Analyzing spread of COVID-19 using Graph Neural Networks

By

Omkar Ranadive, Nikita Raut, Anupam Tripathi

Theme: The aim of this project is to provide users with an end-to-end pipeline for processing data related to COVID-19, create graph structure from that data and allow users to easily manage node features and edge weights.

Index:

Links:

Youtube Video: https://youtu.be/P4u7emR7l9k

Github Repository: https://github.com/Omkar-Ranadive/COVID-19-GNN

Introduction

The aim of this project is to try and help alleviate the impact of COVID-19 by analyzing its spread using Graph Neural Networks. This project delivers an end-to-end pipeline to process COVID-19 data, form a graph out of it and apply various Graph Neural Network algorithms on it. The codebase is extensible and can be easily modified to create different graph representations.

Our project has the following major components:

- **Data Loader:** A data loader class to easily load population data, flight data and form graph nodes. It can also handle timeseries data.
- **Graph Loader:** A graph loader to create the graph representation in the form of a Pytorch-Geometric object. It has other features like creating graph edges, creating edge weights, filtering edges and creating node features.
- **Models:** The repository comes with three models – Graph Convolution Network, Message Passing Network, Sage Convolution Network. Alternatively, users can easily add their own Graph Networks.
- **Visualizer:** A visualizer is provided to visualize the graph formed with functionality like identifying important nodes and generating heat maps.

The Graph Structure

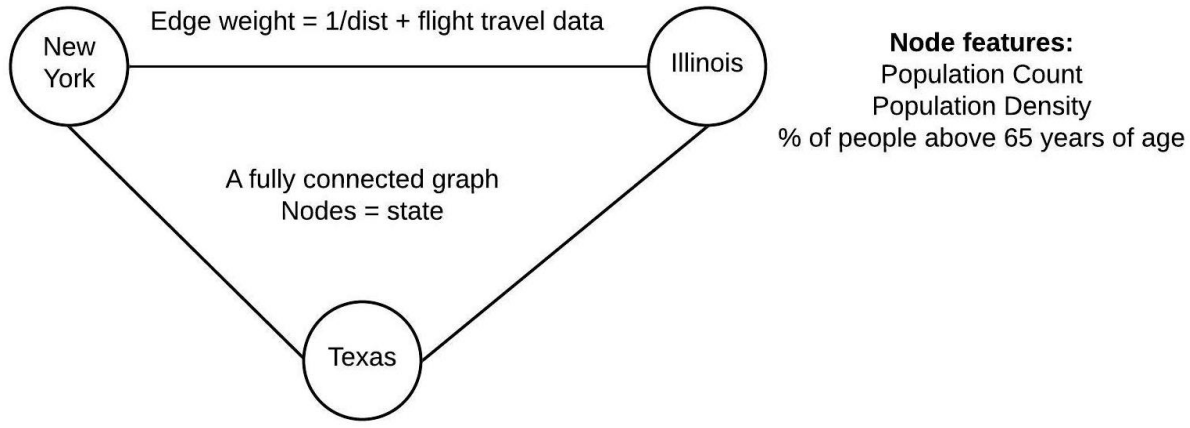

Fig 1: A basic graph representation of the data

Node representation:

The nodes in our graph are represented as “places”. These places can be of different granularity – i.e, a town, a province, a state. The user can decide the granularity which they want. In our implementation, we chose a node to be a state as it was easy to find state level information on the internet; allowing us to create rich node features. The node features which we have selected are as follows:

- **For non-timeseries data:** Population Count, Population Density, Percentage of people above the age of 65. Total features = 3. (Features are normalized)
- **For timeseries data:** Number of infected people in the past 14 days + population count, population density and % of people above 65 years of age. Total features = 17. (Features are normalized)

Node labels:

The node labels or the node output is the number of infected people in that state.

Edge representation:

We assume a fully connected graph structure. That is, each node will be connected to every other node. Hence, there will be a total of $n * \frac{n-1}{2}$ edges.

Every edge is weighted based on two parameters: Inverse of distance + Flight count between the nodes of the two edges. We calculate the distance by calculating the geodesic distance between the states using the latitude and longitude co-ordinates of the state.

Hence, nodes (states) which are closer by will have a higher inverse distance value and nodes with more flights going to and fro between them will have a higher value. This way, we get large edge weight values for related nodes (states). Additionally, we provide a filter function which can delete edges based on a threshold value (i.e., if edge weight < threshold, delete the edge).

In brief, a graph neural network algorithm will try to make use of information of a node's neighboring states and use that to predict the number of infected people for that given node (state). For the sake of simplicity during demonstration we have only considered data from USA. So, the graph structure is one in which the nodes are states of USA.

Model Architectures

We have implemented three model architectures in this project. They are as follows:

Graph Convolution Network

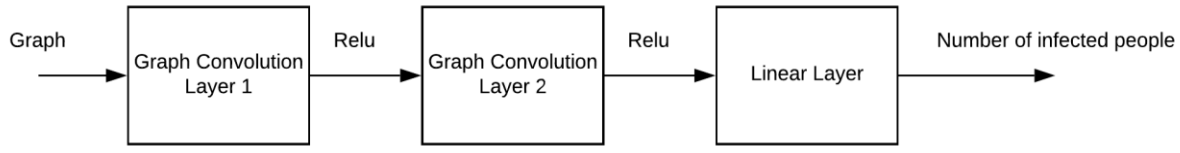

Fig 2: Graph Convolution Network Architecture

In the first network, we have implemented a two-layer graph convolution network as shown in figure 2.

Message Passing Network:

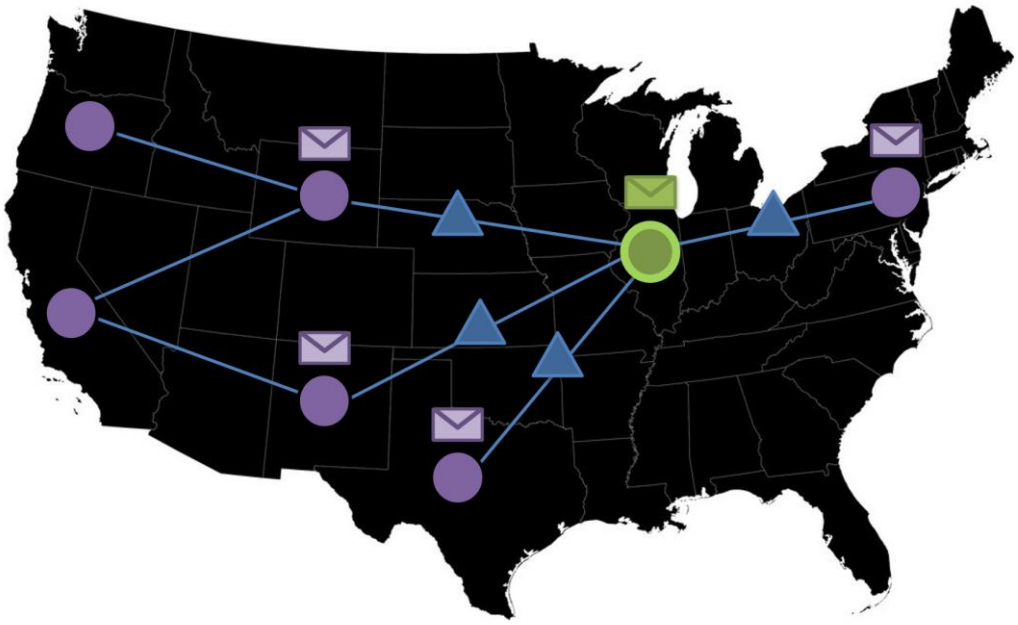

Fig 3: Message Passing Network graphical representation

For the message passing network, we filter out weak edges (hence, the graph is no longer fully connected). Information at each node can be represented in the form of a message. Messages of all such nodes are passed to its neighbors. As seen in figure 3, the green node denotes the active node which is receiving messages in that iteration. The messages of its immediate neighbors (1-hop neighbors) will pass through a MLP (denoted by a triangle) and then they would be aggregated together. We continue this process for multiple hops, which means information from entire graph is passed to every connected node.

Graph Sage Network:

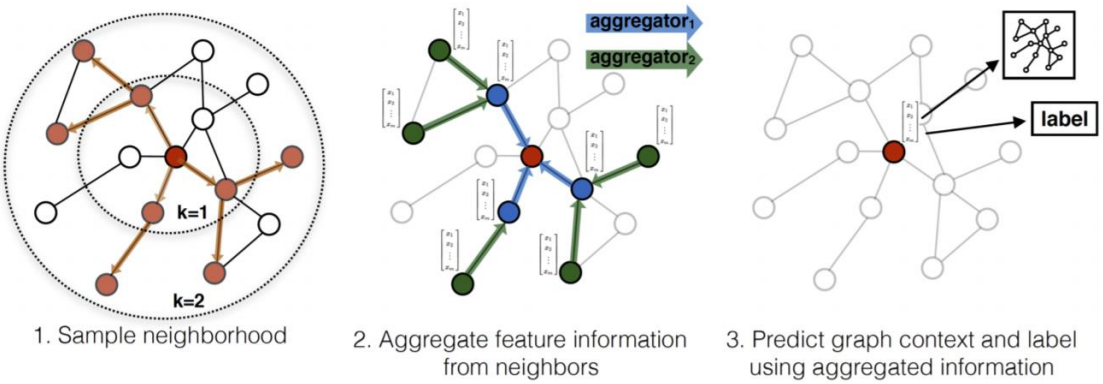

Fig 4: Sample and Aggregation Apporach in GraphSage (Image Courtesy of GraphSage Paper)

The Graph Sage network builds upon the idea of message passing network. The network learns to generate node embeddings. This is useful because now the graph can be dynamic;

i.e., we can scale it to unseen data. This makes sense in the case of Coronavirus dataset if we want to find node embeddings for some place where no case has been recorded yet.

Results

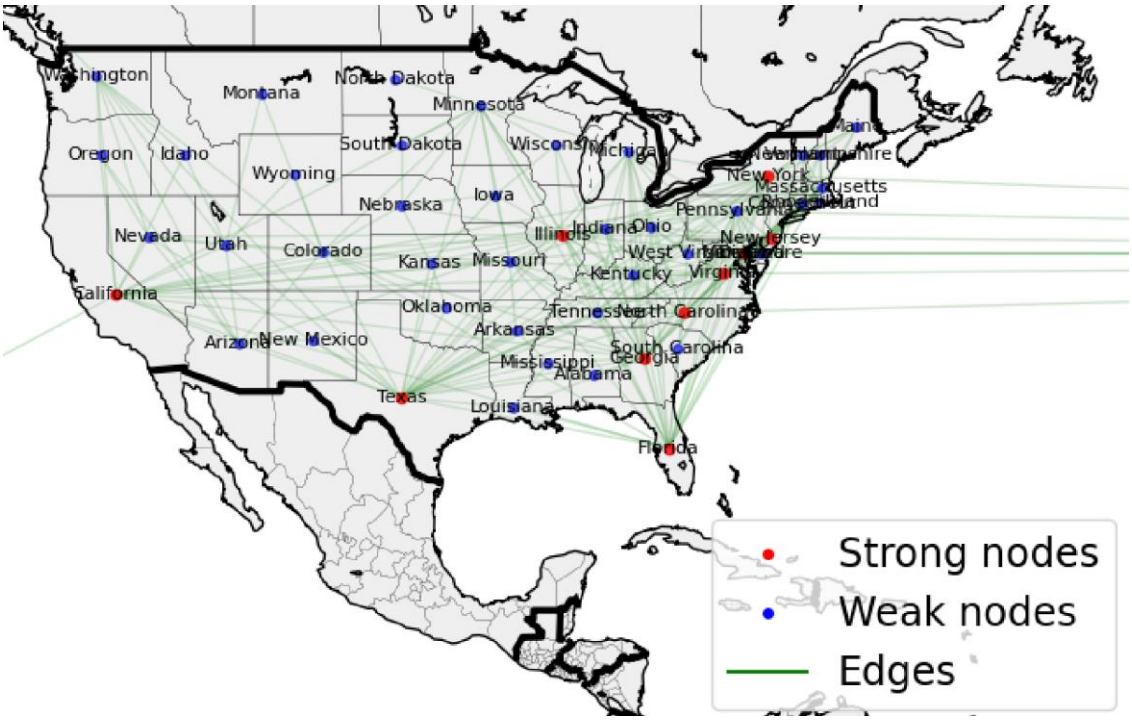

Fig 5: Visualization and identification of the strong/weak nodes using our visualizer

From figure 5, we can see that the red nodes like Illinois, New York, Florida and others have been identified as strong nodes. This makes sense as these nodes are strongly connected to other states via flights and they have neighboring states who also have strong infection rates. One interesting thing to see is that the state of Washington wasn't identified as a strong node even though it has a huge count of infected cases. But algorithmically, it makes sense that it was classified as a weak node because if we see its immediate neighbors (Idaho, Montana, Oregon etc.) we can see that these states have a low infection count. Moreover, the state of Washington has fewer flights frequency as compared to other major states.

MSE Loss: We got a Mean Squared Error of **0.0377.** The results seemed to follow actual trend (places with high number of cases were given a high value, places with low number of cases were given a low value).

Conclusion

This project delivers a pipeline for easy analysis of COVID-19 spread with the help of a data loader, graph loader, graph models and visualizer functions. The results we obtained were quite promising, especially considering the limited data we used. We believe the results can be improved significantly by using more complete data and adding more features like train travel data and number of businesses in each state. In conclusion, Graph Neural Networks are a powerful tool to analyze the spread of COVID-19.