Report for Assignment 1

Data-Intensive Computing SS2024

Lukas Mahler (11908553)   Julian Flür (11807481)

Contents

Introduction

In this exercise we are tasked with performing apreprocessing step for a text classification task. We are going to calculate the chi-square value for words in the text corpus. The data we are working on are reviews from various Amazon aritcles, divided in disjoint categories.

Problem Overview

Data set

We are working on a data set of Amazon reviews. While there are 10 attributes, such as **helpful** or a **reviewTime**, we are only interested in two of them. Namely:

- **category**: the category that the product belongs to
- **reviewText**: the content of the review; this is the text to be processed

Each review is in exactly one category and requires no preprocessing.

The review text on the other hand has to be split into unigrams where each token is one word. We split on whitespaces as well as a list of special characters and digits. Also all the text is cast to lower case and certain stopwords are ignored for the analysis.

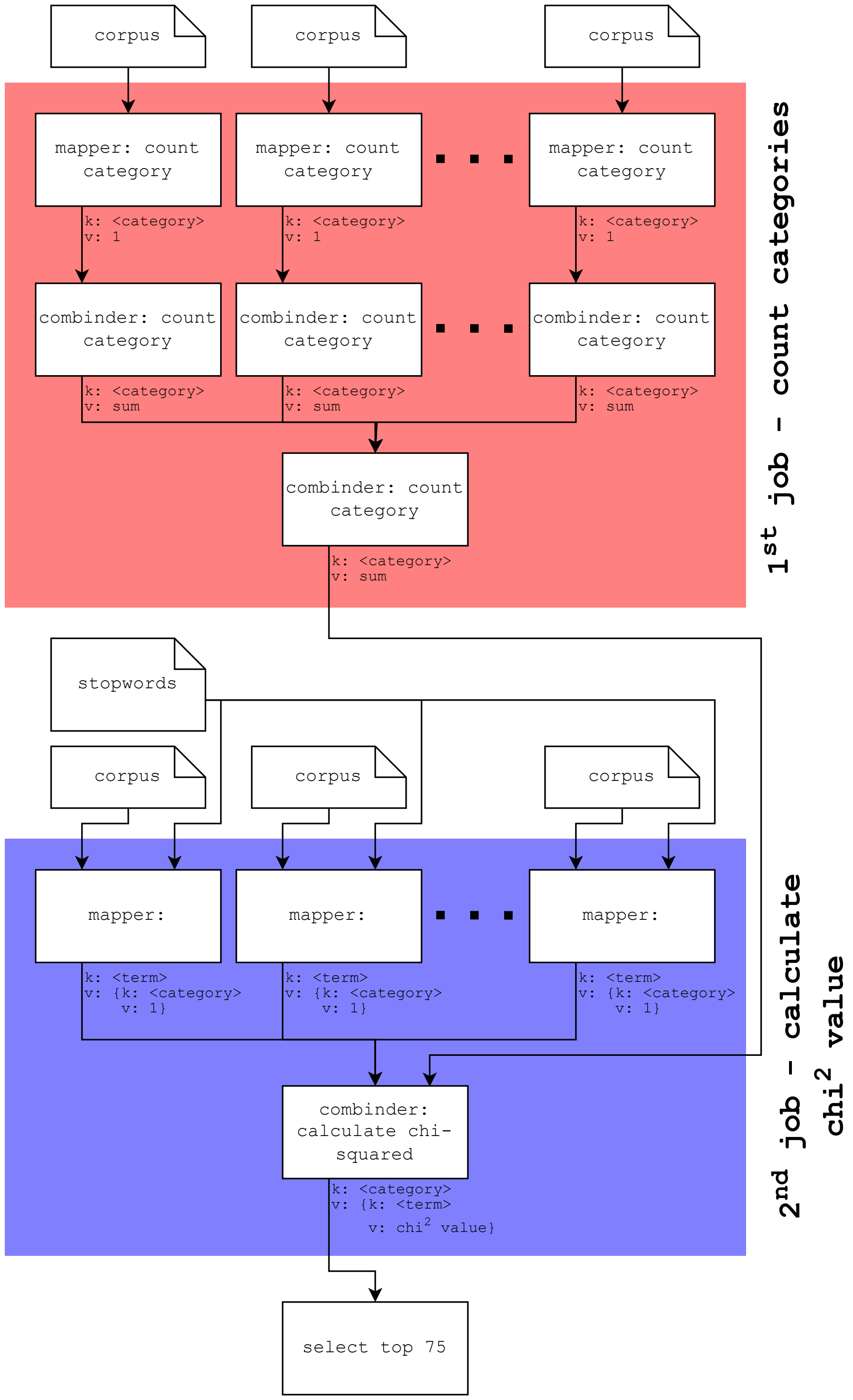
Chi-square value

The chi-square value is a metric, expressing the dependence between a token and a category. Essentially the more often a term is used in a category and the less it is used in reviews from other categories the more important it is.

$$\chi^2_{\mathrm{tc}} = \frac{N(AD - BC)^2}{(A+B)(A+C)(B+D)(C+D)}$$

Methodology and Approach

Essentially we split the problem into two map reduce jobs. In a first step the amount of reviews for each category is calculated, while the second job counts the amount of reviews a term occurs in per category. With the result of the first jobs we are able to calculate the chi-square value for each term and category. As a final step we can select the top 75 terms for each category and return the result.


corpus
corpus
corpus
mapper: count category
mapper: count category
mapper: count category
k: <category> v: 1
combinder: count category
combinder: count category
combinder: count category
k: <category> v: sum
combinder: count category
k: <category> v: sum
1st job - count categories
stopwords
corpus
corpus
corpus
mapper:
mapper:
mapper:
k: <term> v: {k: <category> v: 1}
combinder: calculate chi-squared
k: <category> v: {k: <term> v: chi2 value}
2nd job - calculate chi2 value
select top 75

First MapReduce Job

This map reduce job is very straight forward.

mapper: For each review check the category and append a key-value pair of the form {k: <category>, v: 1} to the result.

combiner: On each worker we can sum up the amount of reviews per category and return {k: <category>, v: # of reviews}.

reducer: In this step we sum up the amount of reviews over all the workers and the result is {k: <category>, v: # of reviews}.

Second MapReduce Job

The second job is similar to the first, but since a toke can occure in multiple categories we work with dictionaries as values. Adding a combinder for this job did not result in a shorter runtime, thus it is not implemented.

mapper: For each review iterate over the tokens and return {k: <term>, v: {k: <category>, v: 1}}.

reducer: During initialization we read the result of the first map reduce job. Then we sum up the amount of reviews per term and category. With this information we can calculate the chi-square value and return {k: <category>, v: {k: <term>, v: χ^2 value}}

Calculate the final result

Finally we can order the collection of terms for each category and select the top 75 terms. This is implemented as a third map reduce job for a better structure but only a reducer step is implemented.

Conclusions

The total runtime of the cluster is **ToDo**, which is below the threshold of 20 minutes. Of course the task could be solved by a single map reduce job, but the interim results of each mapper would be a more complicated data structure, essentially reducing the simplicity of the task.

Runtime detailed

ToDo