Project Statement for Milestone 2

Chasing Waterfalls

Subham Behera, Ayden Armstrong, Gabe Righi

1. **Data Preparation and Data Reduction:**

   a. Describe data cleansing and data transformation steps you have performed so far. Include pseudo-code you have implemented for these steps.

Our dataset consists of 5 interrelated dataset. These datasets are: airlines.dat, airpots.dat, countries.dat, planes.dat, and routes.dat. Throughout this section I will explain how we cleaned each one.

Airlines.dat:

Airlines.dat has around 6000 entries, and each entry contains the following values: Airline ID,Name,Alias,IATA,ICAO,Callsign,Country,Active. The IATA,ICAO, and Active are optional. To clean our dataset, we needed to remove any entries that were missing fields besides the aforementioned optional ones. We converted the .dat to csv file by adding a header describing the columns and changing the file extension.

There were 15 entries that didn't have their country specified. We can't just remove airlines that don't have a country listed because the airlines are referenced in routes.dat and we wouldn't want to remove an airline that is referenced elsewhere. To fix this we either added NONE as the country of origin, or looked up the country of origin if it was available. For example Aerojet de Costa Rica was missing the country of origin, and unsurprisingly it is based out of Costa Rica.

There were 800 airlines that did not have a callsign. We elected to leave their airlines in and just added \N as their callsign, which represents null.

Airports.dat:

Airports.dat has around 7500 entries, and each entry contains the following values:

Airport ID,Name,City,Country,IATA,ICAO,Latitude,Longitude,Altitude,Timezone,DST,Tz database time zone,Type,Source. Just as before, IATA and ICAO are optional. We found that all entries had every column value except for 49 of the entries that did not have a city listed. It seems essential for an airport to have a city associated with it, so we will remove all entries that are missing a city, and will remove any entries from the following datasets that referenced the removed airports. The reason we did not manually add the city like we did the country in airlines.dat, is 50 entries is just too many to go through.

Countries.dat:

Countries.dat has around 260 entries, and each entry contains the following values:

Name,Iso_code, and Dafif_code. All entries had all values, except for one entry that was missing a dafif_code. Since the dafif_code is for historical purposes, we will be leaving the entry in. We needed to make no changes to this dataset besides converting to csv.

Planes.dat:

Countries.dat has around 260 entries, and each entry contains the following values:

Name, IATA code,ICAO code. All entries had all data in planes.dat, so no changes were necessary besides converting to csv.

Routes.dat:

Routes.dat has around 6700 entries, and each entry contains the following values:

Airline,Airline ID,Source airport,Source airport ID,Destination airport,Destination airport ID,Codeshare,Stops,Equipment. All entries had all data, except 18 of the entries were missing Equipment, which describes the plane type. This isn't that relevant for our purposes so we left in the 18 entries that were missing the equipment. We also filtered out the entries that had a source or destination airport id that was not present in airports.dat. This removed around 100 entries.

The pseudo code for all the files was basically the same:

file = "<somefile>"

columns_to_check = [columns]

Open the file with the name "{file}.dat" for reading as input_file

Open a new file with the name "{file}.csv" for writing as output_file

Write the column headers, joined by commas, to the output_file followed by a newline

For each line in the input_file:

Write the line to the output_file

Read the CSV data from "{file}.csv" into a DataFrame named dataset (using pandas)

Calculate the number of missing values for each column in dataset[columns_to_check] and store it in missing_values

If there are any missing values in missing_values:

Print "There are missing values in the following columns:"

Print missing_values

Else:

Print "All columns have complete data."

Save dataset to a CSV file named "{file}.csv" without including the index

b. It is recommended that you reduce the data to a reasonable size for faster development and testing. Described the reduced data set using basic statistics - number of files, storage size (KB/MB/GB), number of records (rows), number of attributes (columns), etc.

I choose the first 5000 routes from routes.csv at (67000 line file). I will call a new csv file with these first 5000 lines reducedRoutes.csv. I then used overlapping airport IDS and airline IDs to shrink airports.csv and airlines.csv to only contain routes mentioned in the reducedRoutes.csv. This means that

Filename	Line Count/ Rows	Size	Columns
routes.csv	67000	2.2MB	9
redicedRoutes.csv	5000	164 KB	9

Filename	Line Count/	Size	Columns

	Rows		
airports.csv	7650	963 KB	14
redicedAirports.csv	1200	158 KB	14

Filename	Line Count/ Rows	Size	Columns
airlines.csv	6100	316 KB	8
redicedAirlines.csv	90	4.3 KB	8

I left countries.csv and planes.csv the same as both files are small, around 8 KB each.

c. You may have developed a parser to transform the raw data into the format/tools you are using. Briefly describe the functions of the parser you have implemented so far.

All we have implemented in a short .ipnyb file that has the ability to take a .dat file and some headers, and convert that to a CSV file.

2. **Data Model:**

d. Describe the data model you are using to represent the dataset? Justify why the data model is an appropriate one for the dataset. Note: You should be using a non-relational data model for this project.

Since we are using Neo4j, the model we are going to be using is the property graph model. This will allow for ease of use in displaying the models in a whiteboard-friendly format, while also carrying the benefit of being able to take advantage of edges having properties. We can assign airports, airlines, planes and country entities as nodes, and routes as the edges, with attributes detailing these relationships.

e. Report the following statistics for your (reduced) dataset:

ii. If you are using a graph data set: how many nodes and edges? How many attributes are there for the nodes/edges? Is it labeled? Directed?

In our graph dataset representing the airlines data. We have 1810 nodes and 5000 edges. The node entities represent either airports, airlines, counties, or planes. Airports have 14 properties[Airport ID, Name ,City , Country, IATA , ICAO, Latitude, Longitude, Altitude, Timezone, DST, Tz database time zone, Type, Source] while the airlines have 8 properties with the following values [Airline ID, Name, Alias, IATA, ICAO, Callsign, Country, Active]. Counties have 3 properties with values [Name,Iso_code, Dafif_code], and planes also have 3 properties of [Name, IATA code, ICAO code].

Each edge in the graph represents a route and has 10 properties with the following values [Airline,Airline ID,Source airport,Source airport ID,Destination airport, Destination airport ID, Codeshare, Stops, Equipment]. It will be both labeled and directed, as the nodes can be labeled either an airport, airline, country, or plane and the direction signifies the direction of the flight route.

Nodes: Airlines, airports, countries, planes :

$$90 + 1200 + 260 + 260 = 1810\, nodes$$

Edges: Routes:

5000 edges

3. **Database:**

   a. What database are you considering for storing the reduced data? Would it scale for storing and processing the original dataset?

      We are using NEO4j to store the data. It would scale to store the original dataset because we just removed 50 out of 7500 entries from Airports.dat and 100 out of 6700 from Routes.dat. The percentage of removed entries is such a small number that it should easily handle the original dataset.

4. Source Code:

   a. Provide source code of your data preparation, data reduction and data transformation steps in a Zip file.