Tool Link:

VizHub:https://vizhub.com/Khaninsi/e1b050ad56cb4c7688835bf745c60a2d?edit=files&file=config.json
GitHub: https://github.com/Khaninsi/Force-DirectedHierarchyGraph

Why using this tool?

This tool leverages node2vec on the original network to effectively detect communities within the network. The primary goal is to break down massive amounts of data, thereby reducing information overload and allowing users to interact with a more manageable size of information.

Features:

- Each node is interactive, allowing users to drag nodes around the screen for better visualization.
- Clicking on a node provides further investigation. If the node represents a community, the graph will display the nodes and edges inside the community. Otherwise, the node will be highlighted along with its neighbors.
- Users can zoom in and out of the screen by scrolling, enabling a more detailed view of the network.
- The thickness of edges indicates the number of connections between nodes.
- Community size reflects the number of nodes within each community.
- The color of a node will follow the color of its community.

To customize the visualization, please access and modify the 'config.json' file. Below, you'll find explanations for each parameter.

Configuration explanation:

```
  "maxNumNodes":5000,
  "numNodePerCluster": 300,
  "numEdgePerCluster": 700,
  "numAdjNodePerCluster": 150,
  "numAdjEdgePerCluster": 300,
  "numWalks":    50,
  "walkLength":  5,
  "dimensions":  20,
  "numClusters": 15,
  "maxIterations": 500,
```

```
  "id2nameURL": "https://raw.githubusercontent.com/Khaninsi/D3-visualization/main/data/user.json",
  "edgesURL": "https://raw.githubusercontent.com/Khaninsi/D3-visualization/main/data/enron_edges.txt",
  "communityFile": " https://raw.githubusercontent.com/Khaninsi/D3-visualization/main/data/enron-community.json"
```

Parameters for Visualization Control: The first five configuration variables are designed to control the quantities of nodes, edges, and clusters displayed within the visualization.

- **“maxNumNodes”:** This represents the maximum number of nodes that are displayed within the visualization. The nodes are sorted based on their degrees, which typically refer to the number of edges connected to a node. The nodes with the highest degrees are prioritized for inclusion. The actual nodes displayed will be selected based on this maximum value, favoring those with higher degrees.
- **“numNodePerCluster”:** This defines the maximum number of nodes that can belong to the same cluster. Clusters are groups of nodes that share similar attributes or properties and are represented with the same color.
- **“numEdgePerCluster”:** This sets the maximum number of edges that can exist within a cluster. Edges are the connections or relationships between nodes. Specifically, this limit applies only to edges that connect nodes within the same cluster and share the same color.
- **“numAdjNodePerCluster”:** This specifies the highest number of adjacent nodes that can be associated with a single cluster. Adjacent nodes are nodes that are connected directly to a cluster's nodes but do not belong to the cluster itself.
- **“numAdjEdgePerCluster”:** This sets the maximum number of adjacent edges that can connect nodes from the cluster to adjacent nodes. These edges extend from the cluster's nodes to the adjacent nodes that are not part of the cluster.

Node2vec parameters

- **"numWalks":** This parameter defines the number of random walks to be generated for each node in the graph. A random walk is a sequence of nodes traversed by moving from one node to a neighboring node based on certain criteria. Generating multiple random walks for each node helps in capturing different perspectives of node relationships.
- **"walkLength":** The walk length determines how many steps (node transitions) are taken in each random walk. Longer walk lengths can capture more distant relationships in the graph, while shorter walk lengths focus on local neighborhood structures.
- **"dimensions":** This parameter indicates the dimensionality of the embeddings produced by Node2vec. Embeddings are low-dimensional vector representations of nodes that capture their graph-related characteristics. The "dimensions" value determines the length of these vectors.

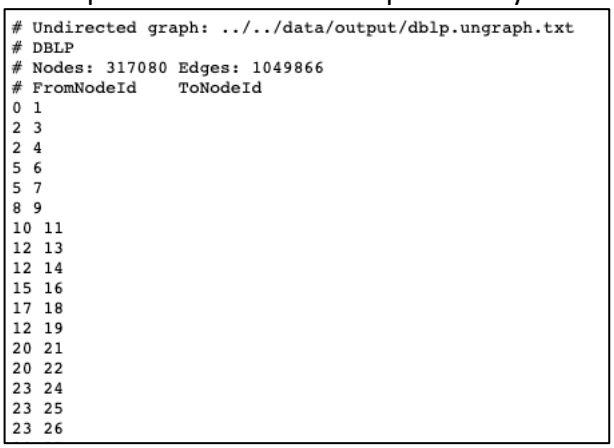
- **"numClusters":** The number of clusters you want to partition the nodes into after obtaining embeddings. Clustering groups similar nodes together based on their embedding vectors, helping to discover communities or groups in the graph.
- **"maxIterations":** This parameter defines the maximum number of iterations or optimization steps that the algorithm will perform. Node2vec uses optimization techniques to learn node embeddings. The algorithm iteratively adjusts the embeddings to improve their representation of the graph structure. The "maxIterations" value sets an upper limit on how many times this optimization process will be repeated.

Graph Visualization: Providing Input URL; The easiest way to provide the link is to upload to GitHub and put the link into these three variables.

- **"id2nameURL": (Optional)** If your nodes are represented by numerical IDs and you wish to translate them into meaningful entity names, please include the link that facilitates the mapping of IDs to entity names.
  Support format: https://raw.githubusercontent.com/Khaninsi/D3-visualization/main/data/user.json
  Brief example:

  ```
  {"": 0, "rfranson czn": 1, "duluth": 2, "jay": 3, "zhou zou": 4, "greenberg": 5, "sbutler nutrastar csbutler": 6, "kheal": 7, "dalton mcgrath": 8, "ffelvey csipaper": 9, "owner hsa users highschoolalumni": 10,…}
  ```

- **"edgesURL": (Required)** This parameter specifies the URL containing the edges connecting pairs of nodes. The system supports both SNAP and JSON formats.
  1. **Example for SNAP format:** https://raw.githubusercontent.com/Khaninsi/D3-visualization/main/data/email-Eu-core2.txt
     Brief example: It's acceptable for the initial lines not to represent edges. The connections between each pair of nodes can be separated by either spaces or tabs.



```
# Undirected graph: ../../data/output/dblp.ungraph.txt
# DBLP
# Nodes: 317080 Edges: 1049866
# FromNodeId    ToNodeId
0 1
2 3
2 4
5 6
5 7
8 9
10 11
12 13
12 14
15 16
17 18
12 19
20 21
20 22
23 24
23 25
23 26
```


  2. **Example for JSON format:** https://raw.githubusercontent.com/Khaninsi/D3-visualization/main/data/miserables.json

Brief example:


```
{
"nodes":[{"id":"Myriel","group":1},{"id":"Napoleon","group":1},{"id":"
Mlle.Baptistine","group":1}, …],

"links":[{"source":"Napoleon","target":"Myriel","value":1},{"source":"
Mlle.Baptistine","target":"Myriel","value":8}, …]
}
```


- **“communityFile”: (Optional)** If you possess your own community detection results, you can simply input the link to visualize them. Otherwise, this visualization will execute Node2Vec and autonomously detect communities.
  Expected format: For the key, use the community's name, and for the values, list the nodes within each community. If an "id2nameURL" is provided, **ensure that the node names are represented as their actual names rather than IDs.**
  Brief example:


```
{
    "Com.0": [
        "ssims bechtel",
        "kim onofrio ckim mooneyinsurance",
        "aguilera peon maria teresa",
        "gpg general",
        "dmaxey",
        "brobinhold ftenergy"
    ],
    "Com.1": [
        "ena ranigel",
        "priscilla ebabilonia orionpowerny",
        "ae",
        "ofanjul omega capital",
        "edmund cooper"],
    …
}
```


If encounter any problems, please email to:

- sisaengs@usc.com, lastcalled.t@gmail.com