# Review Report: a5-ankita-ushang-xia

*Tirthraj, Shreysa, Pranav*

## Contents

**Code Repository:** https://github.ccs.neu.edu/pdpmr-f17/a5-ankita-ushang-xia

The commit id reviewed is b9b9ae11bae35cd99fbb01f1ae434061ec52646d which was checked before deadline.

## Code execution

We did the following to run the code: *Cloned the repository* Changed the hadoop path *Following the readme file: make setup gives error, placed input directory in the folder but it still gives error. Read the Makefile and make setup was not required* Then executed make which runs the program. The phase1output folder has some entries but the final output folder had reducer outputs which are always empty. (we used 323.csv as input and 2001, 1,1, BOS, LAX as query )

## Review

*The README.md file has some details about the K neighborhood program which can be omitted.

- No need to use only try block in main function of `Driver` class. You are already throwing `IOException`. Also, since it doesn't make sense to continue execution if input file is not read, throwing exception is a better choice.
- Also, you shall close the reader once you are done.

```
try(BufferedReader br = new BufferedReader(new InputStreamReader(in))) {
  ...
}
```

- No need to merge array elements as done for `input` since you already have line which is of the same format.

```
String[] array = line.split(",");
...
String input = array[0]+","+array[1]+","+array[2]+","+array[3]+","+array[4];
```

- `FlightPredictionReducer` is inside `FlightPredictionMapper`. Code below shows how it is used because of that. It should have been inside `FlightPrediction` as reducer is not part of mapper.
- Also, making job input non splittable degrades the performance.

```
j.setReducerClass(FlightPrediction.FlightPredictionMapper.FlightPredictionReducer.class);
j.setInputFormatClass(NonSplitableTextInputFormat.class);
```

- In `UtilityHelper`, instead of using parseLine, using CSVRecord to parse line could have been more efficient. CSVRecord creates only one string and when you call its `get` method with column index (starting with 0), it will return a substring. In contrast, former approach will create String for all columns and you are not needed to use all columns.

```
String[] inVal = csvParser.parseLine(value.toString());

// instead of above line try using following
CSVRecord r = new CSVRecord(value.toString());
// then read column with its index as follows
String val = r.get(0);
```

- Functions in `UtilityHelper` can be private.
- In following code from `isRecordValidAndRequired` function of `UtilityHelper`, instead of catching `Exception`, `NumberFormatException` shall be catch. You should catch only anticipated exceptions. Catching with `Exception` will not flag any undesirable behavior. This kind of use can be seen in various methods of `UtilityHelper`.
- Furthermore, it could have been better if you check for integer in a separate function, and then check for year value. That will better to modularize your code.

```
try{
  if(Integer.parseInt(record[0])>=Integer.parseInt(year)
```

```java
     && Integer.parseInt(record[0]) < 1989)
    return false;
}
catch(Exception e){
  return false;
}
```

- See below suggestion.

```java
if(checkIfNonZero(record) && checkIfNotEmpty(record) && timezoneCheck(record))
            return true;
return false;

// better way
return (checkIfNonZero(record) && checkIfNotEmpty(record) && timezoneCheck(record));
```

- Incorrect calculation of timezone. Perform 24hour format time difference. Use `Date` class provided by Java.

```java
// timezone = CRS_ARR_TIME - CRS_DEP_TIME - CRS_ELAPSED_TIME
int timeZone = Integer.parseInt(record[40])
                 - Integer.parseInt(record[29]) - Integer.parseInt(record[50]);
```

- Following line in `map` function of `SingleHopFlightsMapper` does not tell what fields are extracted. Function comments also do not specify the same. This basically make key of `HashMap<String, DelayWritable> flights` unclear.

```java
String outKey = inVal[14]+"_"+inVal[23]+"_"+inVal[29]+"_"+inVal[40]+"_"+inVal[2]+"_"+inVal[8];
```

- You are writing in `cleanup` method of `SingleHopFlightsMapper` class. What if your mapper fails? You can't make your mapper fault tolerant if you write in cleanup method.
- Mapper and reducer comments are not clear enough to explain what exactly is being done. Along with writing what it does, you also shall write about its input and output.
- For `predictionJob`, `j.setInputFormatClass(NonSplitableTextInputFormat.class);` making job input non splittable degrades the performance.
- Since you want false for same boolean values and true for different boolean values, you can use XOR as shown below. This will make your code compact and more readable.

```java
public static boolean isRouteRequired(String[] record, String[] input){

  if(!record[4].equals(input[1]))
    return false;

  if(record[0].equals(input[3]) && record[1].equals(input[4]))
    return false;

  if(record[0].equals(input[3]) || record[1].equals(input[4]))
    return true;

  return false;
}

// After simplification above function can be written as follows.
public static boolean isRouteRequired(String[] record, String[] input) {
  return record[4].equals(input[1]) &&
          (record[0].equals(input[3]) ^ record[1].equals(input[4]));
}
```

- Please auto-format code to avoid long lines as seen at line 203-206 in `UtilityHelper`. It isn't shown here for better readability.
- Invalid calculation of layover in `UtilityHelper.checkWithinTimeLimits` as shown below. Use `Date` to take time difference of 24hours time formats.
- Also, you are checking with actual arrival time instead of taking predicted arrival time. The job was to predict delay of first flight so that you can predict its arrival time. Then with predicted arrival time and departure time of next flight, calculate layover.

```
double layoverTime = dept - arr - delay;
```

- Hashmap condition is not explained properly-what is first hop and what is second hop till now,the 1st mapper only contained flights aggregated with the mean delays. Some commenting does not give any detail and can be removed
- Value of mapper 1 is not mentioned,only key is mentioned.
- Report has a few typos for eg: dealy instead of delay, phrase instead of phase, reuslt instead of result and so on.
- Overall, because of the points mentioned above, the results are not correct as per the requirement. The code could have been clearer and compact. NonSplittable input could have been ignored and that would in turn enforce to avoid using static HashMap. Current solution is degrading performance by taking more time and memory.
- The report contains enough amount of implementation details and relevant experiment information. However, it fails to conclude the experiment and provide execution time.