Report for Assignment 1

Data-Intensive Computing SS2024

Lukas Mahler (11908553) Julian Flür (11807481)

Contents

Introduction

In this exercise we are tasked with performing a preprocessing step for a text classification task. We are going to calculate the chi-square value for tokens in the text corpus for each of 24 categories. The data we are working on are reviews from various Amazon articles, 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.

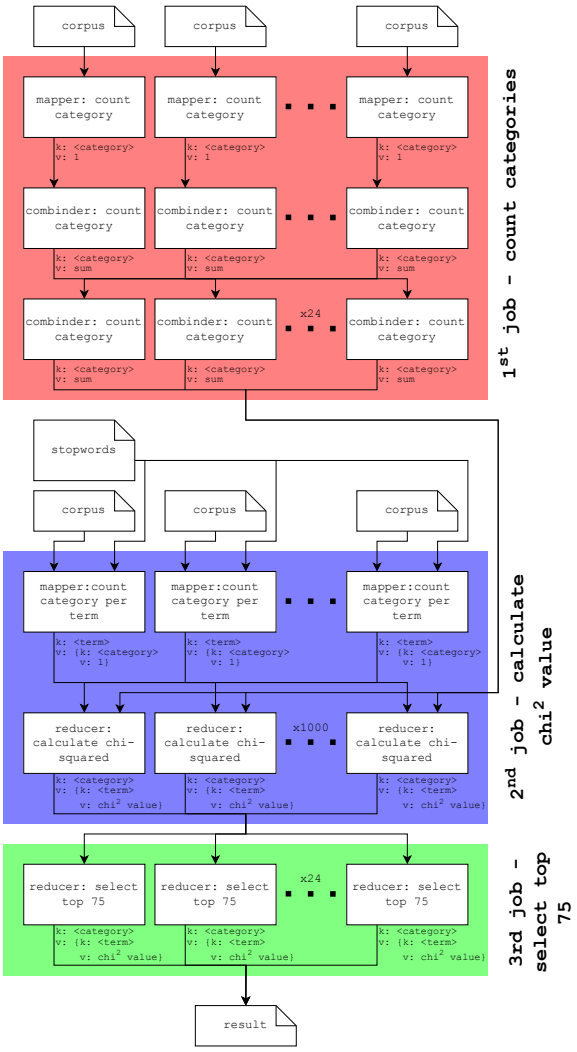
Chi-square value

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.

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

Methodology and Approach

We split the problem into two map reduce jobs. In the only step of the first job the amount of reviews for each category is calculated. With the result of the first job we are able to calculate the chi-square value for each term and category in the reducer of the first step of the second job. The final step sorts the terms and yields the top 75 terms for each category.



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 the sum for each category (key) is calculated and

returned {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}.

We specify 24 reducers according to the amount of categories. Spawning more reducers will cause no benefit, as they will not be used.

Second MapReduce Job

Step 1

The second job, similarly to the first one, also counts occurrences per review. However, this time the occurrences of tokens inside the review text are counted. Since there are multiple tokens per review, the counts of tokens are stored inside a map (dictionary) as value. Adding a combiner for this job resulted in longer runtimes, hence it is implemented but not used.

mapper: For each review iterate over the unique tokens (done by using a set) and return {k: <term>, v: {k: <category>, v: 1}}.

reducer: During initialization we read the result of the first map reduce job, as the result of the first job is of a small and constant size (only 24 categories). Then we sum up the amount of reviews per term and category. With this information we calculate the chi-square value and return {k: <category>, v: {k: <term>, v: χ^2 value}}

For this job we spawn 1000 reducers. A few tests have shown, that a larger amount of reducers decreased the runtime significantly, which is the expected behaviour. We did check for 1, 10, 400 and 1000 reducers, the latter one achieving the best performance.

Step 2

Finally we can order the collection of terms for each category and select the top 75 terms. This is achieved by utilizing a map reduce step containing only a reducer.

reducer: For each key (category) we order the results and select the top 75 terms.

Post Processing

The result of the second map reduce job is transformed into the final output not as map reduce job, but rather directly on the host which starts the jobs, as the result of the second job is already relatively small (75 tokens each for 24 categories) and using a map reduce job would be overkill.

Once the output is formatted, it is written to the file system to the 'out' directory, as well as printed to standard output.

Conclusions

The total computation time on the cluster is about 12 minutes, which is below the threshold. We decided on the structure of three map reduce jobs for a simple structure of the interim results. This way the keys are always of the same type and a straight forward pipeline can be used.

Result

Since it is not possible to manually check our results we heuristically check them. For example the top 5 terms for the category "Book" are: author, reading, characters, written and reader. All of these terms are obviously related to books and expected to show up in this list. It is worth noting that the term characters also shows up in other categories such as "Kindle Store". Again this does not come as a surprise since the categories are related, and characters being a topic of a review is also not surprising.

Runtime

In the table below the runtime is given.

step	runtime
count review per category	57 sec
calculate chi-square value	8 min 42 sec
select top 75 terms	1 min 58 sec
total	**11 min 37 sec**