Big Data Project – Report

Laurian Duma (s1091563)
June 2024

Problem Foundation and Motivation

Objective

The main objective of this project is to apply data analysis techniques, namely word count and frequency analysis, to investigate online content from different web pages. The focus is on terms associated with **environmental issues** such as **climate change**, **pollution**, and **renewable energy**. Hence, the main goal of this project is quite simple: identify, count, and analyze how frequent these terms appear online. This analysis has potential applications in several areas, such as enhancing public engagement strategies for environmental advocacy groups, aiding policymakers in understanding public concerns about environmental issues, and assisting marketers in crafting more effective Search Engine Optimization strategies.

Personal Motivation

My interest in environmental issues has been the inspiration for this project. As someone deeply concerned about sustainability and ecological well-being, I was curious to discover which terms related to the environment are most prevalent online. To conduct this analysis, I created a list of keywords commonly associated with sustainability and environmental issues, which served as the basis for my study. These keywords include: **Climate Change**, **Global Warming**, **Renewable Energy**, **Pollution**, **Sustainability**, **Carbon Emissions**, **Greenhouse gases**, **Solar Power**, **Wind Power**, **Fossil Power**, **Climate Action** and **Recycling**.

It is important to note that this list of terms is not exhaustive, and initially, I considered adding more words to expand its scope. However, given the significant computational impact that this specific aspect might have, I made the decision to limit the size of this list.

Process

As instructed in the guidelines of this project description, I adopted a divide-and-conquer approach to manage the complexity of analyzing WARC files. Following the recommendations, I began with a modest scope, utilizing the example provided in the Zeppelin notebook. Initially, I downloaded web content from three specific websites into designated WARC files for subsequent analysis, namely **env1.warc.gz**, **env2.warc.gz** and **env1.warc.gz**. To achieve this, I employed the **wget** command, which proved to be both efficient and swift. In addition, I followed several cells that were present in the

example notebook, as they proved helpful in my subsequent analysis. For example, the code snipped from below, allowed me to load WARC files into Apache Spark for processing.

```
val warcs = sc.newAPIHadoopFile(
  warcFilePath.mkString(","),
  classOf[WarcGzInputFormat],
  classOf[NullWritable],
  classOf[WarcWritable]
)
```

After conducting an initial analysis on 3 WARC files, I developed a thorough understanding of their data structures and inherent limitations. This knowledge enabled me to think about a strategic approach for parsing HTML content. I noticed a consistent pattern: environmentally-related terms were primarily located within the bodies of web pages. Consequently, I adjusted my focus to parse only this section, ignoring the other parts of the HTML content. With this refined approach in mind, I then evaluated the effectiveness of using Spark and Jsoup for data parsing. This phase was crucial for understanding the HTML structure, extracting relevant body content and implementing straightforward keyword matching techniques to quantify the prevalence of environmental terms. One of the challenges was ensuring that entries with null URLs or HTML bodies were filtered out. These entries are non-informative for the subsequent analysis and could potentially skew the results if included (in case a graphical representation of the results is employed).

An integral phase of this project was the formulation of a regular expression pattern designed to capture the environmental terms defined in the variable "**environmentalKeywordsPattern**". Employing this regex pattern enabled the precise scanning of webpage body contents and helped computing individual counts of these specific terms. This process served as a base to later analyze the relevance of a webpage to environmental topics. It operates under the premise that web pages with a higher frequency of terms like "**climate change**" or "**renewable energy**" are more significantly engaged with environmental matters. Further thoughts on the extent to which this premise applies, will be discussed in the final section of this report.

```
        val environmentalKeywordsPattern = "\\b(([Cc]limate [Cc]hange)|([Gg]lobal [Ww]arming)|
([Pp]ollution)|([Rr]enewable [Ee]nergy)|([Ss]ustainability)|([Rr]ecycling)|([Cc]limate [Aa]ction)|
([Ee]co-friendly)|([Cc]arbon [Ee]missions)|([Gg]reenhouse [Gg]ases)|([Ss]olar [Pp]ower)|([Ww]ind
[Pp]ower)|([Ff]ossil [Pp]ower))\\b".r
```

Following the completion of the environmental related terms counting, the next objective was to rank webpages according to their relevance to environmental issues and not only. In order to sort the web pages accordingly, I used SparkSQL, which enhanced the sorting operation and also reduced the runtime, thus boosting the overall performance of the program.

In the analysis, I transformed the processed data into a DataFrame using Spark's DataFrame API, which is more optimized and expressive for handling big data compared to traditional RDDs. This transformation streamlined the management of data and enhanced performance during analytical computations. Once converted into a DataFrame, the data was registered as a temporary view, enabling the use of advanced querying capabilities through Spark SQL. In this way, I was able to count and analyze the occurrences of specific environmental terms, as well as consider their proportions relative to the overall content length of each webpage

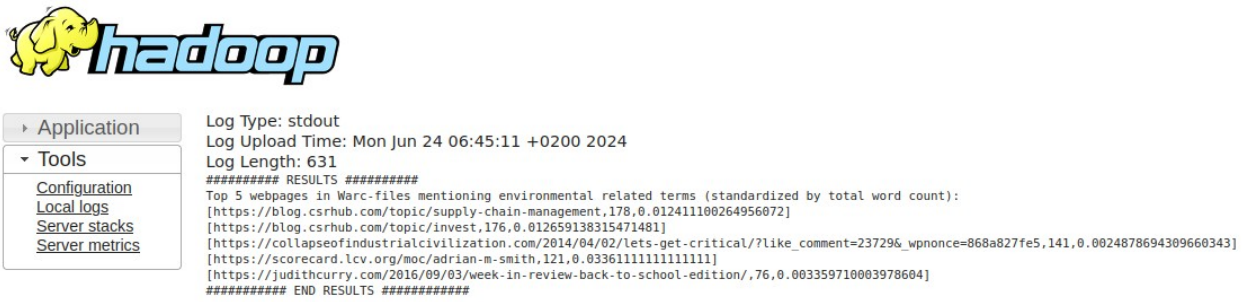
```
val df = environmentalImpactAnalysis.toDF("url", "pattern_count", "standardized_wordcount")
df.createOrReplaceTempView("df")

val results = spark.sql("SELECT url, pattern_count, standardized_wordcount FROM df ORDER BY pattern_count DESC LIMIT 5").collect()
```

Scaling up to multiple WARC files

After achieving successful results with three WARC files in the Zeppelin notebook, the next step was to scale the project to handle multiple files on the cluster. Utilizing external libraries like Jsoup for HTML parsing required packaging the application and its dependencies into a single assembly jar. This was accomplished using the **sbt assembly** command, which compiled everything into a portable jar file. The jar was then deployed to the cluster using the **spark-submit --deploy-mode cluster --queue default target/scala-2.12/RUBigDataApp-assembly-1.0.jar** command. This ensured the application ran across multiple nodes, taking advantage of the cluster's capabilities to manage the increased workload effectively. This whole process was not as straightforward as I initially expected, because the Jsoup depenencies had to be included in the **build.sbt** file**.** However, the second part of this process has been quite easy as the necessary commands for creating and deploying the jar had been previously outlined in the project documentation.

Analysis of results

The project successfully processed 3 WARC files from the segment, to identify and rank web pages based on the frequency of environmentally-related terms. The results of the analysis can be observed in the figure below, which presents the top five web pages with the highest frequency of environmentally-related terms.



In analyzing the environmental term frequencies across webpages, an interesting pattern emerges that highlights the nuanced difference between absolute and relative frequencies of keyword occurrences. While the first-ranked webpage shows the highest absolute count of environmental terms, it is the third-ranked webpage that presents a greater proportion of these terms relative to its overall word count. This observation suggests that although the first webpage contains a larger number of environmental terms, the third webpage dedicates a higher percentage of its content to these topics, indicating a more focused and possibly more intense discussion on the environmental topic.

I am confident that with more time to run the program across additional WARC files from the cluster, or perhaps the entire segment, the results would have been more consistent. This expanded analysis could potentially reveal other significant web pages that might qualify for the top 5 ranking. An extension would not only validate the findings with a broader dataset, but could also uncover a wider range of key players in environmental discourse online.

Other optimization

The design and implementation of this project involved significant optimizations to ensure efficient processing of large-scale web archive data using Apache Spark. In this section, I will talk about some of them, briefly:

1. **Memory Management:** By setting spark.driver.memory to "3g" and spark.executor.memory to "5g", the application was equipped with sufficient memory to handle the intensive operations involved in parsing and analyzing WARC files. These settings ensure that both the driver and executors have adequate memory allocated, reducing the likelihood of memory overflows which are common in data-intensive applications. Fortunately, I remembered from the lectures that this is an important aspect to consider when dealing with out-of-memory issues in clusters and some TAs have been kind enough to help me with this.
2. **OffHeap Memory Utilization:** Enabling spark.memory.offHeap.enabled and setting spark.memory.offHeap.size to "4g" allowed the application to use off-heap memory which has the incredible advantage of better dealing with large datasets as it reduces garbage collection overhead and improves performance. This concept has

been presented in the lecture where we discussed about Datasets and Dataframes in Spark.

3. **Serialization with Kryo:** The use of Kryo serialization (spark.serializer set to "org.apache.spark.serializer.KryoSerializer") optimized the serialization process. Kryo is faster and more compact than Java serialization, which is beneficial for Spark applications that require serializing large amounts of data for shuffling and caching. As mentioned earlier, several concepts have been used from the example Zeppelin notebook and this is one of them, as well.

Reflection and Further Improvements

The project efficiently analyzed 3 WARC files from the segment. The objective was to identify and rank web pages based on the prevalence of environmentally-related terms. As mentioned above, the list of terms was selected from commonly used environmental terminology.

This analysis has yielded several insightful observations. However, as mentioned above, the top results might not align with expectations for pages with the highest frequency of environmentally-related terms. This discrepancy could be due to the use of such terms in contexts unrelated to direct environmental advocacy or discussion. To cope with this, in a future analysis, I would incorporate Natural Language Processing techniques to better deal with the context of terms usage. This should allow for a more targeted identification of web pages that are genuinely focused on environmental issues, rather than merely featuring high occurrences of related terms.

Memory management was crucial throughout this project. The initial difficulties highlighted the importance of effective resource allocation for big data processing. Optimizing memory allocation for Spark’s driver and executor, enabling offHeap memory, and utilizing Kryo serialization significantly enhanced the data processing capabilities.

Another limitation noted was the project’s scope to process only 3 WARC files, due to the time constraints I encountered. With more time and further development of the project, this limitation could potentially be expanded to include a larger dataset, which would allow for more comprehensive and representative results.

Notes for the reader

Please see the results of the analysis described in this report at link.