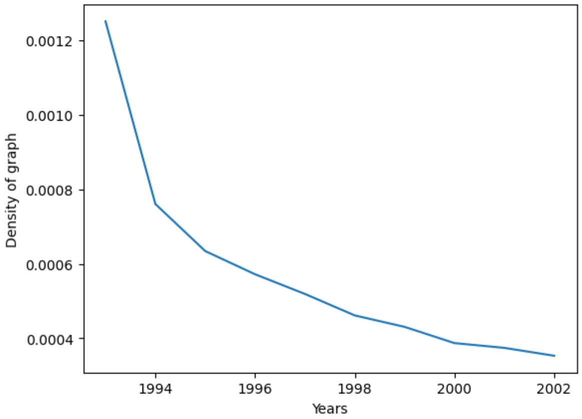


Figure 4: Density of citation network over years

Let us analyze this plot. The decrease in density for the citation network over years is expected as number of papers(nodes) and number of citations(edges) in real-world networks do not increase as much as they can. Max number of edges possible in a graph is of the order n^2 where n is the number of nodes. So the denominator in 1 and 2 grows by a huge factor while the numerator does not grow as much as the denominator, leading to this decrease.
Thus we can say that real-world citation networks grow in number of nodes and edges over years however the density decreases over the years.

2.2.4 In-degree Centrality

The in-degree centrality of a node in a graph is given by :

$$IndegreeCentrality_v = \frac{IncomingEdges_v}{TotalNodes - 1} \tag{3}$$

This metric can tell us which paper has directly affected research for most papers in the citation network over years. Again, we use a built-in function from networkx that returns the in-degree centrality for a given node. An interesting insight we can get from this metric is as follows:

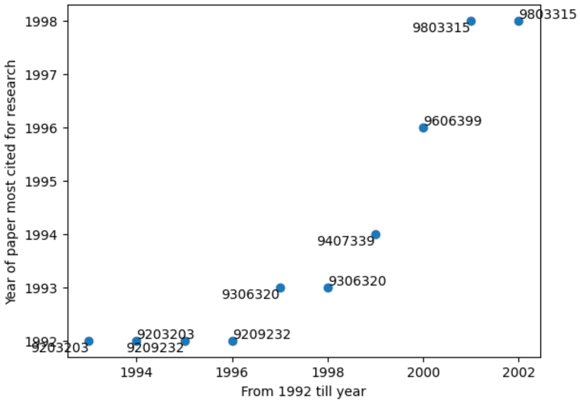


Figure 5: Year of most cited paper v/s the year

An important point to note here is that the scatter plot never goes down on the y-axis with increasing x. This tells us that most of the research that happens upto a year tend to use related work that is done closer in the past, which shows that there is build-up on past work. Branching into new domain from past work is very rare.
Another point to note is that until year 1998, it took about 2 years for the most cited paper until that time to change. However, this changed after 1998, from which we can deduce that the amount of diverse research increased and the amount overall might have increased leading to such results.

2.2.5 Closeness Centrality

Closeness centrality of a node in a graph can be calculated as follows:

$$ClosenessCentrality_v = \frac{1}{avg_d(u\ to\ v)} \tag{4}$$

where $u \epsilon U$ and U is the set of all nodes that can reach v. An insight we can draw from this metric is how much does a particular paper which is directly or indirectly cited by other papers affect the research in that domain on its own. Figure 6 shows year of paper with maximum closeness centrality in the citation network over years.

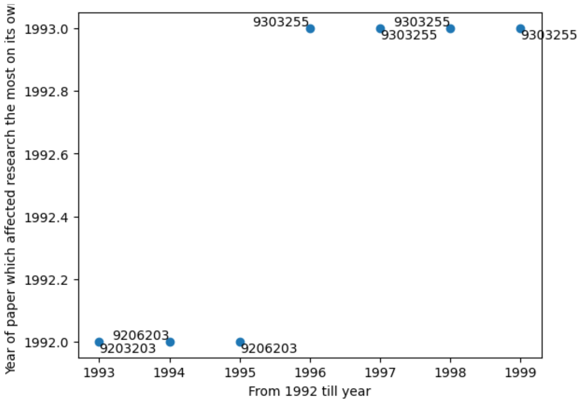


Figure 6: Year of paper which affected research most on its own v/s the year

As we can see from equation 4, closeness centrality depends on two factors - no. of direct/indirect connections to the node and length of those connections. As we can see from the plot, max. closeness centrality has changed only once over 7 years. This shows that a paper which influenced research a lot for a year would continue to be the most influential paper over a long time period. It is not easy for a paper to suddenly influence many papers over time. We can approximate influential papers with this metric. [1]

2.2.6 Eigenvector Centrality

It is not just the centrality of a node that is important to decide how influential a paper is in the network. It is important to consider how influential a node's neighbours are as well in order to decide how influential a community is as a whole. Eigenvector centrality is one such measure that can help us with this. Eigenvector centrality of a node v is the v^{th} element of a vector x s.t :

$$Ax = \lambda x \tag{5}$$

where A is the adjacency matrix and λ is some constant. All entries of some x are definitely positive as per the Perron-Frobenius theorem if λ is the largest eigenvalue for A. Figure 7 shows year of paper with maximum eigenvector centrality in the citation network over years.

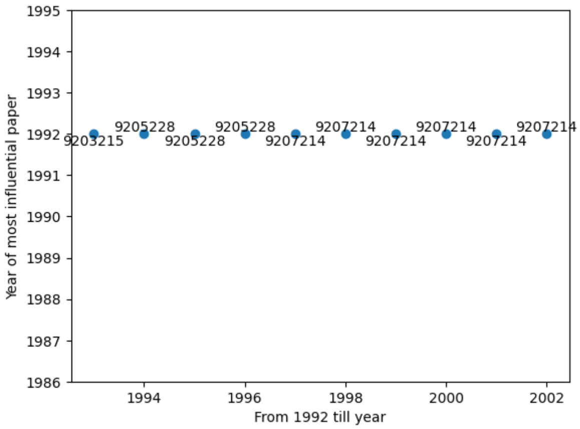


Figure 7: Year of most influential paper considering neighbour's influence v/s the year

[1] It is still important to note that this metric might be misleading. Consider the case that a paper has been cited by just one paper and that paper has indegree of 0, this means that the closeness centrality for that paper would be 1. But the paper is not that influential as it has been cited by just one paper. However for large and dense real-world networks like this, it is a rare case to happen. For this network, max. closeness centrality ranges from 0.03 to 0.15 from 1993 to 1999.

As mentioned previously, we can calculate centrality for nodes using centrality of the node's neighbours from this metric. Figure 7 shows that from 1993 to 2002, only papers from year 1992 have neighbour's that have been highly influential. Papers which cited those papers gained importance over the next few years. There are just two changes in the paper with highest eigenvector centrality from 1993 to 2002.
Till 1993, paper ID 9203215 was cited by the most influential papers in the network.
From 1994 to 1996, paper ID 9205228 was cited by the most influential papers in the network.
From 1997 to 2002, paper ID 9207214 was cited by the most influential papers in the network.

2.2.7 Strongly Connected Components

A strongly connected component in a directed graph is any subgraph such that starting from any node in the subgraph, you can reach any other node in the subgraph.
A strongly connected component in a citation network can only exist if there is some cycle in the graph. In a citation network, for every edge $u \to v$, it means that paper u cites paper v which would mean that paper v was submitted before paper u. This implies that a citation network cannot have cycles. However on analysis of the graph, it is found that the citation network does have cycles.
Some of the reasons as to why cycles can exist are as follows:

- Collaborative research, shared topics and mutual influence leading to local cycles
- Citing of different versions of the same paper

Figure 8 shows the number of strongly connected components in the citation network over years.

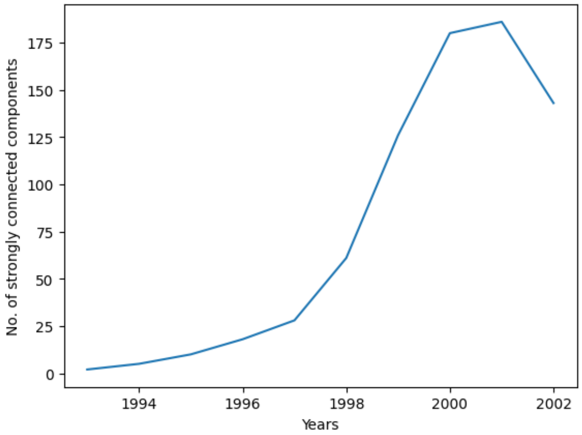


Figure 8: No. of strongly connected components in the network v/s the year

As we can observe, number of strongly connected components in the citation network increases over the years as new papers get added to the network. This can be because of increase in small to medium sized local cycles in the citation network. The decrease in the number of strongly connected components from 2001 to 2002 can be because of merging of two different strongly connected components.
This can again happen when there is some kind of interdisciplinary research happening along with citation of different versions of the same paper.
Another thing to observe is the number of strongly connected components. The number is around 5-180 over years from 1993 to 2002. This number is very small as compared to the size of the entire network.

2.2.8 Number of papers v/s the number of citations

For the final citation network, we plot the number of papers v/s the number of citations. Figure 9 shows this plot.

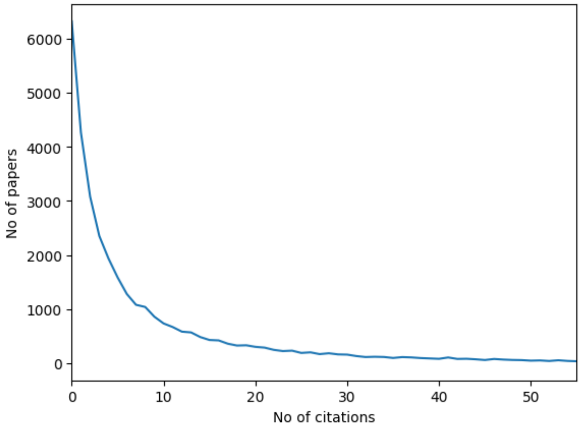


Figure 9: Number of papers v/s the number of citations

As we can clearly see from the above plot, the number of papers with lesser citations are more, while number of papers with higher number of citations are less. Papers with higher citations maybe having higher importance for related research work, the plot tells that such papers are lesser in number.

2.2.9 Number of papers v/s the number of references

For the final citation network, we plot the number of papers v/s the number of references used. Figure 10 shows this plot.

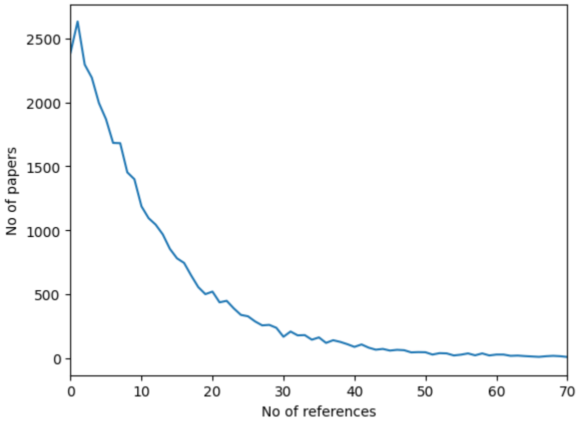


Figure 10: Number of papers v/s the number of references

As we can observe from this plot, most papers refer to about 2-4 papers and papers with higher references are lesser (overall a decreasing trend after reaching global maxima). Papers with no references also exist. It is possible that these papers are based on some new research from the scratch.

3 Community Detection and Clustering

Community detection and clustering are methods to identify communities in a graph that are more closely connected to each other among themselves and sparsely connected to other nodes. We use two methods to perform this task - one is a classic algorithm, Louvain community detection and the other one is based on ML - a Graph Neural Network that gives node embeddings in some lower dimensional space.

3.1 Louvain Community Detection

3.1.1 What is the algorithm?

The algorithm is based on modularity measure M:

$$M_{undirected} = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j) \tag{6}$$

The change in modularity by adding a node i to community C for undirected graph is given by:

$$\Delta Q_{undirected} = \frac{d_i^C}{2m} - \frac{\sum_{tot} d_i}{2m^2} \tag{7}$$

This modularity can be extended to that for directed graph (as mentioned in this paper) as:

$$M_{directed} = \frac{1}{m}\sum_{i,j}[A_{ij} - \frac{d_i^{in} d_j^{out}}{m}]\delta(c_i, c_j) \tag{8}$$

So, the change in modularity for adding node i to community C is as:

$$\Delta Q_{directed} = \frac{d_i^C}{m} - \frac{\sum_{in} d_i^{out} + \sum_{out} d_i^{in}}{m^2} \tag{9}$$

- We initially assign one node to each community so initially, number of communities = number of nodes.
- We iterate through all the nodes, for each node, we try to assign that node to the community of each neighbour. If the modularity change overall after this is positive, we put the node in the community of that neighbour.
- We continue to do this as long as there is no gain in modularity for all the nodes.
- After the above process is done, we get the communities of nodes.

An important thing to note is there is no overlapping of communities. One node belongs to one community only.

3.1.2 Why this algorithm?

The algorithm is simple, easy to understand and even easy to implement. It is a greedy algorithm and effective because it tries to maximize the modularity for each assignment we try.

3.1.3 Implementation and results

A naive implementation of this algorithm (which is basically the Girvan-Newmann algorithm [2]) is provided in CommDetection.ipynb on the GitHub repository. A max of 10 iterations happen to see if there can be an increase in modularity. Figure 11 shows the number of communities in the citation network v/s the year.

We perform temporal analysis on the citation network from 1992 to 1999 by observing community growth and

[2] Girvan-Newmann algorithm is O(m^2n) complexity where m is number of edges and n is number of nodes, Louvain is an improvement over this with complexity of O($nlogn$)

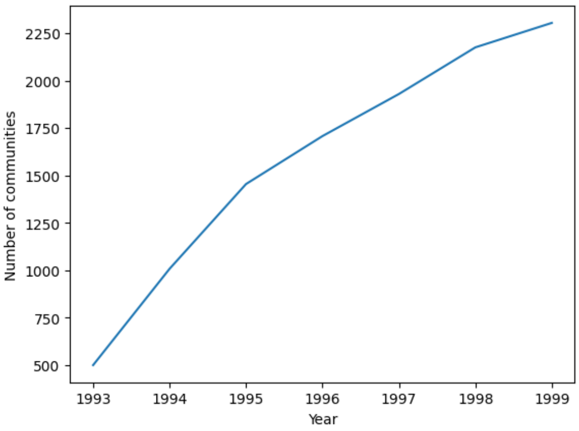


Figure 11: Number of communities v/s the year (Girvan-Newmann implementation)

As we can see, the number of communities increases over years as expected. However the rate of increase decreases over years, which tells us that the overall community structure is stabilizing meaning that there is maturity in the field of study. Establishment of new subfields may occur but formation of entirely new communities slows down.
This might also point towards increased specialized research among existing communities.

Figure 12 shows a plot of number of communities v/s the year in the citation network as given by the in-built louvain communities function in networkx. This algorithm is a time-efficient implementation of the Girvan-Newmann algorithm which can be used for large networks as well.

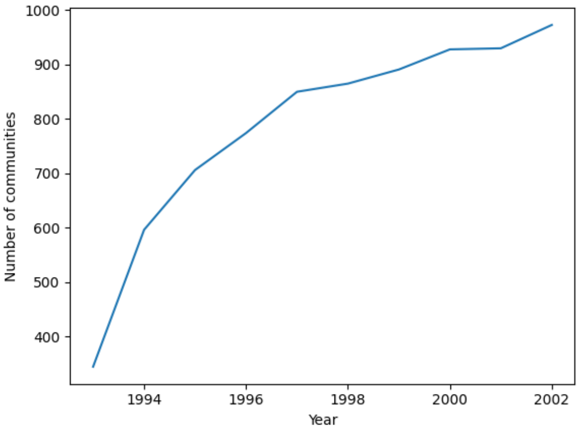


Figure 12: Number of communities v/s the year (Louvain implementation)

Note the similarity in the two graphs. Citation network after 1999 is too big to implement Girvan-Newmann efficiently (even 10 iterations for the first step take a lot of time). In this case, Louvain algorithm works the best!
A comparative basis for Louvain v/s Girvan-Newmann can be the time it takes to distinguish communities in the citation network of year 1999 (21627 nodes and 201485 edges). While it takes 135 min for the Girvan-Newmann algorithm to take 10 iterations to check modularity gain, it takes just 5-7 secs for the Louvain algorithm to do that.

3.2 Graph Neural Network (GNN)

3.2.1 Basics

Graph Neural Network is an ML tool that we can use to generate node embeddings for all the nodes in the graph. A graph is passed as input to the GNN and the GNN outputs node embeddings (vectors in a lower dimensional space). These node embeddings can be used for various tasks like node classification, link prediction, etc.

3.3 An attempt using Supervised Learning

The dataset we have doesn't have any labels for the nodes apart from the date of submission of paper. Networkx offers some node-specific properties that might be used as labels for the model. Some examples are clustering coefficients, pagerank, etc. GNNCommDet.ipynb in the GitHub repo provides code for the GNN that uses clustering coefficient as the label.
The GNNStack is as follows:

- one layer of GCNConv with dimension 1x101
- 5 layers of GCNConv with dimension 101x101
- layer normalization layers in between GCNConvs

The feature vector x is just all ones(number of nodes x 1). As no node-features are known for the graph. Activation function used is ReLU(Rectified Linear Unit). In order to avoid overfitting, we perform dropout for output of each GCNConv. Loss function used is the negative log likelihood function.
Output embeddings is in a 7 dimensional space.

3.3.1 Issues faced

- Since most of the nodes in the network have clustering coefficient very close to 0, it becomes difficult for the model to differentiate nodes.
  Attempted solution: Scaled the coefficient by multiplying with a large number (say, 100). This solved the problem to some extent however, many unrelated nodes still had similar labels. A very large number is needed to actually solve the issue. But doing so is not computationally feasible as that would mean a higher hidden dimension for the GCNConv layers thus increasing the computation and training time.
- Clustering coefficient in itself is not really a good measure to perform community detection as two completely unrelated nodes can have similar clustering coefficients since it is a structural feature.
  Attempted solution: Tried training an unsupervised model that would learn the node labels by itself (through GAE).

Figure 13 shows the embeddings for citation network of year 1992 to 1994 after training it over network from 1992 to 2000.

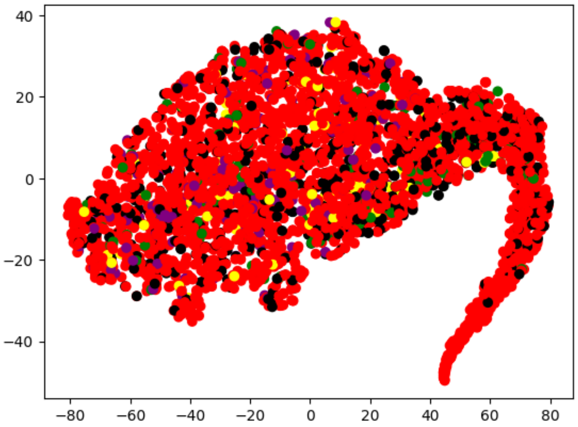


Figure 13: Supervised model's embedding

3.4 Another attempt using Supervised Learning

In this part, we try to train a GNN model using year of submission of paper as label. All GNN parameters remain the same except that the dimension of hidden layer is 32. (since labels are just 10, 32 should work fine).
We train the model on the entire citation network. Now, on passing graph/subgraph structure of any year, the model can predict the range of years the graph belongs to. Figures 14 and 15 show an example for the same. Code for the same can be found in BasedOnYear.ipynb in the GitHub repo.

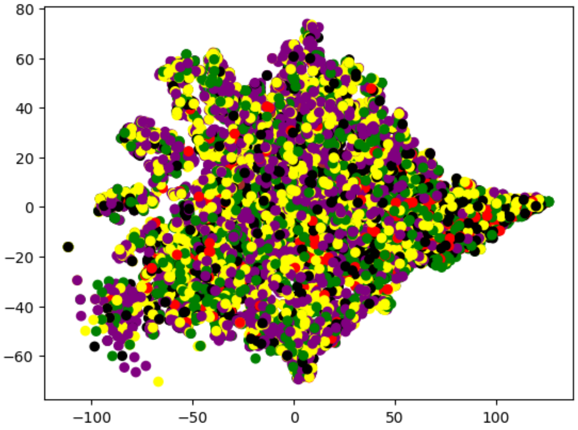


Figure 14: 1992 to 1996 graph node embeds(years as labels)

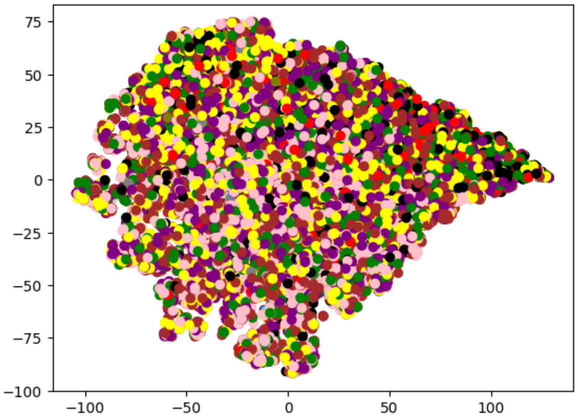


Figure 15: 1994 to 1998 graph node embeds(years as labels)

As per the color-year map for the nodes shown in table 1, we find that the output of the GNN is correct. The GNN is correctly predicting the range of years, the graph belongs to.

Red	1992
Black	1993
Green	1994
Yellow	1995
Purple	1996
Brown	1997
Pink	1998
Gray	1999
Olive	2000
Cyan	2001
Orange	2002

Table 1: Color-year map

3.5 Unsupervised Learning model

Since no label worked well for node classification tasks in supervised learning model, we try training an unsupervised model.
This uses GAE(Graph AutoEncoder), we code up a custom Encoder. It is a 3-layered GCNConv stack, the default decoder it uses is the inner product (dot product) of the embeddings.

- Input dimension is 7
- Hidden layer dimension is 32
- Output dimension is 16

The model reduces the binary cross entropy for training set edges and tunes parameters according to that.
Figure 16 shows the node embeddings visualization for the citation network from 1992 to 1998.

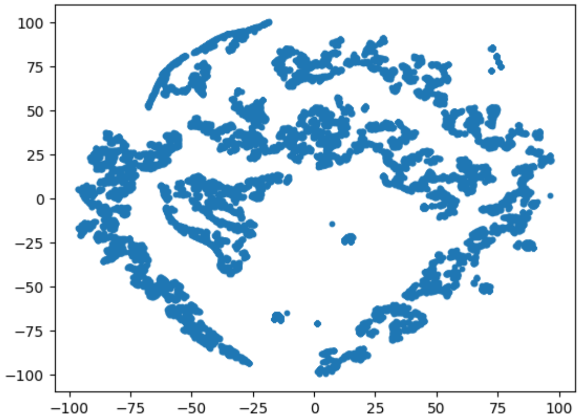


Figure 16: 1992 to 1998 graph node embeds(unsupervised)

Same community nodes tend to have similar node embeddings(they are closer together in the plot) as the decoder is the inner/dot product for the GAE. Cosine similarity works well in order to determine clusters in the citation network. We split the dataset into test edges and train edges. The code for this can be found in Unsup.ipynb in the GitHub repo.
Figure 17 shows node embedding visualizations for citation network from 1997 to 2002, after training the model on 1992 to 1997 data.

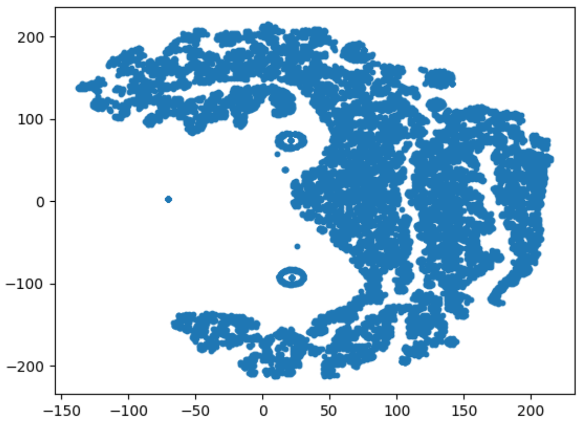


Figure 17: 1992 to 1998 graph node embeds(unsupervised)

3.6 Link Prediction

3.6.1 Basics

Link Prediction is an important category of study that helps us predict missing edges or future edges in the graph. This can be done making use of node embeddings of nodes in the graph by calculating the edge embeddings from the node embeddings. There are two ways to test this - one by removing some edges in the dataset and training it on the rest and then finally using the model to predict those missing edges, the other one is to train the model on dataset from 0-T time and then predicting edges that appear after time T.

3.6.2 node2vec algorithm

The node2vec algorithm is a classic algorithm proposed in this paper. It uses biased random walk to learn about the neighbourhood of nodes, it learns about the network patterns unique to the network as well unlike GNNs that focus on the rigid notion of structural neighbourhood.
An implementation of node2vec algorithm can be found in node2vec.ipynb in the GitHub repo. After the node embeddings are obtained from the node2vec algorithm, we calculate the edge embeddings and use it for the link prediction tasks. After we get the node embeddings, we calculate edge embeddings by simply multiplying the node embeddings element-wise.

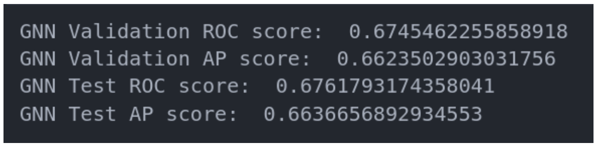
Now, we train a logistic regression model that will help us predicting links that are missing or will appear in the future depending on which dataset we pass for testing. The reference for the same is here.

3.6.3 Link prediction from embeddings by GNN

The unsupervised GNN we trained gave node embeddings for all nodes of the network. We then use the same procedure as mentioned above to get edge embeddings and use it for link prediction tasks.

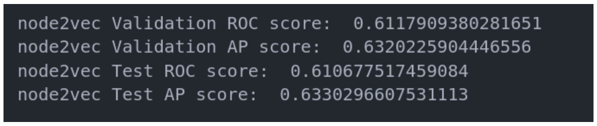
3.6.4 Comparisons between the two methods

We test the two methods on two datasets - 1996 to 2002(after training on 1992 to 1996), and 1994-1998(after training on 1992 to 1994).



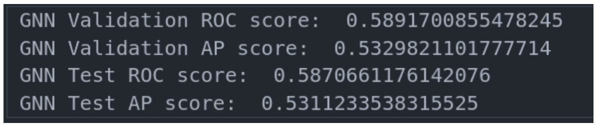
```
GNN Validation ROC score:  0.6745462255858918
GNN Validation AP score:  0.6623502903031756
GNN Test ROC score:  0.6761793174358041
GNN Test AP score:  0.6636656892934553
```

Figure 18: 1996 to 2002 by GNN



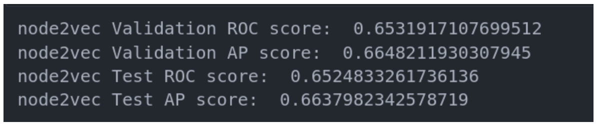
```
node2vec Validation ROC score:  0.6117909380281651
node2vec Validation AP score:  0.6320225904446556
node2vec Test ROC score:  0.610677517459084
node2vec Test AP score:  0.6330296607531113
```

Figure 19: 1996 to 2002 by node2vec



```
GNN Validation ROC score:  0.5891700855478245
GNN Validation AP score:  0.5329821101777714
GNN Test ROC score:  0.5870661176142076
GNN Test AP score:  0.5311233538315525
```

Figure 20: 1994 to 1998 by GNN



```
node2vec Validation ROC score:  0.6531917107699512
node2vec Validation AP score:  0.6648211930307945
node2vec Test ROC score:  0.6524833261736136
node2vec Test AP score:  0.6637982342578719
```

Figure 21: 1994 to 1998 by node2vec

As we can see, node2vec gives better performance on 1994 to 1998 dataset as compared to GNN, while giving similar performance on the other dataset. This implies that node2vec is able to draw more information about the graph than GNN due to its biased random walk constituent. Even the paper talks about this performance improvement of node2vec over GNN.

4 References

Most of the references used have been included in the report at relevant places. Other than those, I have used the following references for learning about Graph Neural Networks, node2vec and their reference implementation:
https://colab.research.google.com/drive/1DIQm9rOx2mT1bZETEeVUThxcrP1RKqAnscrollTo=m-R$_{E}AYAz5kk$
https://github.com/aditya-grover/node2vec
For getting to know about the basics of graph neural networks:
https://web.stanford.edu/class/cs224w/index.html