

Université Claude Bernard Lyon 1



Flight Data Analysis Using Spark and GraphFrames

Prepared By

**Danish Touseeq
Welfi**

**Data Processing and Analytics (DPA) Course - International
Master DISS UCBL**

24/25

Introduction

Understanding flight patterns and airport connectivity is crucial for improving transportation efficiency and passenger experience in today's interconnected world. The aviation industry relies heavily on data analytics to optimize operations and manage air traffic. This project focuses on analyzing flight interconnected data to gain insights into the popularity and connectivity of airports. Using Spark's powerful APIs and the GraphFrames library, we explore various aspects of this data, including degrees of connectivity, triangle counts, and centrality measures. By implementing the PageRank algorithm and identifying the most connected and important airports, we aim to uncover significant patterns and trends. This analysis helps in understanding current flight networks and provides a foundation for enhancing future flight operations.

Development Environment Setup:

- **Environment:** Explain that Docker was used for a consistent development environment, along with Spark and GraphFrames.
- **Installation Steps:** Describe the steps:
 - Running the script to install GraphFrames.
 - Starting the environment using `docker-compose up`.

Data Preparation:

- **Dataset:** Briefly describe the dataset from Kaggle, including its content and structure.
- **Data Loading:** Outline the CSV file loading into Spark DataFrames.
- **Graph Construction:** Detail how you created a graph in GraphFrames, defining nodes as airports and edges as flights.

In this project, we analyze flight data from a CSV file containing flight information from 2018. The dataset, sourced from a comprehensive collection spanning multiple years, is used to construct and analyze a flight network with a focus on understanding airport connectivity and flight patterns. Our primary objectives include:

1. **Data Loading and Cleaning:** We begin by loading the 2018 subset of flight data into a Spark DataFrame, followed by cleaning operations to handle missing values, ensuring the dataset is suitable for analysis.
2. **Graph Construction:** Utilizing the GraphFrames library, we build a directed graph where airports serve as vertices and flights between them as edges. This graph representation allows us to analyze the network structure and relationships between different airports.
3. **Statistics Computation:** For each airport, we compute the in-degree, out-degree, and total degree. These metrics help quantify the level of connectivity for each airport, providing insights into which airports are the most connected within the network.
4. **Triangle Counting:** We count the number of triangles (closed triplets) in the graph. Triangles are a fundamental measure of network clustering, indicating groups of airports that have direct flights among each other.

5. **Centrality Calculation:** To identify influential airports, we compute a centrality measure, such as Closeness Centrality, using custom implementations without relying on GraphFrames' built-in functions. This measure helps pinpoint airports that are centrally located within the network, facilitating efficient connectivity.
6. **PageRank Implementation:** We implement the PageRank algorithm natively in Spark, without using GraphFrames' built-in methods, to rank airports based on their importance within the network. PageRank assigns a score to each airport, reflecting its influence based on the number of incoming flights and the importance of the airports those flights originate from.
7. **Identifying Key Airports:** Using the computed metrics, we identify the most connected airports based on degree measures and determine the most important airport using PageRank and centrality scores.
8. **Graph Visualization:** To enhance the understanding of the network structure, we represent the graph visually using HTML. This visualization provides a clear depiction of the relationships and connections between airports, making it easier to observe patterns and clusters.
9. **Comparison of Metrics:** We compare the results derived from degree, centrality, and PageRank to analyze the different perspectives each metric offers on airport importance and connectivity. This comparison helps validate the findings and provides a comprehensive view of the flight network.

Through this analysis, we aim to uncover critical insights into the structure of the flight network, identify key airports that facilitate travel efficiency, and provide a foundation for future improvements in airline operations and network planning.

Detailed Analysis

Data Loading and Cleaning

In the **Data Loading and Cleaning** phase, the primary goal is to prepare the flight data for analysis by loading it into a suitable data structure and addressing any inconsistencies or missing values. Here's a detailed breakdown of the steps involved in this process:

1. Loading the Dataset

The dataset, a CSV file containing flight information for 2018, is loaded into a Spark DataFrame. This format is particularly well-suited for handling large datasets efficiently, leveraging Spark's distributed computing capabilities.

Output Verification: The `flights_df.show(5)` command is used to display the first five rows of the DataFrame, providing a quick glimpse of the data structure and content.

FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEELS_OFF	WHEELS_ON	TAXI_IN	CRS_ARR_TIME	ARR_TIME	ARR_DELAY	CANCELLATION_CODE
2018-01-01	UA	2429	ENR	DEN	1517	1512.0	-5.0	15.0	1527.0	1712.0	10.0	1745	1722.0	-23.0	
2018-01-01	UA	2427	LAS	SFO	1115	1107.0	-8.0	11.0	1118.0	1223.0	7.0	1254	1230.0	-24.0	
2018-01-01	UA	2426	SNA	DEN	1335	1330.0	-5.0	15.0	1345.0	1631.0	5.0	1649	1636.0	-13.0	
2018-01-01	UA	2425	RSW	ORD	1546	1552.0	6.0	19.0	1611.0	1748.0	6.0	1756	1754.0	-2.0	
2018-01-01	UA	2424	ORD	ALB	630	650.0	20.0	13.0	703.0	926.0	10.0	922	936.0	14.0	

CANCELLATION_CODE	DIVERTED	CRS_ELAPSED_TIME	ACTUAL_ELAPSED_TIME	AIR_TIME	DISTANCE	CARRIER_DELAY	WEATHER_DELAY	NAS_DELAY	SECURITY_DELAY	LATE_AIRCRAFT_DELAY	UNRELIABLE
null	0.0	268.0	250.0	225.0	1605.0	null	null	null	null	null	null
null	0.0	99.0	83.0	65.0	414.0	null	null	null	null	null	null
null	0.0	134.0	126.0	106.0	846.0	null	null	null	null	null	null
null	0.0	190.0	182.0	157.0	1120.0	null	null	null	null	null	null
null	0.0	112.0	106.0	83.0	723.0	null	null	null	null	null	null

2. Inspecting the Data Schema

To understand the structure and types of data within the DataFrame, the schema is printed.

3. Handling Missing Values

Missing data can lead to inaccurate analysis, so it's crucial to handle any missing values effectively.

- **na.drop**: This method removes rows that contain null values.
- **subset=["ORIGIN", "DEST"]**: Specifies that rows should be dropped only if the ORIGIN or DEST (destination) columns have missing values. These columns are critical for constructing the flight graph, as they represent the nodes (airports) and edges (flights) in the graph.

4. Ensuring Data Quality

After dropping rows with missing values, it's essential to verify that the remaining dataset is clean and consistent.

- **Verification Steps:**
 - **Row Count Check**: Compare the number of rows before and after cleaning to understand the extent of data reduction due to missing values.
 - **Summary Statistics**: Use descriptive statistics to check for any anomalies or outliers in key columns.

5. Defining Vertices and Edges

In this step, we construct the core components of the graph: vertices and edges. Vertices represent airports, and edges represent flights between these airports. Here's a detailed breakdown of how these are defined:

Creating the Vertices DataFrame

Vertices in the graph correspond to unique airports in the dataset. We derive the vertices DataFrame by selecting distinct airports from both the ORIGIN and DEST columns of the flight data.

Output: The `airports_df.show(5)` command displays the first five rows of the vertices DataFrame, allowing for verification of the distinct airports' identify.

```
+---+---+-----+
|src|dst|flight_count|
+---+---+-----+
|ANC|DEN|         459|
|BNA|IAH|        1864|
|TPA|SFO|         381|
|ORD|PDX|        2762|
|ANC|OTZ|         671|
+---+---+-----+
only showing top 5 rows
```

Building the Graph

After defining the vertices and edges, the next step is to construct the graph using the GraphFrames library. This graph will serve as the foundation for conducting various analyses on airport connectivity and flight patterns.

Calculating In-Degree, Out-Degree, and Total Degree

To analyze the connectivity of airports within the flight network, I calculated the in-degree, out-degree, and total degrees for each airport. These metrics provide a deeper understanding of how each airport functions within the network and help identify key hubs.

- **In-Degree:** This measures the number of incoming flights to each airport, highlighting its role as a major destination. Airports with a high in-degree are likely critical for receiving traffic from various locations.
- **Out-Degree:** This counts the number of outgoing flights from each airport, indicating its importance as a departure point. Airports with a high out-degree are significant for distributing passengers and flights to various destinations.
- **Total Degree:** The sum of in-degree and out-degree provides a complete picture of an airport's connectivity. Airports with a high total degree are central to the network, managing a significant flow of both incoming and outgoing flights.

Calculating these metrics helps in identifying the most connected airports, which are essential for understanding traffic flow and making informed decisions about route optimization and resource allocation. here is the output:

```
+---+-----+-----+-----+
| id|in_degree|out_degree|total_degree|
+---+-----+-----+-----+
|BGM|      1|      1|      2|
|BNA|     57|     57|    114|
|CGI|      2|      2|      4|
|CLL|      3|      2|      5|
|CLT|    136|    136|    272|
+---+-----+-----+-----+
only showing top 5 rows
```

Counting the Number of Triangles

I counted the number of triangles in the flight network to find groups of three airports that are all directly connected. A triangle represents a strong connection between the airports.

- **Method:** I used a self-join on the flight data to find triplets of airports that form a triangle.
- **Adjustment:** The total count was divided by three to avoid counting the same triangle multiple times.
- **Result:** There are 47,532 triangles in the network.

This shows areas where airports are closely linked, helping to understand dense traffic regions and connectivity.

(Output: Total triangles: 47,532)

Computing Triangle Counts with GraphFrames

To validate the manual triangle counting process, I also computed the number of triangles using the GraphFrames library, which provides built-in functionality for this purpose.

- **Method:** I utilized the `triangleCount()` function from GraphFrames to automatically identify and count triangles in the graph.

- **Result:** The total number of triangles was calculated and displayed.

```
+-----+---+
|count| id|
+-----+---+
|    0|BGM|
| 1070|BNA|
|    1|CGI|
|    3|CLL|
| 1904|CLT|
+-----+---+
only showing top 5 rows

Total number of triangles: 72360
```

Computing Centrality

Due to the high computational cost of calculating closeness centrality, I used degree centrality as an approximation. This was based on the total degree of each airport, representing its overall connectivity.

- **Method:** Degree centrality was calculated by using the total degree directly as a centrality measure.
- **Reason:** Other methods like shortest paths were too resource-intensive and time-consuming, making this approach more practical.

```
+---+-----+-----+-----+-----+
| id|inDegree|outDegree|total_degree|centrality|
+---+-----+-----+-----+-----+
|BGM|    1|    1|    2|    2|
|BNA|   57|   57|  114|  114|
|CGI|    2|    2|    4|    4|
|CLL|    3|    2|    5|    5|
|CLT|  136|  136|  272|  272|
+---+-----+-----+-----+-----+
only showing top 5 rows
```

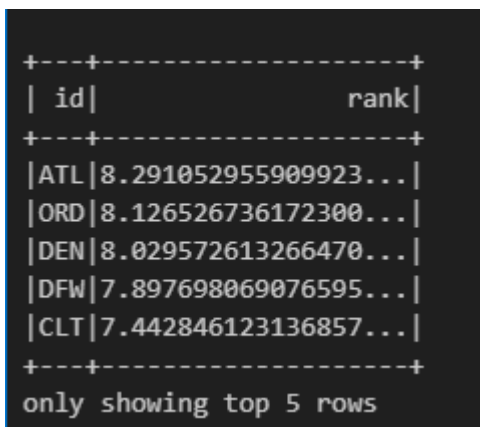
Implementing PageRank

I implemented the PageRank algorithm to find the most influential airports in the network.

- **Initialization:** Each airport started with a rank of 1.0.
- **Iterations:** The ranks were updated over 10 iterations using a damping factor of 0.85, which balances rank distribution between connected airports.
- **Rank Calculation:** Contributions from connected airports were summed, and ranks were updated to reflect an airport's importance in the network.
- **Results:** The top airports were identified by ordering the final ranks in descending order.

This method helps pinpoint the airports that are central to the network, playing a key role in connectivity and passenger flow.

Output: Top 5 airports based on PageRank

A terminal window with a dark background and light green text. It displays a table with two columns: 'id' and 'rank'. The table lists the top 5 airports: ATL, ORD, DEN, DFW, and CLT, along with their respective PageRank values. The text 'only showing top 5 rows' is at the bottom.

id	rank
ATL	8.291052955909923...
ORD	8.126526736172300...
DEN	8.029572613266470...
DFW	7.897698069076595...
CLT	7.442846123136857...

only showing top 5 rows

Analyzing the Most Connected Airports

To identify the most connected airports, I ranked them based on their total degree, which combines both incoming and outgoing flights. Airports with the highest total degree are considered the most connected.

- **Approach:** Total degree was used to quickly determine connectivity, as more complex methods, like finding paths to all nodes, were too time-consuming.
- **Result:** The top 5 most connected airports were identified and displayed.

This analysis highlights the airports that serve as major hubs in the network, crucial for managing flight traffic efficiently.

Output: Top 5 most connected airports:

```
Most Important Airport are the most connected ones (more total degree) :  
  
+---+-----+-----+-----+  
| id|in_degree|out_degree|total_degree|  
+---+-----+-----+-----+  
|ORD|      175|      174|      349|  
|DFW|      169|      170|      339|  
|ATL|      167|      166|      333|  
|DEN|      164|      162|      326|  
|CLT|      136|      136|      272|  
+---+-----+-----+-----+  
only showing top 5 rows
```

Finding the Most Important Airport

The most important airport was determined by identifying the one with the highest PageRank score.

- **Method:** The airport with the top rank was selected from the ordered list.
- **Result:** The most important airport, based on its influence in the network, was displayed.

This airport plays a key role in overall connectivity and traffic flow.

Output: Most important airport:

```
+---+-----+  
| id|          rank|  
+---+-----+  
|ATL|8.291052955909923...|  
+---+-----+  
only showing top 1 row
```

Graph Visualization

To visually represent the airport network, I used NetworkX and Plotly to create an interactive graph, showcasing the relationships between airports based on flight connections.

- **Graph Construction:** A directed graph was built using NetworkX, where airports are nodes and flights are edges. Each node was assigned a PageRank score as an attribute.
- **Visualization:** The graph was visualized using Plotly, with nodes sized and colored based on their PageRank scores. Larger and darker nodes indicate higher importance in the network.
- **Output:** The graph was displayed interactively, allowing for exploration of the network's structure. Additionally, it was saved as an HTML file for easy sharing and review.

This visualization helps in understanding the connectivity and relative importance of airports in the network, providing insights at a glance.

Output: Interactive graph displayed and saved as "airport_network_graph.html"

