LAB EXERCISE 4

Aim: Stop word elimination problem:

Input: A large textual file containing one sentence per line. A small file containing a set of stop words (One stop word per line)

Output: A textual file containing the same sentences of the large input file without the words appearing in the small file.

Theory:

Stop Words:

In computing, stop words are words which are filtered out before or after processing of natural language data (text). Though "stop words" usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even use such a list. Some tools specifically avoid removing these stop words to support phrase search.

Any group of words can be chosen as the stop words for a given purpose. For some search engines, these are some of the most common, short function words, such as the, is, at, which, and on. In this case, stop words can cause problems when searching for phrases that include them, particularly in names such as "The Who", "The", or "Take That".

Procedure:

Step1) Create two file one named "largefile.txt" in which we have our sentences, and second is named "stopword.txt" in which we have list of all stop words. And upload them on Hadoop HDFS using command,

>>> hadoop fs -put "input file path" /input

Step 2) Open IntelliJ and add required dependencies in "pom.xml" file.

```
<dependencies>
        <dependency>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>hadoop-common</artifactId>
                <version>3.3.3</version>
        </dependency>
        <dependency>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>hadoop-mapreduce-client-core</artifactId>
                <version>3.3.3</version>
        </dependency>
        <dependency>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>hadoop-hdfs</artifactId>
                <version>3.3.3</version>
        </dependency>
</dependencies>
```

Step 3) After adding dependencies create 3 java files named "StopWordMapper.java", "StopWordReducer.java" and "StopWordElimination.java" and there code as mentioned below,

For StopWordMapper.java

```java
package org.priyanshu;

importorg.apache.hadoop.io.LongWrita;
import org.apache.hadoop.io.Text;
import
org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.URI;
import java.util.HashSet;

public class StopWordMapper extends Mapper<LongWritable, Text, NullWritable, Text>

  { private final HashSet<String> stopWords = new HashSet<>();

  @Override   protected void setup(Context context) throws java.io.IOException
    { URI[] stopWordsFiles = context.getCacheFiles();
    if (stopWordsFiles != null && stopWordsFiles.length >
      0) { Path path = new Path(stopWordsFiles[0]);
      FileSystem fs = FileSystem.get(context.getConfiguration());

      // Use try-with-resources to ensure resources are closed
      try (FSDataInputStream fis = fs.open(path);
        BufferedReader reader = new BufferedReader(new InputStreamReader(fis))) {

        String word;
        while ((word = reader.readLine()) != null) {
          stopWords.add(word.trim().toLowerCase());
        }

      }

    }

  }

  @Override
  public void map(LongWritable key, Text value, Context context) throws java.io.IOException, InterruptedException
    { String[] words = value.toString().split("\\s+");
    StringBuilder filteredSentence = new StringBuilder();

    for (String word : words) {
      if
        (!stopWords.contains(word.toLowerCase()
        )) {
        filteredSentence.append(word).append(" ");
      }

    }

    context.write(NullWritable.get(), new Text(filteredSentence.toString().trim()));
  }

}
```

For StopWordReducer.java

```java
package org.priyanshu;

import org.apache.hadoop.io.*;
import
org.apache.hadoop.mapreduce.*;
import java.io.IOException;


public class StopWordReducer extends Reducer<NullWritable, Text, NullWritable, Text> {
    @Override   protected void reduce(NullWritable key, Iterable<Text> values, Context context) throws
IOException,  InterruptedException {
        for (Text sentence : values) {
            context.write(NullWritable.get(), sentence);
        }

    }

}
```

For StopWordElimination.java

```java
package org.priyanshu;

import
org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import
org.apache.hadoop.io.NullWritable;
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 java.net.URI;

public class StopWordElimination {

    public static void main(String[] args) throws
        Exception {  if (args.length < 3) {
            System.err.println("Usage: StopWordElimination <input path> <output path> <stop words
            file>");  System.exit(-1);
        }

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Stop Word Elimination");

        job.setJarByClass(StopWordElimination.class);
        job.setMapperClass(StopWordMapper.class);
        job.setReducerClass(StopWordReducer.class);

        job.setOutputKeyClass(NullWritable.clas
        s);  job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, new
        Path(args[0]));  FileOutputFormat.setOutputPath(job,
        new Path(args[1]));

        // Add stop words file to the distributed cache
        job.addCacheFile(new URI(args[2]));

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

}
```
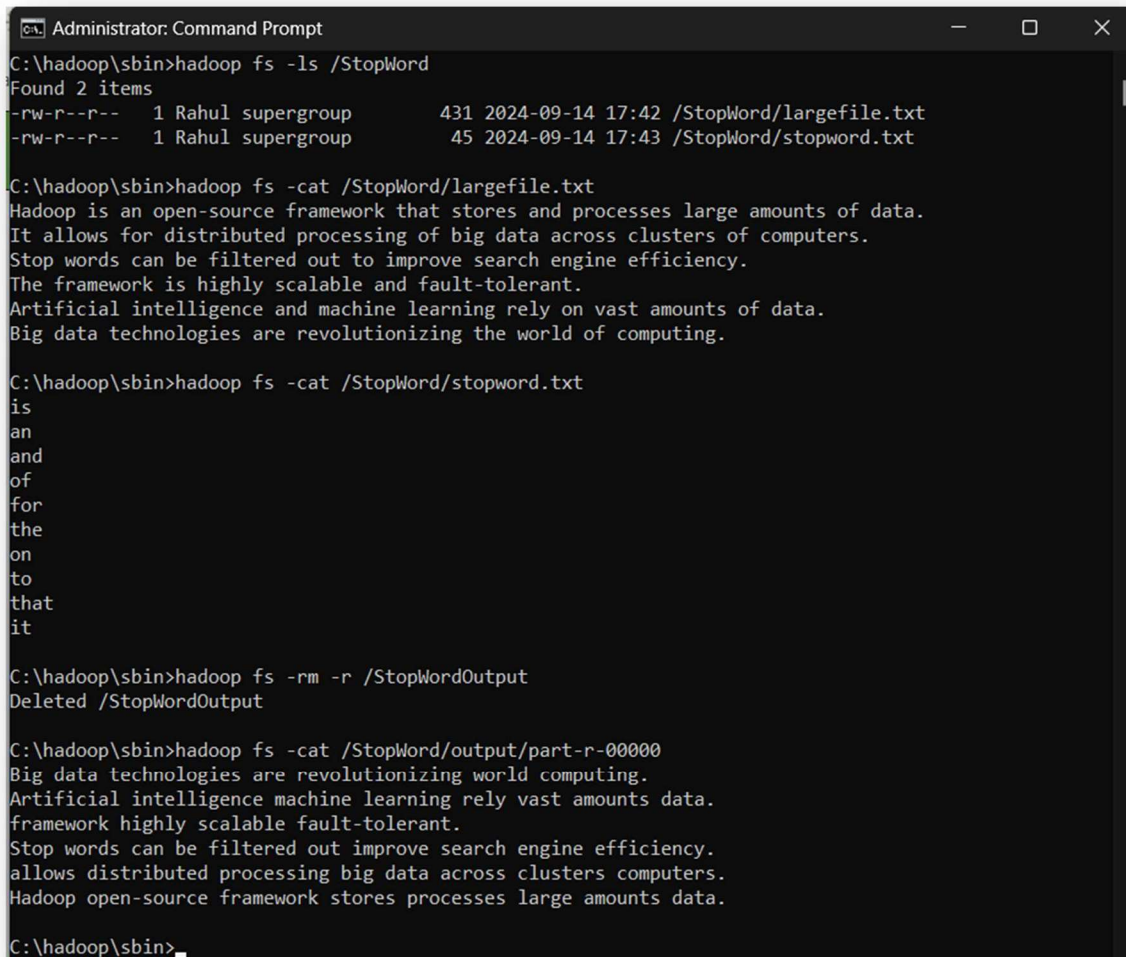
Step 4) After making above 3 java file open maven terminal and first clean it and then install, it will generate jar file needed to run the StopWordelimination.

>>> hadoop jar target/StopWord-1.0-SNAPSHOT.jar org.example.StopWordElimination "hdfs large file path" /

output "hdfs stop word file path"

Output:

```
Administrator: Command Prompt                                          —    □    ×

C:\hadoop\sbin>hadoop fs -ls /StopWord
Found 2 items
-rw-r--r--   1 Rahul supergroup        431 2024-09-14 17:42 /StopWord/largefile.txt
-rw-r--r--   1 Rahul supergroup         45 2024-09-14 17:43 /StopWord/stopword.txt

C:\hadoop\sbin>hadoop fs -cat /StopWord/largefile.txt
Hadoop is an open-source framework that stores and processes large amounts of data.
It allows for distributed processing of big data across clusters of computers.
Stop words can be filtered out to improve search engine efficiency.
The framework is highly scalable and fault-tolerant.
Artificial intelligence and machine learning rely on vast amounts of data.
Big data technologies are revolutionizing the world of computing.

C:\hadoop\sbin>hadoop fs -cat /StopWord/stopword.txt
is
an
and
of
for
the
on
to
that
it

C:\hadoop\sbin>hadoop fs -rm -r /StopWordOutput
Deleted /StopWordOutput

C:\hadoop\sbin>hadoop fs -cat /StopWord/output/part-r-00000
Big data technologies are revolutionizing world computing.
Artificial intelligence machine learning rely vast amounts data.
framework highly scalable fault-tolerant.
Stop words can be filtered out improve search engine efficiency.
allows distributed processing big data across clusters computers.
Hadoop open-source framework stores processes large amounts data.

C:\hadoop\sbin>
```

Conclusion:

In this, we learnt about map-reduce working and how to write code in Hadoop and java and how to read and write file from HDFS. Also learnt about working of stop word and how to perform java code in Hadoop to remove stop word from a file containing large data.