Thomas Larsen

12/15/2024

DS210

Final Project Write-Up

Data Set Link:

https://www.kaggle.com/datasets/parulpandey/us-international-air-traffic-data?resource=download

GitHub Link: https://github.com/tlarsen03/DS_210_Final_Project

This project has evolved considerably since the project proposal. This project originally started using Border Crossing data from the US government, treating each border crossing point as a node to understand how they are interconnected. Unfortunately, after I constructed the graph I realized there were only 116 nodes because there were only 116 different border crossing points. I tried adjusting the data to increase nodes, for example, treating each day as an individual node, but none of the transformations created a product that made sense. With this sad realization, I searched for data that I could adjust the code I had already created, looking for data that was interesting to me and satisfied the requirement of fitting my existing code. After much searching, I found U.S International Air Traffic data from 1990 to 2020. This data came from the US International Air Passenger and Freight Statistics Report and analyzes data on all flights between US gateways and non-US gateways irrespective of origin and destination. As someone who flies a considerable amount (to visit family, go on vacation, study abroad, etc) I was curious

to understand which airports are the busiest in the international US network and which airports are the best connected/most important for US international travel.

To answer this question I created a graph where each node represents an airport and each edge represents direct flight connections with the weight of each route corresponding to the total flight count. With these goals in mind, I started creating my code. For my dependencies, I brought in a CSV reader and initialized HashMap, HashSet, and VecDeque from the standard collections, Error handling, and CSV reader. I created a FlightData structure following the setup of the CSV table. I then created a read_csv function which created a vector of FlightData structs. Next, I created the build graph function which created an undirected weighted graph. I elected to use an undirected graph because, for connectivity and centrality analysis, directionality is not essential, especially because most airports have bidirectional flights, making the graph naturally undirected. This graph relied on helper functions, most importantly add_edge. This added an edge to the graph. The function created the graph so each airport is a node and each flight between two airports is an edge with a weight equal to the total flight count.

Next, I utilized some graphing algorithms. I used breadth-first search to calculate the shortest path (in terms of edges) from a starting node to all other reachable nodes. I used breadth-first search because I was looking at the shortest path in terms of the number of edges, not the weights on the edges because most of my work looks at the closeness of nodes and weight is less important. In BFS I use a Hashmap to keep track of the shortest distance to each node, and a queue that stores nodes to be processed, each item is a tuple. I use a while loop to run as long as there are nodes in the queue and use the pop_front function to call each node. I also make sure to mark each node as visited, record distance, and process all the neighbors, using the adjacency list.

In my first iteration of the code, I wanted to implement closeness centrality to understand which airports act as important nodes, but no matter my adjustments I kept on getting bizarre results. After a lot of searching around, I identified the issue. Closeness centrality does not work well with disconnected graphs because it relies on the assumption that all nodes in the graph are reachable from the node being analyzed, when a graph has disconnected components it considers the distances of those separate as infinite leading to undefined or extremely high closeness centrality scores for nodes in different components. This would explain the bizarre results I kept getting. To check my hypothesis I created and ran a connected components function. This function is designed to identify clusters of airports that are reachable from others helping understand how the international network works. The function uses a Hashset to track the visited nodes and a vector to hold the components. Then it iterates for all the nodes in the adjacency list, checks if the node has been visited, performs DFS to analyze the neighbors of each node, and then stores the completed component. Running this function found that there were 10 different connected components, of which there was one major one composed of 2670 out of 2689 nodes. This answered the question of why my centrality analysis was not working, but not wanting to start over again, I went searching for a function that would perform a similar analysis to centrality analysis but be able to handle disconnected components. After much searching, I found Harmonic centrality. Harmonic centrality handles disconnected nodes as zero as opposed to closeness centrality which considers them to be infinite. Harmonic centrality directly sums the inverse distances which emphasizes proximity while closeness centrality normalizes by total distances and penalizes nodes that are farther from others. In my code, we implement this by first initializing a vector to hold centrality scores and then iterating through every node in the graph adjacency list. I calculated distances using bfs_shortest_paths and then calculated the harmonic

sum by dividing 1 by the distance d and storing the result. I then sort through the scores so the highest appear first and take the top five results.

The next function I create is the top_busiest_airports which finds the airports with the highest total flight counts, this is to identify major hubs and complements the centrality analysis. The function simply iterates through all the flights in the CSV file, sums up the total flights for each airport, using a hashmap to store the data, and then returns the top five.

Finally, we get to the main function. This function reads the CSV file, constructs the graph and then outputs, the top 5 busiest airports, the graph statistics, the number and sizes of connected components, and the harmonic centrality for the five largest components. Looking at the results the top five airports are:

```
Top 5 Busiest Airports:
MIA: 5016561 flights
JFK: 3780640 flights
LAX: 3013661 flights
ORD: 2366023 flights
EWR: 2042525 flights
```

The top five airports by harmonic centrality are:

```
Top 5 Airports by Harmonic Centrality (Largest Component):
YYZ: 1465.5333
MIA: 1435.5667
JFK: 1387.3500
YVR: 1348.4167
YUL: 1337.0833
```

From this analysis, we see that the world flight network is very well connected, with almost every airport being connected to one another. The small disconnected components are likely small regional airports of little importance. The top five by busiest departures is Miami International Airport, JFK Airport, LAX Airport, Chicago O'Hare International Airport, and then

Newark Liberty International Airport (my home airport!). Interestingly the top five airports by Harmonic Centrality do not entirely line up. YYZ (Toronto Pearson International Airport) is the top with a score of 1465 indicating it is very highly connected and is a crucial airport for US international travel. This does make sense as it has a strategic geographic location. Interestingly, for my study abroad flight to Copenhagen, I connect through YYZ even though my home airport is EWR which is a major east coast hub. Next is MIA which also makes sense as it acts as a major gateway to Latin America and the Caribbean. JFK makes sense because it handles a lot of traffic and is located on the East Coast. The last two are YBR, acting as a gateway to Asia from the Pacific Northwest, and YUL, which is another Canadian airport. I was surprised by the prominence of Canadian airports in this list which demonstrates their importance in domestic and international travel. This information can also be used to plan which airport is best to connect to.

This data analysis only scratches the surface of what can be done with this data set. Additional research and code could analyze how air travel varies by season, which regions are the best connected, and what happens to networks if key hubs are removed. The purpose of this project was to understand how the network is connected at a fundamental level, these additional questions would take this analysis further.