Data Aggregation, Big Data Analysis and Visualization

Introduction

Sports has been a very important part of our daily lives. The amount of data related to sports on the web is huge. It has a very wide range of sources ranging from personal comments on social media, to blogs and to official websites to live videos. With this wide array of sources we can understand the patterns and can analyze the data.

Implementation

In this project the data pipeline has been set up by gathering the information sources. The data has been collected in three ways

1. Twitter
2. NY Times articles
3. Common Crawl data

Twitter Data Collection

To gather the tweets, I used tweepy api of Twitter. I used python script to achieve this. I used tweepy AppAuth Handler to increase the tweet limit. Then I used the following search queries for Tweet Collection

1. American football	#americanfootball, #football
2. Basketball	#basket ball, #nba
3. Baseball-softball	#softball
4. Soccer	#soccer, #soccerteam
5. wresting	#wwe, #wresting.

The tweets were collected from 16/03/2019 to 17/04/2020 and saved them in a txt file.

The number of tweets collected were in the range of 40,000. The script extracts the tweet's text and stores them into a txt file.

NY Times data Collection

To gather the articles, I used the API provided by NY Times. I used python script to retrieve the articles content. I collected the data in 2 phases.

Phase 1: Begin date = 20190316

End data = 20190331

Phase 2: Begin date = 20190401

End date = 20190418

The search terms included were the following ["americanfootball", "football","nba","softball","soccer", "wwe", "#wresting"]

We used urlib request of python to hit the NY Times search article api, and fetched the result from the corresponding url and then from the docs. BeautifulSoup library is used to extract the paragraph content. Then that data has been send to the file to written to the text file.

Common Crawl Data Collection

Common Crawl the data has been extracted by using the CC News article. The data is available on AWS S3 in the commoncrawl bucket at /crawl-data/CC-NEWS/. Then using this command

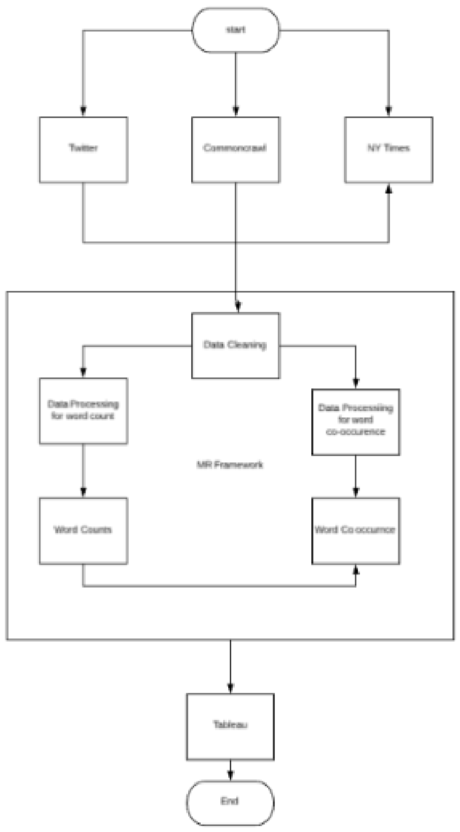
```
aws s3 ls --recursive s3://commoncrawl/crawl-data/CC-NEWS/
```

The listed WARC files are extracted using the AWS Command Line Interface then the WARC file that I used is the s3://commoncrawl/crawl-data/CC-NEWS/2016/09/CC-NEWS-20190419211809-00000.warc.gz

In the next step we extracted the warc file and then we iterated through the entire the entire urls of the warc file. I used requests.get(link) to hit the url and then used BeautifulSoup and then I iterated through the text to find any keywords and stored the url's which provided me the best result.

In the second step, I iterated through that url's and fetched the result of the paragraph content from the url's I above collected and stored them in a CSV file.

Flow Chart



Word Count using the MapReduce Framework

MapReduce is a framework using which can be used to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce framework to count the number of times a word occurred in the tweets or articles or in common crawl.

Mapper: In the Mapper, I clean the data to keep only the words essential for our analysis. I remove stop words to make the data meaningful. Stop words are natural language words which have very little meaning, such as "and", "the", "a", "an" etc. These are commonly occurring words and will distort our analysis.

The Mapper them emits (outputs) a key value pair for each word in the article or tweet. The key in this case is the word and value is 1. I will later, in Reducer, aggregate these 1 for every key (unique word) to find the count of that word.

Reducer: Here I find the actual counts. Output of the mapper is fed as an input to the Reducer. The reducer collects <key,value> pairs which have the same key. The <key, value> pair are of the type <word, 1>. It then sums over all the values received for this key. This generates the count for that key. The reducer then emit (output) this value.

Co-occurrence using the Map Reduce Framework

I find out the co-occurring words in each article/ tweet and from common crawl url. The aim is to find the pair of words which occur together most frequently. For this I use the top ten most frequently occurring words captured after running the Word Count on tweets and articles. The context for co-occurrence is chosen as a tweet or an article. The approach followed is similar to the one used in word count using Map Reduce. The Mapper and Reducer are implemented as follows.

Mapper: The first step of the mapper is to clean the data (just like in Word Count). I do this by removing the stop words and other common words which do not reflect the data. I then select pairs of words from the tweet/article, such that at least one of those words lies in the top ten words which I have identified. This word-pair, consisting of two words, is emitted by the Mapper in <key, value> format as <word-pair, 1>

Reducer: In the Reducer, I collect the all the <key, value> pairs that are emitted by the Mapper. All the values (1) associated with the same key (word-pair) are aggregated to get the count of the word-pair. This is emitted by the Reducer as the output. This procedure is applied to every unique word-pair sent as the key. The resulting output gives us the frequency of co-occurrence of the word-pairs so that I can identify the co-occurring words with the highest frequencies.

Word Cloud Generation

To show the comparison of top occurring words in Twitter data, NY Times data and Common crawl data. I used Tableau as a visualization tool for generating word clouds.

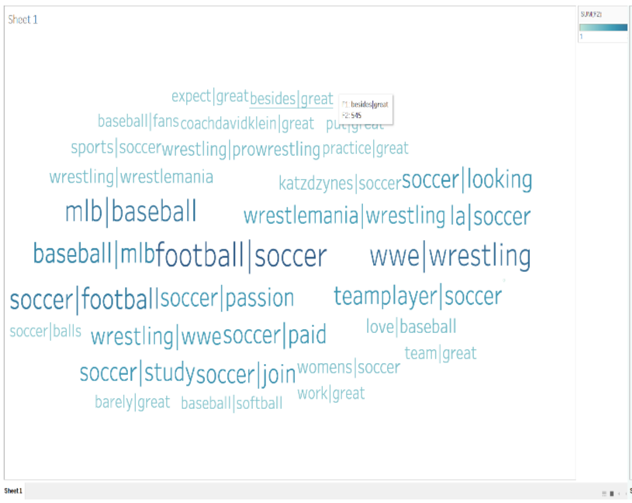
I generated a work book in Tableau in for seeing the word cloud visualizations. Input to the Tableau is given as csv file containing the top 25 words. This csv is fed as an input to Tableau and word cloud is generated.

Parameters used:

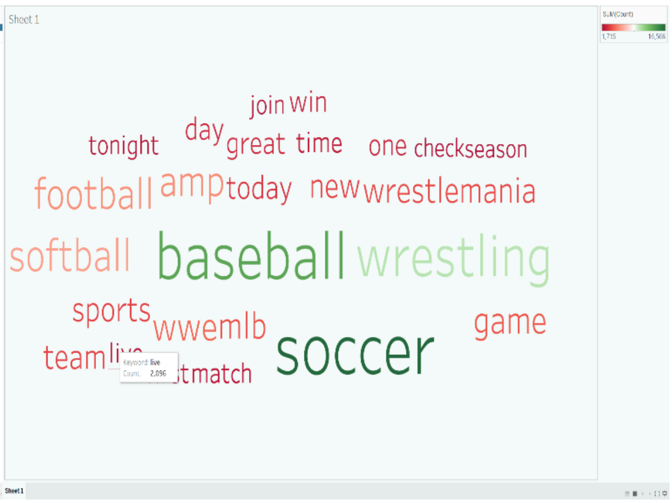
The parameters used are the keyword and Number of values. Key word is given as a text and count is given as input for size and color. Then by changing the view style to automatic we can generate the word cloud.

Visualization for Twitter

Word Count



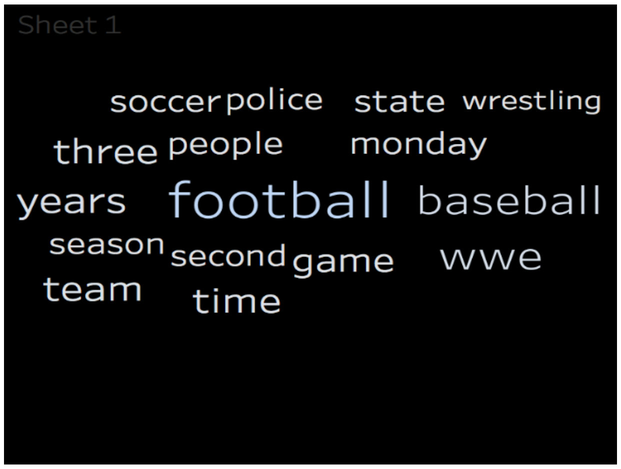
Word Co-occurrence

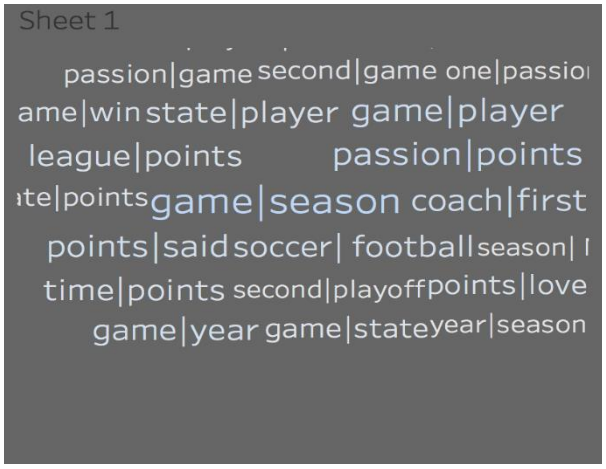


Visualization for Common crawl

Word count

Word co occurrence

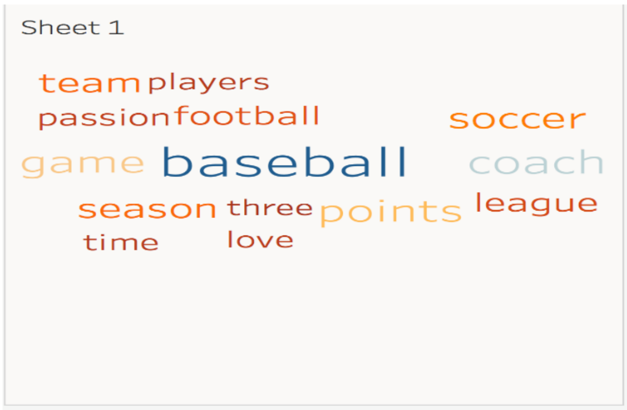


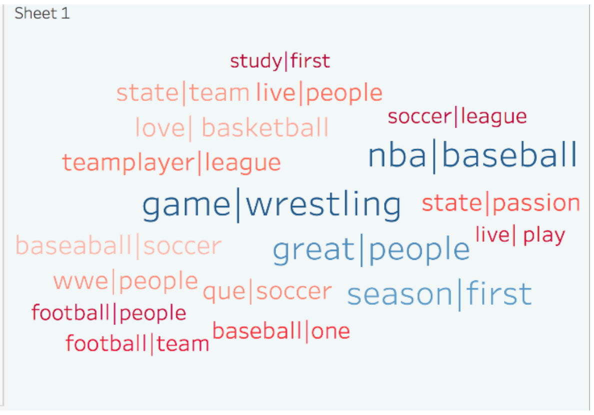


Visualization for NY Times

Word Count

Word Co-occurrence

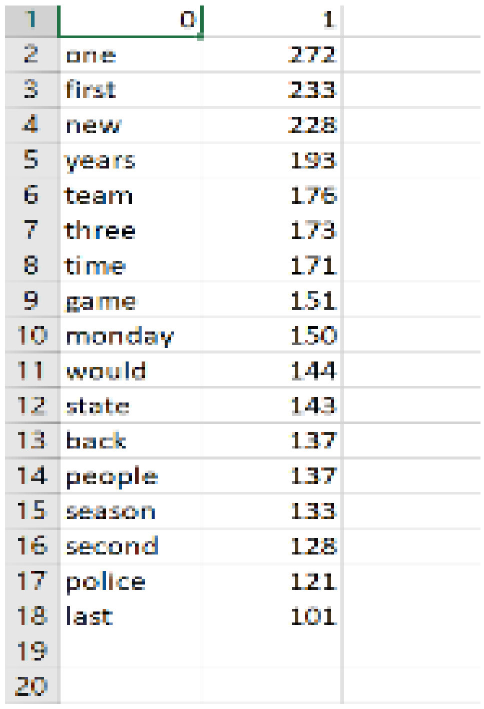




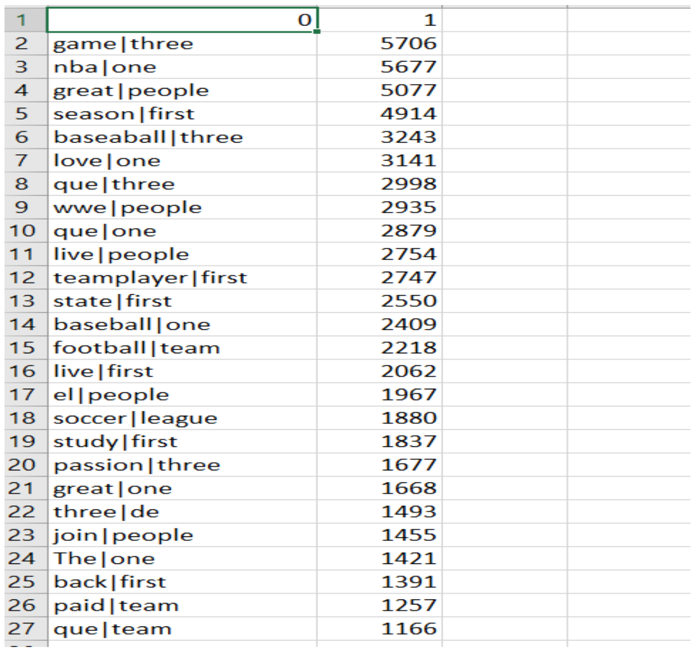
Analysis and Results

Common crawl

Top 25 Keywords of Common crawl word count and word co-occurence



	0	1
2	one	272
3	first	233
4	new	228
5	years	193
6	team	176
7	three	173
8	time	171
9	game	151
10	monday	150
11	would	144
12	state	143
13	back	137
14	people	137
15	season	133
16	second	128
17	police	121
18	last	101
19		
20		



	0	1
2	game|three	5706
3	nba|one	5677
4	great|people	5077
5	season|first	4914
6	baseaball|three	3243
7	love|one	3141
8	que|three	2998
9	wwe|people	2935
10	que|one	2879
11	live|people	2754
12	teamplayer|first	2747
13	state|first	2550
14	baseball|one	2409
15	football|team	2218
16	live|first	2062
17	el|people	1967
18	soccer|league	1880
19	study|first	1837
20	passion|three	1677
21	great|one	1668
22	three|de	1493
23	join|people	1455
24	The|one	1421
25	back|first	1391
26	paid|team	1257
27	que|team	1166

Top word count and co-occurrence of NY Times and Twitter are included in the submission files.

Reference link for Video

Reference Link: https://buffalo.box.com/s/g7dea1kkhbwr6kkdkzvbxssvcfch9w9k/

References:

Python Mapper and Reducer: http://www.michael-noll.com/tutorials/writing-an-hadoopmapreduce-program-in-python/

https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas/16476974