Assignment 5: Routing

Ankita, Ushang, Xia

10/19/2017

Objective

Given historical airplane on time performance data, our task is to offer suggestions for two-hop flights that minimize the chance of missing a connection.

Dataset

We have used the BTS's Airline On-time Performance Data that contains scheduled and actual departure and arrival times reported by certified U.S. air carriers that account for at least one percent of domestic scheduled passenger revenues.It contains monthly data from 1989-2015.

Preparing Data

- Remove all the corrupt data by performing the given sanity test on each row of dataset.
- While doing the check, we also see if the flight is cancelled, then we set the delay time to 720 mins i.e 12 hours. This guarantees that the connecting flight is missed.
- Remove the data for all years that is greater than input prediction year. For example, if we are predicting for year 2012, we only consider data from 1988 to 2011 for predictions.

Model

- Custom model - Maximum Likelihood Estimation
- "In statistics, maximum likelihood estimation (MLE) is a method of estimating the parameters of a statistical model given observations, by finding the parameter values that maximize the likelihood of making the observations given the parameters." –Wikipedia
- In our case, we are given observations from the previous years, and the parameter values here is the delay minutes for each flight. If the training data is big enough – which in this case it is – then the mean will be closed to the real value. For a single flight, its delay minutes will always be less or equal to our calculated mean (less because we take cancellation as 12 hours delay).
- Reasons for using this model: this approach is straightforward and simple, which allows us to put more effort in the mapreduce part.
- A more robust model will be having mean and variance for each flight, and then use 2-sigma principle to estimate the probability of missing next flight. However, our current model works well in the validation, so we decide not to go further as for now.

Predictive Task

In order to create a predictor for predicting whether a flight will be delayed, we are building a model with the features Month, Carrier, Source, Destination, Arrival-time, Departure-time and Delay-time. We will also include the average departure delay for the carrier for each flight as another feature. So our model, based on all these features will then predict the likelihood of delay for all the possible 2 hop flights between the input source and destination.

Approach 1:

- Initially we tried to predict all the flights for a given source and destination irrespective of what month we are looking for. By doing so we approximately got 10,000+ predictions for each of the given route. The problem with the approach was we were mapping each single hop flight to every other single hop flight irrespective of whether it operated in that particular month or not. Hence our results had a lot of false positives and were not that accurate.

Approach 2:

- In order to achieve higher accuracy and to make the results more relevant, we are now finding the flight predictions only for the month mentioned in the input tuple. Along with that we also take the flight timings into considerations so that we can see the trends for the morning , evening and late night flights.By doing this we remove the false positives thus getting more relevant results. One of the measure of this relevance is the decrease in the number of predicted flight to ~3000 flights.

Implementation

Our solution implements 2 Map-reduce jobs that finds all the single hops flight between the given source and destination and also predict the flights for the given year.

Stage 1: Get all the single hop flights with mean delays

Map:

- The map function in the first stage reads the airline input dataset and the year for which we want to predict the flights and prepares the training data set by removing all the data entires greater than or equal to the given year.
- It then does the sanity test on this data set and along with that, for all the cancelled flights, we make the delay to 720 minutes i.e 12 hours, hence ensuring that while predicting the flights, this one misses the connection.
- The output of the mapper is a Key-Value pair where the key is of type Source_airport, Destination_airport, Departure_time, Arrival_time,month, airline and delay timings.

Partition:

- In this phase, the partition class, partitions the key from the mapper into twelve different reducers(1 for each month) as we are calculating the mean delays for each month and each flight time.

Reduce:

- The reduce function gets as key and list of delays for that month and a given flight as value.
- It adds the mean delays for each flight for each month and writes it to HDFS.

Stage 2: Joins and Predictions

Map:

- The input to this mapper function is the output of phase-1.
- Then map function first checks, if the flight is required by doing the following tests:
  1. Consider the flight only if the month matches with the prediction input month.
  2. Reject the flight if its a direct flight.

3. Consider only those files who's either source or destination matches with our input source and destination.

- Once the flights clear the above tests, we make two hashmaps, one containing the destination of the first hop as the key and the other containing the origin of the second hop.
- After these hashmaps are formed, i.e all map functions are executed, we join the maps based on the keys. All the similar keys are mapped and the layover time condition between the two flight is checked i.e Connections must have at least 45 minutes between landing and takeoff, and no more than 12 hours. Here we also add the delay of the first flight to this calculation in order to check if it misses the conneciton.
- If the second hop flight reaches the final destination on time, we score it a positive 1 and if there is a delay, we give it a penalty of -100. While giving this score, if the average delay of this flight is between 0-1, we ignore this delay.
- The output of this map is a key-value where the key is the itinerary and the value is the score.

Reduce:

- The reduce function the reducer aggregrates all the score and outputs only those flights for which the score is >0 i.e the flights are on time.

Performance and accuracy of the solution

- Accuracy: We had a seperate Mapreduce job, called FlightValidation, to validate our prediction. The basic idea is to find the "real" flights in that day and check their dealy.

-FlightValidation workflow: - We wrote a list of two-hop flights that we want to validate into configuration. In the mapper phrase, it goes through all the records in big corpus (otherwise wouldn't work because partial corpus might not contain the information in that specific day), filtering out everything except the flight information in the two-hop flights list and happend in specifically the same year, month, date as input. Once we had those information, in clean-up, we check if the layover time between first and second flight is within the range of 45 mins and 12 hours, and if the second flight is delay. If any one of the above condition fails, we wrote "-100", otherwise "1".

-Validation result: For the given input: 2001,2,1,BOS,LAX, our model yields a few thousand possible-two-hop flights, so we randomly picked 5 of them:

- ((2001,2,1,1015,1329,US,BOS,MCO),(2001,2,1,1849,2137,UA,MCO,LAX))
- ((2001,2,1,1015,1331,B6,BOS,MCO),(2001,2,1,1849,2137,UA,MCO,LAX))
- ((2001,2,1,1100,1427,DL,BOS,MCO),(2001,2,1,1750,2028,UA,MCO,LAX))
- ((2001,2,1,1140,1457,DL,BOS,MCO),(2001,2,1,1745,2028,UA,MCO,LAX))
- ((2001,2,1,1200,1502,DL,BOS,ATL),(2001,2,1,1758,2004,DL,ATL,LAX))
- And the validation reuslt is:
- 2001,2,1,1015,1329,US,BOS,MCO@2001,2,1,1849,2137,UA,MCO,LAX 1
- 2001,2,1,1015,1331,B6,BOS,MCO@2001,2,1,1849,2137,UA,MCO,LAX 1
- 2001,2,1,1100,1427,DL,BOS,MCO@2001,2,1,1750,2028,UA,MCO,LAX 1
- 2001,2,1,1140,1457,DL,BOS,MCO@2001,2,1,1745,2028,UA,MCO,LAX 1
- 2001,2,1,1200,1502,DL,BOS,ATL@2001,2,1,1758,2004,DL,ATL,LAX 1

Ideally we shoud check all predictions, however due to the fact that this job is incomplete, meaning we have to hard-code the flight-list that we want to test, it's nearly impossible to test all. The randomly picked result gives limited insight.

Input Format

The input to the program is as below. It takes Year,Month,Date,Source,Destination .

```
2001,2,1,BOS,LAX
2001,9,11,DEN,DCA
2001,3,1,DEN,LAX
2001,12,12,MCO,LAS
```

Output Format

The predictions are of the following format. For each tuple we get an output file of the format [(First-hop detials),(Second-hop details)]

```
[(2001,2,1,1030,1323,DL,BOS,ATL),(2001,2,1,1855,2107,DL,ATL,LAX)]   1
[(2001,2,1,1040,1357,DL,BOS,MCO),(2001,2,1,1745,2028,UA,MCO,LAX)]   1
[(2001,2,1,1040,1357,DL,BOS,MCO),(2001,2,1,1825,2100,UA,MCO,LAX)]   1
[(2001,7,1,930,1216,B6,JFK,AUS),(2001,7,1,2200,2245,WN,AUS,DAL)]    1
[(2001,7,1,1720,2020,AA,JFK,AUS),(2001,7,1,2200,2245,WN,AUS,DAL)]   1
```

AWS Execution Environment Specifications

- Master Node
  - m4.xlarge
  - 8 vCPU
  - 16 GiB memory
  - EBS only storage
  - EBS Storage:100 GiB
- Data Nodes(2 or 4)
  - m4.2xlarge
  - 16 vCPU
  - 32 GiB memory
  - EBS only storage
  - EBS Storage:100 GiB

Local Execution Environment Specifications:

- Macintosh 2.5Ghz i7 Quad Core
- 16 GB RAM
- macOS Sierra Version 10.12.6
- Java 8 Update 144

Work Distribution:

- Logic and program flow - Ankita, Ushang, Xia
- Report - Ankita, Ushang, Xia
- MakeFile and readMe - Ushang
- SingleHopFlights.java - Ankita
- FlightPrediction.java - Ushang
- FlightValidation.java - Xia