Large Scale Data Management - Project 1

Athens University of Economics and Business
M.Sc. in Data Science
Theofanis Nitsos- p3352325

For the purposes of this project the virtual machine and vagrant file were not used as there were some issues due to the M1 processor causing calculations to take a very long time. Instead, the docker was used directly installed on the Mac (outside the Virtual machine) along with Hadoop.

Part 1

1. We choose the book "Shipwrecks on Cape Cod" from Gutenberg (https://www.gutenberg.org/cache/epub/72960/pg72960.txt)
2. In the file Driver.java we change the name of the input txt to Shipwrecks.txt to match the name of the book we downloaded.
3. After executing the commands as indicated in the programming project briefing, adjusting for different file names and paths, the execution logs are saved as word_count_logs.txt.

Part 2

For the second part, the song with the maximum danceability as well as the average danceability of songs for each country, year-month combination has to be calculated. First the data needs to be filtered based on country & year-month combination and the song name & danceability score have to be retained for the average and max calculations.

1. The mapper takes as input a **key value** pair, where the **key** represents the line number of the csv file provided and the **value** is a text containing the comma separated values.
2. The first row of the csv is neglected as it is the header.
3. Then the rows are parsed using the provided regular expressions to separate the values for each column.
4. The year-month data along with the country are concatenated to become the **key** for the output of the map function and the input to the reducer function.
5. The song name along with the danceability score are stored as the **value** in a composite type of DancableWritable class and are used as the output of the map function and input of the reducer.
6. The DancableWritable is defined in a separate .java file provided with the code.
7. The same **key** is used as the output of the reducer function; the country-year-month combination.
8. The concatenated string of the song name, the max and the average danceability for the specific **key** are used as the **value** output of the reducer function.

Part 2 – with combiner

This part is implemented using java code. Similar to above:

1. The mapper takes as input a **key value** pair, where the **key** represents the line number of the csv file provided and the **value** is a text containing the comma separated values.
2. The first row of the csv is neglected as it is the header.
3. Then the rows are parsed using the provided regular expressions to separate the values for each column.
4. The year-month data along with the country are concatenated to become the **key** for the output of the map function and the input to the combiner function.
5. The song name along with the danceability score are stored as the **value** in a composite type of DancableWritable class and are used as the output of the map function and input of the combiner. (The same object is used for the output of both the mapper and the combiner).
6. The combiner function is executed locally at each mapper decreasing the network traffic required to move information around. In the combiner function the max danceability along with the name of the corresponding song, the sum of danceability and the count of songs are stored as the output **value** of the combiner and the input to the reducer.
7. The same **key** is used as the output of the combiner function; the country-year-month combination.
8. The concatenated string of the song name, the max and the average danceability for the specific **key** are used as the **value** output of the reducer function.
9. The DancableWritable is defined in a separate .java file provided with the code.
10. The same **key** is used as the output of the reducer function; the country-year-month combination.
11. The concatenated string of the song name, the max and the average danceability for the specific **key** are used as the **value** output of the reducer function.