Distributed Computing and Storage Architectures: Project MapReduce

Prof. Nikolaos Deligiannis
{*ndeligia@etrovub.be*}

Assistants:
Xiangyu Yang Yiming Chen
{{*xyanga, cyiming*}*@etrovub.be*}

2020-2021

Project Overview

In this project you will practice your skills on MapReduce algorithms in Python using MRJob[1]. The data you will be working on includes movie titles and ratings from IMDB[2], annual retail exports, meta data on Arxiv[3] scientific papers and large matrices. For each of the provided data you are asked to perform a set of tasks such as counting, matrix manipulation and similarity searching.

MRJob

MRJob enables us to write MapReduce jobs in Python and run them on several platforms, being it locally, a Hadoop cluster, EMR, Dataproc, or spark jobs on a Hadoop cluster. For installation we refer to the documentation manual[4]. Below is an example where the mapper will receive one line from a text document and emit a lowercase version of each word in the line and a 1.

```
from mrjob.job import MRJob
import re
WORD_RE = re.compile(r"[\w']+")

class MRWordFreqCount(MRJob):
        def mapper(self, _, line):
        for word in WORD_RE.findall(line):
                yield word.lower(), 1

        def combiner(self, word, counts):
                yield word, sum(counts)

        def reducer(self, word, counts):
                yield word, sum(counts)

if __name__ == '__main__':
        MRWordFreqCount.run()
```

Using the command below we can feed the map reduce jobs our files and test our algorithm locally.

[1] https://github.com/Yelp/mrjob
[2] https://datasets.imdbws.com/
[3] https://arxiv.org
[4] https://mrjob.readthedocs.io/en/latest/

```
python3 word_counter.py in.txt
```

1 IMDB

In the **1_IMDB** folder you will find:

1. **title.basics.tsv**: Basic information on movies, tv-shows and shorts from IMDB[5]. Parameters include: [tconst titleType primaryTitle originalTitle isAdult startYear endYear runtimeMinutes genres]

1.1 TASK 1: Find the 50 most common keywords for all movies and shorts

For all entities for the type movie and short, find the top 50 most common keywords used in the primary titles using MapReduce. Try to avoid auxiliary verbs, prepositions, articles and conjunctions as keywords. You can use libraries like NLTK to help you with the latter part.

1.2 TASK 2: Top 15 keywords for each movie genre

For each possible **genre** of the type **movie** (this excludes tv-shows, etc...) find the top 15 most common keywords within their primary titles using MapReduce. Try to avoid auxiliary verbs, prepositions, articles and conjunctions as keywords. You can use libraries like NLTK to help you with the latter part.

2 Online Retail

In the **2_RETAIL** folder you will find the retail sales of one individual company for two years. Original file can be found on the archive of ICS UCI[6]

1. **retail0910.csv**: CSV dataset of retail sales for 2009-2010. Parameters include: [Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer ID,Country]

2. **retail1011.csv**: CSV dataset of retail sales for 2010-2011. Parameters include: [Invoice, StockCode, Description, Quantity, InvoiceDate, Price, Customer ID,Country]

2.1 TASK 3: Top 10 best customers for each retail year

Using MapReduce find the top 10 customers of **each retail year** that bought the most in terms of total revenue (quantity * price).

2.2 TASK 4: Best selling product

Using MapReduce find the best selling product, once in terms of total quantity, and once in terms of total revenue for both retail years.

[5]https://www.imdb.com/
[6]https://archive.ics.uci.edu/ml/machine-learning-databases/00502/

3 Similar paper recommendations

In the **3_TEXT-SIMILARITY** folder you will find:

1. **arxivData.json**: A JSON archive of meta-data on 20.000 scientific papers on Arxiv

3.1 TASK 5: Cosine Similarity

Given a random paper summary find the paper with the highest Cosine Similarity score [1]. You will have to implement the Cosine Similarity function yourself for text, if you rely on an external library purely for this function, points will be deducted. In order to vectorize your texts you can mak euse of TF-IDF, Word2Vec, Bag of Words any other option that will allow you to convert text to vectors.

4 Matrix Multiplication

In the **4_MATRIX** folder you will find:

1. **A.txt**: A matrix of 1000 rows and 50 columns
2. **B.txt**: A matrix of 50 rows and 2000 columns
3. **C.txt**: Dot product of A and B

4.1 TASK 6: Matrix dot product

Implement the matrix dot product in MapReduce. You can use the provided matrices to validate your code. The files can be loaded using numpy.

Comments

All of the produced code should be written in **Python** and has to be accompanied with explanatory comments. All tasks are required to be implemented within the MapReduce framework in a distributed fashion. As it is a group project, we encourage you to utilise source control. Copying from other groups equals plagiarism.

Requirements

In this project, you will work in teams of **2 people**. You can form teams by yourselves, and inform us by the **8 November 2020**. If teams of 2 cannot be formed or if an internal issue prevents your team from continuing, please don't wait until the deadline, and contact the professor.

Evaluation

For this project, you will be evaluated based on following criteria:

- Your written report (explicitly mentioning the contribution of each member of the team) and your source code (with explanatory comments).

Deliverables

Upon finishing the project, you need to submit:

- A report describing your work and results for each task in the project;
- The full source code of your project, with clear comments;
- A detailed guideline to run your code, including the dependencies, the commands to run, etc...;
- A description of the contributions of each team member in the project.

Deadline

The deadline for the submission of the project report and material is **23:59 CET, Sunday 10 January 2021**. Any submission after this deadline is considered invalid. You will have to combine the deliverables in a .zip folder and submit them through email to svandenb@etrovub.be

References

[1] A. Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56, 2008.