Project Statement for Milestone 2

Airline Search Engine

CPTS-415_BMW

Brian Joo, Noah Waxman, Mitchell Kolb

In this report, the team should focus on the dataset preparation and data model description. They should also provide appropriate statistics on the reduced and transformed data.

Report Topics:

The report should cover the following subtopics and answer the questions listed:

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.

    To cleanse our data, we are using two different python libraries, pandas and numpy. Using pandas, we can load our CSV file containing our airport data and remove duplicates and handle missing values. More specifically in the airport data, there are several columns where the value is “NaN” or “\N”, so they need to be replaced with 0.

    ```
    # Check and replace "\N" values with 0
    df.replace(r'\\N', 0, inplace=True, regex=True)
    ```

    We also remove any duplicates that may appear in the dataset.

    ```
    # Removes the duplicates
    df = df.drop_duplicates()
    ```

    Additionally, there are some columns that are necessary for data analysis. For instance, in our airport data, there are columns labeled ‘Source’ and ‘Type’. This is data that is useless to the analyses we will inevitably perform on our final dataframe, so we will remove it.

    ```
    # Remove the last 2 columns
    df = df.iloc[:, :-2]
    ```

    We check to ensure that we have the correct data types when it comes to latitude, longitude, and altitude.

```
# Convert latitude, longitude, and altitude columns to
float, float, integer
df.iloc[:, 6] = pd.to_numeric(df.iloc[:, 6],
errors='coerce')
df.iloc[:, 7] = pd.to_numeric(df.iloc[:, 7],
errors='coerce')
df.iloc[:, 8] = pd.to_numeric(df.iloc[:, 8],
errors='coerce').astype(int)
```

We standardized and cleaned all strings in the airport data.

```
# Clean and standardize string columns
def clean_and_standardize_string(s):
    if isinstance(s, str):
        return s.replace('[^\w\s]', '').title().strip()
    return s

# Apply the function to all string columns
df = df.applymap(clean_and_standardize_string)

# Randomly sample 5000 entries
sample_size = 5000
if len(df) > sample_size:
    df = df.sample(n=sample_size, random_state=42)
```

To ensure consistency in our data, we are forcing our data to be lowercase

```
df['column_name'] = df['column_name'].str.lower()
```

We are also reducing our data, so we implemented a feature to take a random sample of any number of entries from the CSV. This is beneficial for testing as well because we can run our parser an infinite number of times and obtain different samples for our data.

```
sample_size = 5000
if len(df) > sample_size:
    df = df.sample(n=sample_size, random_state=42)
```

Finally, our cleaned data will be exported into a new CSV for us to analyze.

```
df.to_csv('cleaned_data.csv', index=False)
```

Now, all of these steps apply for the airport.csv file, but they can be applied to the other datasets (airport.csv, routes.csv, planes.csv, and country.csv.) that will require cleansing and transformation with minor modifications. Eventually, each dataset will need to be combined to form one aggregate dataset, with redundant columns removed.

Specifically, there are 4 datasets which share columns with each other, such as, “Name”, “IATA Code”, and “ICAO Code”, among several others. This is data that is redundant and will only lead to less efficient analytical performance and larger file sizes when it comes to development and testing. With redundant columns removed, this will allow for us to “group” columns with related fields and reduce the final dataset file size.

For now, the rough idea is that our final dataset will consist of 4 different sections, with each section having its own set of columns: Route Information, Airport Information, Plane Information, Country Information. Each row in the dataset represents a combination of the route information, source airport information, destination airport information, plane information (if available), and country information (if available).

Route Information:

- Airline
- Source Airport
- Destination Airport
- Codeshare
- Stops
- Equipment

Airport Information:

- Airport ID
- Airport Name
- City
- Country
- IATA Code
- ICAO Code
- Latitude
- Longitude
- Altitude
- Timezone
- DST
- Timezone (TZ database format)

Plane Information:

- Plane Name
- IATA Code for Plane
- ICAO Code for Plane

Country Information:

- Country Name
- ISO Code
- DAFIF Code

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

   The total size of the unaltered data is 1.1MB, and only one file. There are a total of 14,000 entries and 14 columns. We can accomplish all necessary analytical goals with fewer rows, so we will be reducing our dataset to ~5,000 rows. Seeing as two of the columns are not useful to our data analysis, we will only be using 13 of the given columns in our final dataframe. With these reductions being applied to all 4 datasets, they should reduce our final dataset to ~2.5MB.

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.

   Our Python parser is capable of a couple functions. Currently, it can read our CSV data and convert it into a dataframe, a 2-dimensional structure capable of containing columns of different types. Our parser is also capable of cleaning the data by removing duplicates, handling missing values, and correcting any formatting inconsistencies. We have also implemented a data transformation step where we reduced the size of the original CSV data via random sampling and removed unnecessary columns. Finally, our parser is capable of exporting the final dataset into a CSV format.

2. Data Model:

   a. 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.

The data model that we have decided on to represent our openflights dataset is the graph model. Graphs are useful for modeling complex relationships between entities, which is the case in this dataset. The graph would have nodes representing airports, airlines, and cities, and edges representing routes, codeshares, and other relationships. What made the graph model appealing in the first place is that our dataset is fully composed of named relationships with specific properties which is what graph models are exceptionally good at. Our other choice that was in the running was a document based data model but after analyzing our data closer we decided that our dataset wasn't formatted in such a way that the key-value collection that the document data model can effectively take advantage of. Although our data does have lots of index-centric values which made the document model appealing in the first place.

When using the graph model we think it is best that each node would have properties such as the airport's ID, name, city, country, IATA code, ICAO code, latitude, longitude, altitude, timezone, and type. For airlines, properties would include the airline's ID, name, alias, IATA code, ICAO code, callsign, country, and active status. For cities, properties would include the city's name and country. Edges would represent routes, codeshares, and other relationships. For routes, properties would include the airline's ID, the source airport's ID, the destination airport's ID, and the number of stops. For codeshares, properties would include the airline's ID, the source airport's ID, the destination airport's ID, and whether the flight is a codeshare.

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

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?

For storing the reduced data, a graph database like Neo4j that we got introduced with in HW2 would be a good choice. Neo4j is a highly scalable, high-performance, native graph database that can handle large amounts of data and complex queries. It would be able to handle the original dataset and exponentially more data because it is designed to scale.

In Neo4j, the nodes and edges could be represented in this way for our projects' dataset:

- Nodes: Airport: This node would represent an airport. Properties would include the airport's ID, name, city, country, IATA code, ICAO code, latitude, longitude, altitude, timezone, and type. Airline: This node would represent an airline, properties could include the airline's ID, name, alias, IATA code, ICAO code, callsign, country, and active status. City: This node would represent a city. Properties could include the city's name and country.

- Edges: The route edge would represent a route. Properties would include the airline's ID, the source airport's ID, the destination airport's ID, and the number of stops. The codeshare edge would represent a codeshare. Properties would include the airline's ID, the source airport's ID, the destination airport's ID, and whether the flight is a codeshare.

The database would also include indices on the properties that are queried most frequently, to speed up the queries. This would be useful in the event where our openFlight dataset is x10 or x100 the size. For example, an index could be created on the airline's ID, the source airport's ID, and the destination airport's ID for the Route edge, and on the airline's ID, the source airport's ID, and the destination airport's ID for the Codeshare edge.

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

        The team has decided to go with neo4j. One of the biggest functions in our assignment is to show connections or relationships, such as what airports travelers can take if they want to go to multiple places. One of the biggest strengths of neo4j is that it makes displaying connected data extremely easy. We are also working with multiple csv files, where each one holds a respectable amount of data. Neo4j makes it easy to upload data. The program also allows the team to even put parameters on what data we want to upload.

        We also believe these factors will allow us to give us more complex yet human readable data than information from a different database such as MongoDB.

Source Code:

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

Peer Evaluation:

All team members should complete the CATME Peer Evaluation survey.

Grading:

- 15 pts: Team has successfully performed data preparation, data reduction and data transformation steps. Team has provided accurate description and statistics of the reduced dataset.
- 5 pts: Project milestone document provides all information with relevant diagrams, pseudo-code and sample dataset.