# FLIGHT DATA SEARCH ENGINE AND ANALYSIS

Abhishek Ayachit, Gowtham Tumati, Sudip Bala, Lovepreet Singh Dhaliwal
Department of Computer Science, University of California - Riverside
aayac001@ucr.edu,gtuma002@ucr.edu,sbala027@ucr.edu,ldhal001@ucr.edu

## ABSTRACT

In this project, the aim is to build a historic flight data repository which includes collecting data and merging it so as to make a huge(big-data) dataset. The project includes four major tasks or modules. These are Data Cleaning, Data Integration, Search Engine and Data Analysis and Visualisation. First the data that has been collected from multiple sources need to be cleaned and the attributes are normalized. A new sub-module of Airline Ranking is added to the project. This is a part of Data Cleaning and Integration. In this part, the different airlines are ranked on the basis of delays they have made over the years. This is used by the search engine in order to recommend the flights. The search engine takes the details about the flight and searches the big data in order to retrieve the results faster. Two databases, AsterixDB and MongoDB are used for this purpose. The last module analyses the clean data and visualizes it in order to get clear insights on the data.

## 1. INTRODUCTION

Flights are one of the main options that a person considers while going to places. For this to happen in a smooth manner, there must be proper search algorithms that can effectively deliver results that relate the travel origin and destinations, and match them to all the available airline carriers. Taking all the results, a final suggestion must reflect in the search query that can help an individual to take a quick decision on which flight to choose.

In this project, we aim at creating a historic flight data repository, that has over 10 years of information gathered from various sources. Extending this repository is an application that was done with the data. We ranked the airlines in an order, taking certain parameters into consideration. We made use of the document database MongoDB and even created a kind-of search engine. Using this, we can predict the best flight when a travel plan is made.

## 2. LITERATURE SURVEY

### 2.1 Data Cleaning and Integration

The paper written by Nan Tang described different methods to clean the data. The methods provided included, Constraints based RDF cleaning, Master data, Interactive Cleaning. The Big RDF Data cleaning is used to clean the big data, which uses an existing database engines or the distributed systems. The main point of the paper is that it describes the different methods for cleaning the data and proposes the one that is mentioned above[1]. To integrate the data from multiple data sources were unified to create an uneven schema which can be converted into a global schema for all the data sources. To compile the data from multiple sources, big data integration techniques are discussed which can cope up with the huge volume of data coming at a great velocity and from a variety of sources. The paper also addressed the need of the Big Data Integration techniques over traditional record linkage techniques. The paper also addressed the handling of the incomplete and redundant data as the flight safety anomalies and

erroneous data needs to be handled carefully. Another aspect of cleaning the flight data is to reduce the data generated from all the sensors. The main drawback about the paper is that it failed to mention the actual technical details about the techniques that can be used on the big data[2].

### 2.2 Search Techniques for Big Data

Unlike the traditional databases, the amount of time required to access the data in the big data is very high. Vatsal Jatakia et. al provides different techniques that can be used in order to query in the big data. Among all this, considering the advantages and disadvantages of all the searching techniques, the last one, i.e., Genetic Algorithm or GA shows the best results as it is scalable, flexible and it is robust and to the noisy evaluation functions. Another important point is that any part of the algorithm can be changed and customized[3].

The paper [5] provides a different method for indexing to create a search engine on the big data. The indexes were created along with the Page Rank which can be used on category based search engine. A new way of Key-hash indexing is used to create hash-key folders by using total unique search term in the search engine. The advantage of this method is that when there are concurrent requests, the detch time of the result does not change. The method also works well with distributed architecture. The paper also evaluates the proposed method by evaluating the method.

### 2.3 Big Data Analytics and Data mining

After successfully integrating and cleaning the data, to get inferences from the big data, different data mining techniques need to be used. Different techniques can be used for different use cases. One such use case is mentioned in the [8]. The use case is to identify the patterns of the passengers in the aviation industry. To achieve this, the paper mentions Bayesian network as it is flexible with noise free as well as incomplete datasets. Bayesian Maps are used to in supplying the missing information which uses Bayes theorem in order to find the certainty factors.

Another method is experimented and evaluated in [6]. It uses C5.0, CART, CHAID to build a decision tree, dividing datasets into subsets and using chi-square to identify the optimal split respectively. It divides the original dataset into sets, training and testing datasets. To evaluate the model, accuracy, recall and specificity is calculated.

### 2.4 Big Data Visualization

Data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. There are different ways of visualizing different kinds of data. A few of these type are discussed and compared with respect to visualizing the big data in [7]. The techniques used to visualize the data includes following: Line Chart, Bar Chart, Pie Chart, Table, Scatter Plot, Bubble Chart, Parallel Coordinates, Tree Map, etc. While comparing these when it comes to visualizing the big data, all the techniques are compared on the basis of Volume, Velocity and Variety. According to [7], all the techniques mentioned above except Line Chart and bar Chart are suitable for big volume of data. For high variety of data, all the techniques except Pie Chart and Tree Map are suitable. And finally, for the high velocity of the data, all the techniques except Tree Map are suitable.

## 3. DATA PREPROCESSING

Data has been collected from two major sources, transtats.bts.gov[9] and packages.revolutionanalytics.com[10], that made some complications from the initial stage in the project. There were a total of 44 attributes that were available in the data. Things were not the same, when data from another source was compared to this. To solve this, we can integrate the data into a single piece. Before this can be done, the cleaning process has to be done. The procedure would include streamlining the attributes, followed by attribute selection, creation and deletion. Later, missing values must be handled either by handling them or dropping them. The penultimate part would be formatting of the whole data, and then comes the integration. This is the sequence in which the tasks are categorized and implemented.

To begin the process, all the attribute formats were analyzed and studied. We have sorted them out, minimising the effort in streamlining the data. After this, there was a very wide range of data types. A list of all the missing data in the whole set was sorted and analyzed. Attributes with the least percentage of missing fields were shortlisted and taken into consideration for the cleaning process. The remaining attributes are not considered anymore in the procedure, so we put them aside. Now, in the selected attributes that missing values are present, the values must be filled with any function or with zeros. At this point, all the data is properly shortlisted. With this, the data cleaning process and integration is completed, that also makes the data repository part. The next phase would be ranking of the airlines.

The following things have been used for data preprocessing:

**SparkSQL**

Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrames and can also act as a distributed SQL query engine. It enables unmodified Hadoop Hive queries to run up to 100x faster on existing deployments and data. SparkSQL supports all the popular relational operators. SparkSQL can be intermixed with RDD operations. It uses the Dataframe API as an enhancement to RDD API. Dataframes are basically SparkSQL counterpart to relations or tables in RDBMS. Dataframes consists of various rows and columns. A dataframe is not in 1NF. It can be created from various data sources like csv file, json file, mySQL database, hive, etc. Lazy execution occurs in sparkSQL using dataframes . Also using dataframes, Spark is aware of the data model and query logic.

**PySpark**

PySpark enables scalable analysis and ML pipelines. Since we are already familiar with Python and Pandas, then much of our knowledge can be applied to Spark. The key type used in PySpark is Spark Dataframe. Since we wanted to distributed computation using PySpark, so we had to perform operations on Spark dataframes and not on any other python data types. It is also possible to use Pandas dataframes when using Spark, by calling toPandas() on a Spark dataframe, which returns a pandas object. However, this function should generally be avoided except when working with small dataframes, because it pulls the entire object into memory on a single node. One of the key differences between Pandas and Spark dataframes is eager versus lazy execution. In PySpark, operations are delayed until a result is actually needed in the pipeline. For example, you can specify operations for

loading a data set from S3 and applying a number of transformations to the dataframe, but these operations won't immediately be applied. Instead, a graph of transformations is recorded, and once the data is actually needed, for example when writing the results back to S3, then the transformations are applied as a single pipeline operation. This approach is used to avoid pulling the full data frame into memory and enables more effective processing across a cluster of machines. With Pandas dataframes, everything is pulled into memory, and every Pandas operation is immediately applied. In general, it's a best practice to avoid eager operations in Spark if possible, since it limits how much of your pipeline can be effectively distributed. One of the first steps to learn when working with Spark is loading a data set into a dataframe. Once data has been loaded into a dataframe, you can apply transformations, perform analysis and modeling, create visualizations, and persist the results. In Python, you can load files directly from the local file system using Pandas. In PySpark, loading a CSV file is a little more complicated. In a distributed environment, there is no local storage and therefore a distributed file system such as HDFS, Databricks file store (DBFS), or S3 needs to be used to specify the path of the file. Many databases provide an unload to S3 function, and it's also possible to use the AWS console to move files from your local machine to S3.

**PyArrow**

Apache Arrow is an ideal in-memory transport layer for data that is being read or written with Parquet files. The Apache Parquet project provides a standardized open-source columnar storage format for use in data analysis systems. It was created originally for use in Apache Hadoop with systems like Apache Drill, Apache Hive, Apache Impala (incubating), and Apache Spark adopting it as a shared standard for high performance data IO. We have been concurrently developing the C++ implementation of Apache Parquet, which includes a native, multithreaded C++ adapter to and from in-memory Arrow data. PyArrow includes Python bindings to this code, which thus enables reading and writing Parquet files with pandas as well. Apache Arrow is a cross-language development platform for in-memory data. It specifies a standardized language-independent columnar memory format for flat and hierarchical data, organized for efficient analytic operations on modern hardware. It also provides computational libraries and zero-copy streaming messaging and interprocess communication. Languages currently supported include C, C++, C#, Go, Java, JavaScript, MATLAB, Python, R, Ruby, and Rust. Apache Arrow enables execution engines to take advantage of the latest SIMD (Single instruction, multiple data) operations included in modern processors, for native vectorized optimization of analytical data processing. Columnar layout is optimized for data locality for better performance on modern hardware like CPUs and GPUs. The Arrow memory format supports zero-copy reads for lightning-fast data access without serialization overhead. Arrow acts as a new high-performance interface between various systems. It is also focused on supporting a wide variety of industry-standard programming languages. C, C++, C#, Go, Java, JavaScript, MATLAB, Python, R, Ruby, and Rust implementations are in progress and more languages are welcome.

**3.1 Airline Ranking**

In the selected attributes, there are some specific attributes that can help in one of our applications with the data repository. Taking two of them into consideration, we rank the airlines and suggest them to the user. We consider the departure delays of the airlines to rank them. First we divide the delays in 3 parts: on time($t < 5$ min), small delay ($5 < t < 45$ min), and large delay( $t > 45$ min). We will ignore flights on

time and also departure delays of flight less than 5 minutes since it is of no use to consider them in our ranking. Delays less than 5 minutes can be covered during the flight journey. Now let us consider the small departure delays and the large departure delays and the number of flights handled by the carrier. We took the mean delays of the carriers using the small delays and large delays. The contributions of these attributes helped in ranking the carriers.

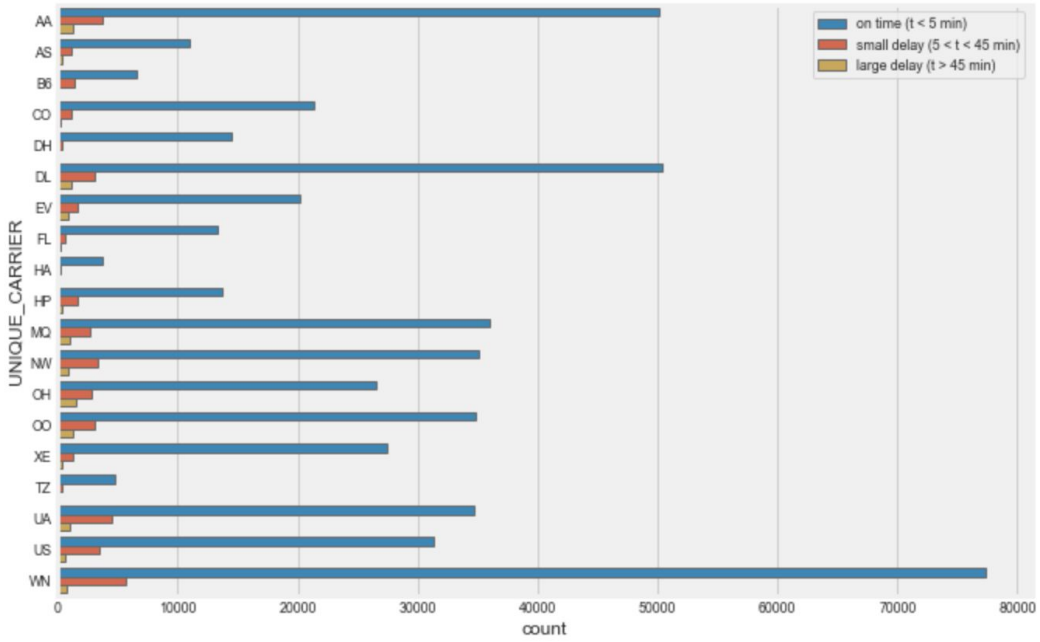


Fig 1: Delay

| UNIQUE_CARRIER | |
|---|---|
| B6 | 1.0 |
| WN | 2.0 |
| TZ | 3.0 |
| US | 4.0 |
| HP | 5.0 |
| UA | 6.0 |
| CO | 7.0 |
| DH | 8.0 |
| NW | 9.0 |
| XE | 10.0 |
| MQ | 11.0 |
| AS | 12.0 |
| AA | 13.0 |
| DL | 14.0 |
| HA | 15.0 |
| OO | 16.0 |
| EV | 17.0 |
| OH | 18.0 |
| FL | 19.0 |

Fig 2: Airline Ranking

## 4. Search Engine

A search engine is an information retrieval system that helps find information stored on any source based on the requirement and functionality. In the earlier application, we ranked the available airlines based on certain parameters. The implementation that we intend to work requires human participation. With the help of the airline ranking, we have extended the application to a kind-of search engine, by using MongoDB and AsterixDB. In there, the source and destination are entered, and based on the user inputs and the ranking, an airline is suggested to the user.

MongoDB is a JSON document datastore. The main advantage with it lies in the fact that objects can be stored with nested data in one collection. Also, it works well with the kind of data structure that is going to evolve over time, as its schema-less operations allows the developer to update data on the fly. It has the ability to represent hierarchical relationships, which is, store arrays and other more complex data structures easily.

Searching is also made easy in it, since supports the operation by fields, range queries, and regular expressions. There is also indexing, to improve the performance of the searches. The speed of the search is also controlled, since the database uses the concept of sharding to scale horizontally by splitting data across multiple instances. The load is balanced across the system to keep it up in case of a failure.

In a similar way, another database that has been used is AsterixDB. It is a highly parallel and highly scalable database system. It has its own semi-structured data model, as well as a custom query interface that uses more of SQL's functionality and new features as well. Data is partitioned based on hashing and controls a custom query execution engine. It has a share of its features with MongoDB which also makes use of JSON for data modelling.

**AsterixDB:**

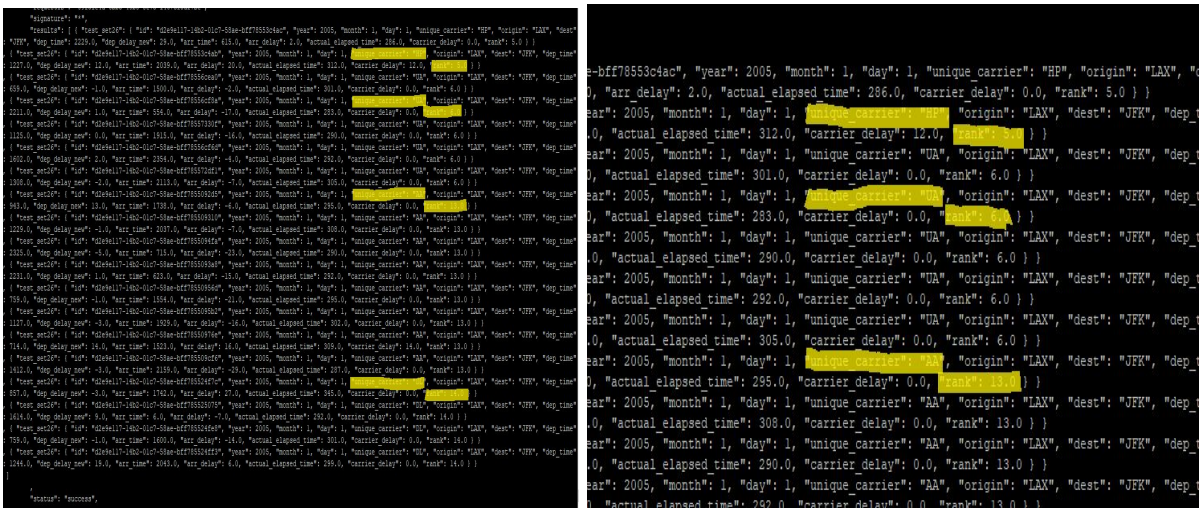

Fig 3: Query Processing using AsterixDB

**MongoDB:**

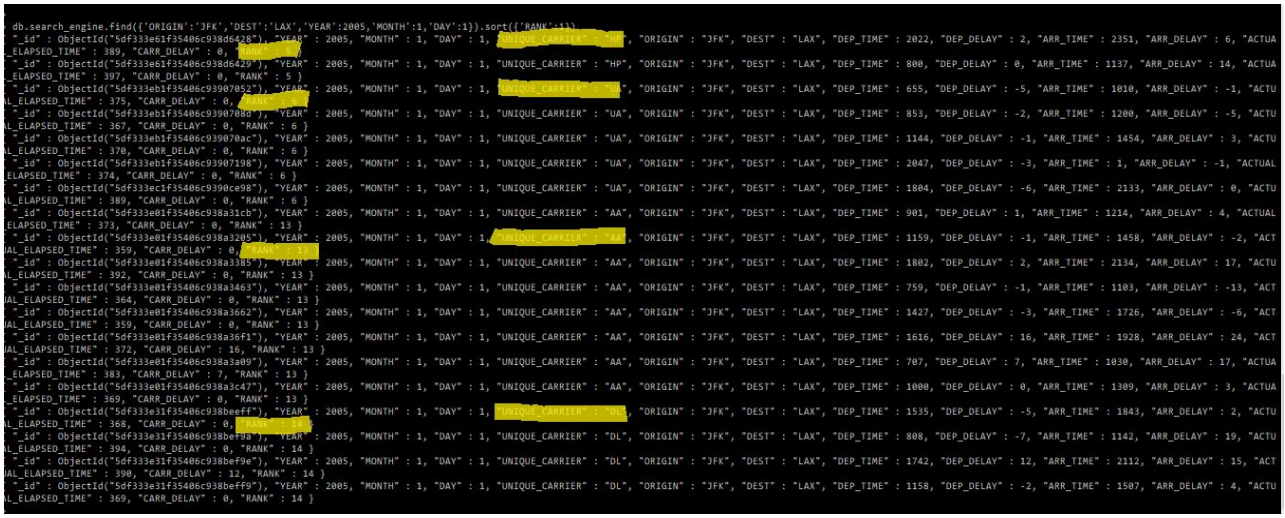


Fig 4: Query Processing using MongoDB

## 5. DATA ANALYSIS AND VISUALISATION

As a first step, we consider all the flights from all unique carriers. Here, the aim is to classify the airlines with respect to their given parameters mostly on the basis of their punctuality and for that purpose, we compute a few basic statistical parameters. Different types of visualisation are done by us which includes various pie charts and histograms.

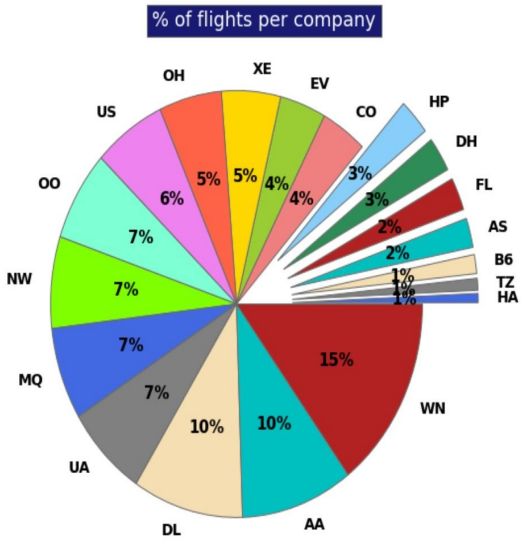


Fig 5: Percentage Flights per Company

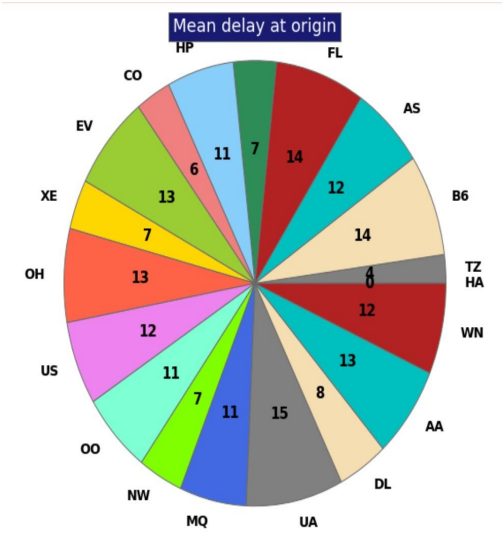


Fig 6: Mean Delay at Origin Airport

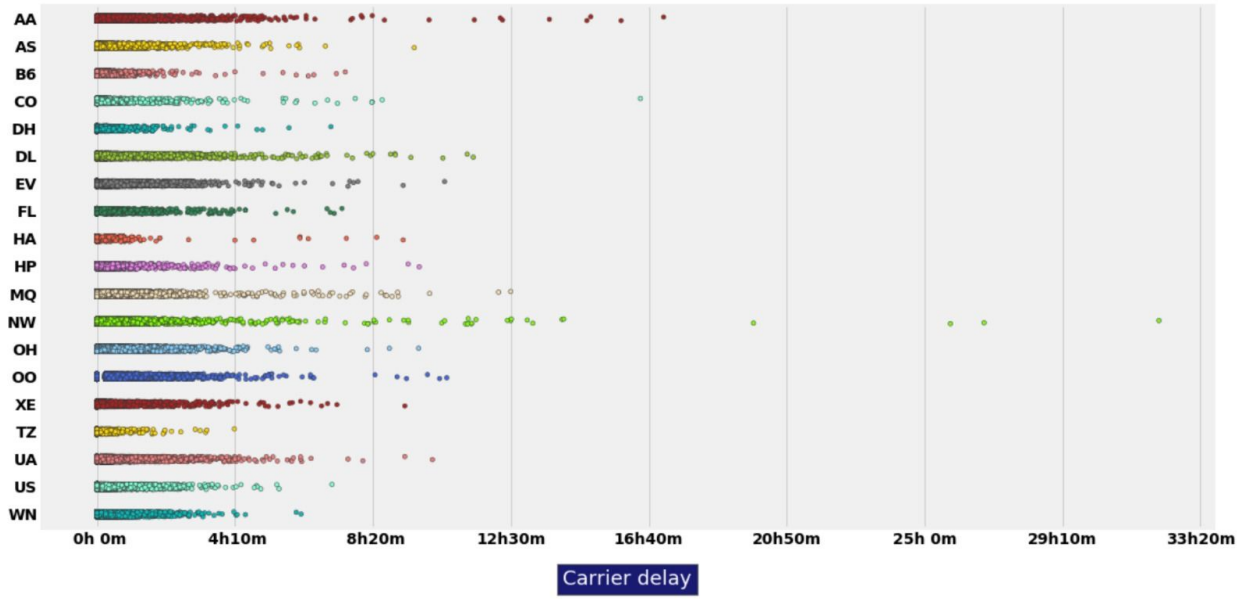


Fig 7: Carrier Delay

Considering the first pie chart that gives the percentage of flights per airline, we see that there is some disparity between the carriers. For example, Southwest Airlines accounts for ~20% of the flights which is similar to the number of flights chartered by the 7 tiniest airlines. However, if we have a look at the second pie chart, we see that here, on the contrary, the differences among airlines are less pronounced. Excluding Hawaiian Airlines and Alaska Airlines that report extremely low mean delays, we obtain that a value of ~11±7 minutes would correctly represent all mean delays. Note that this value is quite low which means that the standard for every airline is to respect the schedule! Finally, the figure at the bottom makes a census of all the delays that were measured in January 2015. This representation gives a feeling on dispersion of data and put in perspective the relative homogeneity that appeared in the second pie chart. Indeed, we see that while all mean delays are around 10 minutes, this low value is a consequence of the fact that a majority of flights take off on time. However, we see that occasionally, we can face really large delays that can reach a few tens of hours!

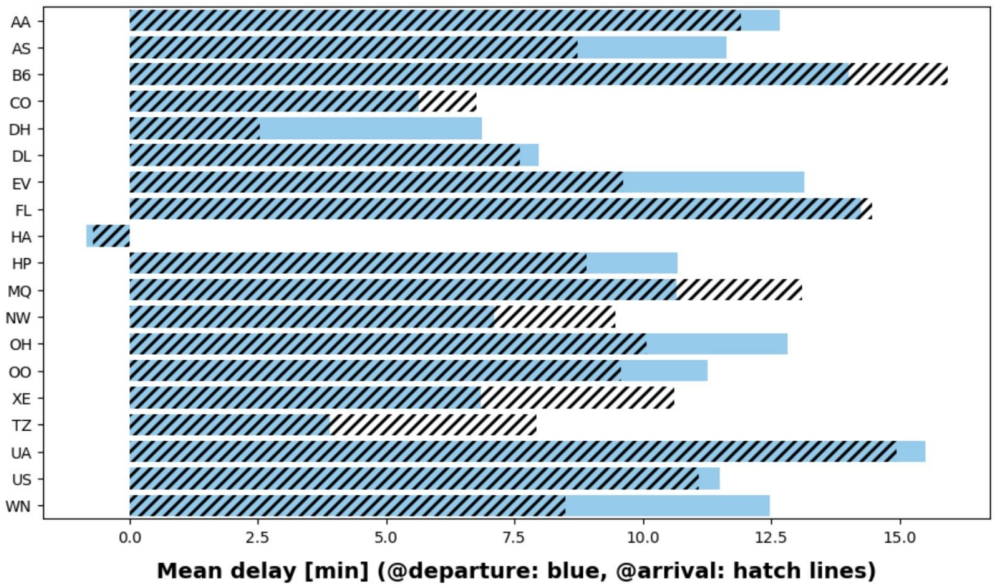


Fig 8: Mean Delay

On this figure, we can see that delays at arrival are generally lower than at departure. This indicates that airlines adjust their flight speed in order to reduce the delays at arrival. In what follows, we will just consider the delays at departure but one has to keep in mind that this can differ from arrival delays.

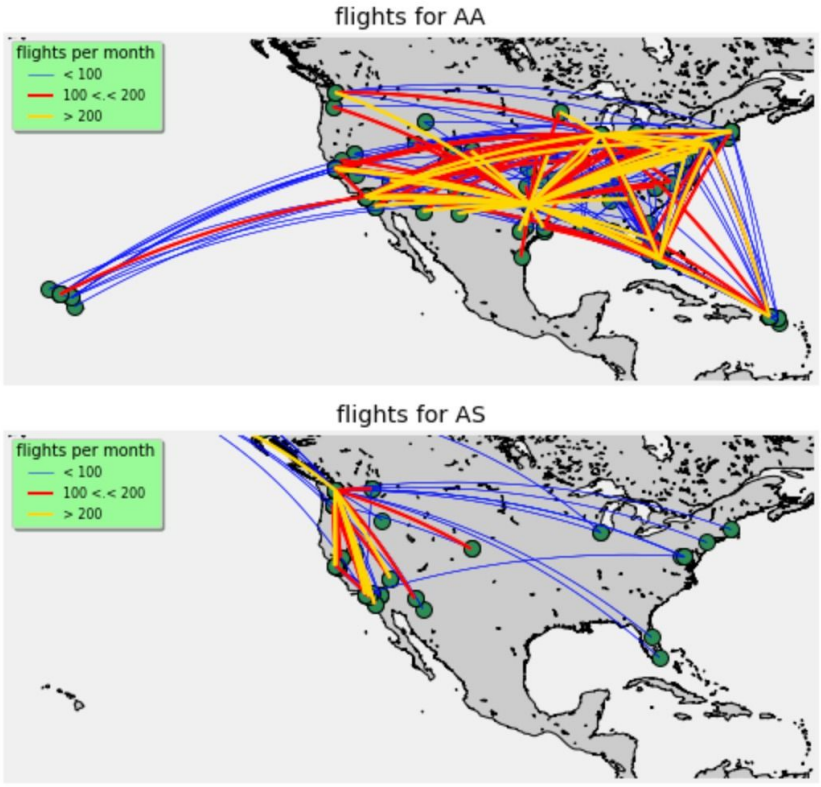


Fig 9: Flights w.r.t. Airport

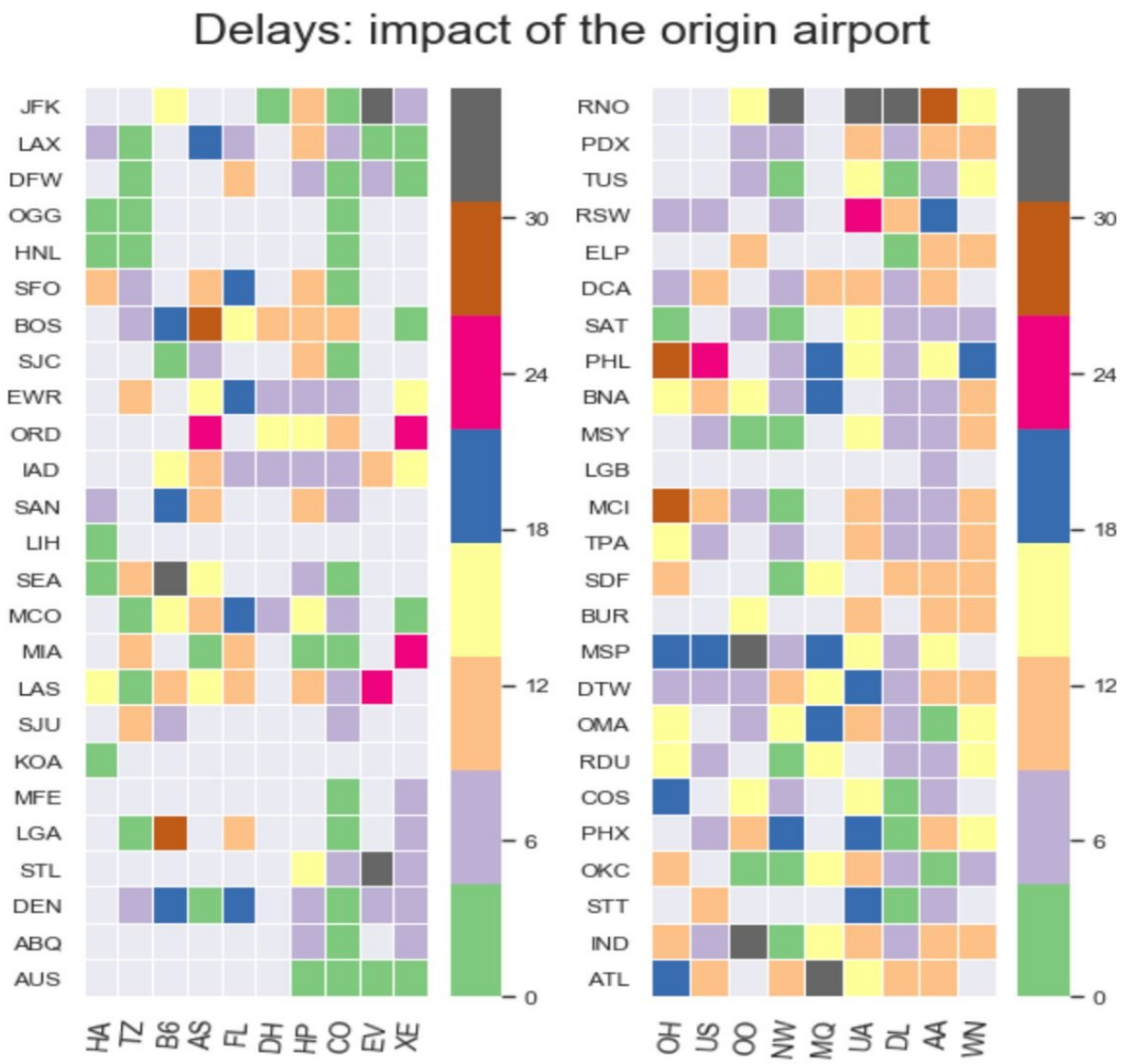


Fig 10: Delays due to Airport

This figure allows to draw some conclusions. First, by looking at the data associated with the different airlines, we find the behavior we previously observed: for example, if we consider the right panel, it will be seen that the column associated with American Eagle Airlines mostly reports large delays, while the column associated with Delta Airlines is mainly associated with delays of less than 5 minutes. If we now look at the airports of origin, we will see that some airports favor late departures: see e.g. Denver, Chicago or New York. Conversely, other airports will mainly know on time departures such as Portland or Oakland.

Finally, we can deduce from these observations that there is a high variability in average delays, both between the different airports but also between the different airlines. This is important because it

implies that in order to accurately model the delays, it will be necessary to adopt a model that is specific to the company and the home airport.

| DATE | UNIQUE_CARRIER | ORIGIN | DEST | DEP_TIME | DEP_DELAY | ARR_TIME | ARR_DELAY | ACTUAL_ELAPSED_TIME | CARR_DELAY | RANK |
|---|---|---|---|---|---|---|---|---|---|---|
| 2005-01-01 | AA | JFK | LAX | 901 | 1.0 | 1214 | 4.0 | 373.0 | 0.0 | 13.0 |
| 2005-01-02 | AA | JFK | LAX | 855 | -5.0 | 1236 | 26.0 | 401.0 | 0.0 | 13.0 |
| 2005-01-03 | AA | JFK | LAX | 901 | 1.0 | 1238 | 28.0 | 397.0 | 0.0 | 13.0 |
| 2005-01-04 | AA | JFK | LAX | 856 | -4.0 | 1306 | 56.0 | 430.0 | 0.0 | 13.0 |
| 2005-01-05 | AA | JFK | LAX | 857 | -3.0 | 1154 | -16.0 | 357.0 | 0.0 | 13.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2005-01-31 | WN | TUS | LAX | 1235 | 0.0 | 1307 | 2.0 | 92.0 | 0.0 | 2.0 |
| 2005-01-31 | WN | TUS | LAX | 805 | 0.0 | 830 | -5.0 | 85.0 | 0.0 | 2.0 |
| 2005-01-31 | WN | TUS | SAN | 1925 | 0.0 | 1930 | -5.0 | 65.0 | 0.0 | 2.0 |
| 2005-01-31 | WN | TUS | SAN | 1135 | 0.0 | 1155 | 5.0 | 80.0 | 0.0 | 2.0 |
| 2005-01-31 | WN | TUS | SAN | 825 | 0.0 | 827 | -13.0 | 62.0 | 0.0 | 2.0 |

568031 rows × 10 columns

Fig 11: Airline Ranking

## 6. FUTURE WORK

In the initial attempt of gathering and streamlining data and making the most of it, there's a miniature version of a search engine and suggestion preferences to the user. These features could be extended fully with the use of real-time data, using the full functionality of the document databases that are currently in use. With collaborations from sources that work on the related topic and implementation, this project has the potential to be converted into a fully scaled search engine.

In addition to this, an algorithm can be integrated into the system that helps in cleaning and sorting the data, so that more focus can be put on implementing the search engine in a full scale. There is a way that the traditional search engines can be replaced and that is by the use of a search aggregator which uses the data that we have converted as a repository. The reason for considering this type of an approach to be even better is due to the ranking and gathering features that are a part of the engine, by default.

## REFERENCES

[1] N. Tang, "Big RDF data cleaning," *2015 31st IEEE International Conference on Data Engineering Workshops*, Seoul, 2015, pp. 77-79.

[2] Burmester G., Ma H., Steinmetz D., Hartmannn S. (2018) Big Data and Data Analytics in Aviation. In: Durak U., Becker J., Hartmann S., Voros N. (eds) Advances in Aeronautical Informatics. Springer, Cham

[3] V. Jatakia, S. Korlahalli and K. Deulkar, "A survey of different search techniques for big data," *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Coimbatore, 2017, pp. 1-4.
[4]Pooja Anand , Dr. Sandeep Maan, “A Study on Big Data with Indexing Technique for Searching and Retrieval of Data Fastly”, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 6, Issue 1, January 2018
[5] N. Ragavan, "Efficient key hash indexing scheme with page rank for category based search engine big data," *2017 IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing (INCOS)*, Srivilliputhur, 2017, pp. 1-6.
[6] A. Lukáčová, F. Babič and J. Paralič, "Building the prediction model from the aviation incident data," *2014 IEEE 12th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herl'any, 2014, pp. 365-369.
[7] Ajibade, Samuel & Adediran, Anthonia. (2016). An Overview of Big Data Visualization Techniques in Data Mining. 4. 105-113.
[8] Rajendra Akerkar , “ANALYTICS ON BIG AVIATION DATA: TURNING DATA INTO INSIGHTS”, International Journal of Computer Science and Applications, Vol. 11, No. 3, pp. 116 – 127, 2014
[9] transtats.bts.gov
[10] packages.revolutionanalytics.com
[11]https://www.kaggle.com/usdot/flight-delays/download/sa7WwXvZrss8jtOlpLiO%2Fversions%2FnbLmIcXVboR8JvDVPDVj%2Ffiles%2Fairports.csv?datasetVersionNumber=1