



Map Reduce Local

INF-0617 - Complexidade de Vocabulário

Rafael Fernando Ribeiro
Thiago Gomes Marçal Pereira

Prof. Lucas Wanner

Apresentação do Problema

Neste trabalho, recebemos os dados de livros de vários autores

O objetivo era obter a complexidade do texto de cada autor e ordenar o resultado pelo autor mais complexo.

Solução

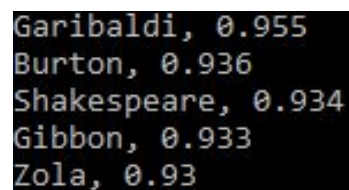
O projeto consistiu em uma implementação de MapReduce, em Python.

Para o mapper, a partir dos livros disponibilizados para a aplicação, foi desenvolvido um script (**map_author_vocabulary.py**) para separar o nome do arquivo, cada palavra e um valor 1 a esta palavra. A primeira coisa feita a cada linha do livro recebido pegamos o nome do livro pela variável de ambiente do hadoop “mapreduce_map_input_file”, depois fazemos um separação no nome do arquivo pela cadeia de caracter “u-” e assim separamos o nome do arquivo do caminho dele no sistema. Na segunda parte do script removemos todos os caracteres que não são alfabéticos, ou seja de a-z, e transformamos toda a cadeia em letras minúsculas conforme requisitado e fazemos a separação da cadeia por cada palavra separadamente. E no final escrevemos na saída do script o nome do arquivo, cada letra e o valor 1 dizendo que ela apareceu no texto.

O segundo passo foi criar um reducer (**reduce_vocabulary.py**) para encontrar as palavras mais comuns em todos os livros. Para isso nosso reducer recebeu como entrada a saída do mapper descrito anteriormente e para cada palavra encontrada ele adicionava em um dicionário, caso a palavra já estivesse no dicionário o valor era incrementado, o nome do livro foi ignorado neste script. No fim, ordenamos o dicionário pela valor da quantidade que cada palavra apareceu e escrevemos na saída as 3000 palavras que mais apareceram nos livros e conforme requisitado no problema.

O passo final foi criar um novo reducer (**reduce_author_complexity.py**) para processar a complexidade de cada autor, para isso, o script inicialmente carregou um arquivo com o nome de cada autor para cada nome de arquivo de livro disponível utilizando o arquivo fornecido “master_list.csv” e criou um dicionário com o nome do arquivo e o autor que o escreveu. Em seguida, carregamos o dicionário criado no script anterior com as 3000 palavras mais comuns. Posteriormente, este reducer recebeu como entrada o mesmo arquivo gerado pelo script de mapper gerado no início e para cada linha lida deste arquivo, contendo o nome do arquivo do livro, a palavra fizemos o processamento a seguir, procuramos o nome do arquivo no dicionário de autor, caso não encontrado pulamos para uma nova linha, assim eliminamos os autores considerados como desconhecidos, após isso criamos um novo dicionário que continha o nome do autor como chave e as palavras mencionadas por ele em qualquer livro, depois olhávamos se a palavra mencionada pelo autor já estava no dicionário, caso afirmativo, ela era ignorada, caso contrário verificamos se a palavra constava no dicionário de palavras comuns, se ela estivesse nesse dicionário adicionamos ela com o valor 0, caso não constasse era adicionada com o valor 1. Por fim, para cada autor no dicionário criado somamos todos os valores das palavras e dividimos pelo número total de palavras mencionadas pelo autor e com isso obtemos a complexidade de escrita de cada autor. Com este resultado, ordenamos a lista de autores da maior complexidade para a menor e escrevemos na saída este resultado, que é o resultado final requisitado no problema.

Resultado dos 5 primeiros autores com maior complexidade de vocabulário considerando os 250 primeiros livros da biblioteca.



```
Garibaldi, 0.955
Burton, 0.936
Shakespeare, 0.934
Gibbon, 0.933
Zola, 0.93
```

Passos para executar o script

Para executar o trabalho siga estes passos:

1 - dentro da sua pasta data do container docker, extraia o arquivo “**gut.zip**”

2 - confirme que a para gut está em “**/tmp/data/gut/txt**” dentro do container docker

3 - Execute os comandos na sequencia

```
$HADOOP_HOME/bin/hadoop jar
```

```
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.0.0.jar -input /tmp/data/gut/txt  
-output /tmp/data/author_vocabulary -mapper "python map_author_vocabulary.py"
```

```
$HADOOP_HOME/bin/hadoop jar
```

```
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.0.0.jar -input  
/tmp/data/author_vocabulary/part-00000 -output /tmp/data/vocabulary -mapper '/bin/cat'  
-reducer "python reduce_vocabulary.py"
```

```
$HADOOP_HOME/bin/hadoop jar
```

```
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.0.0.jar -input  
/tmp/data/author_vocabulary/part-00000 -output /tmp/data/author_complexity -mapper  
'/bin/cat' -reducer "python reduce_author_complexity.py"
```

5 - para visualizar o resultado:

```
$HADOOP_HOME/bin/hdfs dfs -cat /tmp/data/author_complexity/part-00000
```

Observação:

Tivemos problema de estouro de memória da máquina docker durante o processo de map quando executamos o script com os 595 livros do arquivo gut.zip. Sendo assim, Utilizamos nos nossos resultados os 250 primeiros arquivos ordenados por ordem alfabética