CSC – 501

REPORT

ASSIGNMENT 3: GRAPH DATA

SUBMITTED BY: GROUP C

DIVYANSH BHARDWAJ (V00949736)

ARSHIYA GULATI (V00949938)

SHAIMA PATEL (V00949940)

VENISH PATEL (V00949300)

SUBMITTED TO:

Prof. Sean Chester (schester@uvic.ca)

INTRODUCTION

The data was read from the Social Network: Reddit Hyperlink Network, that provides us with 55863 subreddits, 858490 hyperlinks between subreddits, timespan, edge weights (+1,-1) and edge attributes. Relating it to the graph theory, we can easily comprehend that the subreddits can be considered as nodes which can be related to each other with edge attributes such as Timestamp, Link Sentiment and Post Id. We learnt a lot while mining the data and plotting insights, about the tools that are and could be used on the dataset provided to us.

The database we used is Neo4j which is a graph database platform to load the data. Out of all the databases that were available for loading the graphs, Neo4j seemed to be quite efficient and faster than the rest and also some great visualisations could be drawn from it. Initially, we weren't feeling burdened while looking at the amount of data provided to us because we have worked with the larger datasets than this. This data started showing its real colours when we started loading the data into a graph database (Neo4j, in our case). The whole dataset was taking around one and a half hour to get loaded into the Neo4j. But the challenge was resolved by modelling the data into a comparatively smaller dataset. After having modelled the data it took around 5 to 6 minutes for the data to get loaded in the database. This modelling helped us to load the data quickly and efficiently into the Neo4j. Additionally, the modelling being based on the concepts of graph theory taught in the lectures and suggested in the research papers, helped us to generate the visualisations that were quite closer as compared to the whole data. Furthermore we have also linked our assignment with the provided research papers i.e. used the concepts explained in those papers such as matrix representation (adjacency matrix), algorithmic pre-processing, graph reduction techniques, data filtering, view interaction and many more.

The implementation is done using python, one of the best languages for data scientists to work with, as it provides us with various libraries to work efficiently. The code for visualisation is implemented using these developer-friendly libraries of python such as matplotlib, datetime, Seaborn, Pandas, Py2neo and Numpy.

RUBRIC – 1

DATA MODELLING

The major challenge that we faced was in the case of modelling the data. Storing the dataset in a graph database was the biggest challenge in this assignment. Because, if the data is not modelled, one cannot start working on the insights as while querying the dataframes in python or databases, it would take a lot of time to compute results and which is a sheer waste of time. So, the primary focus was to model the dataset so as to visualise something from that data.

We deduced many approaches to model the data. First we tried to take only the one percent of the data, which would be 5000 rows. But the major problem with this kind of approach is that it loses many nodes that would exist in the database which in turn also leads to the loss of many paths that would otherwise exist in the database. So, the visualisations that would be produced on such database would be nowhere close to the visualisations for the actual data. Then in the second try, we sought to take on a subreddit that had the maximum outdegree and consider the nodes linked to that node only. It again resulted in the loss of nodes and some paths. Moreover, it was nowhere related to some ordered filtering of the nodes.

After having rejected 5 to 6 ideas, we started going through the concepts taught in the class and some research papers. One of the research papers, “Visual analysis of Large Graphs” had some concepts related to filtering the data and processing the graphs in a way that one could make the graphs size scalable. As mentioned in the research paper (2.2 Algorithmic Graph Pre-processing, 2.3 Graph Filtering), two forms of filtering are available: stochastic and deterministic. Stochastic filtering is based primarily on the random selection from the original graph of nodes and edges. Deterministic applications of sorting, a deterministic algorithm for the node / edge range to be eliminated. This filtering can be based on attributes of node / edge, topological values such as centrality of the graph or other graph properties.

Moving further on this route, we planned to filter the dataset in such a way that we retain the paths from the nodes. For an instance, consider a source node or source subreddit “leagueoflegends” which in actual data is connected to 840 other target nodes or subreddits. Now, rather than considering all the 840 paths, we just wanted to select one of it and eliminate all the other edges. So, we decided to do this filtering based on the time stamps. Out of all the 840 edges which were going out of it, we chose the one which had the earliest timestamp. Since it is mentioned in the research paper that you could filter out the edges based on the properties or attributes of the edges, so following the same route, we selected that link which had the earliest timestamp. Modelling the data in this way, we were able to cut down the size of the dataset from near around 200,000 rows to 27,864 rows.

The data has been modelled in such a way that it retains all the source nodes and selects that link where timestamp bears the lesser value. Relating it to the concepts that we studied in the class, it is basically the induced subgraph of the whole graph. The deterministic filtering of the data gave us an induced subgraph which was much easier to store and work with now. Now we had retained all the source nodes through this modelling and also completed the path at the final node where it ended. After modelling the data, we just had to store the 27,864 rows which was much easier to work with now. The total number of nodes that we could use were 29,477 which are relatively much closer to the total nodes which are near around 55,000.

Coming to the discussion related to visualisation, we were able to compare some insights which were easier to make and understand the whole dataset and our new modelled dataset. The inferences from the visualisations were pretty much the same which boosted our model structure further. So, we stored the data in Neo4j graph database, which was done in exactly 5 minutes 47 seconds. Shown below is the view of 50 nodes of the modelled data.

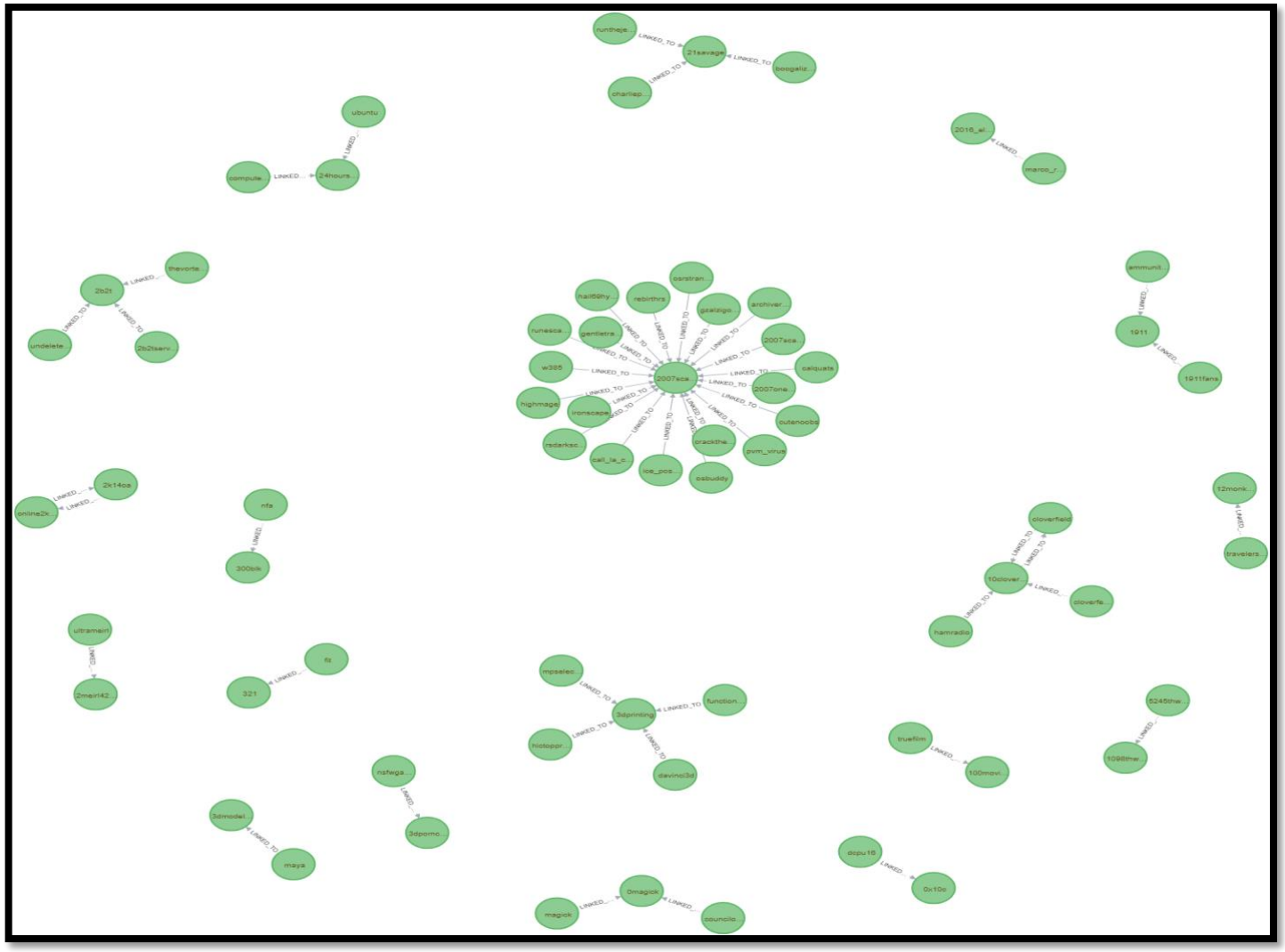


Fig: 50 nodes of the modelled data.

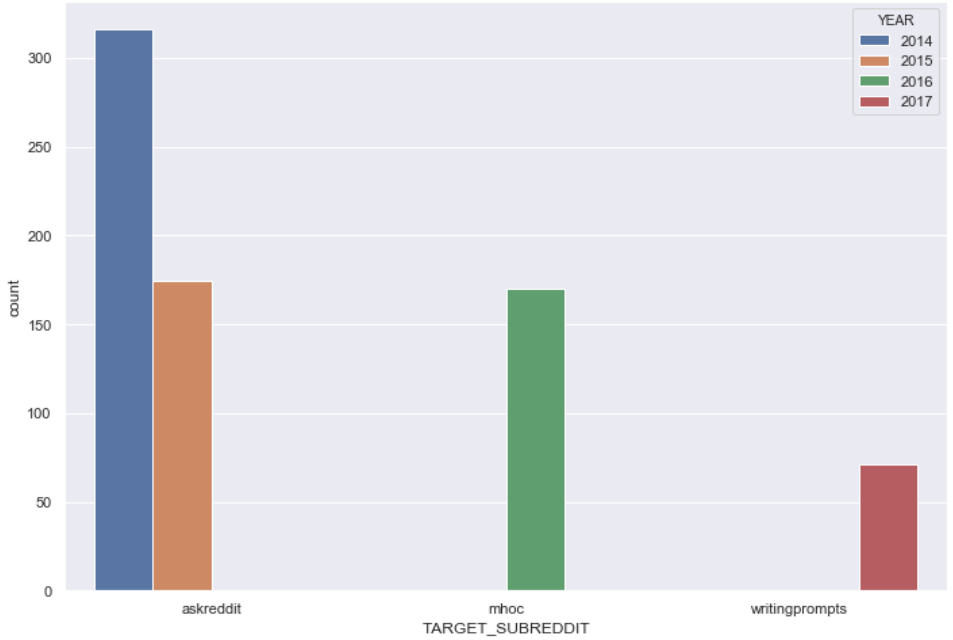
In order to get some visualisations, we were required to make the adjacency list, edge list and adjacency matrix for our assignment. Since we wanted to check which data structure would be the best to store the graph, we implemented all of them. The most efficient in terms of time of all of them was the Edge list. Storing the data in edge list took just a few milli-seconds to load the data which is very fast as compared to the matrix and adjacency list which took more than half a minute to get implemented. The adjacency list is better than adjacency matrix in terms of space efficiency and is used in implementing the algorithms like bfs (breadth first search), dfs (depth-first-search) and dijkstra's algorithm more efficiently.

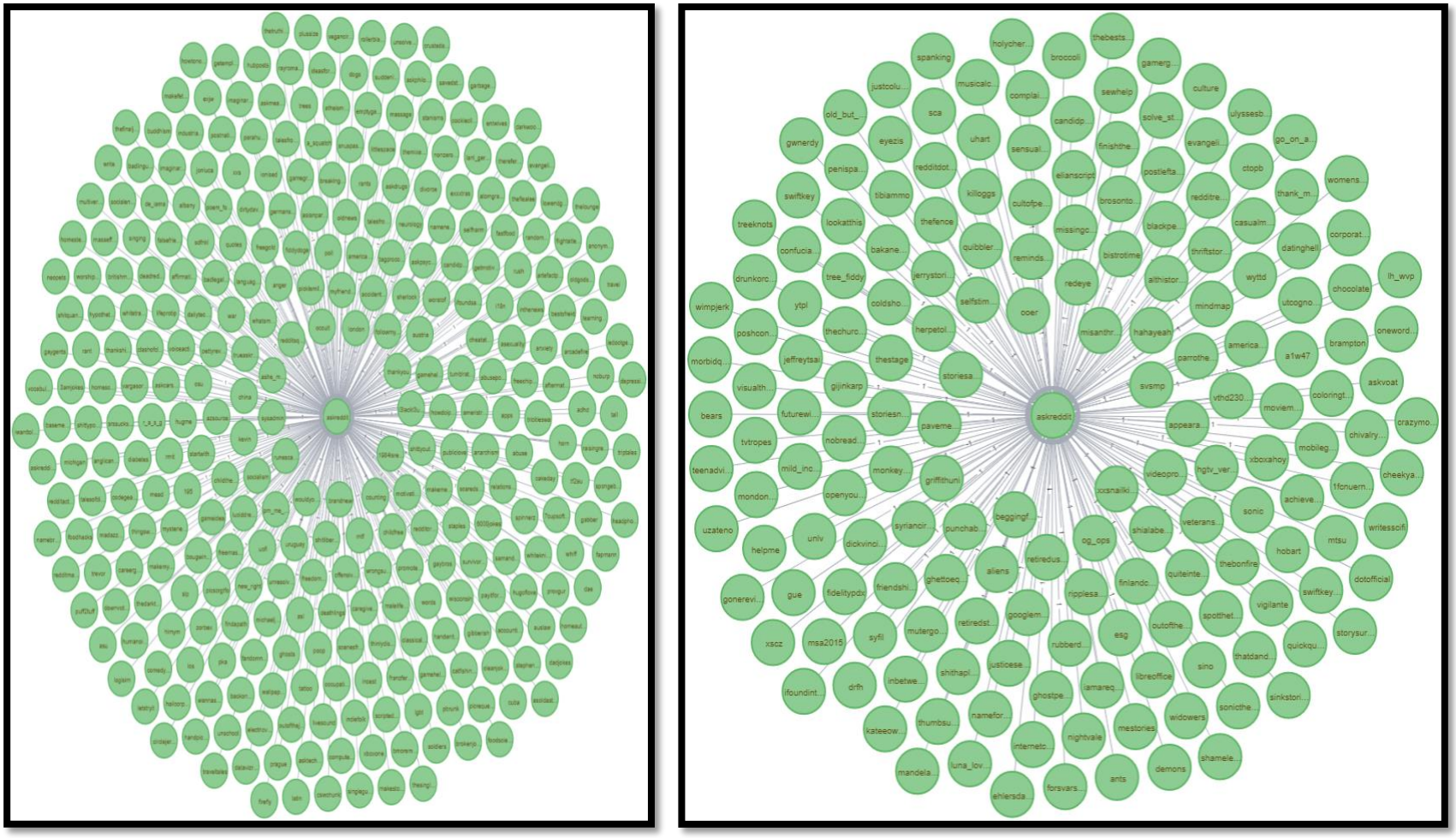
RUBRIC – 2

TRANSFORMING RAW DATA TO INSIGHTS (VISUALISATION)

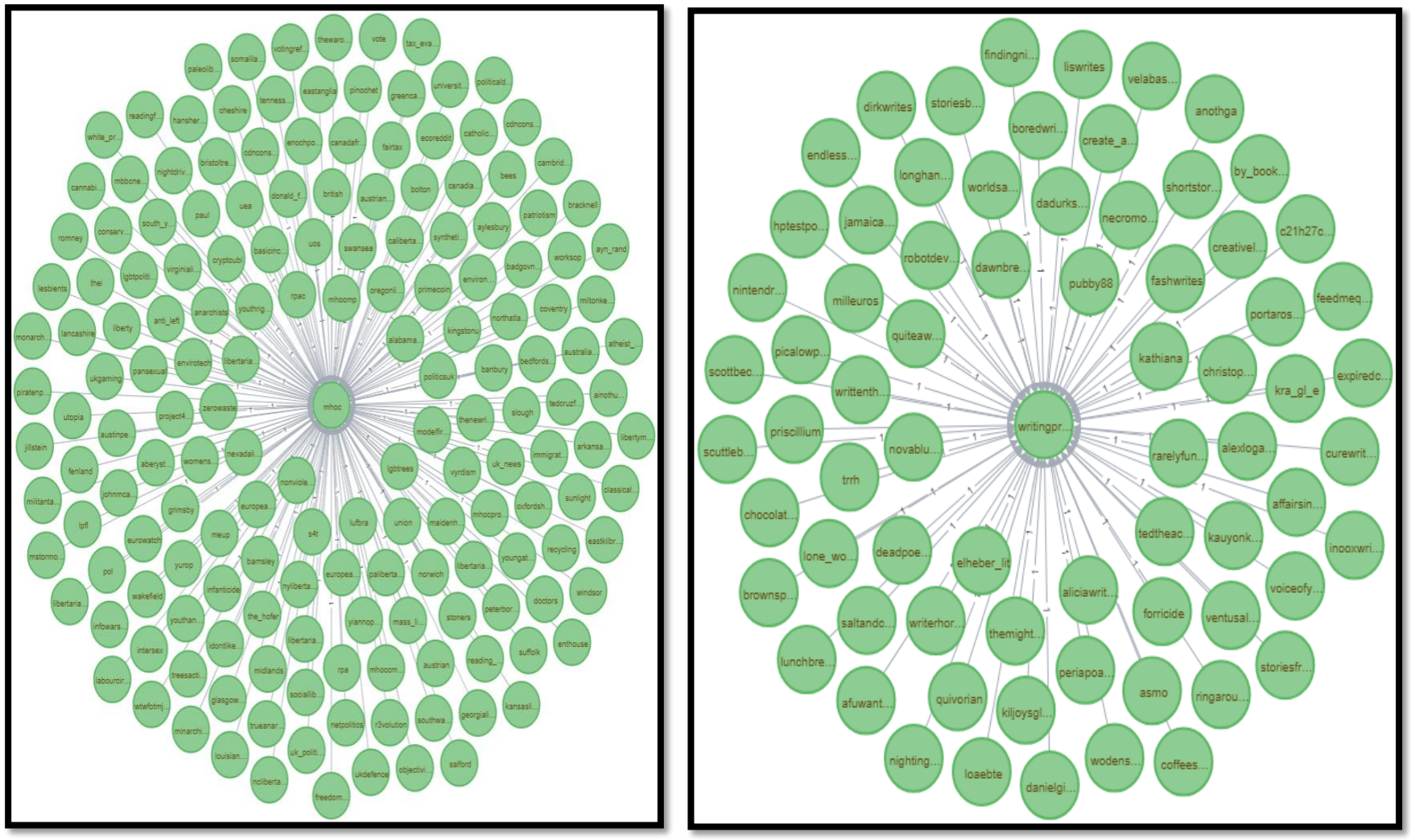
Plot 1: Most targeted (discussed) subreddit per year

As maximum number of the edges are terminating at these subreddits, this implies that posts in these subreddits had the answers to the maximum number of discussions that originated from the source subreddits. By looking at the bar graph, we can infer that subreddit “askreddit” has been the king of all subreddits by having the most incoming edges in the years 2014 and 2015. Whereas “mhoc” (a subreddit related to politics) and “writingprompts”, were most targeted in the year 2016 and 2017 respectively. We have also shown the graphical form of this insight.





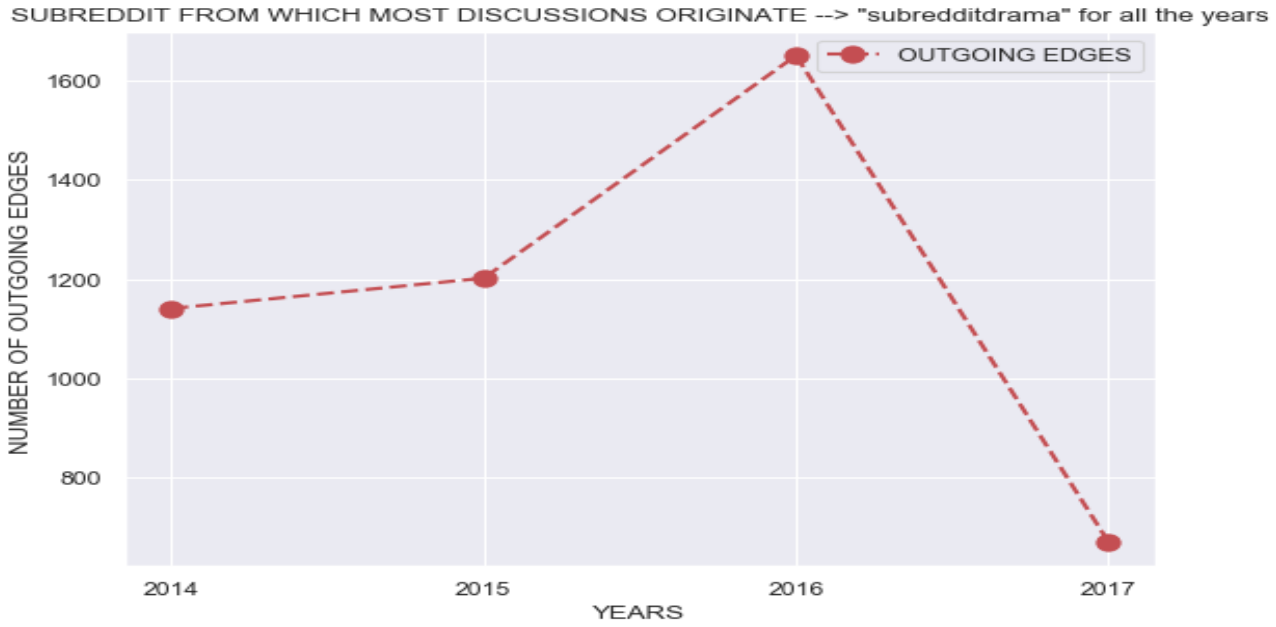
Incoming edges and nodes to the most targeted subreddit, which was “askreddit” for both 2014 and 2015 respectively.



Incoming edges and nodes to the most targeted subreddit, which was “mhoc” and “writingprompts” for 2016 and 2017 respectively.

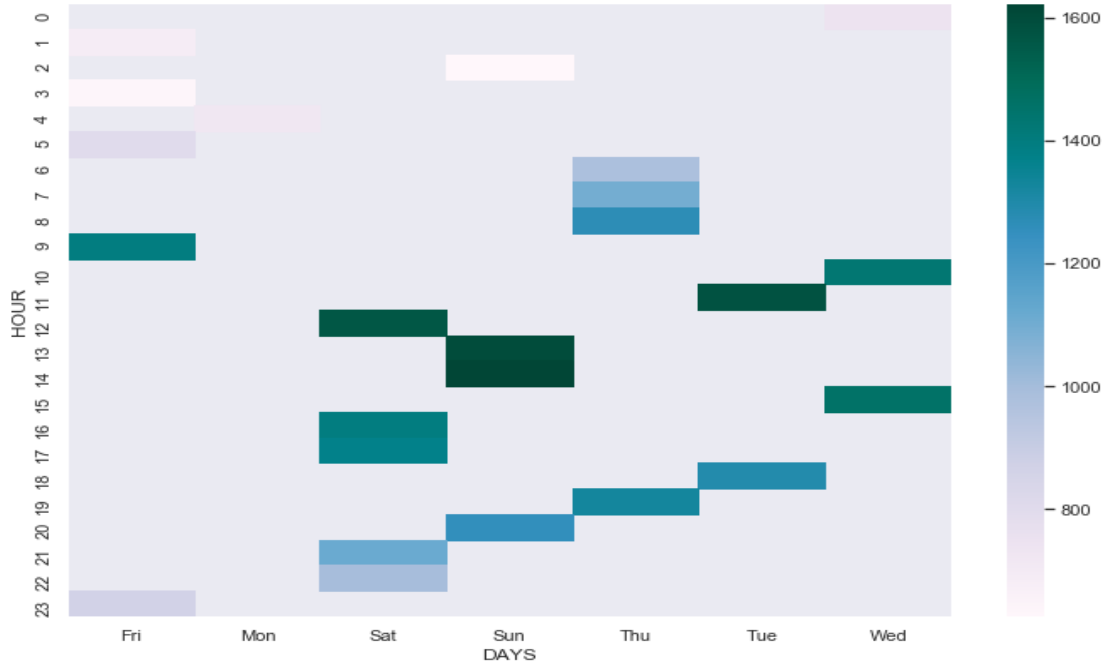
Plot 2: Subreddits from which most discussion originate per year

The following visualisation tells us that subreddit “subredditdrama” has the maximum number of outgoing edges throughout all the years. However, in the year 2016, it has been the maximum. It can be inferred that most of the discussions originate from this subreddit.



Plot 3: Frequency of the number of posts per weekday per hour.

This heat map clearly describes that on weekends, Saturdays and Sundays, maximum number of posts are made in hours 11Am to 2PM. Out of which Sundays being an off-day for all we can see more number of posts being made more than Saturdays. Whereas of weekdays on Tuesdays in the hour 10Am to 11 Am. maximum number of posts are made.

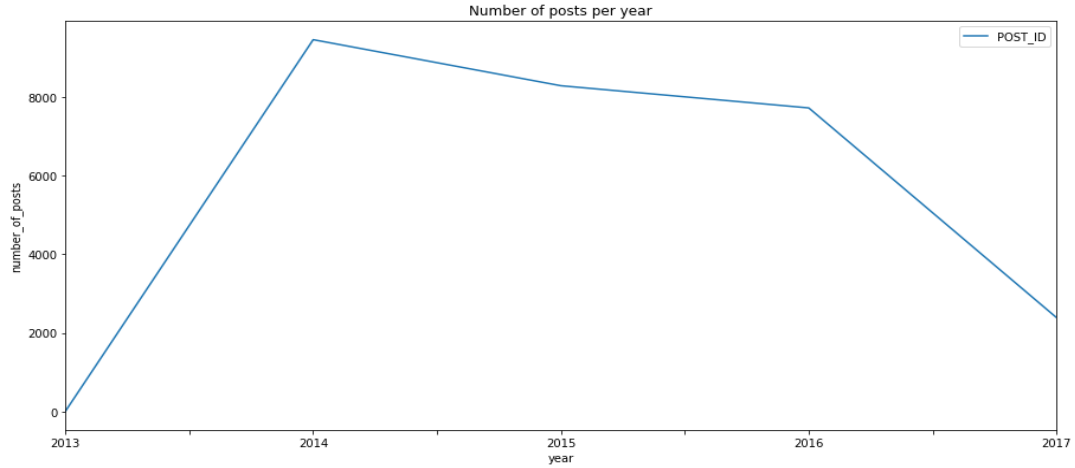


Plot 4: Number of posts per year

From this line graph we can see that maximum number of posts were made in the year 2014, which increased from the year 2013 to 2014. The number of posts decreased over the years after 2014.

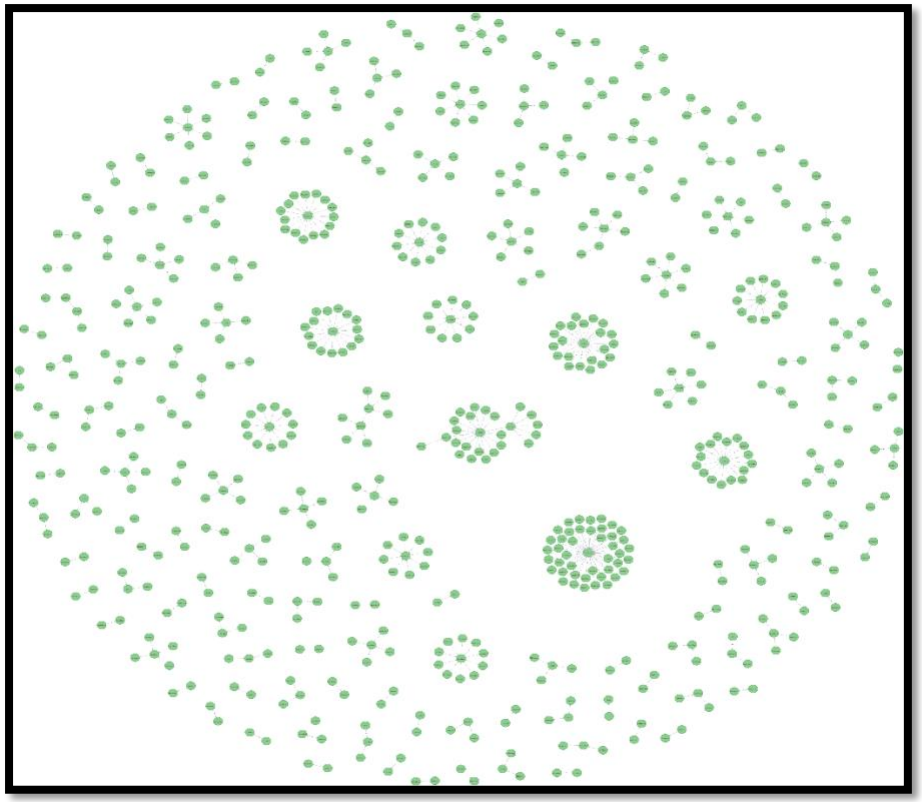
Out[60]:

year	POST_ID
2013	6
2014	9459
2015	8287
2016	7723
2017	2388

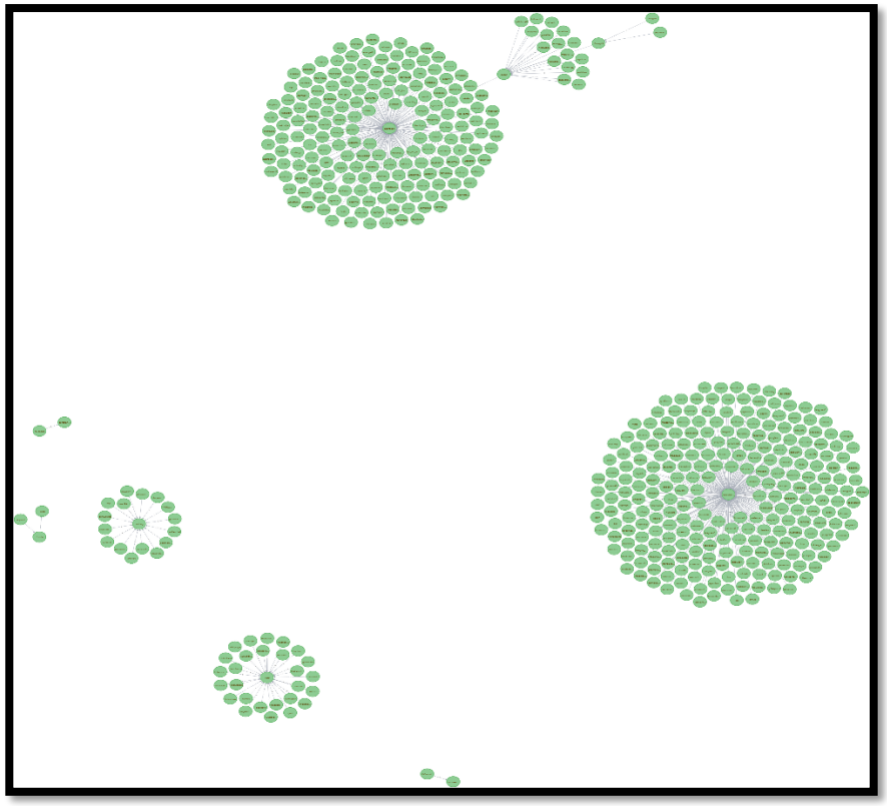


Plot 5: Comparing the nature and structure of subgraphs with link sentiment +1 and -1

We can very vividly see in the structure of the two subgraphs that the one with link sentiment +1 is very densely cluttered and the other one is sparsely distributed. It can be inferred that the subreddits with negative sentiments are very sparsely connected that is the subreddits that are of negative nature are not much coherent. Whereas the the similar subreddits or the subreddits with positive sentiment are very densely clustered or it can be said that similar subreddits are very cohesive in their nature.



Subgraph with link sentiment -1



Subgraph with link sentiment +1

RUBRIC – 3

ALGORITHMIC CONSIDERATIONS

Neo4j is used to visualise the graph data in a better way. As explained in modelling rubric loading the complete data in the neo4j was taking a lot of time which would affect the cost of the model. So we modelled the data in neo4j reducing the number of nodes to 29477, and the average degree correspondingly comes out to be 1.8723.

We also created an **adjacency matrix, adjacency list and an edge lists**. They are created to store the graph data in python in an efficient way. From which we saw that the adjacency lists are better than adjacency matrix in terms of space. Comparing all the three ways to store the graph, we came to the conclusion that for our data, edge lists is the most efficient mean of storing the data.

- **Scalability**
  - Average time taken to read the modelled file: 1.06 sec
  - Average time taken to read the body file: 8.76 sec
  - Average time taken to read the title file: 11.1 sec
  - Average time taken to extract day, time and date from timestamp: 28 ms
  - Average time taken to create an adjacency matrix and adjacency list: 1 min 28 sec
  - Average time taken to create an edge list: 435 ms
  - Average time taken to plot the graphs are:

    Plot 1 Most targeted (discussed) subreddit per year: 485 ms

    Plot 2 Subreddits from which most discussion originate per year: 275 ms

    Plot 3 Frequency of the number of posts per weekday per hour: 1.36 sec

Plot 4 (Number of posts per year): 649 ms

Plot 5 (Comparing the nature and structure of subgraphs with link sentiment +1 and -1): 1.06 sec

It is **scalable as**, even if the size of the dataset increases we can model the data in such a way using neo4j that the cost and time remains efficient.

- **Challenges and Solutions**

Many challenges occurred while we were working through the project but here we are listing some major problems regarding the scalability part along with the solutions:

1. Loading Large Files

We loaded the data into Neo4j so that we could load and get some visualisations out of it in the form of graphs in a better way, which was provided to us in a tabular form. But while loading the complete data we saw that it was taking a lot of time so we had to model the data reducing the number of rows in such a way that the information remains intact.

2. Reducing the Query Time

We created adjacency matrix, adjacency list and edge list to store the graph data effectively, which also reduced the query time to some seconds/ milliseconds.

3. Structure of the actual data : Too many cycles

While working with modelling the data, we came to know by understanding the structure of the data that we have source nodes but the path ultimately ends at one of the source nodes nodes itself. For an instance, let us say that node 1 is connected to 2, which is further connected to 3. Now node 3 is terminating either at source node 1 or source node 2. Hence, while applying algorithms to the data, we encountered many problems due to this. But since we were only considering the one link out of all the possible edges, were able to cut down the cycles by a significant amount.

4. Creating the model for the dataset

While working on the model of the dataset, we had to consider such a model where we could be able to cut down the data to a significant amount keeping in mind that most of the nodes and their paths are preserved and the filtering should be according to some graph related concept. We opted so many ideas, found flaws in them and rejected them. After reading the research paper, we could finalise the model for our dataset with proper data filtering taking the due care of the concepts of the graph theory.

RUBRIC – 4

RELATIONSHIP TO GRAPH DATA CHALLENGES

The modelling part of our assignment took the maximum time to get formed. This was just because we had to model the graph data keeping in mind that modelling could only be done by applying some graph theory concept. Out of all the challenges that we faced, the biggest challenge was to cut down the graph data to an extent that it was easily usable and understand. The whole solution to our problem was lying in the research paper "Visual analysis of Large Graphs". After reading the research paper, we came to an understanding that in order to model the data, we need to cut down the size of the dataset while maintaining the graph structure. There are mainly two approaches for the graph size reduction: Graph filtering and graph aggregation.

The second biggest challenge that we were facing was that the original data contained too many cycles. Let us say that I have some source node as gaming subreddit and it is linked to callofduty subreddit. Also the callofduty is also linked to some other subreddit, say pubg. Now pubg subreddit link will either terminate at leagueoflegends subreddit or callofduty subreddit that is one of the source nodes. Maximum data is linked in this way only. Now this kind of data was giving us problems in implementing algorithms for visualisations. Hence this needed to be taken care of.

We, in our assignment, used Graph filtering as it was making more sense with respect to this assignment. There are two types of Graph filtering as listed in the research paper: Stochastic and Deterministic. Stochastic filtering is when you randomnly choose any nodes or edges from the graph and make a subgraph out of it. Whereas, Deterministic filtering filters the data when we choose the nodes or edges according to some deterministic of well-defined concept related to graph theory.

After reading the article of Data filtering in the research paper, we could relate our data modelling approach to be a bottom up approach as reaching every node we decide which edge or which vertex to be chosen in order to complete the path. This is exactly that we did in our modelling approach as we chose that link emerging from the source node which had the earliest timestamp which in turn is an edge attribute.

For our assignment, Deterministic filtering gave us the proper grounds upon which the whole model could be built. We chose the links that possessed the earliest timestamp and ignored all other edges that originated out of the source node. By doing this, we were able to cut down the size of our dataset to 27,864 rows which was a massive upgrade from 200,000 rows. This also solved the problem of the cycles which were there in the original dataset. Due to elimination of the edges, cycles left were so minimal which could be easily managed during implementing the algorithms for visualisations. Hence, we were able to manage all the challenges that we faced in our assignment during the modelling and solved them with the help of the concepts listed in the research paper.