Big Data – Hadoop Assignment

Name: Abhishek Dubey

Student ID: D20123718

Full time: MSc Data Science 2020-21

Class: TU59 Full Time

Date: 01/03/2021

Dataset Overview

We are using 3 languages books from Project Gutenberg (http://www.gutenberg.org/). Total books we are using in this Assignment is 6.

2 books are chosen using English Language.

2 books are chosen from French Language.

2 books are chosen from Spanish Language.

All books are random, we are suing them to identify the Average number of alphabets counts across the languages.

I have downloaded all files books in .txt format and will load them in hdfs system.

Loading files in HDFS

First, we need to start the Hadoop environment.

Step 1: Check Hadoop Version - *hadoop version*

Step 2: Start HDFS Daemon - *start-dfs.sh*

Step 3: Start Yarn Daemons – *start-yarn.sh*

```
soc@soc-VirtualBox:~$ start-dfs.sh
21/02/26 02:40:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/soc/yarn/hadoop-2.2.0/logs/hadoop-soc-namenode-soc-VirtualBox.out
localhost: starting datanode, logging to /home/soc/yarn/hadoop-2.2.0/logs/hadoop-soc-datanode-soc-VirtualBox.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/soc/yarn/hadoop-2.2.0/logs/hadoop-soc-secondarynamenode-soc-VirtualBox.out
21/02/26 02:40:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/soc/yarn/hadoop-2.2.0/logs/yarn-soc-resourcemanager-soc-VirtualBox.out
localhost: starting nodemanager, logging to /home/soc/yarn/hadoop-2.2.0/logs/yarn-soc-nodemanager-soc-VirtualBox.out
```

Last check what Daemons are running by using jps command. - *jps*

```
soc@soc-VirtualBox:~$ jps
9430 Jps
8916 ResourceManager
8264 NameNode
8746 SecondaryNameNode
8478 DataNode
9137 NodeManager
```

Always remember to close the dfs and yarn after use, it's a best practice so that it will not utilize system resources further.

```
soc@soc-VirtualBox:~$ stop-dfs.sh
21/02/26 02:31:52 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Stopping namenodes on [localhost]
localhost: stopping namenode
localhost: stopping datanode
Stopping secondary namenodes [0.0.0.0]
0.0.0.0: stopping secondarynamenode
21/02/26 02:32:20 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ stop-yarn.sh
stopping yarn daemons
stopping resourcemanager
localhost: stopping nodemanager
no proxyserver to stop
```

Create directory in hdfs

To upload all the files in single directory I have created the input directory with name 'bookinput'.

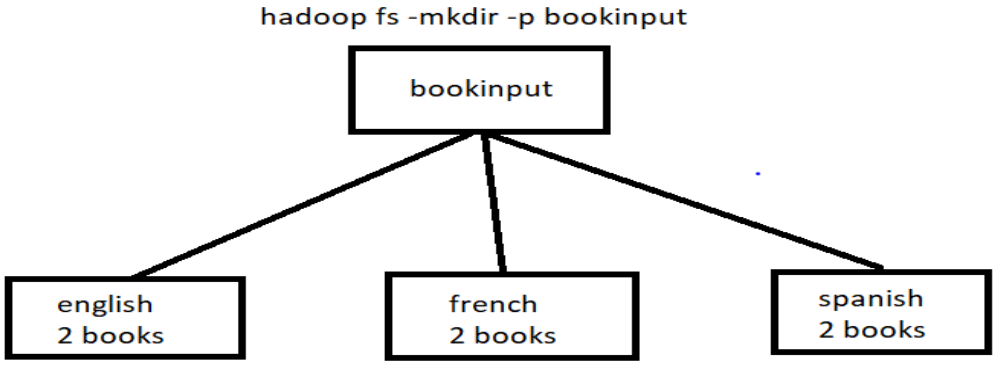
Bookinput will contain 3 different folders specific to our language. All my 6 files from 3 languages are taken as input from these sub directories.

hadoop fs -mkdir -p bookinput

hadoop fs -mkdir -p bookinput/english

hadoop fs -mkdir -p bookinput/french

hadoop fs -mkdir -p bookinput/spanish



Putting all my languages books .txt files in the directory bookinput

Example:

Hadoop fs -put /home/soc/share/englishbook1.txt bookinput/english/engbook1.txt

```
soc@soc-VirtualBox:~$ hadoop fs -put /home/soc/share/Englishbook1.txt bookinput/english/engbook1
21/02/27 21:54:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ hadoop fs -put /home/soc/share/Englishbook2.txt bookinput/english/engbook2
21/02/27 21:55:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ hadoop fs -put /home/soc/share/frenchbook1.txt bookinput/french/frenchbook1
21/02/27 21:55:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ hadoop fs -put /home/soc/share/frenchbook1.txt bookinput/french/frenchbook2
21/02/27 21:56:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ hadoop fs -put /home/soc/share/spanishbook1.txt bookinput/french/spanishbook1
21/02/27 21:56:57 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ hadoop fs -put /home/soc/share/spanishbook2.txt bookinput/french/spanishbook2
21/02/27 21:57:42 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
soc@soc-VirtualBox:~$ hadoop fs -rm -r bookinput/french/frenchbook2
21/02/27 21:58:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
21/02/27 21:58:26 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted bookinput/french/frenchbook2
soc@soc-VirtualBox:~$ hadoop fs -put /home/soc/share/frenchbook2.txt bookinput/french/frenchbook2
21/02/27 21:58:39 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

http://localhost:50070/

Goto : /user/soc/bookinput go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
english	dir				2021-02-27 21:55	rwxr-xr-x	soc	supergroup
french	dir				2021-02-27 21:58	rwxr-xr-x	soc	supergroup
spanish	dir				2021-02-27 21:28	rwxr-xr-x	soc	supergroup

Go back to DFS home

Local logs

Goto : /user/soc/bookinput/spanish go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
spanishbook1	file	301.62 KB	3	128 MB	2021-02-27 22:01	rw-r--r--	soc	supergroup
spanishbook2	file	206.42 KB	3	128 MB	2021-02-27 22:02	rw-r--r--	soc	supergroup

Go back to DFS home

Design of Map Reduce Process

Hadoop MapReduce is a framework used for processing large amount of data.

Instead of sending data to where the program or logic resides, MapReduce executes the logic on the server where the data already resides, to speed up processing.

Working:

MapReduce are two functions Map and Reduce. They are sequenced one after the other.

The Map function takes input from the disk as <key,value> pairs, processes them, and generates as output another set of <key,value> intermediate pairs.

The Reduce function often takes inputs as pairs of <key,value> and generates pairs of <key,value> as the output.

<k1, v1> -> Map() -> list(<k2, v2>)
<k2, list(v2)> -> Reduce() -> list(<k3, v3>)

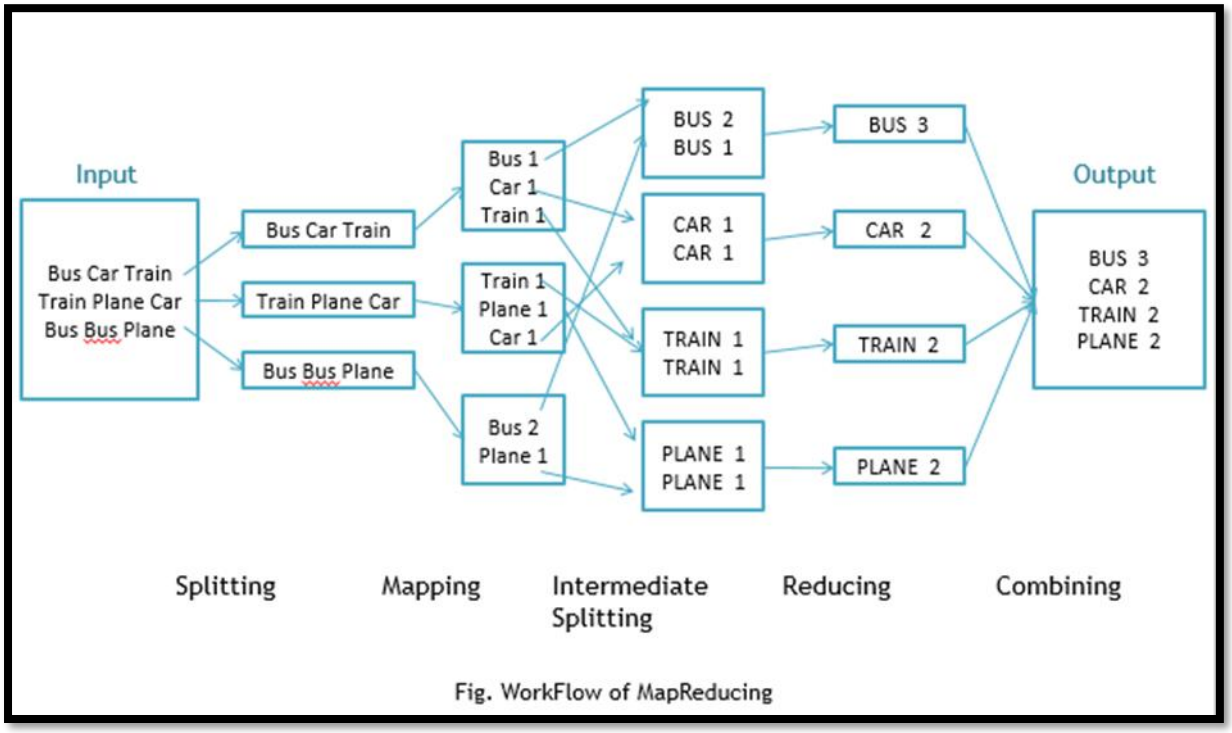


Fig. WorkFlow of MapReducing

Citation: https://dzone.com/articles/word-count-hello-word-program-in-mapreduce

Map Function

The data from the input is first split into smaller blocks. For processing, each block is then allocated to a mapper.

Reduce Function

The framework shuffles and sorts the results after all the mappers stop processing before moving them on to the reducers. While a mapper is still in progress, a reducer will not begin. A single reducer is allocated to all the map output values that have the same key, and then aggregates the values for that key.

Combiner

The method is optional. The combiner is a reducer on each mapper server that runs independently. Before moving it downstream, it lowers the data on each mapper further to a simpler form.

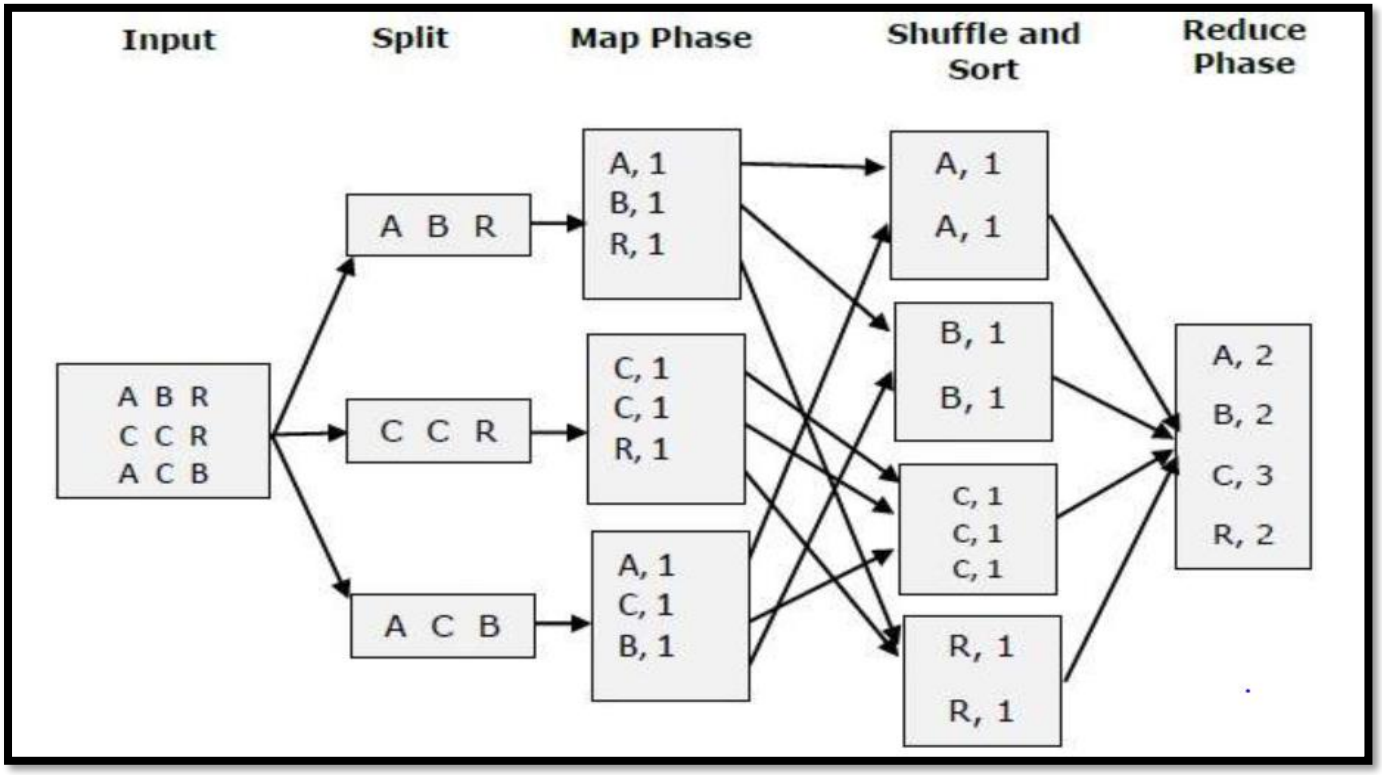
When there is less data to deal with, it makes shuffling and sorting simpler. Sometimes, due to the cumulative and associative functions in the reduction functionality, the combiner class is set to the reducer class itself.

Partition

Partition is the mechanism by which the pairs that result from mappers are converted into another set of pairs to feed into the reducer. It specifies how the data must be addressed to the reducer and assigns it to a specific reducer as well.

The default partitioner sets the mapper's hash value for the key and assigns a partition based on that hash value. As many partitions exist as there are reducers. So, the data from each partition is sent to a particular reducer once the partitioning is complete.

Program Design Alphabet Count:



Mapper receives the input as the .txt file then it will first covert all the text to lower case alphabet. Then process will remove all the special characters from the text.

Later text will process under split method where all the words will get separated to individual alphabets having values equal to 1.

Assumptions

All inputs book files are coming in .txt format.

Only 3 languages can see analyzed using jobs English, French, Spanish.

English Alphabets: 26

5 are vowels: A, E, I, O, U.

The remaining 21 letters are consonants: B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y, Z

French Alphabets: 40

26 characters same as English and 14 special characters Â, À, Ç, Ê, É, È, Ë, Ï, Î, Ô, Œ, Ü, Û and Ù.

Spanish Alphabets: 30

26 characters same as English and 4 special characters "ch," "ll," "ñ," and "rr."

Our question is we need to find out the average count of alphabets used in these 3 languages. **So, for comparison I am taking only 26 English characters which are common in all three languages.**

So, Mapper will split all text in 26 Alphabets for all 3 languages.

```
String s = value.toString().toLowerCase();
String s1 = s.replaceAll("[^a-zA-Z]","");

for (String word : s1.split("")) {
if (word.length() > 0) {
    context.write(new Text(word), new IntWritable(1));
}
}
}
}
```

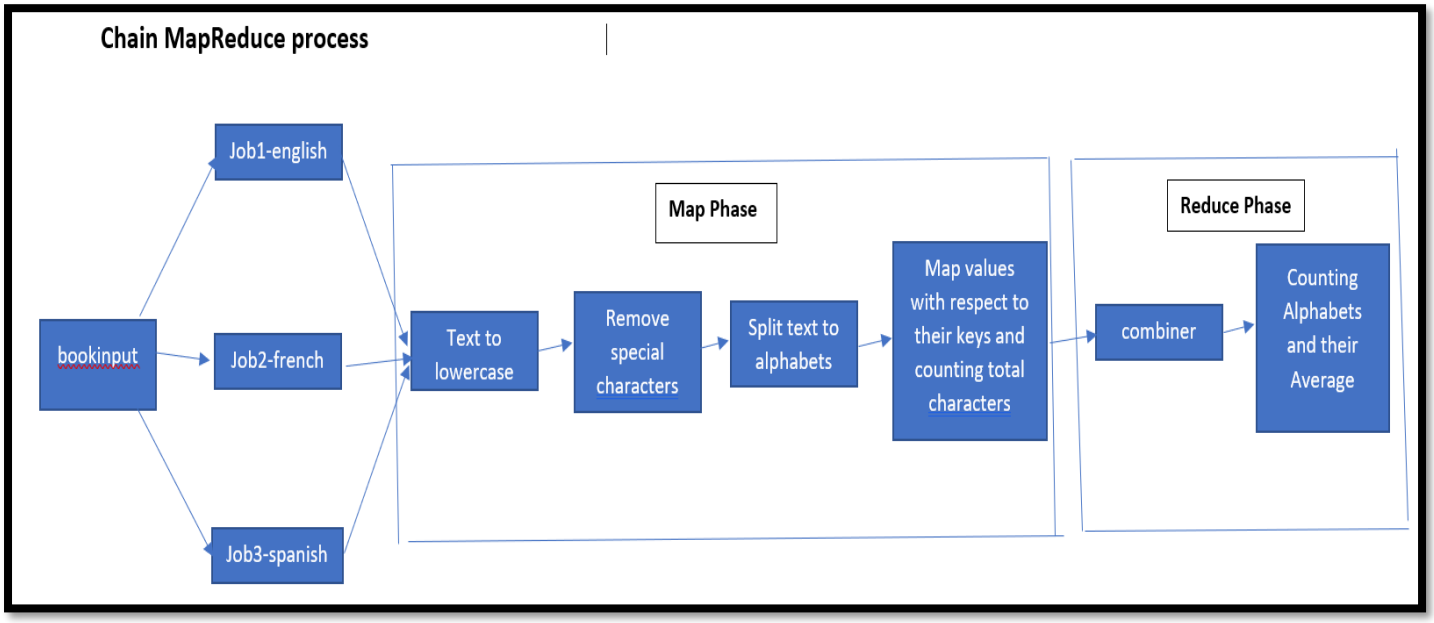
Our will ignore all special characters and annotations in it.

Reducers will count the frequency of alphabets in the text. And Second Reducer process will count the Average of alphabets frequency.

Then output of the program will be stored in .txt format.

Output files consist of Alphabet, Count, Average.

Program Workflow



MapReduce Code

Alphacount.java

//Abhishek Dubey
//Hadoop Driver Class

import org.apache.hadoop.conf.Configuration;

```
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.chain.ChainMapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.jobcontrol.ControlledJob;
import org.apache.hadoop.mapreduce.lib.jobcontrol.JobControl;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.partition.HashPartitioner;

public class alphacount {
public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

//job 1 for English Language
conf.set("Lang", "ENGLISH");
Job job = Job.getInstance(conf, "alphacount");
job.setJarByClass(alphacount.class);
//chain map process alphamap and keymap

Configuration AlphaMapConfig = new Configuration(false);
ChainMapper.addMapper(job, alphamap.class, LongWritable.class, Text.class, Text.class, DoubleWritable.class,
AlphaMapConfig);

Configuration KeyMapConfig = new Configuration(false);
ChainMapper.addMapper(job, keymap.class, Text.class, DoubleWritable.class,Text.class, DoubleWritable.class,
KeyMapConfig);

job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(DoubleWritable.class);
job.setPartitionerClass(HashPartitioner.class);
job.setReducerClass(alphareduce.class);
//for one output file
job.setNumReduceTasks(1);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(DoubleWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
ControlledJob cJob = new ControlledJob(conf);
cJob.setJob(job);
System.out.println("job 1 for English Language");

//job 2 for French Language
conf.set("Lang", "FRENCH");

Job job2 = Job.getInstance(conf, "alphacount");
job2.setJarByClass(alphacount.class);
```

```
Configuration AlphaMapConfig2 = new Configuration(false);
//chain map process
ChainMapper.addMapper(job2, alphamap.class, LongWritable.class, Text.class, Text.class, DoubleWritable.class,
AlphaMapConfig2);

Configuration KeyMapConfig2 = new Configuration(false);
ChainMapper.addMapper(job2, keymap.class, Text.class, DoubleWritable.class,Text.class, DoubleWritable.class,
KeyMapConfig2);

job2.setMapOutputKeyClass(Text.class);
job2.setMapOutputValueClass(DoubleWritable.class);
job2.setPartitionerClass(HashPartitioner.class);
job2.setReducerClass(alphareduce.class);

//for one output file
job2.setNumReduceTasks(1);
job2.setOutputKeyClass(Text.class);
job2.setOutputValueClass(DoubleWritable.class);
FileInputFormat.addInputPath(job2, new Path(args[2]));
FileOutputFormat.setOutputPath(job2, new Path(args[3]));

ControlledJob cJob2 = new ControlledJob(conf);
cJob2.setJob(job2);
System.out.println("job 2 for French Language");


//job 3 for Spanish Language
conf.set("Lang", "SPANISH");

Job job3 = Job.getInstance(conf, "alphacount");
job3.setJarByClass(alphacount.class);

Configuration AlphaMapConfig3 = new Configuration(false);
//chain map process alphamap and keymap

ChainMapper.addMapper(job3, alphamap.class, LongWritable.class, Text.class, Text.class, DoubleWritable.class,
AlphaMapConfig3);
Configuration KeyMapConfig3 = new Configuration(false);
ChainMapper.addMapper(job3, keymap.class, Text.class, DoubleWritable.class,Text.class, DoubleWritable.class,
KeyMapConfig3);

job3.setMapOutputKeyClass(Text.class);
job3.setMapOutputValueClass(DoubleWritable.class);
job3.setPartitionerClass(HashPartitioner.class);
job3.setReducerClass(alphareduce.class);

//for one output file
job3.setNumReduceTasks(1);
job3.setOutputKeyClass(Text.class);
job3.setOutputValueClass(DoubleWritable.class);

FileInputFormat.addInputPath(job3, new Path(args[4]));
FileOutputFormat.setOutputPath(job3, new Path(args[5]));
```

```
ControlledJob cJob3 = new ControlledJob(conf);
cJob3.setJob(job3);

System.out.println("job 3 for Spanish Language");

//creating job control for all jobs
JobControl jobctrl = new JobControl("jobctrl");
jobctrl.addJob(cJob);
jobctrl.addJob(cJob2);
jobctrl.addJob(cJob3);

Thread jobRunnerThread = new Thread(jobctrl);
jobRunnerThread.start();

//running job control in while loop

while (!jobctrl.allFinished()){
        System.out.println("MapReduce Process is running in background - It will take some time according to the
processing power");
        Thread.sleep(5000);
}
jobctrl.stop();
int code = jobctrl.getFailedJobList().size() == 0 ? 0:1;

System.out.println("Output Generated, Thanks for Running Abhishek Dubey Code for Alpha Count");
System.exit(code);
}
}
```

Alphamap.java

```
//Abhishek Dubey
//Mapper Class

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.DoubleWritable;
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 alphamap extends Mapper<LongWritable, Text, Text, DoubleWritable> {
public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
String s = value.toString();
//putting all text to lowercase
//replacing all special characters
s = s.toLowerCase();
s = s.replaceAll("[^a-zA-Z]","");
```

```
StringBuilder str = new StringBuilder(s);
double count = 0.0;
double strlen = str.length();
//counting all characters in string
for(int i = 0; i < str.length(); i++){
        if(Character.isLetter(str.charAt(i))){
                count = 1;
                for(int j = i+1; j < str.length(); j++){
                if (str.charAt(i)==str.charAt(j)){
                        count += 1.0;
                        str.setCharAt(j, ' ');
                }
        }

context.write(new Text(Character.toString(str.charAt(i))), new
DoubleWritable(Math.round((count/strlen)*100.0)/100.0));

        count = 0;

}
}
//Ending Mapper 1

}
}
```

Keymap.java

```
//Abhishek Dubey
//key map with values

import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.io.DoubleWritable;
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 keymap extends Mapper<Text, DoubleWritable, Text, DoubleWritable> {
public void map(Text key, DoubleWritable value, Context context)
throws IOException, InterruptedException {
        Configuration conf = context.getConfiguration();
        String langCode = conf.get("Lang");
//extracting output file in language + key + value format
        context.write(new Text(langCode+"\t"+"\t"+key), value);

//Ending Mapper 2
}
}
```

Alphareduce.java

```
//Abhishek Dubey
//Reducer Class
import java.io.IOException;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class alphareduce extends Reducer<Text, DoubleWritable, Text, DoubleWritable> {
public void reduce(Text key, Iterable<DoubleWritable> values, Context context)
throws IOException, InterruptedException {
//initializing variables for count and total
double alphaCount = 0;
int total = 0;
for (DoubleWritable value : values) {
        alphaCount += value.get();
        total += 1;
}
//Average Count of Alphabets
context.write(key, new DoubleWritable(Math.round((alphaCount/total)*100.0)/100.0));
//Ending Reducing Class
}
}
```

Output from MapReduce process

In Hadoop Web-interface 'http://localhost:50070' the output files can be checked under directory '/user/soc/'. Also by clicking on the Output directories the '_SUCCESS' and 'part-r-00000' files are created.

Stage 1 output in alphamap:
Changing all text to lowercase

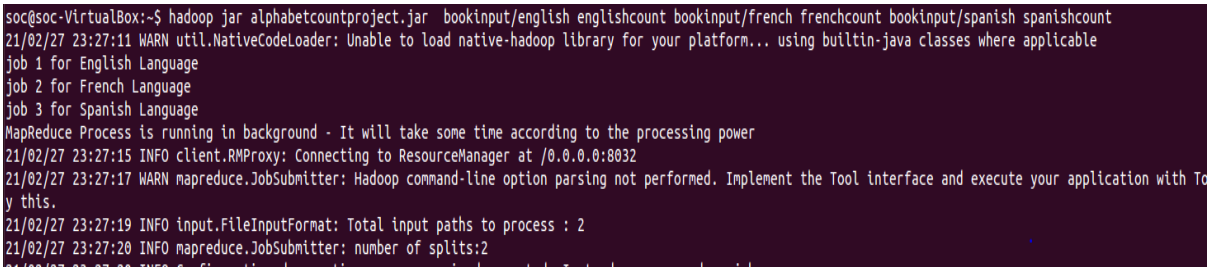
```
s = s.toLowerCase();
```

Stage 2 output in alphamap:
Removing all special character

```
s = s.replaceAll("[^a-zA-Z]","");
```

Final MapReduce output from each language:

English, French, Spanish



File: /user/soc/frenchcount/part-r-00000

Goto : /user/soc/frenchcount go

Go back to dir listing
Advanced view/download options

FRENCH	a	0.09
FRENCH	b	0.03
FRENCH	c	0.05
FRENCH	d	0.06
FRENCH	e	0.15
FRENCH	f	0.03
FRENCH	g	0.04
FRENCH	h	0.04
FRENCH	i	0.09
FRENCH	j	0.03
FRENCH	k	0.03
FRENCH	l	0.07
FRENCH	m	0.04
FRENCH	n	0.08
FRENCH	o	0.07
FRENCH	p	0.04
FRENCH	q	0.03
FRENCH	r	0.07
FRENCH	s	0.08
FRENCH	t	0.08
FRENCH	u	0.06
FRENCH	v	0.03
FRENCH	w	0.03
FRENCH	x	0.03
FRENCH	y	0.03
FRENCH	z	0.03

Download this file
Tail this file

File: /user/soc/spanishcount/part-r-00000

Goto : /user/soc/spanishcount go

Go back to dir listing
Advanced view/download options

SPANISH	a	0.12
SPANISH	b	0.03
SPANISH	c	0.05
SPANISH	d	0.05
SPANISH	e	0.13
SPANISH	f	0.03
SPANISH	g	0.03
SPANISH	h	0.03
SPANISH	i	0.06
SPANISH	j	0.03
SPANISH	k	0.03
SPANISH	l	0.06
SPANISH	m	0.04
SPANISH	n	0.07
SPANISH	o	0.1
SPANISH	p	0.04
SPANISH	q	0.03
SPANISH	r	0.07
SPANISH	s	0.08
SPANISH	t	0.05
SPANISH	u	0.05
SPANISH	v	0.03
SPANISH	w	0.03
SPANISH	x	0.02
SPANISH	y	0.03
SPANISH	z	0.02

Download this file
Tail this file
Chunk size to view (in bytes, up to file's DFS bloc

File: /user/soc/englishcount/part-r-00000

Goto : /user/soc/englishcount go

Go back to dir listing
Advanced view/download options

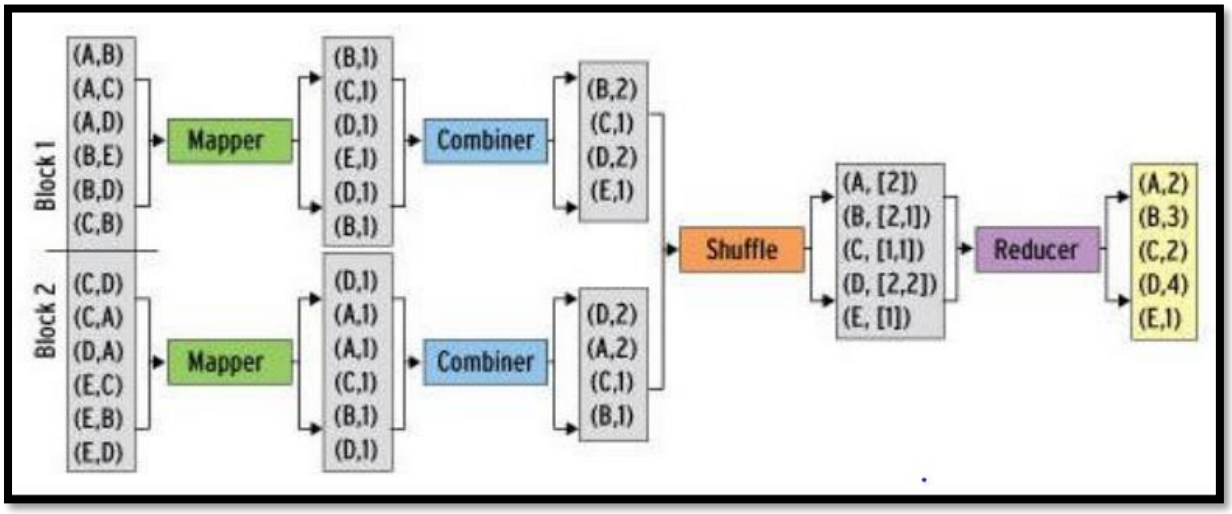
ENGLISH	a	0.08
ENGLISH	b	0.03
ENGLISH	c	0.04
ENGLISH	d	0.05
ENGLISH	e	0.12
ENGLISH	f	0.04
ENGLISH	g	0.03
ENGLISH	h	0.06
ENGLISH	i	0.08
ENGLISH	j	0.03
ENGLISH	k	0.03
ENGLISH	l	0.05
ENGLISH	m	0.04
ENGLISH	n	0.08
ENGLISH	o	0.08
ENGLISH	p	0.04
ENGLISH	q	0.03
ENGLISH	r	0.06
ENGLISH	s	0.07
ENGLISH	t	0.1
ENGLISH	u	0.04
ENGLISH	v	0.03
ENGLISH	w	0.03
ENGLISH	x	0.03
ENGLISH	y	0.03
ENGLISH	z	0.03

Download this file
Tail this file
Chunk size to view (in bytes, up to file's DFS block

MapReduce Additional Features

Combiners

Semi-reducers are combiners. It's an optional class that takes input from the Map class and sends the key-value pair to the Reducer class. It's used in the center of the Map and Reducer groups. Combiners can help you save a lot of money.



In Alphacount program. I have used keymap class which works a combiner. The output key and value from alphamap is fed into keymap class as an input. The output of keymap class is fed as an input further into alphareduce class.
As a consequence, the Map-Reduce architecture enables huge scalability across large data sets. The system is available for use. The programming model is independent of the programming language.

Chain MapReduce

We can run multiple jobs in one attempt. It provide high range of scaibility.
Here is our project we are using 3 languages as input by using three different jobs we can scale it further to more languages, just by adding new jobs

Configuration AlphaMapConfig3 = new Configuration(false);

ChainMapper.addMapper(job3, alphamap.class, LongWritable.class, Text.class, Text.class, DoubleWritable.class, AlphaMapConfig3);

Configuration KeyMapConfig3 = new Configuration(false);

ChainMapper.addMapper(job3, keymap.class, Text.class, DoubleWritable.class,Text.class, DoubleWritable.class, KeyMapConfig3);

Using setupJob function

ControlledJob cJob3 = new ControlledJob(conf);
cJob3.setJob(job3);

System.out.println("job 3 for Spanish Language");

//creating job control for all jobs
JobControl jobctrl = new JobControl("jobctrl");

```
jobctrl.addJob(cJob);
jobctrl.addJob(cJob2);
jobctrl.addJob(cJob3);
```

Python Code for Visualization

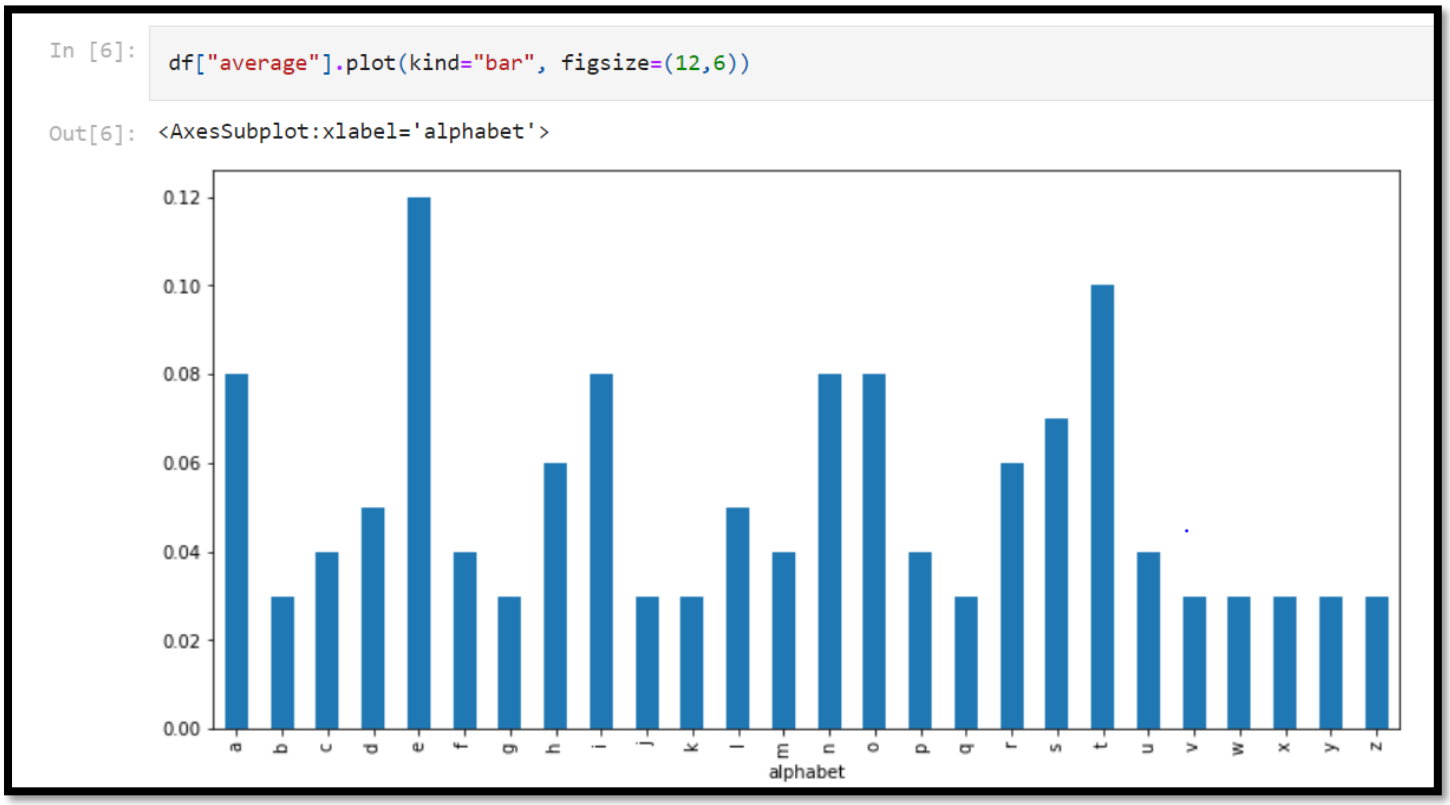
For English Language

```
#importing python libraries
Import pandas as pd
import matplotlib.pyplot as plt

#creating pandas data frame and creating column names
df = pd.read_csv(r"C:\Users\abhis\Desktop\hadoop book\_user_soc_englishcount_part-r-00000", sep='\t', names=["language", "alphabet", "average"])

df
#setting index to alphabet for x axis
df=df.set_index('alphabet')

#plotting graph
df["average"].plot(kind="bar", figsize=(12,6))
```



Insights based on 2 English books:

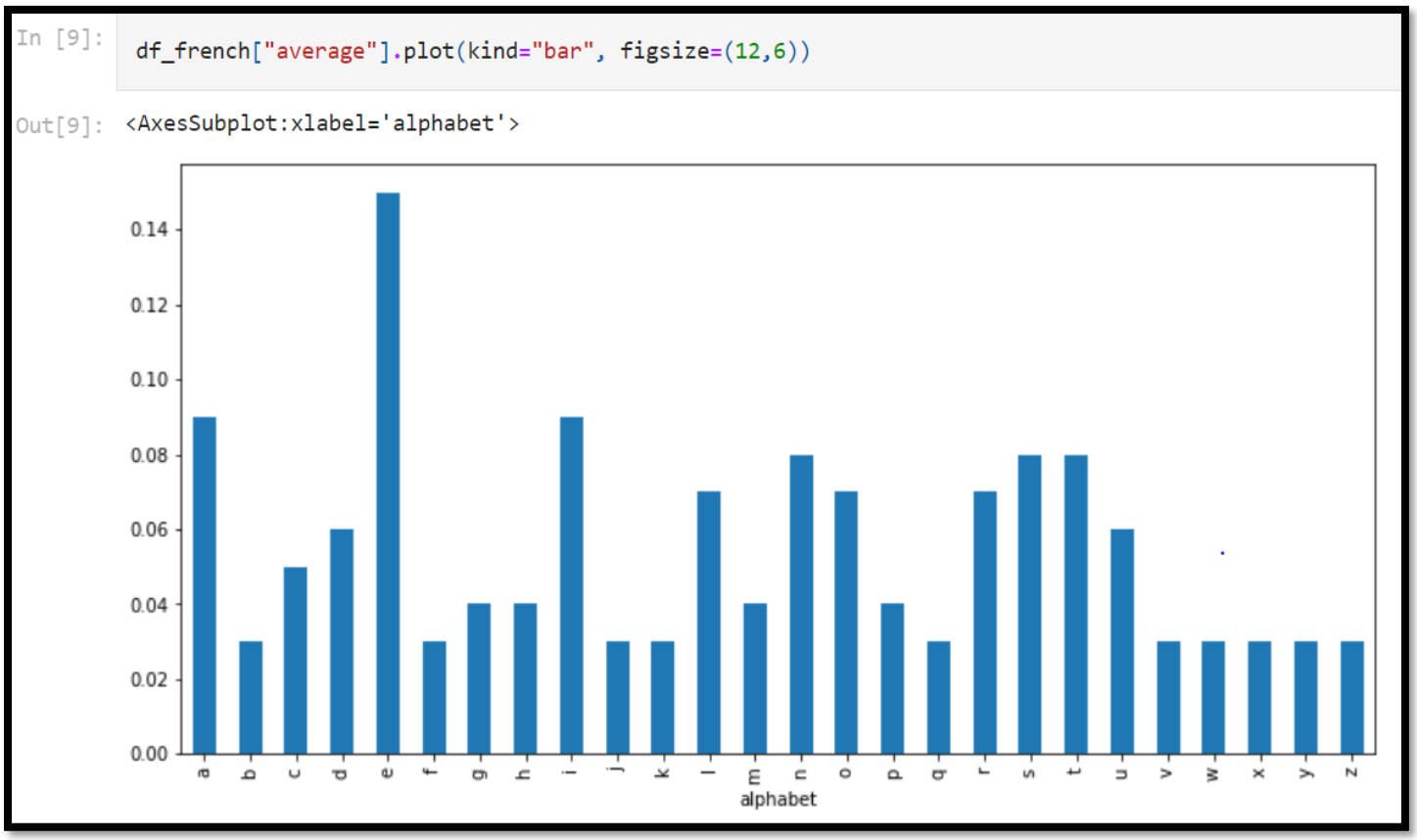
- Alphabet e has highest average frequency in English with value of 0.12.
- Alphabets b,g,j,k,q,v,w,x,y,z looks like they are least using in English with having average of 0.03.
- Alphabet t is the second most used letter in English language with frequency 0.10.

For French Language

#creating pandas data frame and creating column names
df_french = pd.read_csv(r"C:\Users\abhis\Desktop\hadoop book_user_soc_frenchcount_part-r-00000", sep='\t', names=["language", "alphabet", "average"])

#setting index to alphabet for x axis
df_french = df_french.set_index('alphabet')

#plotting graph
df_french["average"].plot(kind="bar", figsize=(12,6))



Insights based on 2 French books:

- Alphabet e has highest average frequency in French with value of 0.15.
- Alphabets b,f,j,k,q,v,w,x,y,z looks like, they are least used in French with having average of 0.03.
- Alphabet a,i is the second most used letter in French language with frequency 0.09.

For Spanish Language

#creating pandas data frame and creating column names
df_spanish = pd.read_csv(r"C:\Users\abhis\Desktop\hadoop book_user_soc_spanishcount_part-r-00000", sep='\t', names=["language", "alphabet", "average"])

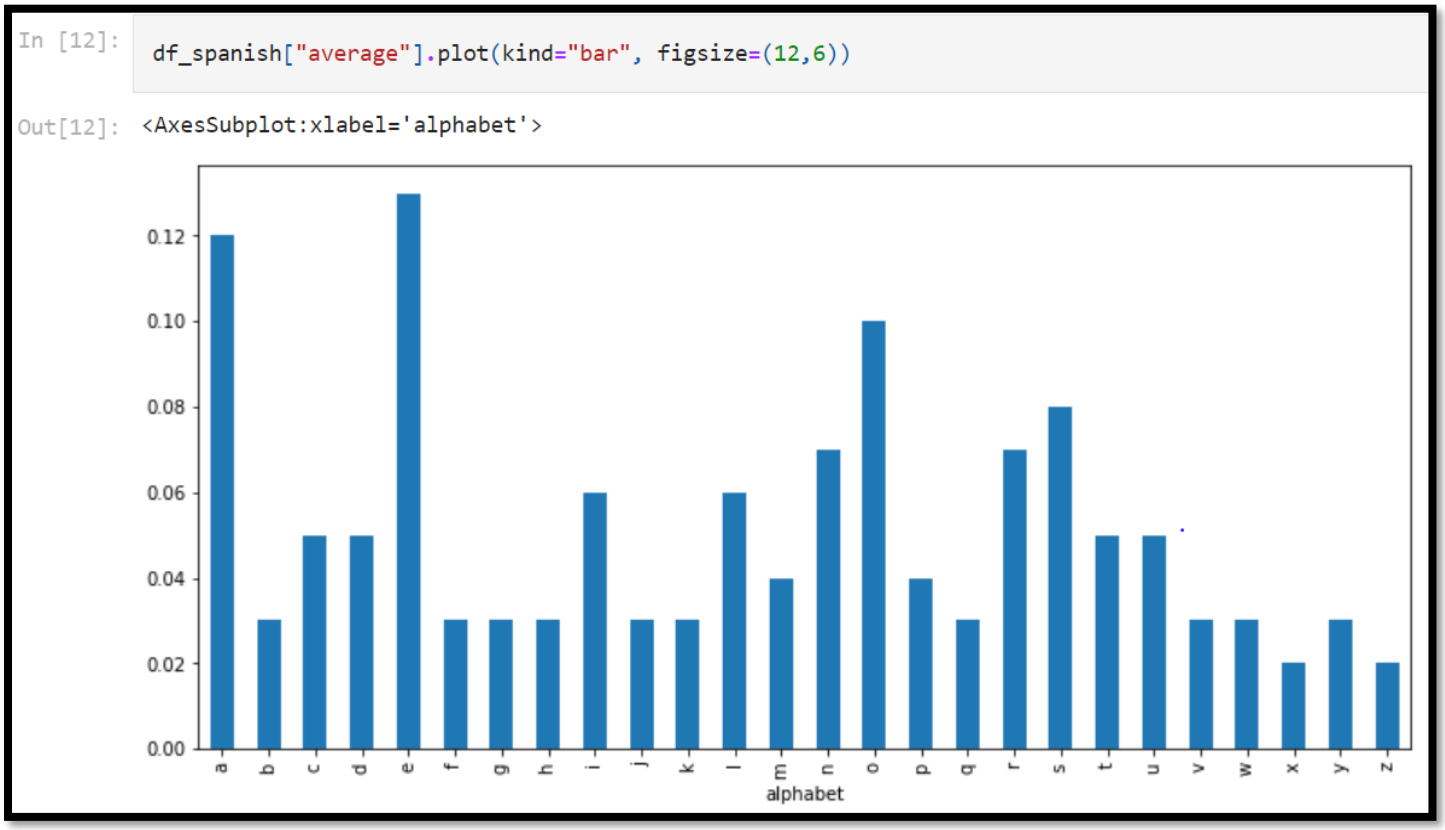
#setting index to alphabet for x axis
df_spanish = df_spanish.set_index('alphabet')

#plotting graph

df_spanish["average"].plot(kind="bar", figsize=(12,6))

```
In [12]: df_spanish["average"].plot(kind="bar", figsize=(12,6))

Out[12]: <AxesSubplot:xlabel='alphabet'>
```



Insights based on 2 Spanish books:

- Alphabet e has highest average frequency in Spanish with value of 0.13.
- Alphabets x,z looks like, they are least used in Spanish with having average of 0.02.
- Alphabet a is the second most used letter in Spanish language with frequency 0.12.

Final overall comparison between languages

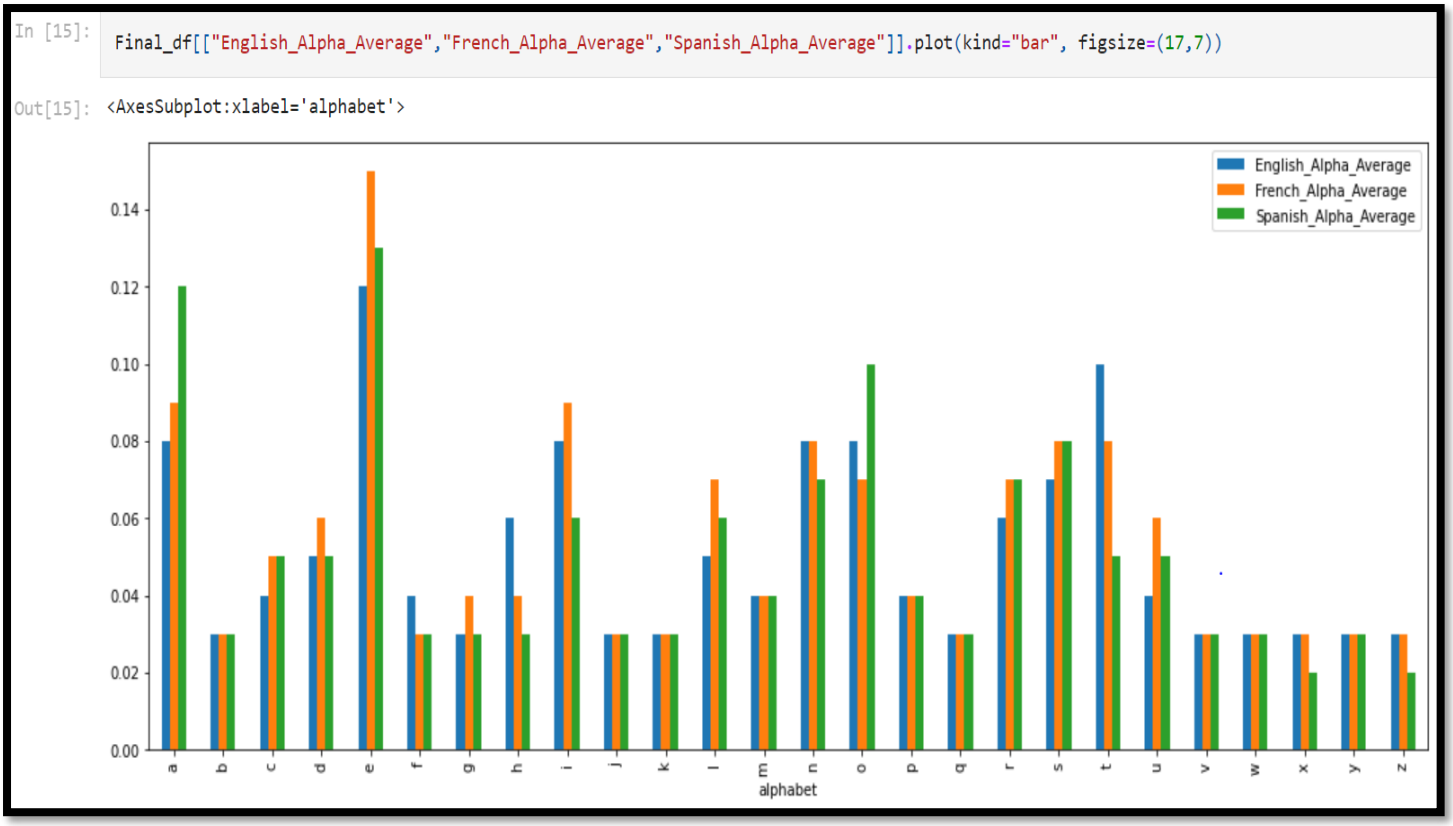
```
#renaming column names
df=df.rename(columns={"average": "English_Alpha_Average"})
df_french=df_french.rename(columns={"average": "French_Alpha_Average"})
df_spanish=df_spanish.rename(columns={"average": "Spanish_Alpha_Average"})

#adding all data frame together
Final_df=pd.concat([df,df_french, df_spanish],axis=1)

#plotting
Final_df[["English_Alpha_Average","French_Alpha_Average","Spanish_Alpha_Average"]].plot(kind="bar",
figsize=(17,7))
```

```
In [15]: Final_df[["English_Alpha_Average","French_Alpha_Average","Spanish_Alpha_Average"]].plot(kind="bar", figsize=(17,7))

Out[15]: <AxesSubplot:xlabel='alphabet'>
```



Insights based on 6 books (2 English, 2 French, 2 Spanish):

- Alphabet e from french language is having highest average frequency of 0.15.
- In all languages alphabet e has been used the greatest number of times.
- Alphabets x,z are least used in Spanish language.
- Alphabets b,j,k,q,v,w ,y share the equal average frequency distribution among languages.
- There is a very small different in average frequency of a,e for Spanish languages of 0.01.