

Big Data Wrangling with Google Books Ngrams

Daniel Mortensen

16-Sep-21

Introduction

For this report, I am setting up a cloud-based computer network for managing a large dataset. First, I will be setting up an EMR cluster using AWS and importing data related to Google's Ngrams dataset. I will then use pyspark to analyze this dataset and extract all rows corresponding to the word "data" into a new data frame. This new "data" data frame will be uploaded to a bucket on AWS and analyzed on my local PC to determine the relationship between year and how frequently the word "data" appears in the text.

EMR Cluster Creation and Connection

The cluster was created in Amazon's AWS using the EMR service.

- In EMR, a new cluster is created by selecting the "Create cluster" button.



- Once inside the new cluster setup page, "Go to advanced options" was selected:

[Go to advanced options](#)



General Configuration

Cluster name

☒ Logging ⓘ

S3 folder ⓘ

Launch mode ☒ Cluster ⓘ ☐ Step execution ⓘ

- The release version was set to "emr-5.29.0". Modules selected included Hadoop, JupyterHub, Hive, and Spark.

Software Configuration

Release ⓘ

<input checked="" type="checkbox"/> Hadoop 2.8.5	<input type="checkbox"/> Zeppelin 0.8.2	<input type="checkbox"/> Livy 0.6.0
<input checked="" type="checkbox"/> JupyterHub 1.0.0	<input type="checkbox"/> Tez 0.9.2	<input type="checkbox"/> Flink 1.9.1
<input type="checkbox"/> Ganglia 3.7.2	<input type="checkbox"/> HBase 1.4.10	<input type="checkbox"/> Pig 0.17.0
<input checked="" type="checkbox"/> Hive 2.3.6	<input type="checkbox"/> Presto 0.227	<input type="checkbox"/> ZooKeeper 3.4.14
<input type="checkbox"/> MXNet 1.5.1	<input type="checkbox"/> Sqoop 1.4.7	<input type="checkbox"/> Mahout 0.13.0
<input type="checkbox"/> Hue 4.4.0	<input type="checkbox"/> Phoenix 4.14.3	<input type="checkbox"/> Oozie 5.1.0
<input checked="" type="checkbox"/> Spark 2.4.4	<input type="checkbox"/> HCatalog 2.3.6	<input type="checkbox"/> TensorFlow 1.14.0

- All other configuration options were left unchanged.

- The cluster was named “BigDataCluster”

General Options

Cluster name

- The security key was set to a preexisting key pair.

Security Options

EC2 key pair ⓘ

☒ Cluster visible to all IAM users in account ⓘ

- The cluster was then created using the “Create cluster” button.

- Successfully creation of the cluster was then confirmed by checking that it appeared in the list of clusters in the EMR service with a status of “waiting.”

☐ ▶ ☒ [BigDataCluster](#) j-1WYG3W38WOR5X Waiting
Cluster ready

Creation of the SSH Tunnel

A connection to the head node of the BigDataCluster cluster was established via an SSH Tunnel. This was accomplished by opening Git Bash on my local PC. Then, the following command was input into the terminal:

```
ssh -i [path to local pem file] hadoop@ec2-34-211-87-207.us-west-2.compute.amazonaws.com
```

Importing the Ngram Data

A version of the Ngram data prepared by BrainStation was imported into HDFS in the /user/hadoop/eng_1M_1gram directory using the following command:

```
hadoop distcp s3://brainstation-dsft/eng_1M_1gram.csv  
/user/hadoop/eng_1M_1gram/eng_1M_1gram.csv
```

Confirmation that the file imported correctly was obtained by checking the contents of the /user/hadoop/eng_1M_1gram directory using the following command:

```
hadoop fs -ls /user/hadoop/eng_1M_1gram
```

Execution of the above command resulted in the display of the following information:

```
Found 1 items
```

```
-rw-r--r-- 1 hadoop 5292105197 2021-09-16 01:04
/user/hadoop/eng_1M_1gram/eng_1M_1gram.csv
```

From this result, it can be observed that there is a csv file with the proper file name in the desired folder.

EDA and Data Filtering

Exploratory data analysis and data filtering were conducted using the Jupyter Notebook submitted with this report named “Ngram Notebook.ipynb”. This notebook was created in AWS using the EMR service’s “Notebooks” option.

Notebook Creation

- Once in the Notebooks service, “Create notebook” was selected.

Notebooks

Use EMR notebooks based on Jupyter to analyze data in Amazon S3. Notebooks run free of charge and are secure.

Create notebook

View details

- The notebook was named “BigDataNotebook”.

Name and configure your notebook

Name your notebook, choose a cluster or create one, and customize configuration options if desired.

Notebook name* BigDataNotebook

Names may only contain alphanumeric characters, hyphens (-), or underscores (_).

The notebook was then connected to the BigDataCluster.

Choose a cluster

The listed clusters meet notebook requirements. They are in an EC2-VPC, running EMR 5.18.0 or later, and have Hadoop, Spark, and Livy installed. [Learn more](#)

The notebook can be opened once the cluster is in Waiting or Running status.

Filter: Filter clusters ...

2 clusters (all loaded)

Name	ID	Status
BigDataCluster	j-1WYG3W38WOR5X	Waiting Cluster ready

- The “Create notebook” button was selected to create the notebook.

Cancel

Create notebook

- Successfully creation of the notebook was confirmed by checking that it appeared in the list of notebooks in the EMR service with a status of “Ready.”

Filter:	All notebooks ▾	Filter notebooks ...	5 notebooks (all loaded)	
	Name	Status	Cluster	Creation time (UTC-6)
	BigDataNotebook	Ready	j-1WYG3W38WOR5X	2021-09-15 18:59 (UTC-6)

The BigDataNotebook was then opened in Jupyter Lab, where the “pyspark” kernel was used.

EDA

The Ngram data were imported as the “ngram_df” data frame. This data frame has five columns:

- “token”: the word of interest
- “year”: the year of interest
- “frequency”: total number of times the word was used in the given year
- “pages”: the number of pages containing the word of interest for the given year
- “books”: the number of books containing the word of interest for the given year

All these columns contain “string” type data. There are 261,823,225, or just over a quarter billion, rows of data in this data frame.

The ngram_df data frame was then filtered into a new data frame containing only the rows from ngram_df where the token column is equal to the word “data.”. This was accomplished using the following line of code:

```
data_df = spark.sql(" \
SELECT * \
FROM ngram_view \
WHERE token = 'data'\
")
```

This new “data_df” data frame has 316 rows of data, meaning that the word data has been found in books published in 316 different years. This new data frame was then exported back into HDFS using the code:

```
data_df.write.csv('/user/hadoop/eng_data_1gram/eng_1M_1gram_data_token.csv',
header=True)
```

In order to confirm that the data were exported successfully as a csv file into the /user/hadoop/eng_data_1gram directory, the following command was entered into the Git Bash command console:

```
hadoop fs -ls /user/hadoop/eng_data_1gram
```

Which resulted in the following output:

```
Found 1 items
```

```
drwxr-xr-x - livy hadoop      0 2021-09-16 15:14  
/user/hadoop/eng_data_1gram/eng_1M_1gram_data_token.csv
```

The last line in that result is a confirmation that the data were exported successfully. This csv file was then collected into a single csv file on the local drive of the head node using the following command:

```
hadoop fs -getmerge /user/hadoop/eng_data_1gram/eng_1M_1gram_data_token.csv  
eng_data_1gram.csv
```

Proper creation of this file confirmed by checking the contents of the local drive using the command:

```
hadoop fs -ls
```

Which resulted in the following information being displayed:

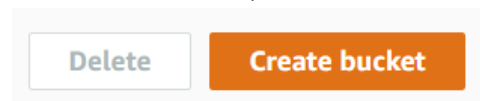
```
Found 2 items  
drwxrwxrwx - hadoop hadoop      0 2021-09-16 01:04 eng_1M_1gram  
drwxr-xr-x - livy hadoop        0 2021-09-16 15:14 eng_data_1gram
```

The filtered data “eng_data_1gram” is clearly seen, confirming the file was created successfully.

Moving the Data to an AWS Bucket

The filtered Ngram data were then uploaded to an AWS bucket. This bucket was created using the AWS service S3.

- Once in S3, the “Create bucket” button was selected.



- The bucket was named “bigdatabucket91621” and set in the “US West (Oregon)” region.

General configuration

Bucket name

bigdatabucket91621

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US West (Oregon) us-west-2

- All other settings were left unchanged.
- The bucket was created by pressing the “Create bucket” button at the bottom of the screen.

Cancel

Create bucket

- Successful creation of the bucket was confirmed by checking that the bucket bigdatabucket91621 appeared in the list of buckets in S3.

	Name ▲	AWS Region ▼	Access ▼	Creation date ▼
<input type="radio"/>	bigdatabucket91621	US West (Oregon) us-west-2	Bucket and objects not public	September 16, 2021, 09:20:54 (UTC-06:00)


Once the bucket was created, the filtered Ngram data were uploaded to this bucket using the following command in the Git Bash command line program:


```
aws s3 cp ./eng_data_1gram.csv s3://bigdatabucket91621/eng_data_1gram.csv
```

After running the above command, the file became visible in the AWS bucket, confirming it was uploaded successfully.


Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can

  Copy S3 URI  Copy URL

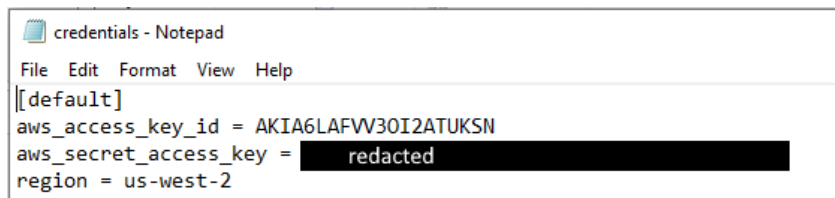
 Find objects by prefix

☐ Name

☐  [eng_data_1gram.csv](#)

Analysis of “data” Data Frame

The filtered data frame, which contains only data corresponding to the token “data” was then analyzed on my local PC by importing the data directly from the S3 bucket it was stored in. In order to do this, s3fs was installed in the conda environment using the command “`conda install s3fs -c conda-forge`.” Proper configuration of the `.aws/credentials` file was also required. This file was given the configurations shown in the image below.

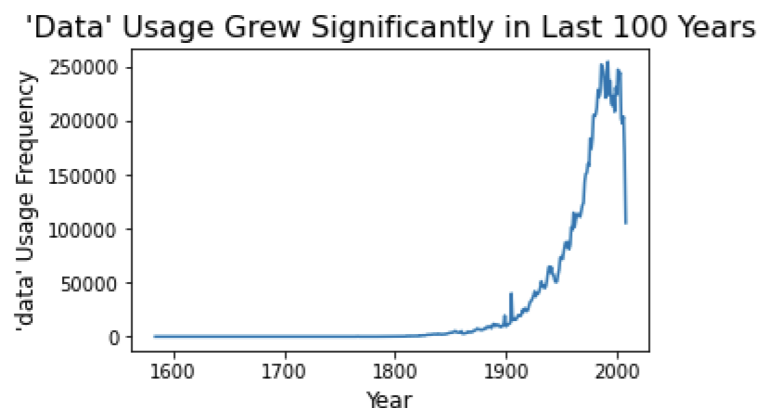


```
credentials - Notepad
File Edit Format View Help
[[default]
aws_access_key_id = AKIA6LAFVW30I2ATUKSN
aws_secret_access_key = redacted
region = us-west-2
```

The data were then analyzed using the Jupyter Notebook submitted with this report named “Data Frequency Analysis.ipynb,” wherein the “python3” kernel was used. The data were imported into this notebook as a new data frame named “df” using the following code:

```
df = pd.read_csv('s3://bigdatabucket91621/eng_data_1gram.csv')
```

Once the data frame was successfully imported, the frequency column was plotted as a function of the year column, resulting in the plot shown below.



In this plot it can be seen that usage of the word “data” was extremely low until about 1800, at which point it started to pick up in usage. From there, usage increased slowly until about 1900, wherein usage of data increased exponentially, including a few anomalous years right after 1900 wherein “data” was used abnormally frequently, until a few years before 2000. The usage of the word “data” appears to be on a downturn since right before the year 2000.

Conclusions

The Ngram data were successfully filtered and analyzed for usage of the word “data” as a function of time. Successful creation of an AWS cluster, notebook, and bucket were

demonstrated, as was the successful import and export of the Ngram data in and out of several file systems and locations.