Big Data Analytics
Lab 02 - Wikipedia

Damien Rochat, Dorian Magnin, and Nelson Rocha
Master of Science in Engineering
University of Applied Sciences and Arts Western Switzerland

April 4, 2019

1 Introduction

The aim of this laboratory is to discover Spark with a large amount of data. In practice, we are going to extract popularity of programming languages, based on provided Wikipedia dataset. Here, a programming language will be more popular if it's referenced in more Wikipedia articles. Finally, the results will be compared with the last RedMonk top-20.

Spark could be run in a cluster, but for now, we'll just use our own computers.

2 Implementation

We filled the provided function as described above.

A specific point is we had to split the text data into words to search matches programming languages name. Here is the dedicated function:

```
def words: Array[String] = text
  .toLowerCase()
  .replaceAll("[!\"$%&'()*,./:;<=>?@\\[\\]^_‘{|}~]", " ")
  .split(" ")
```

The word extraction starts to lowercase the text, replace punctuation characters by a space and split by space. We are able to extract « scala » from « I love Scala so much! » and from « I love Scala! ».

Based on our results, we still miss some references to our list of programming languages. This is because we didn't replace characters #+- with spaces. Those are used by some programming languages name.

2.1 Naive Ranking

The first algorithm, based on the *aggregate* function. The Wikipedia articles are passed through one time per language, which is not efficient. This took 46 seconds to finish.

```
Processing Part 1: naive ranking took 45991 ms.
```

2.2 Ranking with an inverted index

At the second step we managed to get a speedup of a roughly 45%.
This is due to the inverted index use. By providing us with the information of which article contains which language, it spares us the work of parsing all the RDD, for each programming language.
So the results are far quicker to retrieve.

```
Processing Part 2: ranking using inverted index took 25603 ms.
```

2.3 Ranking with reduceByKey

Finally, we achieved a bit more speedup by using the *reduceByKey* instruction.
By reducing before the shuffling, we greatly distribute the computation between workers.
As we are working on our own machines, the beneficial effects of this change are not very important.
But if we were using a remote Spark cluster, the improvements would have been far more visible.

```
Processing Part 3: ranking using reduceByKey took 22242 ms.
```

3 Results

We obtained the same results with the three ranking algorithms.

```
List((JavaScript,1769), (PHP,1416), (Java,891), (C#,785), (CSS,580),
  (Python,556), (C++,553), (MATLAB,343), (Perl,317), (Ruby,273),
  (Haskell,128), (Objective-C,111), (Scala,96), (Groovy,62), (Clojure,59))
```

Except for JavaScript, large winner in both cases, our Wikipedia ranking doesn't match RedMonk one. However, the top-7 of the two rankings is almost the same, in different order.

Rank	**Wikipedia Lab**	**RedMonk**
1	JavaScript	JavaScript
2	PHP	Java
3	Java	Python
4	C#	PHP
5	CSS	C#
6	Python	C++
7	C++	CSS
8	MATLAB	Ruby
9	Perl	C
10	Ruby	Objective-C
11	Haskell	Swift
12	Objective-C	TypeScript
13	Scala	Scala
14	Groovy	Shell
15	Clojure	Go
16		R
17		PowerShell
18		Perl
19		Haskell
20		Kotlin

Table 1: Wikipedia lab vs RedMonk january 2019 ranking

The RedMonk ranking uses more sources, like GitHub and StackOverflow.

We can conclude that **JavaScript** is the current most used programming language.