Big Data Project

Frequency of Covid Related Terms

Burak Balci s1073728
Radboud University 24/06/2024

Contents

1 Introduction

In this report, I will explain my experience with this project assignment step by step. The part where I explained what I learnt and my general experience from start to finish is at the very end. That part is more of what I tried to do at first, what challenges I faced, basically like my journal for this project. The next sections until that are more based on my final form of this assignment.

2 Objective

I struggled to find an interesting topic without making comparison on data from different time frames, so I decided on a topic that is not very interesting. At this point I'm hoping to use what I learnt during the course for this project rather than get actual interesting results. My goal is to find WARC records that are related to Covid-19, which was still a common topic back in 2019.

3 Setting Up

To begin with, I picked a random WARC file (`CC-MAIN-20210410105831-20210410135831-00639`), and did my experiment on that. I copied it from the `redbad` container to my local PC, and then to the `big-data` container, where the Zeppelin file is running.

4 Implementation

My Zeppelin file includes explanation of each code block and what I tried to do at each step, but I will still shortly explain my implementation here.
First, I initialized the Spark configuration to process WARC files.

```scala
val sparkConf = new SparkConf()
    .setAppName("RUBigData WARC4Spark 2021")
    .set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")
    .registerKryoClasses(Array(classOf[WarcRecord]))
```

I created a textbox for the input WARC file and split it into WARC records.

```scala
val fname = z.textbox("Filename:")
val warcfile = s"file:///opt/hadoop/rubigdata/${fname}.warc.gz"

val warcs = sc.newAPIHadoopFile(
              warcfile,
              classOf[WarcGzInputFormat],
              classOf[NullWritable],
              classOf[WarcWritable]
    ).cache()
```

Then I filtered the the records to end up with the title, url and number of covid related words in their content for each record. Note that the words I'm looking for are 'covid', 'covid-19', 'corona', 'pandemic' and 'sars-cov'2'. I first filtered out the invalid and not response record. Then I extracted the url of the record by parsing the header. Next, I extracted HTTP body content which will be used to get the content and the title of the record. Using Jsoup parser, I parsed the body content, get the title, and count how many words are there related to the topic. Finally, I eliminated the records that have empty url or title, if there are any.

```
val filteredWarcs = warcs.map{ wr => wr._2 }.
                        filter{ _.isValid() }.
                        map{ _.getRecord() }.
                        filter{ _.getHeader().getHeaderValue("WARC-Type") == "
                            response" }.
                        map { wr => {
                            val url = wr.getHeader().getUrl()
                            (wr, url)
                            }
                        }.
                        map { wr => (wr._1.getHttpStringBody(), wr._2) }.
                        filter { _._1.length > 0 }.
                        map { wr => (Jsoup.parse(wr._1), wr._2) }.
                        map { wr => {
                                val title = wr._1.title().toLowerCase()
                                val url = wr._2
                                val content = wr._1.body().text().toLowerCase()
                                val covidCount = StringUtils.countMatches(
                                    content, "covid")
                                val covid19Count = StringUtils.countMatches(
                                    content, "covid-19")
                                val coronaCount = StringUtils.countMatches(
                                    content, "corona")
                                val pandemicCount = StringUtils.countMatches(
                                    content, "pandemic")
                                val sarsCount = StringUtils.countMatches(
                                    content, "sars-cov-2")
                                (title, url, covidCount+covid19Count+
                                    coronaCount+pandemicCount+sarsCount)
                            }
                        }.
                        filter{ wr => wr._1.length > 0 && wr._2.length > 0 }
```

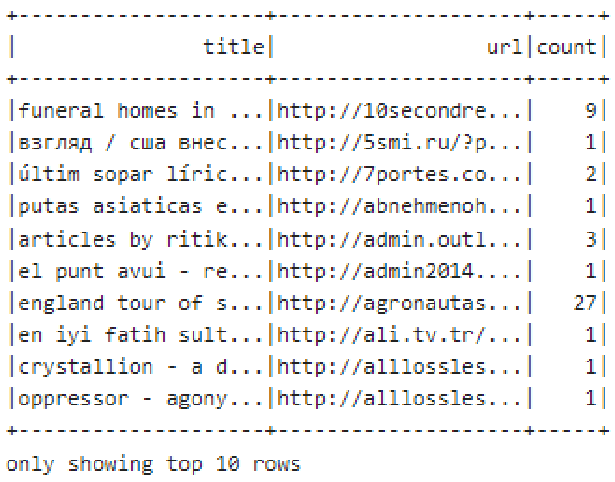
Now that I got the filtered WARC records, I can create a Data Frame to visualize the results.
I created an initial Data Frame, which will then be filtered using SQL queries.

```
val DF = filteredWarcs.toDF("title", "url", "count")
DF.createOrReplaceTempView("DataFrame")
DF.show(10)
```

Using the DF, I filtered it so that only websites related to Covid-19 are left in the DF using SQL. The keywords we are looking for are 'covid', 'covid-19', 'corona', 'pandemic' and 'sars-cov-2'. If a title or url contains one of those words, it will be in the final table. Also, I counted the amount of those words in the content before creating the DF, so if it's more than 0, it will also be in the final table.

```
val filteredDF = spark.sql(
    """
    SELECT * FROM DataFrame
    WHERE title LIKE '%covid%'
    OR title LIKE '%covid-19%'
    OR title LIKE '%corona%'
    OR title LIKE '%pandemic%'
    OR title LIKE '%sars-cov-2%'
    OR url LIKE '%covid%'
    OR url LIKE '%covid-19%'
    OR url LIKE '%corona%'
    OR url LIKE '%pandemic%'
    OR url LIKE '%sars-cov-2%'
    OR count > 0
    """)
filteredDF.createOrReplaceTempView("FilteredDataFrame")
filteredDF.show(10)
```

Now I can see the results table which I'm happy about. The goal of the assignment is accomplished successfully. Final table will show the records related to Covid-19 from a given WARC file.
An example output is given below.



```
+--------------------+--------------------+-----+
|               title|                 url|count|
+--------------------+--------------------+-----+
|funeral homes in ...|http://10secondre...|    9|
|взгляд / сша внес...|http://5smi.ru/?p...|    1|
|últim sopar líric...|http://7portes.co...|    2|
|putas asiaticas e...|http://abnehmenoh...|    1|
|articles by ritik...|http://admin.outl...|    3|
|el punt avui - re...|http://admin2014....|    1|
|england tour of s...|http://agronautas...|   27|
|en iyi fatih sult...|http://ali.tv.tr/...|    1|
|crystallion - a d...|http://alllossles...|    1|
|oppressor - agony...|http://alllossles...|    1|
+--------------------+--------------------+-----+
only showing top 10 rows
```

5 Running In The Cluster

Before submitting my code to the cluster, I need to do a few changes. First, I tested with unedited given `RUBigDataApp` following the instructions from Assignment 5. Instead of a interactive textbox for the input file, I entered the file name manually.

```
val warcfile = s"/opt/hadoop/rubigdata/CC-MAIN
    -20210410105831-20210410135831-00639.warc.gz"
```

In my notebook code, I show two versions of my DF, which is not needed here. Only showing the result table is enough. I used the same WARC file again to make is easier to check if it's working as I want it to be.
I followed the instructions from the fifth assignment after changing the Scala code with mine. I built the `build.sbt` file using `sbt assembly` and ran my code using `spark-submit target/scala-2.12/RUBigDataApp-assembly-1.0.jar`. I think that the way I done it is not exactly how it should be done. We were asked to use `hdfs` I guess. I tried that too. I explained it more detailed in the next section.

6 My Experience

My first plan was to analyse the effect of a certain event on a topic. For example, I believe the release of ChatGPT increased the general interest in terms like 'AI', 'machine learning' and things like that. I wanted to compare the frequency of such terms month by month from six months before the release of ChatGPT and six months after. I think this would've been a really good project, but because we only had access to 2021 crawl data and getting additional data from a different time frame is challenging, I decided to find a different project idea.
Since I have access to crawl from only 2021 April, I think I don't have the environment to make a comparison. After realizing that I can't accomplish my main goal, I decided to find a topic that I can accomplish by just pick a random WARC file and work on it and decided to filter the Covid-19 related records.

My successful implementation is explained in the implementation section, but I tried to do it differently at first. Instead of using SQL to filter titles and urls that containing keywords that I'm looking for, I tried doing it in the filter/map phase. I used `contains()` to do that, but each time I tried that, I faced a serialization error. I later noticed that a blog was included in the tutorial notebook for these type of errors, but I already completed my project and got the results I wanted using SQL.
After getting the table I wanted, I also tried to count the number of rows to check how many records were related to my topic, and wanted to find what percentage of total records are related to Covid-19. For this, I found how many total records are there, which was easy since it's also included in the tutorial notebook.

```
val nHTML = warcs.count()
```

The problem was finding how many records were related to Covid-19. I tried many different methods such as using the COUNT method of SQL, and just using the `count()` function on the filtered DF I have at the end.

```
val count = filteredDF.count()
println(s"Number of records: $count")
```

For some reason, both did not work for me. I also noticed that even if I count the rows manually, there aren't many rows after filtering the data. In the WARC file I worked on, there were only 16 records that are related and comparing that to total of 150181 records does not make sense.
After being done with my code and when finishing up this report, I noticed that maybe filtering for the word 'pandemic' was not a good idea. I forgot the fact that the crawl might include websites about historical events, and Covid-19 was not the only pandemic happened in the history. My thinking was since the records are from a crawl dated 2021 April, if the word 'pandemic' exist, it's probably related to Covid-19. So my final data might include some websites that are not related to Covid-19, but I was too lazy to fix it given that I struggled with finishing this project in time.
Last thing I needed to do is to run it in the cluster. First of all, I want to say that I successfully run my code in the `big-data` container. I copied the WARC file to the container and edited the Scala code to set everything up. Then I did the steps I explained in Running In The Cluster section to run it. One last addition I wanted to do was to instead of printing the results, I wanted to save it into a `csv` file, but it didn't save it, so I just printed the results just like in the notebook.

```
filteredDF.write.csv("/opt/hadoop/rubigdata/")
```

The problem is, I did all of it before noticing that it should've been done using `hdfs`, so I should've put the WARC file to `hdfs:///user/s1073728/`, and edit the Scala code so that it actually uses the input from that path. I'm not sure if it's a must for this project, but I would still like to explain the problem I faced with that.
First I copied the WARC file to the `redbad`. Then I tried using `hdfs dfs -put filename hdfs:///user/s1073728/`. Somehow, that didn't work. So I hope that it isn't a must for this project and running it in the cluster alone is enough.
That sums up what I tried, accomplished and failed, basically my general experience with the project. I also want to talk about what I learned by doing this project.
First of all, this project was a little challenging, but was a good experience for me. Unlike the previous assignments, I actually got to work on this project without many instructions, which is a good practice. I always like to do hands-on practice on concepts I'm trying to learn instead of just studying the theory. With this assignment, I learned the WARC files structure, which has several WARC records that contains a header and content. I parsed the header to get the link and the title of a website, and counted the

words I’m looking for from its content, so I can say it helped me understand what can be done with WARC records. I also built a stand-alone application using my code. How to run such Scala code was taught to us before, but doing it with my code was satisfying. To sum up, this project allowed me to combine everything I learnt during the course and build a final project using those, so I’m happy and hoping that I successfully managed to do that.