

High-Performance Computational Infrastructures

CS5810

Shivangi Dubey

2039934

2021-2022

INTRODUCTION

Hadoop is a framework used for large structured and unstructured datasets. It is scalable, open-source, and runs on commodity hardware and the cloud.

Features that make it different are fault tolerance and multiple libraries and methodologies for extensive data analysis.

MapReduce is also used for data that needs repetitive analysis. It uses a unique structure of mappers and reducers. Mappers produce key-value pairs from the raw data and feed them to the reducers. There is a 'shuffle and sort' operation between mappers and reducers.

Other applications supported by Hadoop for computation are

- Apache Pig
- Apache Hive
- Apache HBase
- Apache Spark

This is not an exhaustive list and can be expanded to include many others.

PROBLEM DESCRIPTION

An over-involvement in social media platforms during the covid times has caused the dissemination of fake news, wrong information, a vibe of uncertainty and negativity. Sentiment analysis can facilitate a better understanding of the situation and a more practical approach to handling delicate situations. (Lin and Moh, 2021)

An attempt has been made to generate word count from the tweets data during covid times, which could further be used for sentiment analysis and consensus study during challenging times.(Rustam et al., 2021)

ASSOCIATED DATASET

The dataset (Aman Miglani, 2020)used is that of tweets collected between the dates 16/03/2020 and 14/04/2020 (inclusive).

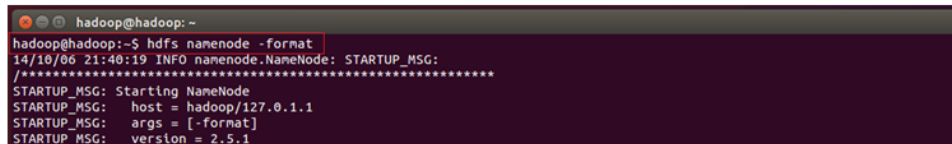
The training file has 12 columns and 41157 rows. The size of the dataset is 5MB.

DESIGN & IMPLEMENTATION

Hadoop's HDFS (Hadoop File Distribution System) is used in conjunction with the MapReduce algorithm to receive the file with the data and henceforth execute the algorithm to get the word count from the tweet data.

The steps involved are:

- Opening the terminal on the Oracle Virtualbox.
- Typing the following commands to start Hadoop



Now, start the Hadoop:

```
hadoop@hadoop-VirtualBox:~$ start-dfs.sh
hadoop@hadoop-VirtualBox:~$ start-yarn.sh
```

To check whether your hadoop servers is on,

```
hadoop@hadoop-VirtualBox:~$ jps
```

You should see something like (numbers might be different):

```
2819 ResourceManager
3002 NodeManager
1938 DataNode
2085 SecondaryNameNode
2349 Jps
1788 NameNode
```

If you cannot see these you can try:

1) Stop hadoop services

```
hadoop@hadoop-VirtualBox:~$ stop-dfs.sh
hadoop@hadoop-VirtualBox:~$ stop-yarn.sh
```

2) Remove the contents in the hadoop tmp directory

```
hadoop@hadoop-VirtualBox:~$ rm -rf /tmp/hadoop-hadoop/*
```

3) Repeat the steps above to remove previous data from Hadoop file systems and to start hadoop

```
hadoop@hadoop-VirtualBox:~$ hdfs namenode -format
hadoop@hadoop-VirtualBox:~$ start-dfs.sh
hadoop@hadoop-VirtualBox:~$ start-yarn.sh
```

Make the HDFS directories required to execute MapReduce jobs:

```
hadoop@hadoop-VirtualBox:~$ hdfs dfs -mkdir /input
```

The steps enumerated above are followed:

- The file from the Kaggle website is a .csv file, and the tweets column is first copied to a .txt file named tweets_corona.txt.
- That file is transferred from the downloads folder on our local system to hdfs by the command:
- **hdfs -put Downloads/tweets_corona.txt /input**
- Here the /input directory has already been created in the previous commands.
- A jar file is created with the following files.

- Word Count Mapper (Mapper class)

```
package org.myorg;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Mapper.Context;

public class WordCountMapper
    extends Mapper<LongWritable, Text, Text, IntWritable>
{
    private static final IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text,
        throws IOException, InterruptedException
    {
        String line = value.toString();
        line = line.replaceAll("&.*?\w+;", " ")
            .replaceAll("[^a-zA-Z0-9 ]", " ")
            .replaceAll("\s+", " ");
        if ((line != null) && (!line.trim().isEmpty()))
        {
            String[] words = line.split(" ");
            String[] arrayOfString1;
            int j = (arrayOfString1 = words).length;

            for (int i = 0; i < j; i++)
            {
                String word = arrayOfString1[i];
                context.write(new Text(word), one);
            }
        }
    }
}
```

- The objects 'word' and 'one' are created from Hadoop related classes "Text" and "IntWritable".
- The context object created allows the mapper and reducer to interact with the Hadoop system.
- Also, the map method that takes the context object as an argument takes three arguments.
- @Override is used to change the actual behaviour of the "map" method present initially in the parent class Mapper.
- Creating a variable line of a data type String.
- The toString and replaceAll tries to convert the text into string and then eliminate all special characters from the tweet data.
- The for loop goes through all the words in the input file.
- Emit key-value pair for each word.
- Adding the key-value pair to the context object and then emitting it.

Word Count Reducer

```
package org.myorg;

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class WordCountReducer extends Reducer<Text, IntWritable, Text, IntWritable>
{
    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException
    {
        int sum = 0;
        for (IntWritable value : values)
        {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }
}
```

- The first step is to import all the classes and methods needed for reducer class.
- A subclass called Wordcount Reducer from a parent class Reducer is created using the keywords */.
- The recurrent data types of Text and IntWritables are for input key and value pairs and output key and value pairs.
- @Override is used here to change the original behavior of the method “reduce” in the parent class “Reducer”
- The reduce method takes three parameters just like the map method in the Mapper class.
- The sum of all the values is stored in the sum variable.
- IntWritable is a list that counts each word.
- The for loop goes through all the values to generate a count.
- the value of key and value is then written to the context object and then emitted.

```

package org.myorg;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.fs.FileSystem;

public class WordCount
{
    public static void main(String[] args) throws Exception
    {
        Configuration conf = new Configuration();
        if (args.length != 3)
        {
            System.err.println("Usage: WordCount<input path><output path>");
            System.exit(-1);
        }

        Job job;
        job=Job.getInstance(conf, "Word Count");
        job.setJarByClass(WordCount.class);

        FileInputFormat.addInputPath(job, new Path(args[1]));
        FileOutputFormat.setOutputPath(job, new Path(args[2]));

        job.setMapperClass(WordCountMapper.class);
        job.setReducerClass(WordCountReducer.class);
        job.setCombinerClass(WordCountReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        // Delete output if exists
        FileSystem hdfs = FileSystem.get(conf);
        Path outputDir = new Path(args[2]);
        if (hdfs.exists(outputDir))

            hdfs.delete(outputDir, true);

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

- The 'conf' object from the configuration class provides access to configuration class. (it is necessary for Hadoop job)
- The if statement in the driver class checks if the hadoop jar command has exactly 3 parameters.
- The 3 parameters ideally to be given are:
 - name of the Driver class
 - input path of the data file
 - output path for storing the results.
- an object of class Job represents a task that needs to be executed by Hadoop.
- Job class in MapReduce API allows user to configure the job, submit it, control its execution and query its state.

- The command following the job class object instantiation are for configuring parameters for the created job.
- The driver class is then defined in the jar file.
- Input and outputs paths are defined for the job using the argument 'new Path (args[1])'
- The next 3 commands define the mapper, combiner and reducer classes for the Job.

job.setOutputKeyClass(**Text**.class);

job.setOutputValueClass(**IntWritable**.class);

- The above two commands are for setting the data types of the input and output key-value pairs. (IntWritable for the value and String for the key)
- We also make sure to delete any pre-existing output folder in the HDFS.
- At last we check the status of the job to ensure its completion.

- The file with the above classes is exported as a .jar file from the editor. And to run the program following command is used
`hadoop jar Downloads/Wordcount.jar WordCount /input/tweets_corona.txt output`
- To copy the output files from the distributed filesystem to the local filesystem.
`hdfs dfs -get output output`
- To get the output, the following path can be inspected.
`hadoop@hadoop-VirtualBox:~$ cd output/output`
and opening the file (part-r-00000)
`hadoop@hadoop-VirtualBox:~/output/output$ cat part-r-00000`
- The output is discussed in the result section.

RESULTS

The result is represented in the screenshot below, and the file is enclosed in the zip file.

AMERICA	1	
AMJoy	1	
AMWRDbLDEX		1
AMZN	4	
AN	3	
ANALERTS		1
AND	26	
ANDUNCONSCIONABLE		1
ANI	1	
ANIMALS	1	
ANNOUNCEMENT		1
ANXIETY	1	
ANY	4	
ANYONE	3	
AOC	1	
A0d5IJUNvJ		1
AP	1	
APPROVE	1	
APS	1	
AR	2	
ARE	6	
ARRESTED		1
ARTSY20	1	
AS	2	
ASAIN	1	
ASANAS	1	
ASAP	5	
ASDA	1	
ASIA	1	
ASPCA	1	
ASSES	1	
ASX	1	
AT	8	
ATBDudley		1
ATM	1	
ATT	1	
ATTN	1	
AUDACITY		1
AWESOME	2	

A second step was executed with the location column of the .csv file, and a screenshot of the result is:

Athens	1	
Athlone	1	
Atlanta	20	
Atlantis		1
Au	1	
Auckland		2
Aurora	1	
Aust	1	
Austell	1	
Austin	17	
Australia		36
Austria	4	
Aviv	2	
Awakabal		1
Aylesbury		1
Ayr	1	
Ayrshire	4	1
BA	1	
BC	7	
BEDS	1	
BNA	1	
BOS	1	
Baghdad	1	
Bahrain	1	
Baker	1	
Bakersfield		1
Baku	1	
Balamb	1	
Ballarat		1
Ballybough		1
Baltimore		7
Bampton	1	
Bangalore		3
Bangladesh		1
Bangor	2	
Bank	1	
Barbara	1	
Barcelona		4
Barnstaple		1
Base	4	
Based	2	
Basilicata		1
Baton	1	

EVALUATION

The screenshot file shows a few meaningful words that can be a solid indicator of sentiment, and their frequency can determine the strength of that emotion within the society. There are, however, mostly garbage words that appear due to the website addresses included in the tweets. They can be ignored that their frequency does not give us an insight into anything.

The second step tells us the location frequency of the tweets to determine which parts of the world are engaging in social media and combined with the previous results, can give us a picture of the dynamics of the emotions during tough times.

REFERENCES

- Aman Miglani, 2020. <https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification?resource=download> [WWW Document].
- Lin, H.Y., Moh, T.S., 2021. Sentiment analysis on COVID tweets using COVID-Twitter-BERT with auxiliary sentence approach, in: Proceedings of the 2021 ACMSE Conference - ACMSE 2021: The Annual ACM Southeast Conference. <https://doi.org/10.1145/3409334.3452074>
- Rustam, F., Khalid, M., Aslam, W., Rupapara, V., Mehmood, A., Choi, G.S., 2021. A performance comparison of supervised machine learning models for Covid-19 tweets sentiment analysis. PLoS ONE 16. <https://doi.org/10.1371/journal.pone.0245909>
- White, T. (2015). *Hadoop: the definitive guide; storage and analysis at internet scale*. Beijing O'Reilly Media.
- Lam, C. (2011). *Hadoop in action*. Greenwich, Conn.: Manning Publications.