



INDRAPRASTHA INSTITUTE *of*  
INFORMATION TECHNOLOGY  
DELHI

Department  
of  
Computer Science & Engineering

CSE557 | Big Data Analytics

**Prof. Vikram Goyal**

Assignment-1:  
Information Fusion

Shrugal Tayal  
2020408

Sunishka Sharma  
2020137

## Part 1

For this assignment, we were tasked with working on two datasets. The first dataset was downloaded and preprocessed, containing three columns: paperid, title, and subtitle. We replaced any NaN values in the subtitle column with whitespaces, concatenated the columns title and subtitles, dropped the column subtitles, and duplicated the data into an iterative list using Pandas. Next, we opened and loaded the second dataset, which contained 10K files.

We explored various online libraries to parse through the documents and extract titles, references, and other metadata related to the files. Specifically, we evaluated grobid, science parse version 1 and 2, and PyPDF2. Our findings revealed

1. grobid worked the best but unfortunately returned data in formats that required more overhead processing hence delaying required outputs
2. science parse version 2 was very accurate but extremely slow and unfeasible to use
3. science parse version 1 came at a close second, it gave by far the fastest and the most accurate results but seemed to reach its cap at 6000-7000 pdfs after which it failed to respond. We tried achieving better results with docker, but that also created more issues.
4. Hence we ended up using science parse version 1, which gave us the best results thus far

We used PyPDF2 on batches of one thousand files to avoid the bottleneck we encountered with science parser version 2. We then created a nested dictionary format list called d2 that contained the references and titles for every paper. We parsed through this list using fuzzy string matching to identify common titles between the two databases. This was done to ensure that there were no incoherence or inconsistencies attributing to extra words, whitespaces and or other additions or reductions made to reference titles while being read or simply during publishing.

We set our baseline to a 90 percent match and proceeded to add matching elements to a list.

Finally, we went through the match list to check if any references matched more than 90 percent with the titles in d2, and we created our final output list. All stages of outputs were pickled.

Citation-graph object reported: [Part2\\_citation\\_graph.pickle](#)

Number of PDF papers found in D1-intersection-D2 -> D1: 82 papers

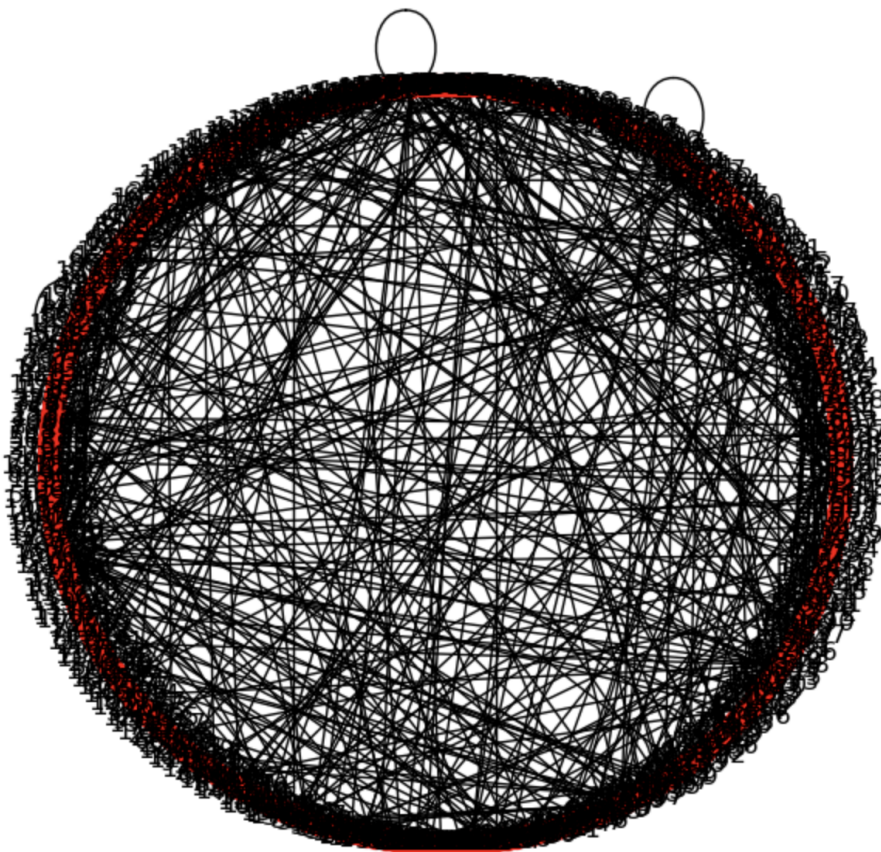
## Part 2

In this project, we utilized our previously pickled dataset and iterated over it to create a graph where each paper in our dataset was represented as a node, and their interconnections were attributes that reflected one paper citing another. To accomplish this, we utilized the NetworkX Python software package, which allows for the creation, manipulation, and investigation of the structure, dynamics, and function of complex networks.

NetworkX was particularly useful for this task, as it allowed us to investigate big, complicated networks represented as graphs with nodes and edges. Additionally, we were able to load and save complex networks, generate various random and traditional networks, study network structure, construct network models, create new network methods, and draw networks.

After creating a list of nodes using titles and ids, we searched for these ids in our citation dictionary, which we had created previously. By utilizing key-value pairs, we were able to understand the connections between papers in our dataset and plot the graph of the intersection of d1 and d2.

Figure-1: D1-intersect-D2 -> D1 citation graph



By creating this graph, we were able to visually represent the relationships between papers in our dataset and gain a better understanding of the network structure. This information could be used to further analyze the dataset and potentially identify key papers or areas of interest.

Citation-graph object reported: [Part2\\_citation\\_graph.pickle](#)

Number of Nodes- 493

Number of Edges- 442

```
print("Average degree: ", sum(degrees)/len(degrees))
```

Average degree: 1.793103448275862

```
# average clustering coefficient for nodes
```

```
print("Average clustering coefficient: ", nx.average_clustering(citation_graph.param))
```

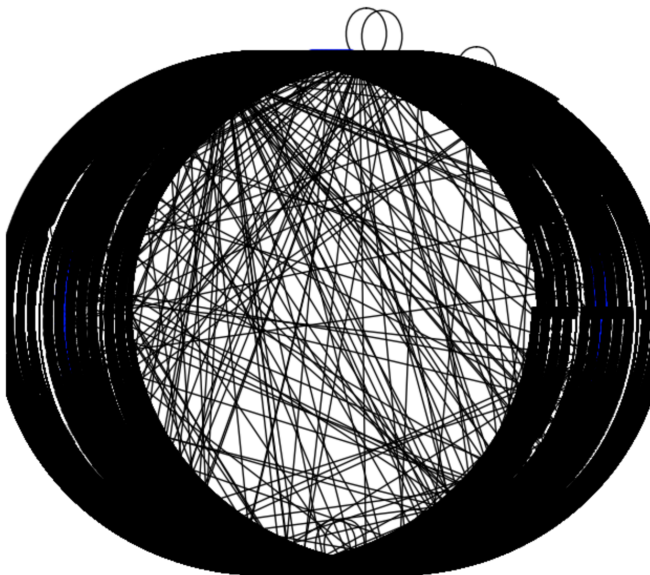
Average clustering coefficient: 0.002260214430599826

## Part 3

For this portion of our assignment, we reloaded our initial d2 dataset of 10,000 PDFs and extracted metadata in batches using relevant libraries, similar to the first part of the assignment. However, this time we also took note of the file names in addition to the metadata.


To accomplish this, we created two dictionaries: one that contained titles and file names, and another that contained references and file names. The file names served as a common key that allowed us to easily match the data in each dictionary. We created these dictionaries by iterating over a master list of titles and references, with citations being made using the file name as the key.

Figure-1: D2 -> D2 citation graph



It's worth noting that we did not consider any references that were not present as a document in d2. This allowed us to focus solely on the relationships between papers in our dataset.

Using the blueprint established in part 2 of the assignment for creating our graphs, we ended up with the following result of matching database 2:



By organizing the data in this manner and visualizing the connections between papers, we gained a better understanding of the relationships between papers in our dataset. This information can be used for further analysis and potentially uncover insights or patterns that may have been missed otherwise.

Final data structure reported: [Part3\\_citation\\_dict.pickle](#)

Citation-graph object reported: [Part3\\_citation\\_graph.pickle](#)

Number of Nodes- 10K

Number of Edges- 247

```
print("Average degree: ", sum(degrees)/len(degrees))
```

Average degree: 0.0494

```
# average clustering coefficient for nodes
```

```
print("Average clustering coefficient: ", nx.average_clustering(citation_graph))
```

Average clustering coefficient: 0.0