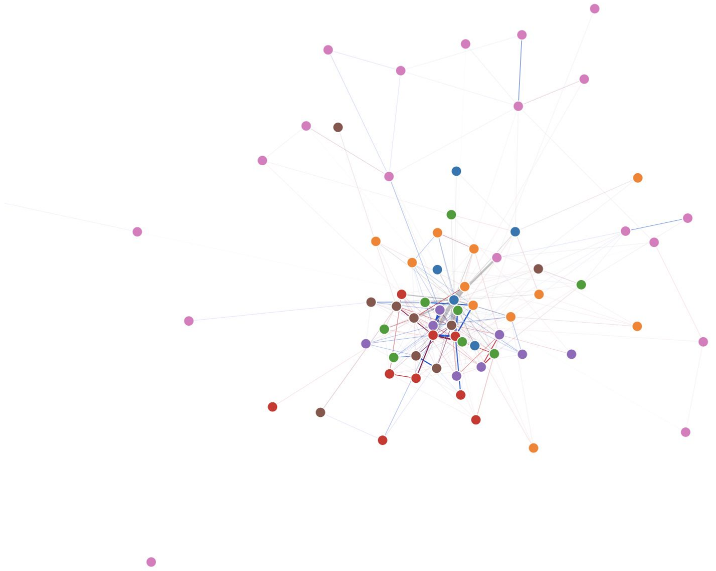
React + Django Network Graph

With AWS S3 and EC2

Brit Dao



The goal of this project

I get very frustrated when trying to make a network graph with javascript, as many javascript libraries either have little documentation or the graphics didn’t suit me. I was also dissatisfied with how they calculate the weighted node graphs. This accounts for how close the nodes would be using numbers, the larger the numbers the closer it would be.

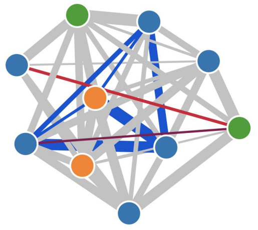
Python’s Networkx library provided positions that satisfies my needs for weighted graphs. I can then use these positions to plot a matplotlib graph. However I wanted some sort of interactivity to change my arguments on demand rather than rerunning and retyping it in the terminal, as well as getting a dynamic graph instead of a still image. I find D3js library useful with its customization options, and works with Reactjs.

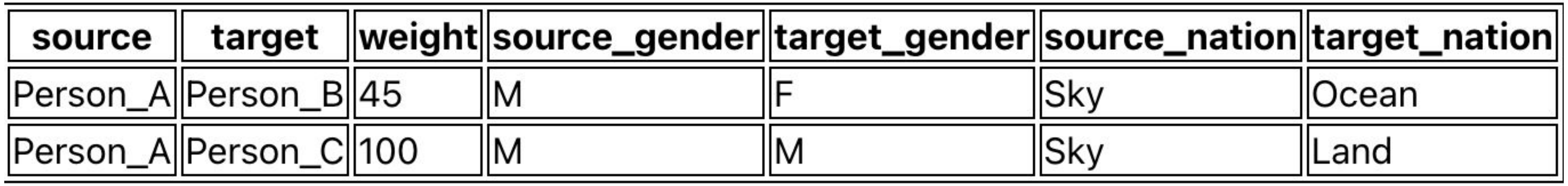
NetworkX
Network Analysis in Python

How The Network Graphs work

In order for this website to work, I need a csv listing the source, target, and weight. I also need the attributes such as gender and nation, as this network graph was used with the purpose of graphing visual relationships between characters.

The colored nodes represents what nation they are in, and the line between them indicates what type of relationship it is.
Blue is M/M , Purple is M/F, Red is F/F, and Grey is other.

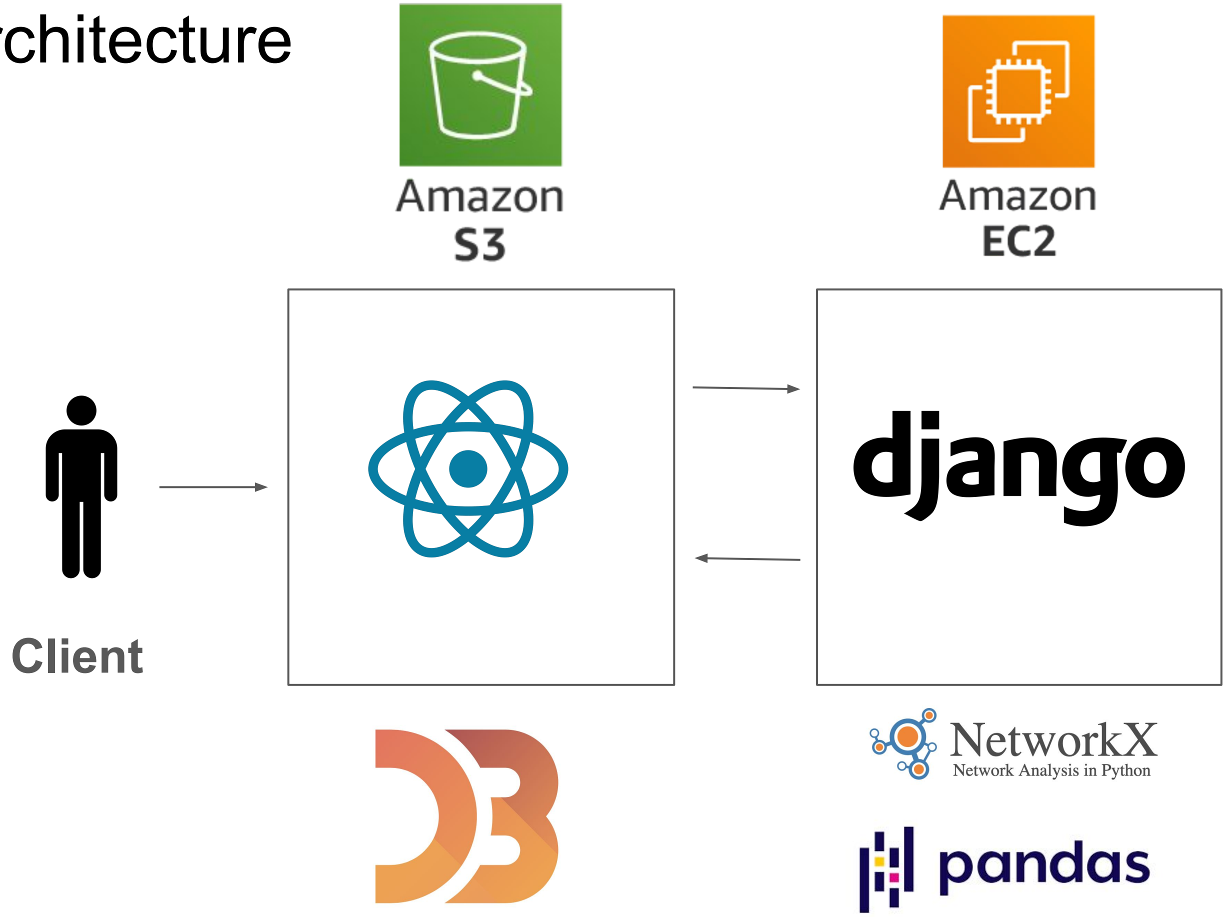




source	target	weight	source_gender	target_gender	source_nation	target_nation
Person_A	Person_B	45	M	F	Sky	Ocean
Person_A	Person_C	100	M	M	Sky	Land

Tech Stack and Architecture

The Tech Stack and Architecture is rather simple, I used AWS S3 as my frontend for React, and AWS EC2 as my backend for Django. I used other libraries like pandas, networkx, and d3js to create the application.



Deployment to AWS

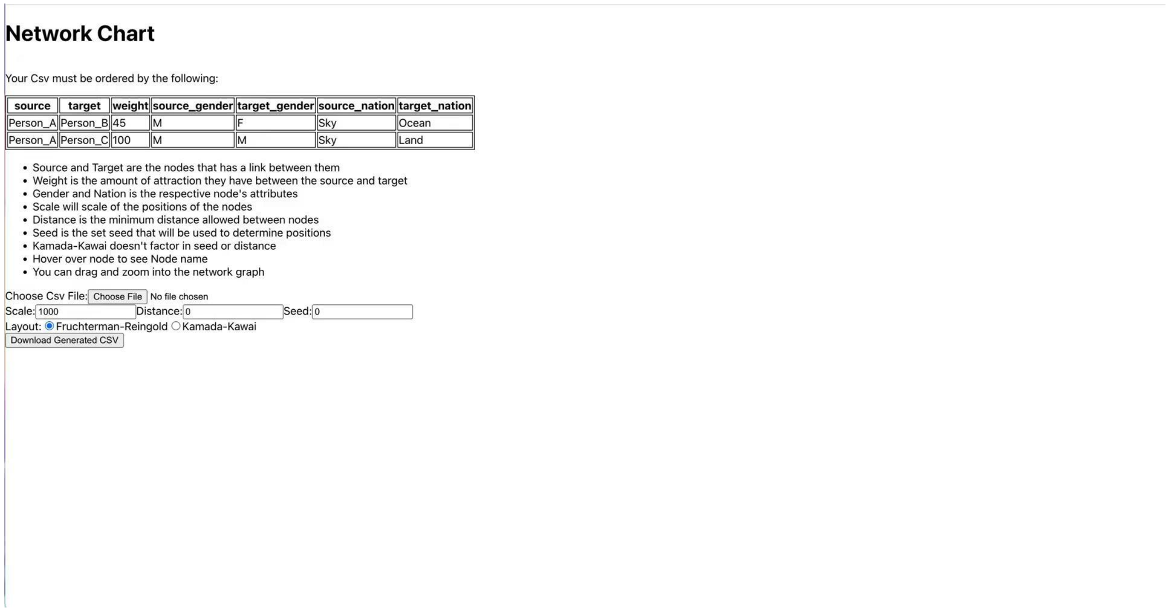
For S3 frontend:

- Created production build of React application
- Create a bucket in S3 with the build production
- Added bucket policy and changed public accessibility
- Choose index.html as my main website

For EC2 backend:

- Import Django Project to Github
- Create a instance and launch it with Ubuntu as the platform
- Git clone Django project to connect to instance with nohup on port 8000
- Change Inbound rules to source from its public IP

Video and Live Demo



Network Chart

Your Csv must be ordered by the following:

source	target	weight	source_gender	target_gender	source_nation	target_nation
Person_A	Person_B	45	M	F	Sky	Ocean
Person_A	Person_C	100	M	M	Sky	Land

- Source and Target are the nodes that has a link between them
- Weight is the amount of attraction they have between the source and target
- Gender and Nation is the respective node's attributes
- Scale will scale of the positions of the nodes
- Distance is the minimum distance allowed between nodes
- Seed is the set seed that will be used to determine positions
- Kamada-Kawai doesn't factor in seed or distance
- Hover over node to see Node name
- You can drag and zoom into the network graph

Choose Csv File: Choose File No file chosen
Scale: 1000 Distance: 0 Seed: 0
Layout: [✓ Fruchterman-Reingold] Kamada-Kawai
Download Generated CSV

Link to video in case the above doesn't render a video: https://docs.google.com/file/d/1vp5FesFtXCTmrzy427RCzPgOpp0adrg-/preview

The main function is to provide you with a mock up csv as well as providing your own csv to graph

You can visit this site to try the demo for yourself: http://networkdemo.s3.us-east-2.amazonaws.com/dist/index.html

FrontEnd Code: https://github.com/BritHD/NetworkFront

BackEnd Code: https://github.com/BritHD/NetworkServer