Konstantinos Ilias
DS210 Project Write-Up

Dataset/Project Description
This project uses data from email communication of employees of Enron, a company that filed for bankruptcy in 2001 because it committed accounting fraud.

I found the data here:https://snap.stanford.edu/data/email-Enron.html

I am using the SNAP Enron dataset(email-Enron (1).txt) which is a dataset of edges where each node is an email address. Although the original email data is directional meaning one user sends a message to another, the SNAP Enron dataset treats the email communication as undirected, meaning an edge exists between two nodes if at least one email was exchanged regardless of its direction.

The SNAP database provides numeric node IDs with no direct email mapping. However, another dataset was provided (enron_mail_20150507.tar.gz), which contains over 500,000 emails organized in employee folders within a maildir directory, where each folder (e.g., lay-k, skilling-j) represents an Enron employee's mailbox. Sender and receiver email addresses are provided as well.

I used python to create another csv file that maps the numerical nodes to the email addresses and employee folders. With python I produced email_to_node.csv, which mapps NodeID to Email to Folder. Emails and folders can be different as only employees have folders but the dataset has a lot of non Enron emails. Therefore, these non Enron emails are saved in folders of Enron employees.

What does the code do, how to run it?
The program performs three types of centrality analysis (Degree, Closeness, Betweenness) and finds clusters to identify representative nodes within each cluster. Given that degree centrality is the least computationally heavy to implement, I calculated closeness and betweenness centrality only on the top 1000 nodes with the highest degree centrality. The code prints the top 10 nodes by: 1) Degree Centrality 2)Closeness Centrality 3)Betweenness Centrality and also prints cluster leaders (top node by degree in each cluster) and cluster the nodes by all centrality measures using k-means. It finally generates the following plots:
1)degree_histogram.png 2)closeness_vs_degree.png 3)betweenness_histogram.png 4) clusters.png

- Degree Centrality tells us how many direct connections a node has, indicating how active an individual is in the network.
- Closeness Centrality measures how close a node is to all others via shortest paths, identifying those who can quickly communicate with everyone else.
- Betweenness Centrality captures how often a node lies on the shortest paths between other nodes. These nodes serve as key connectors or information brokers.

To run the code one needs these two datasets: email-Enron (1).txt, email_to_node.csv, these four rust modules:

- main.rs: Uses all functions to print the top 10 nodes given different centrality measures
- graph.rs: Reads the files, conducts the mapping and computes centrality measures
- cluster.rs: Divides nodes into clusters with the help of BFS and k-means clustering
- plot.rs: Generates plots using the plotters crate

and an environment that supports Rust and cargo. Using the cargo run –release >output.txt command the program takes around 15 seconds to generate the output.txt file which contains the output.

These are some important functions used and what they do:

- ☐ read_file(path: &str) -> Vec<(usize, usize)>: Reads the edge list from the dataset and returns a list of email communication pairs.
- ☐ load_email_mapping(path: &str) -> HashMap<usize, (String, String)>: Maps numeric node IDs to actual email addresses and employee folders.
- ☐ compute_degree(edges: &[(usize, usize)]) -> HashMap<usize, usize>: Calculates the degree (number of direct connections) for each node.
- ☐ compute_closeness(edges: &[(usize, usize)], nodes: &HashSet<usize>) -> HashMap<usize, f64>: Computes closeness centrality by evaluating shortest path distances.
- ☐ compute_betweenness(edges: &[(usize, usize)], nodes: &HashSet<usize>) -> HashMap<usize, f64>: Calculates betweenness centrality by counting shortest paths passing through each node.
- ☐ find_clusters(edges: &[(usize, usize)]) -> Vec<HashSet<usize>>: Identifies clusters of connected nodes using breadth-first search (BFS).
- ☐ kmeans(features: &HashMap<usize, (f64, f64, f64)>, k: usize, max_iters: usize) -> HashMap<usize, usize>: Performs K-Means clustering on degree, closeness and betweenness centrality of each node to assign nodes to clusters.
- ☐ normalize_features(features: &mut HashMap<usize, (f64, f64, f64)>): Normalizes features to ensure equal weighting during clustering.

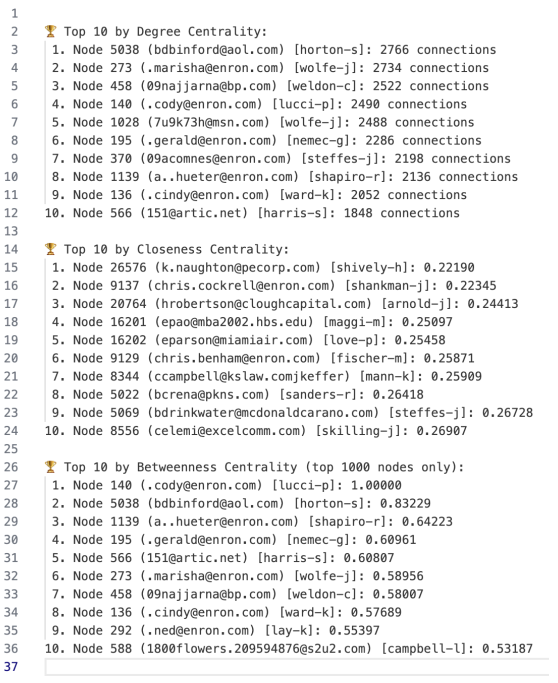
What is the output?

The code produces rankings of the most important individuals in the Enron email network based on three centrality metrics. Nodes with the highest degree centrality (e.g., bdbinford@aol.com, .marisha@enron.com) had the most direct email connections, meaning they were highly active in sending and receiving emails. Nodes with high closeness centrality could reach others quickly in the network. Those with high betweenness centrality often served as connections between different groups, playing a role in information flow and inter-group communication. Additionally, the network was divided into clusters of connected individuals, and the most connected node in each cluster was identified as its leader, revealing the most central figure within each communication cluster. Nodes were also divided into clusters based on their centrality measures using k-means.

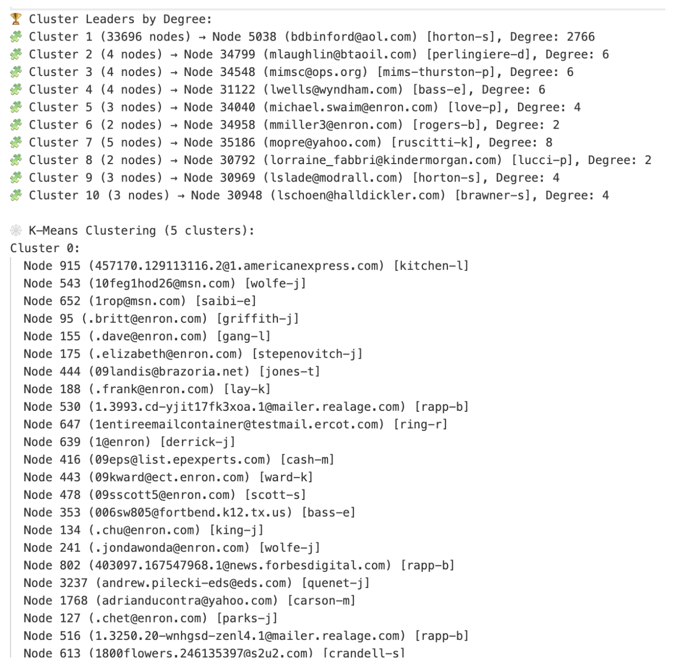
These four executives got convicted of fraud: **Jeffrey Skilling** – Former CEO, **Andrew Fastow** – Former CFO, **Kenneth Lay** – Former Chairman and CEO and **Richard Causey** – Former Chief Accounting Officer.

Based on the output of the code: **Jeffrey Skilling** appears in **Closeness Centrality** and **Cluster 4** based on degree and **Kenneth Lay** appears in **Betweenness Centrality**. On the clusters generated with k-means, emails that belong to their folders appear in multiple clusters. However, most emails on **Skilling**'s folder appear to be concentrated in **Cluster 2**.

Output when using cargo run - -release >output.txt



```
🏆 Top 10 by Degree Centrality:
 1. Node 5038 (bdbinford@aol.com) [horton-s]: 2766 connections
 2. Node 273 (.marisha@enron.com) [wolfe-j]: 2734 connections
 3. Node 458 (09najjarna@bp.com) [weldon-c]: 2522 connections
 4. Node 140 (.cody@enron.com) [lucci-p]: 2490 connections
 5. Node 1028 (7u9k73h@msn.com) [wolfe-j]: 2488 connections
 6. Node 195 (.gerald@enron.com) [nemec-g]: 2286 connections
 7. Node 370 (09acomnes@enron.com) [steffes-j]: 2198 connections
 8. Node 1139 (a..hueter@enron.com) [shapiro-r]: 2136 connections
 9. Node 136 (.cindy@enron.com) [ward-k]: 2052 connections
10. Node 566 (151@artic.net) [harris-s]: 1848 connections

🏆 Top 10 by Closeness Centrality:
 1. Node 26576 (k.naughton@pecorp.com) [shively-h]: 0.22190
 2. Node 9137 (chris.cockrell@enron.com) [shankman-j]: 0.22345
 3. Node 20764 (hrobertson@cloughcapital.com) [arnold-j]: 0.24413
 4. Node 16201 (epao@mba2002.hbs.edu) [maggi-m]: 0.25097
 5. Node 16202 (eparson@miamiair.com) [love-p]: 0.25458
 6. Node 9129 (chris.benham@enron.com) [fischer-m]: 0.25871
 7. Node 8344 (ccampbell@kslaw.comjkeffer) [mann-k]: 0.25909
 8. Node 5022 (bcrena@pkns.com) [sanders-r]: 0.26418
 9. Node 5069 (bdrinkwater@mcdonaldcarano.com) [steffes-j]: 0.26728
10. Node 8556 (celemi@excelcomm.com) [skilling-j]: 0.26907

🏆 Top 10 by Betweenness Centrality (top 1000 nodes only):
 1. Node 140 (.cody@enron.com) [lucci-p]: 1.00000
 2. Node 5038 (bdbinford@aol.com) [horton-s]: 0.83229
 3. Node 1139 (a..hueter@enron.com) [shapiro-r]: 0.64223
 4. Node 195 (.gerald@enron.com) [nemec-g]: 0.60961
 5. Node 566 (151@artic.net) [harris-s]: 0.60807
 6. Node 273 (.marisha@enron.com) [wolfe-j]: 0.58956
 7. Node 458 (09najjarna@bp.com) [weldon-c]: 0.58007
 8. Node 136 (.cindy@enron.com) [ward-k]: 0.57689
 9. Node 292 (.ned@enron.com) [lay-k]: 0.55397
10. Node 588 (1800flowers.209594876@s2u2.com) [campbell-l]: 0.53187
```

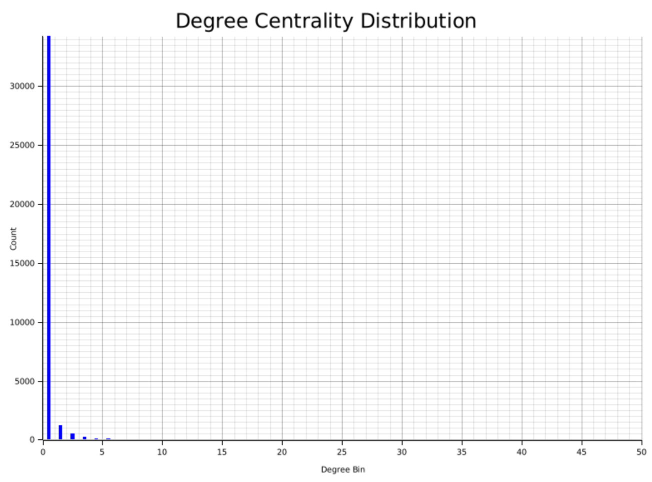


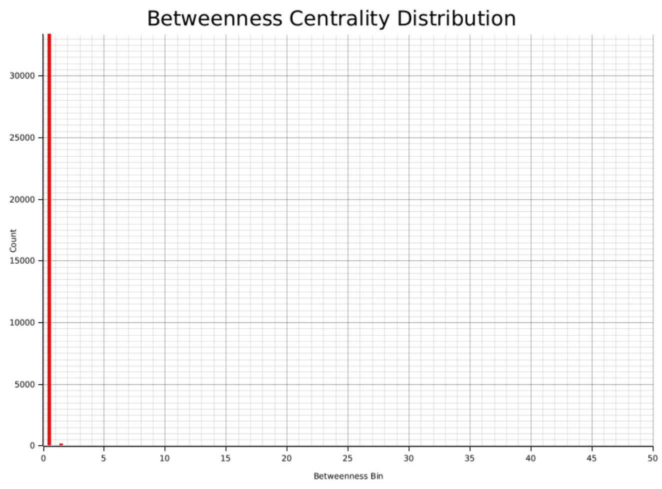
```
🏆 Cluster Leaders by Degree:
🧩 Cluster 1 (33696 nodes) → Node 5038 (bdbinford@aol.com) [horton-s], Degree: 2766
🧩 Cluster 2 (4 nodes) → Node 34799 (mlaughlin@btaoil.com) [perlingiere-d], Degree: 6
🧩 Cluster 3 (4 nodes) → Node 34548 (mimsc@ops.org) [mims-thurston-p], Degree: 6
🧩 Cluster 4 (4 nodes) → Node 31122 (lwells@wyndham.com) [bass-e], Degree: 6
🧩 Cluster 5 (3 nodes) → Node 34040 (michael.swaim@enron.com) [love-p], Degree: 4
🧩 Cluster 6 (2 nodes) → Node 34958 (mmiller3@enron.com) [rogers-b], Degree: 2
🧩 Cluster 7 (5 nodes) → Node 35186 (mopre@yahoo.com) [ruscitti-k], Degree: 8
🧩 Cluster 8 (2 nodes) → Node 30792 (lorraine_fabbri@kindermorgan.com) [lucci-p], Degree: 2
🧩 Cluster 9 (3 nodes) → Node 30969 (lslade@modrall.com) [horton-s], Degree: 4
🧩 Cluster 10 (3 nodes) → Node 30948 (lschoen@halldickler.com) [brawner-s], Degree: 4

🕸 K-Means Clustering (5 clusters):
Cluster 0:
  Node 915 (457170.129113116.2@1.americanexpress.com) [kitchen-l]
  Node 543 (10feg1hod26@msn.com) [wolfe-j]
  Node 652 (1rop@msn.com) [saibi-e]
  Node 95 (.britt@enron.com) [griffith-j]
  Node 155 (.dave@enron.com) [gang-l]
  Node 175 (.elizabeth@enron.com) [stepenovitch-j]
  Node 444 (09landis@brazoria.net) [jones-t]
  Node 188 (.frank@enron.com) [lay-k]
  Node 530 (1.3993.cd-yjit17fk3xoa.1@mailer.realage.com) [rapp-b]
  Node 647 (1entireemailcontainer@testmail.ercot.com) [ring-r]
  Node 639 (1@enron) [derrick-j]
  Node 416 (09eps@list.epexperts.com) [cash-m]
  Node 443 (09kward@ect.enron.com) [ward-k]
  Node 478 (09sscott5@enron.com) [scott-s]
  Node 353 (006sw805@fortbend.k12.tx.us) [bass-e]
  Node 134 (.chu@enron.com) [king-j]
  Node 241 (.jondawonda@enron.com) [wolfe-j]
  Node 802 (403097.167547968.1@news.forbesdigital.com) [rapp-b]
  Node 3237 (andrew.pilecki-eds@eds.com) [quenet-j]
  Node 1768 (adrianducontra@yahoo.com) [carson-m]
  Node 127 (.chet@enron.com) [parks-j]
  Node 516 (1.3250.20-wnhgsd-zenl4.1@mailer.realage.com) [rapp-b]
  Node 613 (1800flowers.246135397@s2u2.com) [crandell-s]
```

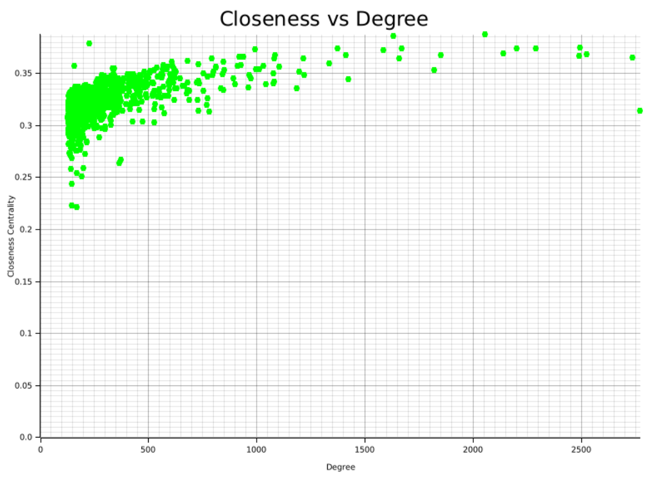
Output when using cargo test

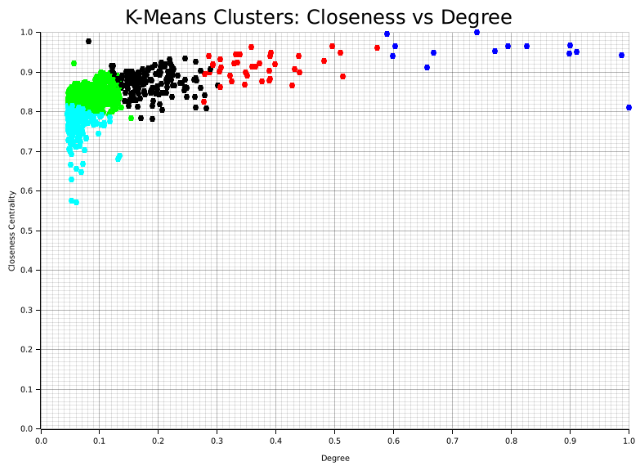
```
running 6 tests
test cluster::tests::test_find_clusters ... ok
test cluster::tests::test_normalize_features ... ok
test cluster::tests::test_kmeans ... ok
test graph::tests::test_compute_closeness ... ok
test graph::tests::test_compute_betweenness ... ok
test graph::tests::test_compute_degree ... ok

test result: ok. 6 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

Graphs









Degree Centrality Distribution: We see that most nodes have high degree, with only a few outliers who are not very connected to the rest of the nodes.

Betweenness Centrality Distribution: Most nodes have a very small betweenness centrality, with only a few nodes serving as communication bridges.

Closeness vs Degree: As expected we see that higher degree is associated with higher closeness centrality, meaning that nodes who are very connected can reach other nodes quickly.

K-Means Clusters: Closeness vs Degree: We see groups of nodes with different patterns of connectivity and influence in the Enron email network. It should be noted that the clustering was done on betweenness centrality as well and that this graph is just the 2D representation.