Documentación de aplicaciones y Manual de Usuario

Preprocesamiento:

Esta aplicación tiene la función de limpiar el dataset original. Primero recibe un dataset en formato txt, luego abre un diccionario, también un txt. Posteriormente recorre el dataset línea por línea, palabra por palabra, para comparar cada palabra. Si dicha palabra esta en el dataset, entonces la borra y reemplaza por un espacio en blanco. El tiempo de ejecución puede durar entre 7-9 minutos dependiendo del tamaño del dataset. Finalmente, la aplicación retorna un txt con el dataset procesado.

```
    public static void cleanDataset(String datasetFile, String dictionaryFile, String outputFile) {
        try {
            ArrayList<String> dictionary = loadDictionary(dictionaryFile);

            try (BufferedReader br = new BufferedReader(new FileReader(datasetFile));
                 FileWriter fw = new FileWriter(outputFile, true)) {
                String line;
                while ((line = br.readLine()) != null) {
                    line = removePunctuation(line);
                    String[] words = line.split("\\s+");
                    for (String word : words) {
                        String cleanedWord = word.toLowerCase();
                        if (!dictionary.contains(cleanedWord)) {
                            fw.write(cleanedWord + " ");
                        }
                    }
                    fw.write("\n");
                }
            }
            System.out.println("Dataset cleaned successfully and saved to " + outputFile);
        } catch (IOException e) {
            System.out.println("An error occurred: " + e.getMessage());
        }
    }
}
```

Wordcount:

Esta aplicación utiliza el MapReduce de Hadoop para analizar la frecuencia de cada palabra en el dataset. La aplicación consiste en tres clases: WC_Runner, WC_Mapper, WC_Reducer. La aplicación cuenta con las siguientes dependencias:

```
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-common</artifactId>
            <version>3.3.3</version>
        </dependency>
        <dependency>
            <groupId>org.apache.hadoop</groupId>
            <artifactId>hadoop-mapreduce-client-core</artifactId>
            <version>3.3.3</version>
        </dependency>
```

Lo cual genera un Jar:

El cual llamaremos desde Hadoop y ejecutaremos con WC_Runner.

Comandos:

Crear repositorio para Hadoop:

hdfs dfs -mkdir /ruta_de_la_carpeta_input/

Poner nuestro Dataset en el Repositorio:

hdfs dfs -put /ruta_del_dataset/ /ruta_de_la_carpeta_input/

Correr nuestra aplicación Wordcount.

hadoop jar /mnt/c/Users/luisc/.m2/repository/com/mycompany/WordCount/1.0-SNAPSHOT/WordCount-1.0-SNAPSHOT.jar com.mycompany.wordcount.WC_Runner /input/datasetF.txt /output

Convertir los resultados en archivo de texto:

hadoop fs -get /output/part-00000 ~/wordcount-results.txt

Wordcount2:

Esta aplicación utiliza el MapReduce de Hadoop para analizar la frecuencia de cada par de palabras (bigramas) en el dataset. La aplicación consiste en una clase: WordCount2. La aplicación cuenta con las siguientes dependencias:

```
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-common</artifactId>
    <version>3.3.3</version>
</dependency>
<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-core</artifactId>
    <version>3.3.3</version>
</dependency>
```

Lo cual genera un Jar:

El cual llamaremos desde Hadoop y ejecutaremos con WordCount2.

Comandos:

Crear repositorio para Hadoop:

hdfs dfs -mkdir /ruta_de_la_carpeta_input/

Poner nuestro Dataset en el Repositorio:

hdfs dfs -put /ruta_del_dataset/ /ruta_de_la_carpeta_input/

Correr nuestra aplicación Wordcount.

hadoop jar /mnt/c/Users/luisc/.m2/repository/com/mycompany/WordCount2/1.0-SNAPSHOT/WordCount2-1.0-SNAPSHOT.jar com.mycompany.wordcount2.WordCount2 /input/datasetF.txt /bigram-output

Convertir los resultados en archivo de texto:

hadoop fs -get /bigram-output/part-r-00000 ~/wordcount-results2.txt

FrequencyAnalysis:

Esta aplicación se encarga de analizar los resultados de Hadoop (para una sola palabra), sortear los resultados que se repitan un mínimo de 5,000 veces para posteriormente ordenarlos. El input de la aplicación es el archivo de texto con los resultados de Hadoop, y el output es freq-results-sorted.txt.

FrequencyAnalysis2:

Esta aplicación se encarga de analizar los resultados de Hadoop (para bigramas), sortear los resultados que se repitan un mínimo de 5,000 veces para posteriormente ordenarlos. El input de la aplicación es el archivo de texto con los resultados de Hadoop, y el output es freq-results-sorted2.txt.