Report Exercise 1:

Text Processing Fundamentals using MapReduce

Fritzer Florian
11802387
e11802387@student.tuwien.ac.at

Gratt Lisa
12017333
e12017333@student.tuwien.ac.at

Schuch Teresa
12007762
e12007762@student.tuwien.ac.at

April 22, 2024

1 Introduction

The aim of this assignment is to develop a MapReduce program for processing large text corpora. For this purpose, the mrjob package in Python is utilized. The primary objective of this exercise is to identify terms that effectively distinguish between various categories within the Amazon Review Dataset 2014. This Dataset contains 142.8 million reviews from 24 product categories. To accomplish this task, several preprocessing steps were undertaken, including tokenization, case folding and stopword filtering in order to gather the terms occuring in the review texts. The MapReduce job is then designed to compute the chi-square values for every term across all categories. Finally, the terms are ordered according to their value per category and the top 75 most discriminative terms per category are stored in a file together with a merged list of these terms over all categories. This report details the implementation of a MapReduce solution, outlining the problem overview, methodology and approach for effectively utilizing the mrjob Python package to accomplish the aforementioned tasks.

2 Problem Overview

The task involves efficiently analyzing vast amounts of textual data from the Amazon Review Dataset 2014, which has a total size of 56 GB. Developing an efficient algorithm is crucial to run the program in reasonable time. Therefore unnecessary overheads and calculations need to be avoided. The goal is to identify terms within these reviews that serve as effective discriminators between the various categories.

3 Methodology and Approach

3.1 χ^2 Statistic

This can be obtained by computing Pearson's χ^2 Statistic, which is a dependency measurement for categorical variables. In the domain of text processing χ^2 is a metric, which allows to evaluate the relationship between the presence of a particular term t in a corpus of documents and its association with a certain category c. Particularly, ${\chi^2}_{t,\,c}$ measures whether the occurrence of a term is significantly dependent on the category which it belongs to. By calculating ${\chi^2}_{t,\,c}$ for each term-category combination, we can determine, how

strongly a term is connected to a category. Terms yielding higher ${\chi^2}_{t,c}$ values are considered to be more typical of a particular category. Therefore the metric is computed for every pairing of category and term, followed by aggregating the results by category.

$$\chi^2{}_{\mathrm{t,c}} = \frac{N(AD - BC)^2}{(A+B)(A+C)(B+D)(C+D)} \tag{1}$$

A...number of documents in c which contain t
B...number of documents not in c which contain t
C...number of documents in c without t
D...number of documents not in c without t
N...total number of retrieved documents

By sorting the ${\chi^2}_{t,c}$ values for each category from highest to lowest, we derive ranked lists of terms based on their dependence between term and category. Those at the top are considered to be the most characteristic for their category.

3.2 MapReduce algorithm

The strategy for solving the exercise involves the implementation of two MapReduce jobs. The first job serves the purpose on computing the count of reviews for every category. This step is straightforward but important for the computation of the χ^2 values in the second job. Subsequently the output containing the top 75 terms within each category is written to a file.

3.2.1 1. Job: Category Counter

The first job is needed to determine the number of reviews in each category. This is done by a mapper-, combiner- and reducer-function. The mapper gets a line of the dataset in JSON format and extracts the category. It yields a key-value pair where the key is the category and the value is $1 < category, 1 >$. This indicates that one occurrence of the category has been encountered. The combiner then locally aggregates these counts for each category. This reduces the amount of data transferred between the mapper and reducer, improving efficiency.

The reducer aggregates the output from the combiners. It receives the key-value pairs, where the key represents a category and the value is a list of counts. It calculates the total number of occurrences by summing up the counts. Finally it emits key-value pairs where the key is the category and the value is the total count $<$ *category, count occurrence* $>$.

3.2.2 2. Job: χ^2-Calculator

The second job calculates the χ^2 values for each term in one category. This serves the goal of identifying which terms are most strongly associated with each category. For this purpose a mapper- and two reducer-functions are implemented.

The mapper_init method initializes the mapper by loading the stopwords from the textfile into a set. The mapper method takes then each line of the dataset in JSON format as input and extracts the category and the review. For preprocessing the review text is tokenized into terms, by separating it at any whitespace, tab, digit and set of characters *{}()[].!?,;:+=-_”’‘ #@&*%€$§\/*. Then all terms are transformed to lowercase and the

stopwords as well as tokens consisting of only one character are filtered out. For each remaining token after filtering, the method yields a key-value pair. The key is the token itself and the value is the category of this review $<$ *token, category* $>$.

In the reducer_init method the category counts from the first job are loaded from the file into a dictionary. Furthermore the total number of reviews is calculated. The first reducer receives a word and a list of categories corresponding to the reviews the word appeared in. It then iterates through these categories and for each unique category, it calculates various statistical measures, which are needed to compute the ${\chi^2}_{t,c}$ value. A is simply the number of times the word appears in one category. B indicates how often the word appears across all categories minus A. To calculate C the output of the first MapReduce job is needed, which was loaded in the init method and provides the counts of occurrence of each category over all reviews. C is then the total number of reviews in one category minus A. D is the sum of all reviews minus the total number of reviews in one category minus C. Finally, The ${\chi^2}_{t,c}$ statistic (1) can be computed. The Reducer yields for each category a key-value pair, where the category is the key and the value is a tuple containing the word and its corresponding χ^2 value $<$ *category, (word, chi-squared value)* $>$.

For the determination of the top 75 terms per category a further reducer step is necessary. While the first reducer aggregates the data and calculates the chi-squared statistics for word-category pairs, a secondary reducer is needed to sort the results and identify the top 75 terms per category. The function receives the category and the word-χ^2-value tuples. It then sorts the tuples based on the χ^2 value in descending order and extracts the top 75 tuples for every category. The function then yields key-value pairs where the category serves as the key, and the corresponding tuple containing the word and χ^2 value is the value.$<$ *category, (word, chi-squared value)* $>$

3.3 Output

In the run script we manage the execution of the two MapReduce jobs in order to obtain the final result which is saved to the file output.txt. After the execution of the Category Counter job the result is written to a file named category_counts.txt which is used in the second job χ^2-Calculator (see above). The script then proceeds with the execution of the second job. Beforehand the file paths for the stopwords and category counts file were specified as they are required by this MapReduce job. Finally, the result is written to an output file named output.txt. This file contains one line for each category with the top 75 terms and their χ^2 values as well as one line containing the merged dictionary of the terms.

3.4 Conclusion

In conclusion, the development of a MapReduce program using the mrjob package in Python has proven effective for processing large text datasets, such as the Amazon Review Dataset 2014. Efficiency, with a focus on minimizing unnecessary overhead and computations, was considered throughout the entire project, especially due to the dataset's size. We decided on utilizing two MapReduce jobs, although that means that the entire dataset needs to be read twice. However, that seemed to be the simplest way to calculate the counts of categories and that job didn't slow down the runtime significantly. For the second job, we decided on using two reducer-functions. First, we calculate all χ^2 values one after another before sorting them. The second reducer then sorts the resulting list of tuples. Given the size of our dataset, it's essential to note that sorting operations, particularly in-memory sorting, can slow down runtime, especially when the data exceeds available memory capacity.

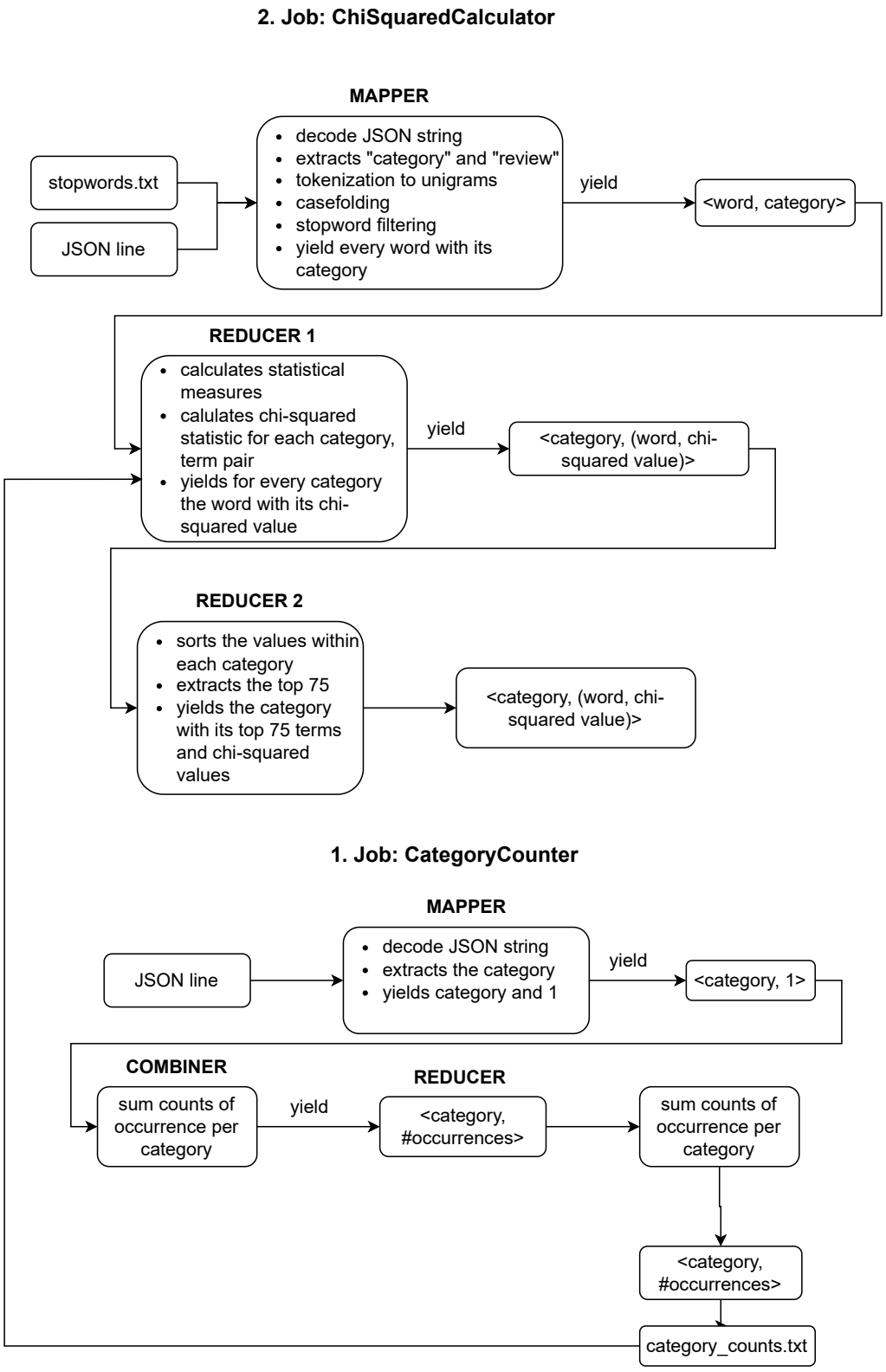

Figure 1: pipeline