Data Intensive Computing

Exercise 1: Text Processing Fundamentals using Mapreduce

Group 29:
Daniel Szakacs, 12229273, e12229273@student.tuwien.ac.at
Ekaterina Blasounig, 51845774, e51845774@student.tuwien.ac.at
Gergely Marton Tanay, 11702428, e11702428@student.tuwien.ac.at

Introduction

Text classification is an important aspect of natural language processing, allowing us to categorize large amounts of text data. With the rapidly increasing amount of large text corpora, it has become essential to use distributed computing techniques like MapReduce to process these datasets efficiently. In this assignment, we will use the previously mentioned MapReduce programming model to perform text classification on a subset of the Amazon Review Dataset 2014, consisting of 142.8 million reviews from 24 product categories. Specifically, our task is to preprocess the data, select discriminating terms for each category using the chi-square test, and output the top 75 terms per category. Our goal is to develop efficient MapReduce jobs that can handle the large-scale data set and produce accurate results. In this report, we will describe our methodology and approach in detail and present the results of our experiments.

Problem Overview

The problem we are trying to solve in this assignment is to select terms that discriminate well between product categories, as a preparation step for text classification. Our task is to calculate chi-square values for all unigram terms for each category, order the terms according to their value per category, and preserve the top 75 terms per category. Then we have to merge the lists over all categories and output a file that contains the top 75 most discriminative terms for each category in descending order. This will allow us to efficiently identify the most significant terms for text classification in a given product category.

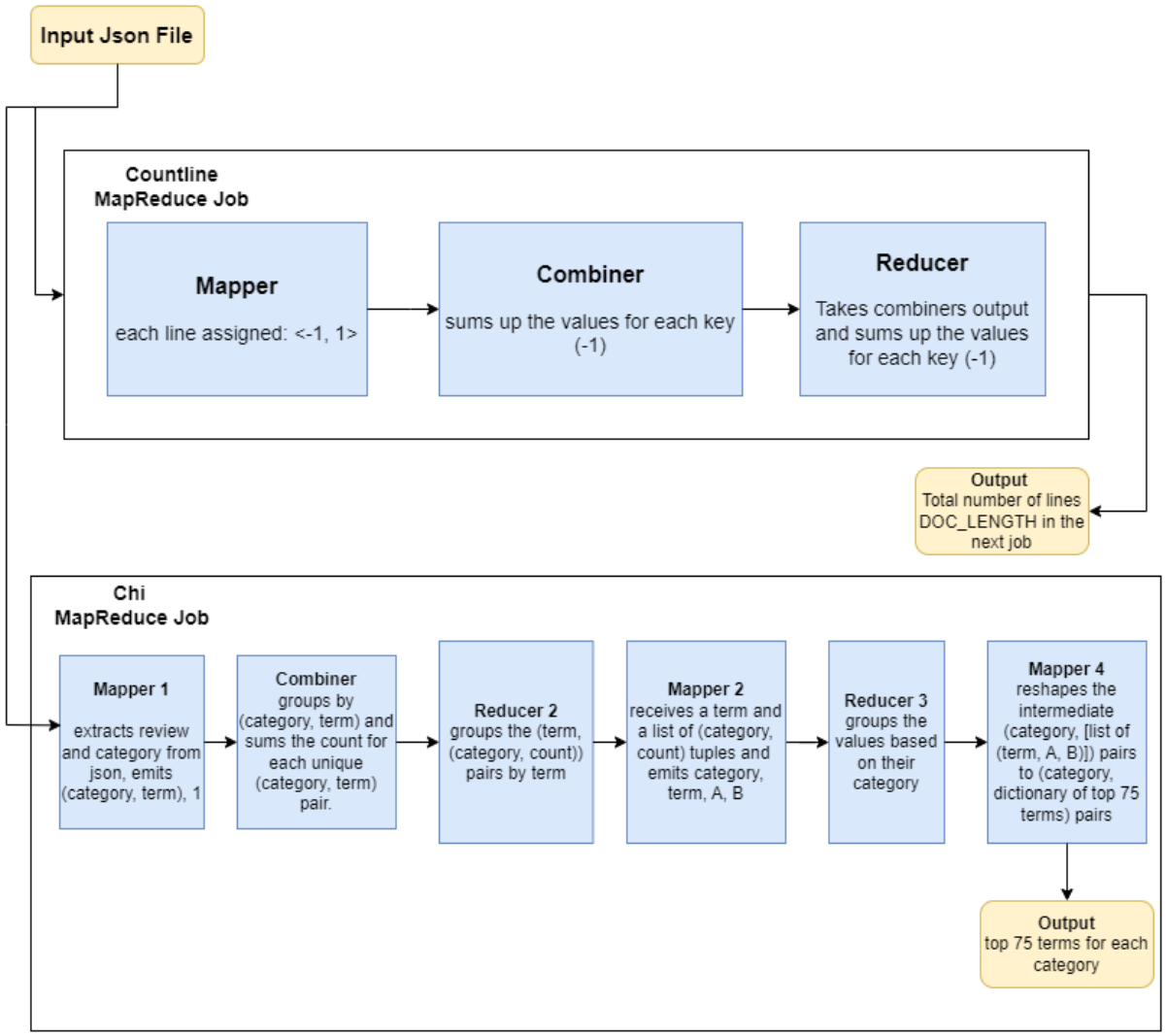
To highlight the efficiency of our implementation, evaluation of the submissions will be performed on the full dataset, which is 56GB. However, for development purposes, we are also provided with a smaller dataset that contains a 0.1% sample of a preprocessed version of the full Amazon review data set combined over all categories and extended with category information in JSON format.

To achieve this, we will tokenize the review text to unigrams using various delimiters, perform case folding, and filter out stopwords and single-character tokens. Then we will calculate the chi-square values for each unigram term in each category, order the terms according to their

value, and preserve the top 75 terms. Finally, we will merge the lists over all categories to generate the output file.

In summary, the problem we are addressing is the need to identify the most significant terms for text classification in a given product category, and we will be using MapReduce programming techniques to accomplish this efficiently on large text corpora.

Methodology and Approach



Our approach was running two Python programs sequentially: countline.py and chi.py. First, the countline.py program counts the number of lines in the JSON file to obtain the value of docs_length. Second, the chi.py program calculates the chi-square statistic for each word in a given category and takes the docs_length as an input. It consists of five steps of mappers, reducers and a combiner, which can be logically aggregated into two complete MapReduce steps which are finalized by a single mapper step. The first MapReduce step consists of a mapper, a combiner, and a reducer. This step gives us the value A and prepares the data structure for the next step.The second MapReduce step has a mapper and a reducer. The main goal here is to calculate B and to prepare the structure again for the final mapper step.

The final step consists only of a mapper because we do not need to change the structure of the lines. Here we already receive a needed construct of data from the previous steps. The mapper and output of the countline.py file allow us to calculate the left values for the chi-square statistic, namely C and D. As a result, we can calculate the final chi-square on the fly. The only thing which is left to receive the final output - sorting and cutting - can be done here as well.

To complete the task, two Python files were created that need to run one after another:

- countline.py
- chi.py

Detailed description of the codes:

Countline.py

This is a MapReduce job to count the number of lines in a JSON file (which is the input file). The job consists of a single step that involves a mapper, combiner, and reducer function. The mapper function emits a key-value pair with key -1 and value 1 for each line in the JSON file. The combiner function then aggregates the counts of each key-value pair emitted by the mapper. Lastly, the reducer function sums up the counts and emits a single key-value pair with key None and value equal to the total number of lines in the JSON. The output of the job is this final key-value pair, which represents the number of lines in the input file.

To sum up:

- Input: A JSON file with n lines.
- Intermediary: A list of key-value pairs with key -1 and value 1 for each line in the input file.
- Output: A single key-value pair with key None and value equal to n, the number of lines in the input file.

Chi.py

This MapReduce job calculates the chi-square statistic for each word in a given category in a set of Amazon product reviews. The job consists of five steps of mappers, reducers and a combiner. The input to the MapReduce job is a set of Amazon product reviews in JSON format and the stopwords txt file.

In the first step, the mapper function extracts the category and the review text from the input file, tokenizes the review text of each product, filters out the stopwords (using the stopwords.txt) and the words with only one character. Then it emits a key-value pair of (category, term) , 1 to count how many terms we have in each category. The purpose of this step is to count how many times each term appears in each category, which is later used to calculate the chi-square statistic.

The combiner function in the first step receives the key-value pairs emitted by the mapper, groups them by term, then sum the counts of each term per category.

In the second step, the reducer2 function collects key value pairs from the combiner and emits a term as a key and the value as a list of tuples. Here the tuples contain the category and the sum of the counts of each term in the category. The purpose of this step is to prepare the data for the next step.

In the third step, the mapper2 function receives the term and the list of tuples (category, count) for each term emitted in the previous step. Then it creates a dictionary where keys are the categories and values are the counts of terms per category. It also creates a new empty dictionary. In this dictionary the first element will be the count (value) and the second element will be the count of all terms in the category minus the count of the term. For each category the mapper2 emits the key value pair where key is the category and the tuple that contains the term and the elements of the new dictionary.

In the fourth step, the reducer3 function groups the key-value pairs emitted by the mapper2 by category and emits the key value pair where key is the category and is the list of all tuples that corresponds to the category. This is done to prepare the data for the final step, where we compute the chi-square statistic.

In the fifth step, the mapper4 function receives the list of (term, A, B) tuples for each category emitted in the previous step and calculates the chi-square statistic for each term in each category with the following steps:

- Firstly it calculates the total number of documents in the category by summing the count of all terms in the category (in sum_pro_category dictionary)
- Secondly, it creates a dictionary where the keys are the terms and the values are the list with the four elements: A, B, C and N. A is the count of the term in the category, B is the count of all terms in the category minus the count of the term, C is the count of all terms in the text minus count of the term in the category, N is the number of documents in the text.
- For each term the mapper4 calculates the chi-square value using the following formula: (N*(A*D-B*C)^2)/((A+B)*(A+C)*(B+D)*(C+D)) where D = N-A-B-C

The resulting dictionary is sorted in descending order by the chi-square values and the top 75 are selected. The output of this step is a key-value pair of the form category, final_dict, where final_dict is a dictionary containing the top 75 words with the highest chi-square values.

Summary and conclusion

In this assignment, we solved the task of text classification using MapReduce programming techniques. The main goal was to identify the most significant terms per category from the Amazon Review Dataset 2014. For developing the MapReduce jobs, we have worked with a smaller dataset that contains a 0.1% sample of a preprocessed version of the full Amazon review dataset. The MapReduce jobs were written using mrjob library in Python.

We managed to run the code on the development dataset on the TU Wien hadoop cluster. The execution time is approximately 3 minutes on the development set, which is longer than on a personal laptop. The corresponding output file can be found in the attached zip file. Unfortunately while running the code on the full dataset we experienced troubles from the cluster side, which did not allow us to receive any output for the full dataset. Even a simple countline.py did not produce any result, although we tried it many times. We assume that the reason lies in the high traffic on the Hadoop cluster.

In conclusion, the use of MapReduce programming techniques enabled us to efficiently process large text corpora and the knowledge obtained in this exercise can be applied by us in the future work.