A02 – REPORT ON ADDITIONAL SPARK LIBRARIES.

Rajbir Bhattacharjee

R00195734

Table of Contents

Exercise 5: Additional Spark Libraries

In this exercise, we discuss two different possible uses of Spark GraphX and Spark Spark MLlib.

Analysis of reachability by public transport: identifying missing links

With the threat of climate change, it is essential that cities move away from a personal automobile dominated plan to one where cities rely mostly on public transport. This also has benefits from the social justice point of view. Cars are expensive and not everyone can afford a car. If inability to afford a car affects accessibility of places of work and education, this has far reaching economic consequences for the individual as well as the society and the country at a large.

Ideally, a bus service for a city should be planned in such a way that people can confidently rely on it enough to avoid buying a car altogether. The environmental cost of a car comes not just from running it but also in producing it.

Trains and buses are, by far, the most economical modes of public transport as they can transport a large number of people at a very low cost. Trains are more economical and efficient than buses, however, running trains to places with low population density may not be the best approach. A combination of buses, trains and trams are an ideal solution to a transportation problem.

Research in developing countries have also shown that planning the last mile connectivity is essential for an effective public transport system. Buses form the ideal mode of transport for last mile connectivity.

The public transport infrastructure in Ireland is far from adequate, and while it is not quite as bad as in the USA, it leaves a lot to desire. In many places there are no bus routes, and where there are bus routes, there may not a good frequency of buses. So much so, that given two locations, it may not be practically possible to travel between the two with public transport because of the presence of missing links.

One exercise that may be done is to predict missing links in bus connectivity which prevent people from using public transport.

The assumptions here are these:

1. Commuters must be able to travel from any bus station in the city to any other bus station in the city.
2. Commuters may have to walk up-to a 500 metres to catch a connecting bus, but no more than that.
3. This must be true for the entire day 24/7

This knowledge can be used by the city planners to add or modify bus routes so that places where either 1 or 2 are invalid also come under the bus transport system. This can be done by adding new bus lines altogether, or by modifying existing bus routes to also serve some new stations along the route.

The third assumption is important here that any two places must be connected by the bus route 24/7. It is often the case that many bus routes reduce their frequency at night or completely stop plying. While the demand during those hours may be low, the unavailability of buses would mean that people cannot rely on them and are forced to buy personal vehicles, thereby incurring all the production cost of all the cars.

The problem essentially is a form of connected components. The idea is to find out subgraphs that are unconnected, and then connect them by changing the bus routes.

Both GraphX and GraphFrames facilities for finding connected components. However, we will prefer GraphFrames as it is supported on both Scala and Python, and will not tie us down to one language only. This is important as availability of talent is one important reason to prefer one one programming language over another, and python talent is easy to find as the language is very popular.

The steps to do so will be as follows:

1. Find a list of all stations by looking at the entire dataset in with regular spark RDDs. This can be found by looking at entries where atStop=1. Cache this set of stations for later use. We need to do this step separately because if we take a certain day some buses may not be running and serving some stations, and then we'll miss those stations entirely. However, it is safe to assume that if a station is active, it will have some bus serving it in the entire dataset, and even if one bus serves it we will have made a note of it.
2. For every hour interval for both weekdays and weekends do the following:
    a. Add all stations as nodes.
    b. Take all physical buses
        i. If a bus travels to two stations in the same line, then add an edge between them.
    c. Take all pairs of stations (regardless of whether there is a bus between them or not).
        i. If the haversine distance between the two stations is less than 500m, then add an edge between them. This is because we have the assumption that commuters may have to walk upto 500 metres to catch a connecting bus. The haversine distance is not accurate as people will not walk as the crow flies, but we don't have any other information in the dataset that will allow us to get the actual on-fot distance, so we will use this as a proxy.
    d. Find connected components using Graph Frames, and then print all the connected sub-graphs. Also, store all sub graph information with the nodes we had cached earlier. Now that we have all the nodes labelled with the connected sub-graph, we can move to the next phase of this algorithm.

The next value addition would be to suggest stations which may be connected easily with the least cost. Two unconnected sub-graphs may be connected by adding just one edge, although adding more edges will be beneficial.

In this context, it will make sense to add edges between the closest points. Once we have the connected components labelled, this can be done using regular Spark RDDs.

1. First a join must be performed between rows which have different sub-graphs. A join may be expensive, but it is expected that the number of stations in a city will still be limited, and the memory requirements will not explode. The next step is to calculate the haversine distance between all such pairs of stations belonging to different sub-groups.
2. The next step is to group by both sub-graph ids, and then sort in ascending order.
3. The last step would be to report the top N connections that can join two sub-graphs.

Predicting delay of buses

If delays can be reasonably predicted ahead of time, this might give the bus transport providers a chance to readjust their fleet and use their resources more effectively. This might also give the providers a chance to alert their riders of possible delays. The riders may the adjust their schedules accordingly.

Some researchers have modelled traffic as a second order chaotic system.[1] Chaotic systems may be of the first order or the second order. A first order chaotic system is one in which any prediction made about the system doesn't affect the outcome of the system itself. Predicting the weather is an example of a first order chaotic system. Any prediction about the weather will not change the weather. By contrast, a second order chaotic system is one in which any prediction about the system is likely to change the system itself. The stock, for example, market is a second order chaotic system. Predictions about the stock market influence people's behaviours in buying and selling stocks, and that in turn affects the market itself. Traffic can also be seen as a second order chaotic system. Predicting traffic accurately can have the effect of changing people's behaviour to avoid peak traffic hours and routes, which will end up reducing traffic on a whole thereby having a positive effect. Therefore, it is easy to see how predicting delays and congestion can be useful.

Congestion and delays may also be highly correlated with weather. For example, people may make a rush to reach home before a storm hits, and that may result in excess traffic and congestion before bad weather. Hence there might be value in correlating other datasets that have weather data and augmenting the Dublin bus dataset with the weather information.

While considering weather events, we will not make any attempt to address natural calamities or very severe storms that cause widespread damage or flooding. This is for two reasons:

1. These are rare events, and there may not be enough data to get any meaningful signal
2. Focusing on regular weather events will make the system more useful, as these are more frequent than rare destructive stormy evens

Met Eirann provides datasets with hourly weather measurements. These datasets include precipitation and wind-speed. The steps to deal with this would be the following

1. Load the bus-eirann dataset and enrich it
    a. For every station and route, group by hour, and reduce such that if any bus reported congestion, mark that (route, hour, station) combination as congested, and if any bus reported a delay, then mark that (route, hour, station) combination as delayed
    b. Traffic patterns on weekdays and weekends may be different. Also traffic patterns on different times of the day will be different. The nature of this is essentially

[1] Disbro, J. E., & Frame, M. (1989). *Traffic flow theory and chaotic behavior* (No. Special report 91). New York (State). Dept. of Transportation.

periodic. The periodicity can be effectively captured by taking sines or cosines. The information can be captured by adding the following columns.
        i. Take the day of the year, and convert it to a cosine
        ii. Take the month of day and convert it to a cosine
        iii. Take the day of week and convert it to a cosine
        iv. Take the minute of day and convert it to a cosine
        v. Take the hour of day and convert it to a cosine

2. Load the Met-Eirann dataset
    a. For every hour, populate the following features, the Met-Eirann dataset will surely allow us to do that
        i. Convert the wind speed to n-categorical numbers, severe, high etc.
        ii. Convert rainfall to n-categorical numbers – severe, high, etc.
3. Join the enriched bus-eirann dataset, and the Met-Eirann dataset
    a. The join will be performed on both time and geolocation
        i. The condition for joining should include weather data 2 hours before and after every bus eirann measurement. This is because weather before and after the current time affects the traffic flow. For example, people may be moving more in anticipation of bad weather.
        ii. Anothe r condition would be that the geolocation of the prediction. We should take the MetEirann data from the closest station to the location of the vehicle. This can be accomplished group and reduce after join such that:
            1. Add a new column for every row of the join that captures the distance between the met-eirann station and the bus location
            2. Reduce to take the minimum distance between the station and any bus Met station.
    b. Group by each vehicleID and timestamp, and add the following features
        i. Precipitation in the same hour as the timestamp
        ii. Precipitation in the past hour, two hours before, next hour and two hours later
        iii. Wind speed in the past hour, two hours before, next hour and two hours later
    c. Finally experiment with various classifiers from mllib.classification to see which classifier behaves the best to first form a baseline, and then tune the hyperparameters with the best classifiers.
        i. A multi-class prediction to predict both congestion and delay can be used

Spark MLLib is dataframe based and is the ideal choice as Spark ML is not under active development. The plan from the Spark community is to first enhance MLLib to achieve parity with Spark ML and then continue developing MLLib further and add no further features to Spark ML. Choosing the platform that has a clear roadmap will be advantageous in terms of maintenance and support, and also to future proof the solution.

Difficulty

This new use-case is not necessarily more difficult than the other questions attempted but has more steps in the process. Also, since two different datasets must be combined, there is some added complexity. However, all of these steps can be easily done in Spark, and are not necessarily difficult, but may be time consuming just because there are more steps involved.