MapReduce: Lab assignment 3

Mustapha Tidoo Yussif and Jones Dari

April 2020

1 Introduction

This document explains our approach to solving the lab exercise three. The purpose of the lab was to use the MapReduce framework for processing big data. Specifically, the assignment required us to implement **(1) a simple word count algorithm and (2) a program to find the K most frequently occurring words in a text document**. In both cases, stop words were excluded. We are using **mrjob**, the python implementation of the MapReduce framework.

2 The Word Count Algorithm

The program inherited **MRJob** class to implement a job that executes the task. This job class required us to implement some methods such as **mapper_init, configure_args, mapper, reducer.**

- The **configure_args** method creates command-line options. We used this to pass the stop words file as a command-line argument.
- The **mapper_init** method initializes or setups the resources that the mapper method needs. We read the stop words and converted the words to a python set in this method.
- The **mapper** method splits every sentence from the text file into words. All the stop words are removed before further processing. The method finally returns a key-value pair consisting of the word as the key and a one as the value for every word. E.g., word, 1
- The **reducer** method in this program acts as a combiner and a reducer at the same time. This method finds the occurrences of each word in the text by summing up the results from the mapper method passed to it. The output of the reducer method is a key-value pair containing a word as the key and the value as the number of times it appears in the text.

3 The Top K Algorithm

Just like the word count program, this program also inherits the **MRJob** class. However, it uses multiple steps. It does this by overriding the default steps method of the MRJob class. The methods implemented in this program include:

- **configure_args** method: creates command-line options for passing the stop words file and the K value (the number of most the frequent words to output).
- **mapper_init:** for initializing resources for the mapper method.
- **mapper_get_words:** for yielding each word in every line from the text file.
- **combiner_count_words:** for finding the number of times each word occurred so far. The number of words seen so far might not be the total number of words in the text file, but a subset of the words from the file.
- **reducer_count_words:** for finding a number of times every word appears in the text.
- **reducer_find_top_k:** finds the k most frequently occurring words in the text. I added all the occurrences of each word to a list. This list is heapify, and only the top K values are maintained. This method uses python's max heap implementation, nlargest.

4 Programs requirements

1. The program must be run with python 3.5 or later.
2. The programs require the correct command-line options passed in the terminals.

Example.

1. **python3 mr_word_count.py ¡ File2ForLab3.txt –stop-words=stop_words.txt**
2. **python3 mr_top_k.py ¡ File2ForLab3.txt –stop-words=stop_words.txt –top-k=4**