

Data Intensive Computing (CSE 587)

Lab-2 (Report)

Submitted by:

Deepak Goyal (dgoyal2)

Vishal Gawade (vgawade)

Topic: We used Sports as our main topic for Data Collection. Subtopic and keywords that we used for collection are baseball, Soccer, Tennis, Rugby, Golf.

Data Collection from below three sources:

1. **New York Times:** To collect data from New York times, we use New York times article search api which gives the articles urls. We stored the 100 articles urls into .csv for every subtopic. After getting the urls we read the urls from .csv one by one and fetch the article data using urllib and requests library & process the data using BeautifulSoup library and store in .txt files. Code for the New York times are placed as part1 -> code -> nyTimes_Articles_Collection.py & part1 -> code -> nyTimes_Data_Collection.ipynb. Collected Data are placed in part1 -> Data -> NYT
2. **Common Crawl:** To collect data from common crawl, we use common crawl warc files to fetch the data. We read the data from warc files; check for the language and search for our keywords in the articles and store the corresponding article data into .txt files if it matches. Warc and langdetect libraries of python are used to reach the extract the relevant data. Code for this are placed as part1 -> code -> common_crawl.ipynb. Collected Data are placed in part1 -> Data -> Commoncrawl
3. **Twitter:** To collect tweets data from twitter search api, we use the R language. We store the collected tweets in the .csv files. After that we process the tweets and remove the irrelevant emojis, special characters and store it in the .txt files. Code for this are placed as part1 -> code -> crawl_tweets.ipynb. Collected Data are placed in part1 -> Data -> Twitter.

After Collecting the data from above three sources, we process the data in mapper and reducer to get the word count of each word in the data files. Sort the word in descending order using word count. Get the top 10 words that repeated most of the times. On the basis of top 10 words, we process the data to get word co-occurrence of these top 10 words. We used AWS to implement and run mapper and reducer.

Explanation of Mapper (Word Count):

- Read the data file (data that are collected from different sources).
- Remove special character, stop words and other irrelevant characters.
- Perform lemmatization on words
- For all words in data Emit (word, 1)

Explanation of Reducer (Word Count):

- Read input data from Mapper.
- Combine the data using word as key and increment the count(value) of each key for every word.
- Finally sort the data in descending order of count. (Word with the highest count stored at first)
- For first 10 words -> Emit the (word, count)

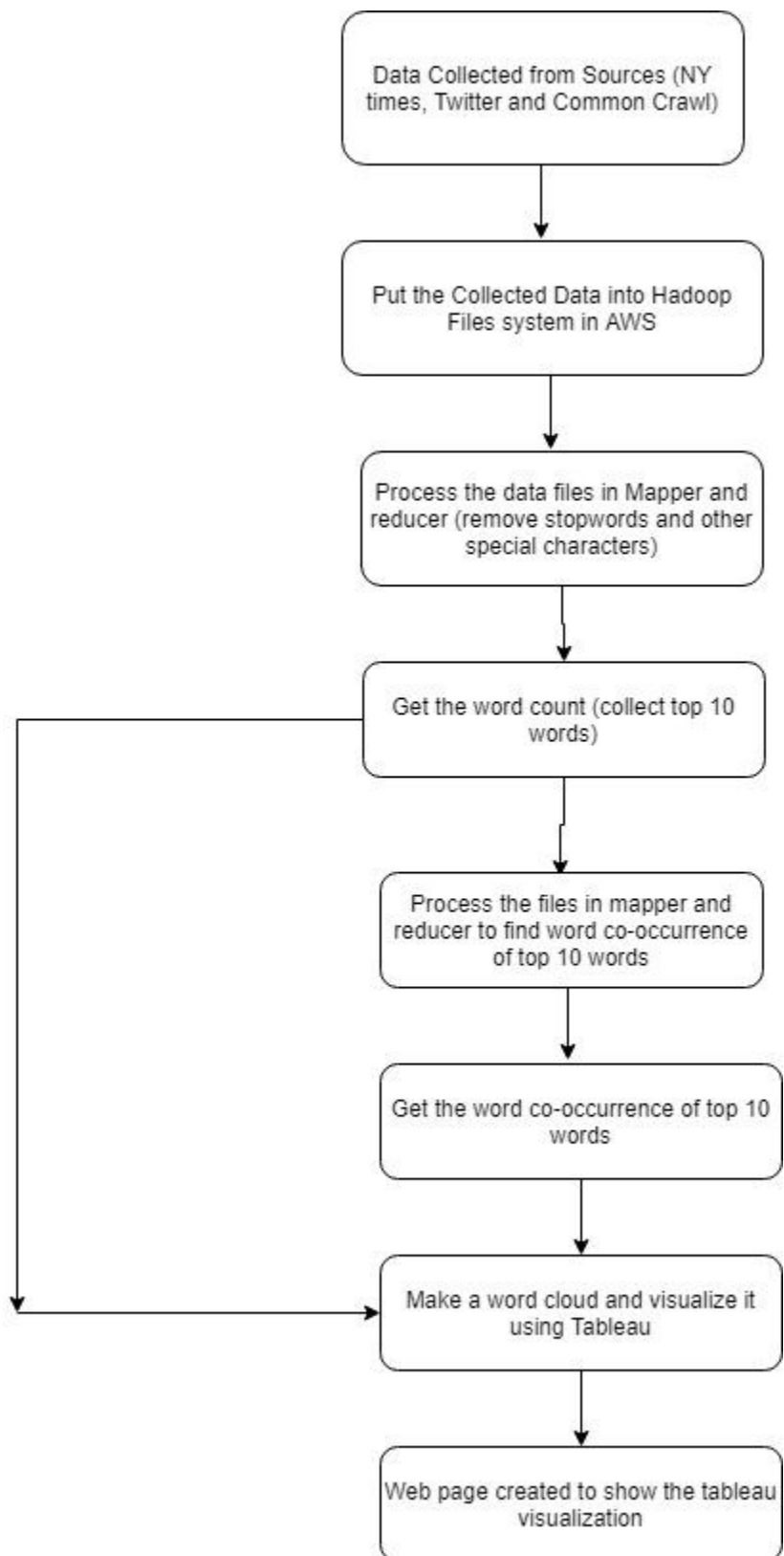
Explanation of Mapper (Word Co-occurrence):

- Read the input (that we get after reducer of word count)
- Remove special character, stop words and other irrelevant characters.
- Perform lemmatization on word
- For all words w1 in doc
 - For all words w2 in Neighbours(w1)
 - Emit (w1-w2, 1)

Explanation of Reducer (Co-occurrence):

- Read input data from Mapper.
- Combine the data using word pair as key and increment the count(value) of each key for every word pair.
- Finally sort the data in descending order of count. (Word pair with the highest count stored at first)
- For first 10 words -> Emit (word pair, count)

Flow Chart:



While doing all this on AWS, we install all dependencies that are required to run the code. To install the dependencies, we made a bash file “install_dependencies.sh” which install all required dependencies. Below are the screenshots to explain the steps.

The screenshot shows the 'Create Cluster - Advanced Options' page in the AWS EMR console, specifically Step 1: Software and Steps. The 'Software Configuration' section has 'Release' set to 'emr-5.22.0'. Under 'Add software settings', 'Hadoop 2.8.5' is selected. In the 'Add steps (optional)' section, a 'Streaming program' step is added with the following arguments: 'hadoop-streaming -files s3://twitterid/Mapers/mapper_latest_cooccurrence_cc.py -3 /twitterid/Reducers/reducer_latest.py -mapper mapper_latest_cooccurrence_cc.py -reducer reducer_latest.py -input s3://twitterid/CC_Input/output -output s3://twitterid/CC_Output/Workd_Co-occurrence'. A green arrow points to the 'Streaming program' step type, labeled 'select streaming option'. Another green arrow points to the 'Arguments' field, labeled 'mapper, reducer, input, output path'. The 'Auto-terminate cluster after the last step is completed' checkbox is checked. The 'Next' button is visible at the bottom right.

The screenshot shows the 'Create Cluster - Advanced Options' page in the AWS EMR console, specifically Step 2: Hardware. The 'Hardware Configuration' section shows 'Instance group configuration' selected. The 'Network' is set to 'vpc-b591a3dd (172.31.0.0/16) (default)'. The 'EC2 Subnet' is set to 'subnet-9f708bd3 (Default in us-east-3c)'. The 'Root device EBS volume size' is set to '10' GiB. Below this, a table lists the instance groups:

Node type	Instance type	Instance count	Purchasing option	Auto Scaling
Master Instance Group	m4.large 4 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB	1 Instances	On-demand Spot Use on-demand as max price	Not available for Master
Core Instance Group	m4.large 4 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB	2 Instances	On-demand Spot Use on-demand as max price	Not enabled

The 'Next' button is visible at the bottom right.

us-east-2.console.aws.amazon.com

My recursively... JAVA O(1) v... EMR - AWS... S3 Manage... tweetL_prepr... CSE-587-D... CSE587.DIC... https://cse... Instances [... CSE 4/587 (... Work with S... map.drawio... +

Services Resource Groups

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security

General Options

Cluster name: My cluster.1

☒ Logging ⓘ
S3 folder: s3://aws-logs-382552567353-us-east-2/elasticmapreduce

☒ Debugging ⓘ
☐ Termination protection ⓘ

Tags ⓘ

Key	Value (optional)
Add a key to create a tag	

Additional Options

☐ EMRFS consistent view ⓘ

Custom AMI ID: ⓘ

Bootstrap Actions

Bootstrap actions are scripts that are executed during setup before Hadoop starts on every cluster node. You can use them to install additional software and customize your applications. [Learn more](#) ⓘ

Bootstrap action type	Name	JAR location	Optional arguments
Custom action	Custom action	s3://twitterdc/install_dependencies.sh	

Add bootstrap action: ⓘ [Configure and add](#)

select written shell script as bootstrap action for installing nltk and necessary libraries

[Cancel](#) [Previous](#) [Next](#)

Feedback English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

us-east-2.console.aws.amazon.com

My recursively... JAVA O(1) v... EMR - AWS... S3 Manage... tweetL_prepr... CSE-587-D... CSE587.DIC... https://cse... Instances [... CSE 4/587 (... Work with S... map.drawio... +

Services Resource Groups

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security

Security Options

EC2 key pair: ⓘ

☒ Cluster visible to all IAM users in account ⓘ
This feature will be deprecated soon.

Permissions ⓘ

☐ Default ☒ Custom
Select custom roles to tailor permissions for your cluster.

EMR role: ⓘ

EC2 Instance profile: ⓘ

Auto Scaling role: ⓘ

Authentication and encryption


EC2 security groups

An EC2 security group acts as a virtual firewall for your cluster nodes to control inbound and outbound traffic. There are two types of security groups you can configure, EMR managed security groups ⓘ and additional security groups ⓘ. EMR will automatically update ⓘ the rules in the EMR managed security groups in order to launch a cluster. [Learn more](#) ⓘ

Type	EMR managed security groups <small>EMR will automatically update ⓘ the selected group</small>	Additional security groups <small>EMR will not modify the selected groups</small>
Master	<input type="text" value="Default: sg-0afe2ec8955f8d0c (ElasticMapReduce-master)"/> ⓘ	<input type="text" value="sg-0fa1e5e4f9d579b8 (launch-wizard-1)"/> ⓘ
Core & Task	<input type="text" value="Default: sg-065c7da29502b764d (ElasticMapReduce-slave)"/> ⓘ	<input type="text" value="sg-0fa1e5e4f9d579b8 (launch-wizard-1)"/> ⓘ

[Create a security group](#) ⓘ

[Cancel](#) [Previous](#) [Create cluster](#)



Feedback English (US)

© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. [Privacy Policy](#) [Terms of Use](#)

After performing the Mapper and reducer, we get the word count and word co-occurrence for all the three sources (NY times, Twitter and Common Crawl). We also perform mapper and reducer on sub topic for each data sources, which gives us word count (top 10 words) and word co-occurrence.

We make 21-word cloud as below:

- 3 for word count (top 10); for each data source (NY times, Twitter and Common Crawl)
- 3 for word co-occurrence (top 10); for each data source (NY times, Twitter and Common Crawl)
- 15 for word count (top 10); for each subtopic and for each data source (NY times, Twitter and Common Crawl) (5 subtopic * 3 data source = total 15)

We used Tableau to perform visualization on each of the above the word cloud. We make web page of this visualization where you can view by selecting different like for word count and word co-occurrence. We have deployed the code; you can click on the below link to view the visualization:

https://vishalgawade.github.io/wordcloud_hadoop/visualization.html