Centrality Analysis of IMDB Dataset Using Hypergraphs

Funso Oje
Computer Science
Washington State University
Pullman, USA
olufunso.oje@wsu.edu

Oluwafemi Ajeigbe
Electrical Engineering
Washington State University
Pullman, USA
oluwafemi.ajeigbe@wsu.edu

***Abstract*—Understanding the centrality of vertices (or edges) is crucial in assessing the significance of vertices in real-world applications, from social networks to power grids to the influence of actors. Literature on centrality metrics computations posits that it is easy to run metrics on graph models using various traditional packages(igraph and networkx). However, when the network becomes too large, it becomes computationally expensive and almost impossible to use these packages. If these large networks are modeled as the s-line graph of a hypergraph, it becomes easier to compute the centrality measures for the resulting s-line graph. In this project, we modeled the IMDB database as a hypergraph and computed its betweenness centrality score using Brandes algorithm. This is a work in progress.**

***Index Terms*—hypergraphs, imdb, s-line graphs, large graphs, betweenness centrality.**

I. Introduction

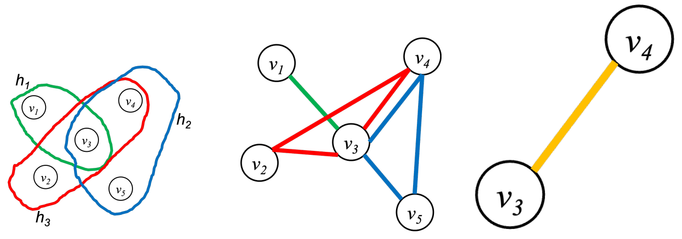
Data collected from complex systems are made up of objects that interact in complex forms. Traditional graphs assume binary relationships among nodes in a network. For example, an edge is present between two nodes if the defined relationship exists between the two nodes. Sometimes, these relationships are clearly defined and can only hold two nodes at a time. In other cases, these relationships are complex and can hold more than two nodes at a time.

A. Modeling a movie as a graph

Let us consider a movie as an example. Numerous actors (or actresses) act in a movie. Also, an actor could have acted in various movies. We could model this as a bipartite graph if we consider both the actors and the movies as nodes, and the edge relationship will be present if an actor acted in the movie.

B. Modeling a movie graph as a hypergraph

Hypergraphs provide a mathematical model of multi-way relationships between objects in a complex network. Hypergraphs have been studied to explore the connectivity of entities, clustering structures, and multi-way relationships of objects in an extensive and complex network structure. Using the movie example, we can model the graph as a hypergraph by using the movie as a hyperedge and the actors and actresses as the nodes in the hyperedge. If we take this model and simplify it in a simple graph in which an edge is considered if two nodes are present in the same hyperedge, we will be able to represent this as a graph. Figure 1 shows the representation of the IMDB database using a few nodes and edges. Figure 1a shows the actors as vertices and the movies as hyperedges. Figure 1b shows the adjacency representation of the hypergraph where an edge is present between two nodes if they share a hyperedge. Figure 1c shows the 2-line graph representation in which only node pairs that have more than two edges between them are represented in the graph. All other nodes are dropped.



(a) Hypergraph (b) Adjacency Edge (c) 2-line graph

Fig. 1: IMDB hypergraph representation

C. Betweenness Centrality

In such a big graph model, such as the movie database, it is important to know which node(actor/actress) has the most influence in the graph. A centrality measure based on shortest paths is known as betweenness centrality(BC) in graph theory. It was devised as a general measure of centrality. Those shortest paths passing through each vertex counts as the vertex's betweenness centrality. Network theory uses betweenness centrality to measure the degree to which nodes are in proximity. For instance, a node with a higher centrality betweenness will exert more control over the network since more information will be passed through it. Essentially, betweenness centrality gives us an idea of who is the most influential node in the network [6].

D. Outline of the paper

The remainder of this paper is organized as follows. Section II describes the dataset and the problem it poses for calculating betweenness centrality. Section III presents the Brandes Between Centrality [2] algorithm used in this project. Section IV presents our implementation approach and the various analysis used in this project. Results are discussed in Section V. Conclusions and future work are presented in Section VII, followed by acknowledgments and references.

II. Problem Definition

A. Dataset: IMDB

The IMDB is also known as the Internet Movie Database. This dataset consists of information about films, movies, TV shows, series, video games, and streaming content online. [5] including the people that performed various roles (actors/actresses, directors, cinematographers, and writers) in these movies and tv shows. The database dates back to 1892, including movies planned up till 2029 – it has over 8 million title entries.

TABLE I: Dataset Summary

Dataset	Movies	Vertices	Edges	Hypergraph
Full IMDB	606,294	1,886,338	4,107,264	3 - 3,213,508
US Region IMDB	89,949	235,743	218,298	2 - 109,149

The IMDB dataset was downloaded from [4]. This data repository consists of seven datasets. The following is a description of each dataset and what it contains:

- **name.basics.tsv**: This dataset contains the names of all the people in the different IMDB datasets, including actors/actresses, producers, writers, soundtrack engineers, etc. This dataset also includes the person's year of birth, year of death (where applicable), and what titles (movies) they are known for. This file has over 11 million records (11,531,349).
- **title.basics.tsv**: This dataset seems to contain the basic details for a movie. Preliminary analysis shows that this file has over 8 million records (8,821,675).
- **title.akas.tsv**: This file contains alternate titles for each title in the title.basics.tsv dataset. It contains "also known as" titles in various languages, e.g., a movie made in the US might have an alternate title for France (FR) or Germany (GE). Each title could have multiple entries in this dataset. There is an ordering column that helps to indicate which is the main title information. This file has over 31 million records (31,626,489).
- **title.crew.tsv**: This dataset contains the crew (director and writer) of each movie title. It has over 8 million records (8,824,341).
- **title.episode.tsv**: This dataset contains the various episodes (season and episode) of TV shows. This file has over 6 million records (6,615,976).
- **title.principals.tsv**: This dataset contains the movie and each individual's roles in the movie (e.g., actor, director, cinematographer, etc.). This file has over 49 million records (49,736,924).
- **title.ratings.tsv**: This file contains the average rating and the number of votes for each title. This file has over 1 million records (1,230,952).

For this project, we used name.basics.tsv, title.basics.tsv, title.akas.tsv, and title.principals.tsv. This is because we were only interested in movies, who acted in them, and the region where the movies were produced.

Considering the volume of the database, it presents quite a challenge in running any centrality metrics on the resulting graph.

B. Computational Time: Betweenness Centrality

Centrality computations are computationally expensive because you have to calculate the shortest path between every node pair while counting all the shortest paths to identify which node lies between the shortest paths. Furthermore, as the graph grows larger, it becomes almost impossible to calculate the betweenness centrality of the resulting graph. According to [2], betweenness centrality has a $O(n^3)$ time complexity and $O(n^2)$ space complexity.

Even with an efficient algorithm, the graph size will always be a factor because of the time and space complexity. Various methods and algorithms have been proposed to tackle this problem. We will be using Brandes [2] algorithm in this paper. In addition to this, we would introduce CPU parallelism to improve the process. The idea here is that if we can compute the centrality metric in parallel, we can use the CPU multiple threaded architectures to achieve our aim faster.

III. Algorithms

For this project, we used Brandes Betweenness Centrality computation to compute each node's betweenness. The algorithm proposed by Brandes [2] is as shown in Algorithm 1.

The algorithm explores accumulating dependencies (rather than dependency summation) as the graph is traversed.

IV. Implementation/Analysis

A. Implementation

1) Environment Setup: To successfully implement this project, we used the graph libraries developed by the PNNL team. To do this we had to clone the following repositories:

- NWGr library [9]: This is the base graph library developed by the PNNL team and publicly hosted on Github. It is a high performance C++ library that consists of multiple sequential and parallel graph algorithms.
- NGHy library [10]: This is the hypergraph library developed by the PNNL team and privately hosted on Github. It is a hypergraph processing framework focused on s-line

Algorithm 1: Algorithm proposed by Brandes [2]

```
$C_B[v] \leftarrow 0, v \in V$;
for $s \in V$ do
    $S \leftarrow$ empty stack;
    $P[w] \leftarrow$ empty list, $w \in V$;
    $\sigma[t] \leftarrow 0, t \in V;\ \sigma[s] \leftarrow 1$;
    $d[t] \leftarrow -1, t \in V;\ d[s] \leftarrow 0$;
    $Q \leftarrow$ empty queue;
    enqueue $s \rightarrow Q$;
    while $Q\, not\, empty$ do
        dequeue $v \leftarrow Q$;
        push $v \rightarrow S$;
        foreach $neighbour\, w\, of\, v$ do
            $//\, w\, found\, for\, the\, first\, time?$
            if $d[w] < 0$ then
                enqueue $w \rightarrow Q$;
                $d[w] \leftarrow d[v] + 1$;
            end
            $//\, shortest\, path\, to\, w\, via\, v?$
            if $d[w] = d[v] + 1$ then
                $\sigma[w] \leftarrow \sigma[w] + \sigma[v]$;
                append $v \rightarrow P[w]$;
            end
        end
    end
    $\delta[v] \leftarrow 0, v \in V$;
    $//\, S\, returns\, vertices\, in\, order\, of\, non-$
      $increasing\, distance\, from\, s$
    while $S\, not\, empty$ do
        pop $w \leftarrow S$;
        for $v \in P[w]$ do
            $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$;
        end
        if $w \neq s$ then
            $C)B[w] \leftarrow C_B[w] + \delta[w]$;
        end
    end
end
```

graph constructions. It also contains various algorithms for calculating centrality metrics of the s-line graphs.

To complete our environment setup, we installed the required libraries (gcc+ version 11 and tbb by Intel), then we cloned the repositories. For this project, we pushed our code changes into a new branch called *imdb-explore* on the NWHy github repository.

2) Data Preprocessing: Taking advantage of the IMDB to MTX converter (also located in the repo), the database was converted to an MTX file, an edge list file containing all the nodes adjacent to each other in the resulting hypergraph. We did not need to do a lot of data pre-processing. However, due to the size of the IMDB database, it took a considerable amount of time to build the hypergraph every time we needed to compute the centrality metric.

3) Code Changes: The basic code functionality of converting the IMDB to a hypergraph, generating one s-line graph, and calculating the centrality metric for that graph, was already provided in the repository. We changed the code to support constructing multiple s-line graphs from an ensemble of s without the need to reload the whole IMDB database into memory.

To achieve this, we modified the code to construct the hypergraph as in the original code, then used a loop to go through the s-ensemble and generate the s-line graph (and its centrality metric) for each s. In addition to this, we also added various switches to the code that will help limit the data. The following command-line filters were added:

- --$styear$: only movies produced after this year will be considered
- --$enyear$: only movies produced before this year will be considered
- -t: number of threads to use for parallelism
- --$sstart$: start value of the s-line graph generation. This should be higher than sstop.
- --$sstop$: end value of the s-line graph generation
- --$sstep$: step value for the s-line graph decremental function
- --$datafolder$: specify the folder where the datasets are located instead of each individual dataset
- --$akas$: added this to load the alternate titles datasets to help determine which region each movies was made
- --$region$: filter the movies by a specific region e.g. US, FR, etc.
- --$norm$: Boolean to indicate normalizing the between centrality

B. Analysis

1) S-Ensemble Analysis: After modifying the code to allow the generation of centrality for an s-line ensemble, we ran the code starting with $s = 130$. This value was chosen after experimenting what the upper bound for s should be. For each s, we noted the top 3 nodes that had the highest centrality. For this, we have centrality results ranging from when $s = 130$ to $s = 2$. We could not generate the centrality metric for when $s = 1$ due to the size of the graph and the computational complexity required. We tried to compute this but gave up after 16 hours.

2) US Region S-Ensemble Analysis: In an attempt to have a good baseline centrality metric (i.e., when $s = 1$) we attempted to reduce the size of the graph so we could compute all values of s. To do this, we filtered the IMDB data using the --$region$ switch. The dataset was filtered with only movies that were produced in the US. This reduced the number of movies from over 600 thousand to just over 89 thousand. For this, we ran the analysis from when $s = 27$ to $s = 1$. The betweenness centrality measures were computed successfully.

3) s-line Plots: To visualize the resulting data and its components, the edge list of each resulting s-line graph was exported to an edge list file. We then loaded this edge using the igraph [3] package and generated the plot of each graph. Each plot had the top three nodes with the highest centrality metric color-coded for reference.

V. Results and Discussion

In this section, we present the result of the work we have done so far.

A. *Using the full IMDB*

First, the betweenness centrality was run using the whole IMDB dataset with an s-ensemble ranging from 130 to 2. Table II shows the result for a few selected s in the ensemble. The entire table is shown in the Appendix.

Our observation is that Adoor Bhasi seems to be an influential actor for higher values of s. As s becomes smaller, the results of the actors begin to vary a lot.

To fully understand the s-line plots for each s look like, we also generated plots for each s in the ensemble. Figure 2 shows some examples of the plots for various values of s.

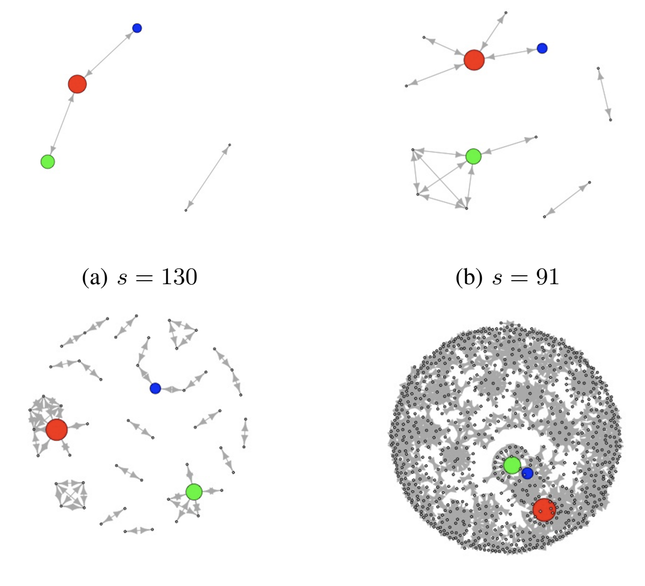

(a) $s = 130$ (b) $s = 91$ (c) $s = 56$ (d) $s = 19$

Fig. 2: Some examples of the s-line plot for various values for s. The red (top 1), green (top 2) and blue (top 3) are the nodes with the highest betweenness centrality scores.

B. *Using the US IMDB*

Due to the size of the 1-line graph of the whole IMDB dataset, we could not get the betweenness centrality for the 1-line graph. We, therefore, opted to limit the amount of data used for the computation. We filtered the movies included in the computation by the movie's region. We used only movies produced in the US, which reduced the data. Table III shows the result for a few values of s in the ensemble.

To fully understand the s-line plots for each s look like, we also generated plots for each s in the ensemble. Figure 3 shows some examples of the plots for various values of s.

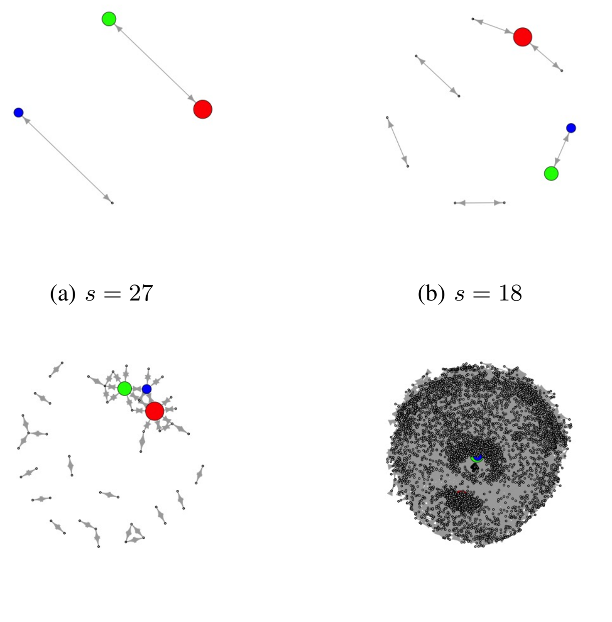

(a) $s = 27$ (b) $s = 18$ (c) $s = 10$ (d) $s = 2$

Fig. 3: Some examples of the s-line plot (using just the US region) for various values for s. The red (top 1), green (top 2) and blue (top 3) are the nodes with the highest betweenness centrality scores.

VI. Related Work

Past research on betweenness centrality [2][11][6] demonstrates the extent to which the betweenness centrality index is essential and reliable in determining the most influential nodes in a network—for example, understanding the behavior or characteristics of the most influential nodes in a social network. Section II of this study states that it is computationally expensive to compute the betweenness centrality measures on large and complex datasets. Motivated by this need to compute centrality indices on complex networks, Brandes [2] proposed an algorithm for faster betweenness centrality computations. Similarly, the study proposed by [11] extends the study by [2]. They proposed a faster algorithm–the HyperBFS algorithm, reported to speed up the forward phase of Brande's betweenness centrality computation for hypergraphs. Their study affirmed the need for betweenness centrality computations. It proposes a framework for computing the most influential nodes in an extensive network with complex objects that are impossible to find without fast computational algorithms. In the same vein, [6] explored how the influence of a team or groups in a social network can be measured using the betweenness centrality computations on modeled hypergraphs. The authors of this paper posited that the extent to which a team or group in a social network is influential is a function of how widely connected a member of the team is.

Network science is largely driven by graph theory, and analyzing hypergraphs has been pivotal in analyzing and un-

TABLE II: Betweenness Centrality Results showing Top 3 actors for some s in the ensemble

s	**Top 1 Actor**	**Top 1 Score**	**Top 2 Actor**	**Top 2 Score**	**Top 3 Actor**	**Top 3 Score**	**Components**
130	Adoor Bhasi	0.0833333	Jayabharati	0	Bahadur	0	2
125	Adoor Bhasi	0.05	Matsunosuke Onoe	0.05	Jayabharati	0	2
101	Adoor Bhasi	0.0138889	Matsunosuke Onoe	0.00231481	Kijaku Ôtani	0.00231481	3
97	Adoor Bhasi	0.0111111	Matsunosuke Onoe	0.0111111	Jayabharati	0	3
90	Adoor Bhasi	0.00549451	Matsunosuke Onoe	0.0032967	Ritoku Arashi	0.0010989	4
86	Adoor Bhasi	0.00549451	Matsunosuke Onoe	0.0032967	Bahadur	0	4
84	Adoor Bhasi	0.00416667	Kijaku Ôtani	0.000833333	Matsunosuke Onoe	0.000833333	5
76	Adoor Bhasi	0.00326797	Matsunosuke Onoe	0.000653595	Kijaku Ôtani	0.000653595	6
75	Adoor Bhasi	0.0021645	Kijaku Ôtani	0.0004329	Matsunosuke Onoe	0.0004329	8
73	Adoor Bhasi	0.00197628	Ritoku Arashi	0.000790514	Kijaku Ôtani	0.000395257	8
71	Adoor Bhasi	0.00132275	Ritoku Arashi	0.000529101	Takeo Azuma	0.00026455	10
70	Adoor Bhasi	0.00132275	Ritoku Arashi	0.000529101	Kitsuraku Arashi	8.81834E-05	10
68	Adoor Bhasi	0.00123153	Matsunosuke Onoe	0.00106732	Ritoku Arashi	0.000492611	10
62	Matsunosuke Onoe	0.000674764	Adoor Bhasi	0.000404858	T.N. Balakrishna	4.49843E-05	13
60	Matsunosuke Onoe	0.000641026	Adoor Bhasi	0.000384615	Aachi Manorama	0.000042735	13

TABLE III: Betweenness Centrality Results (US IMDB only) showing Top 3 actors for some s in the ensemble

s	**Top 1 Actor**	**Top 2 Actor**	**Top 3 Actor**	**Components**
27	Peter North	Tom Bryon	Leo Gorcey	2
20	Peter North	Tom Bryon	Billy Dee	3
15	Tom Bryon	Mike Horner	Billy Dee	5
10	Tom Bryon	Billy Dee	Mike Horner	14
5	Ron Jeremy	Tom Bryon	Jamie Gills	97
1	Eric Roberts	Ron Jeremy	Mickey Rooney	2,979

derstanding complex object interactions in several real-world datasets, e.g., social networks. To this end, the importance of hypergraphs and similar structures is increasingly being identified [1][8][7]. In the study by [1], the authors proposed a library (SimpleHypergraphs.Jl) with the Julia library that allows the efficient analysis of the structural properties of a complex network. The authors tested the library on two relational datasets: the 2019 yelp dataset and the collaboration network of the game of thrones tv series. Compared with traditional graph-based approaches, their study results demonstrated how hypergraphs provided much richer insights into how the objects in these complex networks interacted.

VII. Conclusion and Future Work

We present a faster heuristic on the BC computation by parallelizing Brandes betweenness centrality algorithm. While we achieved the BC values for higher s-values, we still need to validate these results using the BC scores when the s-value = 1. We need to compute the BC scores for when the s-value = 1 for the full IMDB dataset or possibly use the smaller subset of the IMDB dataset and see if the BC values can validate the result across the other values in the s-ensemble. Thus the present study raises several interesting questions we hope to answer in future works – where do these actors with the highest BC score for a particular s-value lie when the s value is 1. Are they in one of the top 10 percent highest BC scores? Does the s-line approximation help to provide faster centrality results?

Acknowledgment

We would like to thank Tony Liu for his help and support throughout this project–the many hours spent troubleshooting the various issues are much appreciated. We would also like to thank the PNNL team for their support and suggestions for the project. Finally, we would like to thank Dr. Assefaw Grebremedhin, the instructor for this class, for giving us the opportunity to work on this project. It's been an exciting journey as it has exposed us to the real-life applications of concepts we have learned in class.

Bibliography

[1] Alessia Antelmi et al. "Analyzing, exploring, and visualizing complex networks via hypergraphs using SimpleHypergraphs. jl". In: *arXiv preprint arXiv:2002.04654* (2020).

[2] Ulrik Brandes. "A faster algorithm for betweenness centrality". In: *Journal of mathematical sociology* 25.2 (2001), pp. 163–177.

[3] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". In: *InterJournal* Complex Systems (2006), p. 1695. URL: https://igraph.org.

[4] *IMDB Datasets*. https://datasets.imdbws.com/. [Online; accessed 4-April-2022].

[5] *IMDB Wikipedia*. https://en.wikipedia.org/wiki/IMDb. [Online; accessed 3-May-2022].

[6] Jongshin Lee et al. "Betweenness centrality of teams in social networks". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.6 (2021), p. 061108.

[7] Xu T Liu et al. "High-order Line Graphs of Non-uniform Hypergraphs: Algorithms, Applications, and Experimental Analysis". In: *arXiv preprint arXiv:2201.11326* (2022).

[8] Xu T Liu et al. “Parallel Algorithms and Heuristics for Efficient Computation of High-Order Line Graphs of Hypergraphs”. In: *arXiv preprint arXiv:2010.11448* (2020).

[9] *NWGr GitHub*. https://github.com/NWmath/NWgr. [Online; accessed 3-May-2022].

[10] *NWHy GitHub*. https://github.com/NWmath/NWhy. [Online; accessed 3-May-2022].

[11] Rami Puzis, Manish Purohit, and VS Subrahmanian. “Betweenness computation in the single graph representation of hypergraphs”. In: *Social networks* 35.4 (2013), pp. 561–572.

Appendix

A. Code for Plots Generation Using R

```
install.packages("igraph")
library(igraph)

t3path <- "top3nodes.csv"

t3 <- read.csv(t3path)

for (row in 1:nrow(t3)) {
  s <- as.integer(t3[row, "s"])
  node1 <- as.integer(t3[row, "node1"])+1
  node2 <- as.integer(t3[row, "node2"])+1
  node3 <- as.integer(t3[row, "node3"])+1

  gPath <- paste("sline/", s, ".edges",
    sep = "")
  jpegP <- paste("/jpegs/", s, ".jpg",
    sep = "")

  g = read_graph(gPath, "edgelist")

  V(g)$color <- ifelse(V(g) == node1,
                    "red",
                ifelse(V(g) == node2,
                    "green",
                ifelse(V(g) == node3,
                    "blue", "gray")))

  V(g)$size <- ifelse(V(g) == node1,
                    25,
                ifelse(V(g) == node2,
                    20,
                ifelse(V(g) == node3,
                    15, 10)))

  jpeg(jpegP)
  plot(g, layout=layout_with_kk,
     vertex.color=V(g)$color,
     vertex.size=V(g)$size)
  dev.off()
}
```

B. Modified Code for hypergraph computations

Code is located here: https://github.com/NWmath/NWhy/blob/imdb-explore/bench/imdbf.cpp

C. Full Table for Centrality Result

Table IV shows the full results.

TABLE IV: Betweenness Centrality Results showing Top 3 actors for some s in the ensemble. The s values that have the same actor as the previous s value were skipped in this table due to space constraint

s	Top 1 Actor	Top 1 Score	Top 2 Actor	Top 2 Score	Top 3 Actor	Top 3 Score	Components
130	Adoor Bhasi	0.0833333	Jayabharati	0	Bahadur	0	2
125	Adoor Bhasi	0.05	Matsunosuke Onoe	0.05	Jayabharati	0	2
101	Adoor Bhasi	0.0138889	Matsunosuke Onoe	0.00231481	Kijaku Ôtani	0.00231481	3
97	Adoor Bhasi	0.0111111	Matsunosuke Onoe	0.0111111	Jayabharati	0	3
90	Adoor Bhasi	0.00549451	Matsunosuke Onoe	0.0032967	Ritoku Arashi	0.0010989	4
86	Adoor Bhasi	0.00549451	Matsunosuke Onoe	0.0032967	Bahadur	0	4
84	Adoor Bhasi	0.00416667	Kijaku Ôtani	0.000833333	Matsunosuke Onoe	0.000833333	5
76	Adoor Bhasi	0.00326797	Matsunosuke Onoe	0.000653595	Kijaku Ôtani	0.000653595	6
75	Adoor Bhasi	0.0021645	Kijaku Ôtani	0.0004329	Matsunosuke Onoe	0.0004329	8
73	Adoor Bhasi	0.00197628	Ritoku Arashi	0.000790514	Kijaku Ôtani	0.000395257	8
71	Adoor Bhasi	0.00132275	Ritoku Arashi	0.000529101	Takeo Azuma	0.00026455	10
70	Adoor Bhasi	0.00132275	Ritoku Arashi	0.000529101	Kitsuraku Arashi	0.0000881834	10
68	Adoor Bhasi	0.00123153	Matsunosuke Onoe	0.00106732	Ritoku Arashi	0.000492611	10
62	Matsunosuke Onoe	0.000674764	Adoor Bhasi	0.000404858	T.N. Balakrishna	0.0000449843	13
60	Matsunosuke Onoe	0.000641026	Adoor Bhasi	0.000384615	Aachi Manorama	0.000042735	13
58	Matsunosuke Onoe	0.000528541	Adoor Bhasi	0.00038055	Kijaku Ôtani	0.0000634249	15
57	Matsunosuke Onoe	0.00042517	Adoor Bhasi	0.000306122	Aachi Manorama	0.000136054	16
55	Matsunosuke Onoe	0.000324675	Shivaji Ganesan	0.00021645	Adoor Bhasi	0.000177096	17
52	Matsunosuke Onoe	0.000219491	Adoor Bhasi	0.0000940675	Shivaji Ganesan	0.0000739102	20
51	Matsunosuke Onoe	0.000185117	Shivaji Ganesan	0.0000708272	Adoor Bhasi	0.0000708272	22
50	Adoor Bhasi	0.000154321	Matsunosuke Onoe	0.000133825	Shivaji Ganesan	0.0000530478	24
45	Adoor Bhasi	0.0000865351	Shivaji Ganesan	0.0000450704	Nagesh	0.0000378591	31
44	Adoor Bhasi	0.0000762776	Nagesh	0.0000284425	Shivaji Ganesan	0.0000206855	33
38	Nagesh	0.0000321007	Gemini Ganesan	0.0000247175	Savitri	0.0000235405	49
30	Shivaji Ganesan	0.0000100462	Nagesh	0.00000549609	S.V. Ranga Rao	0.00000540096	79
29	Shivaji Ganesan	0.00000883143	S.V. Ranga Rao	0.00000545872	Nagesh	0.00000414863	83
28	Shivaji Ganesan	0.00000740434	S.V. Ranga Rao	0.00000497585	Adoor Bhasi	0.00000391566	85
27	Shivaji Ganesan	0.0000062034	S.V. Ranga Rao	0.00000431616	N.T. Rama Rao	0.00000336036	86
26	Shivaji Ganesan	0.0000051071	Adoor Bhasi	0.00000214716	Savitri	0.00000210768	99
25	Shivaji Ganesan	0.00000442282	Gummadi	0.0000036444	Krishna	0.0000035778	101
22	Shivaji Ganesan	0.00000240743	Kaikala Satyanarayana	0.00000149813	Brahmanandam	0.00000140327	131
20	Shivaji Ganesan	0.00000168444	Brahmanandam	0.00000106203	Kaikala Satyanarayana	0.000000813775	145
17	Gummadi	0.000000884148	Jaya Prada	0.000000710024	Jeetendra	0.000000696368	188
16	Jaishankar	0.000000652995	Srividya	0.000000646674	Madhu	0.000000640677	217
15	Jaishankar	0.000000485958	Srividya	0.000000481478	Pandari Bai	0.000000368625	248
14	Srividya	0.000000377536	Jeetendra	0.000000295201	Aachi Manorama	0.00000027993	256
13	Jeetendra	0.000000283084	Ashok Kumar	0.000000196188	Sridevi	0.000000192098	292
12	Jeetendra	0.000000212658	Sridevi	0.000000145769	Helen	0.000000129337	334
11	Jeetendra	0.000000151109	Helen	0.000000114345	Sridevi	0.000000107567	382
10	Jeetendra	0.000000106901	Helen	0.0000000926395	Jaya Prada	0.0000000891955	422
9	Jaya Prada	0.0000000727116	Jeetendra	0.000000072377	Sridevi	0.0000000581119	497
8	Jeetendra	0.0000000463378	Jaya Prada	0.000000042746	Sultan Rahi	0.0000000386357	585
7	Jaya Prada	0.0000000279499	Jeetendra	0.0000000222057	Mumtaz Askari	0.0000000212169	716
6	Sultan Rahi	0.0000000155996	Mumtaz Askari	0.0000000153083	Brahmanandam	0.000000013428	894
5	Junko Miyashita	0.00000000803423	Nobuko Otowa	0.00000000791085	Taiji Tonoyama	0.00000000785657	1112
4	Sessue Hayakawa	0.00000000333873	Misao Seki	0.00000000333529	Kisaburô Kurihara	0.00000000333227	1776
3	Misao Seki	0.0000000010088	Sessue Hayakawa	0.000000000963373	Kazuo Hasegawa	0.000000000757661	3140
2	Vin Diesel	0.000000000141887	Deepika Padukone	0.000000000141673	Richard Harrison	0.000000000122544	8616