DATA AGGREGATION, BIG DATA ANALYSIS AND VISUALIZATION

Group-members

Priyanka Pai (UBIT : 50295903)
Goutham Thota:(UBIT:50286877)

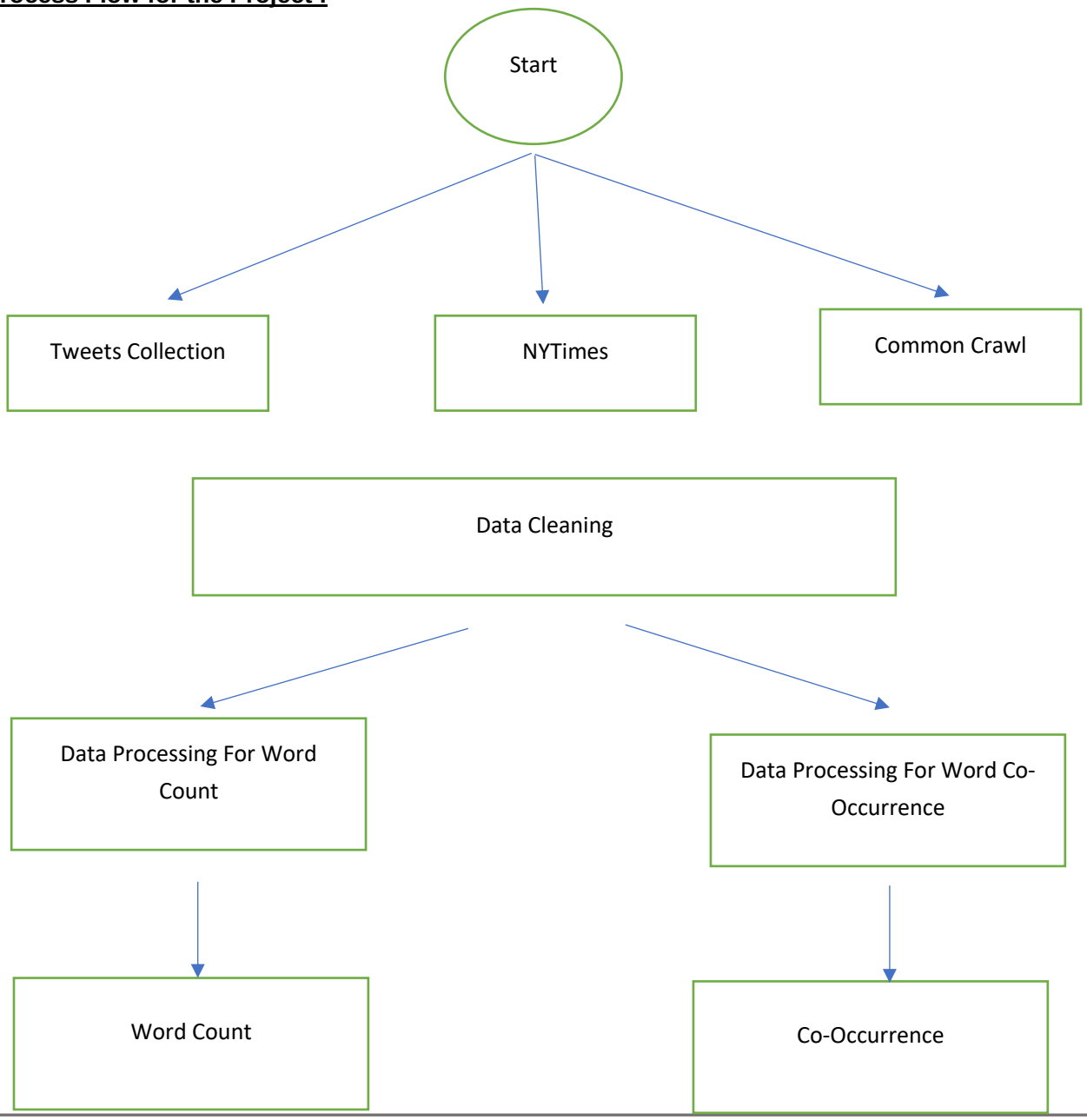
Introduction:

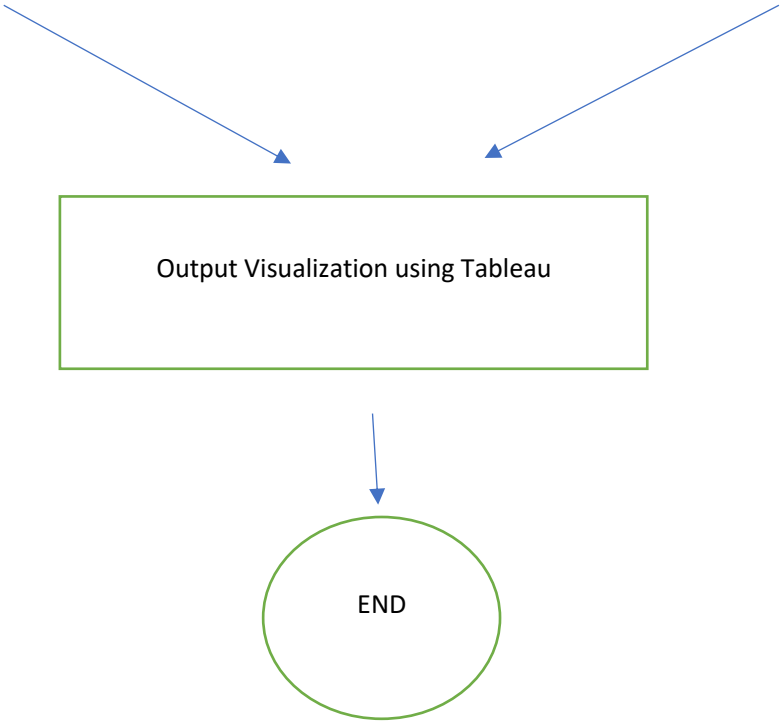
Sports are an evergreen topic on which a lot of data can be collected at any point of time in a year, therefore we concluded that we would have a lot of data to carry out an in-depth analysis of it. It should also be noted that a lot of popular figures also tweet regarding various sports therefore providing another interesting perspective.

Steps for Data Collection :

1) Data aggregation from more than one source using the APIs
2) Applying classical big data analytic method of MapReduce to the unstructured data collected
3) store the data collected on WORM infrastructure Hadoop
4) building a visualization data product.

Process Flow for the Project :





Data Collection : We used 3 platforms to fetch data :

- Twitter
- NewYorkTimes
- Common Crawl

Twitter data collection:

To gather the tweets on my selected topic, we used the twitteR API of Twitter. We wrote code for an R script that makes use of an API key and return tweets for given 'search query' and 'date range'. We will be using a Python library called TwitteR to connect to Twitter Streaming API and downloading the data around 20k tweets for hashtags.

Entered credentials into access_token, access_token_secret, consumer_key, and consumer_secret. The hashtags which were used for searching follow-

#baseball
#basketball
#football
#soccer
#golf

Our R script uses the tweets "screenname" to retrieve user-location and stores meta-data about the tweets into a csv file(eg-twitter_dataxxxx), and the text of the tweets is stored in a text file.

New York Times Data :

To scrape data from NYTimes, we obtained an API key from the New York Times.The API will not return full text of articles. But it will return a number of helpful metadata such as subject terms, abstract, and date, as well as URLs, which one could conceivably use to scrape the full text of articles. Installed nytimesarticle package, which is a python wrapper for the New York Times Article Search API. This allows you to query the API through python. We then used **BeautifulSoup** to prettify the application and generated only the <p> tags from the json output.

BeautifulSoup -

is a Python library for pulling data out of HTML and XML files. Beautiful Soup 4 is published through PyPi, so if you can't install it with the system packager, you can install it with easy_install or pip. The package name is beautifulsoup4, and the same package works on Python 2 and Python 3. Make sure you use the right version of pip or easy_install for your Python version (these may be named pip3 and easy_install3 respectively if you're using Python 3).

$ easy_install beautifulsoup4

$ pip install beautifulsoup4

Common Crawl :

The third source we used was Common Crawl. It contains 210TiB of data that needs to be queried. So we used a Hadoop cluster EMR to run a Hadoop job.

Common Crawl currently stores the crawl data using the Web ARChive (WARC) format.
Before that point, the crawl was stored in the ARC file format.

The WARC format allows for more efficient storage and processing of Common Crawl's free multi-billion page web archives, which can be hundreds of terabytes in size.

This document aims to give you an introduction to working with the new format, specifically the difference between:

WARC files which store the raw crawl data

WAT files which store computed metadata for the data stored in the WARC

WET files which store extracted plaintext from the data stored in the WARC

Word Count using the MapReduce Framework

MapReduce is a framework using which I can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. I use the MapReduce framework to count the number of times a word occurred in the tweets or articles. I do this to find the most frequent words which capture the essence of the topic. Mapper and Reducer are implemented as follows:

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 we find the actual counts. Output of the mapper is fed as an input to the Reducer. The reducer collects pairs which have the same key. The pair are of the type . 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:

We find out the co-occurring words in each article/ tweet. The aim is to find the pair of words which occur together most frequently. For this we 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 <key, value> format as <word-pair, 1>.

Reducer:

In the Reducer, I collect the all the 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 comparison:

Word clouds show you what is emphasized in your text. you can use word clouds of different scenes to compare how your characters feel about the inciting action. A word cloud will help you notice right away if your characters' reactions are similar in both scenes. You'll be able to identify where you should have a new reaction, a new emotion or feeling.

Co occurrence :

A co-occurrence matrix could be described as the tracking of an event, and given a certain window of time or space, what other events seem to occur.

Pairs:

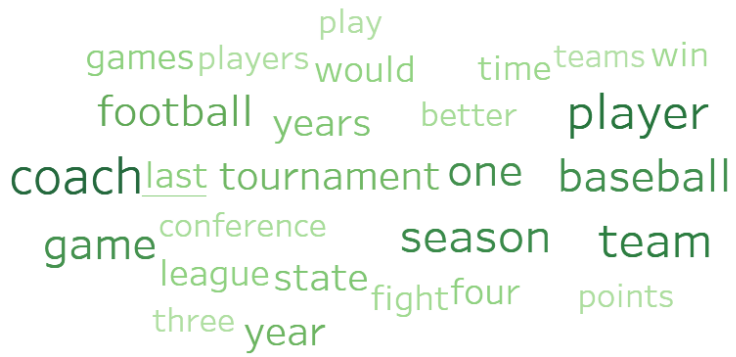
Implementing the pairs approach is straightforward. For each line passed in when the map function is called, we will split on spaces creating a String Array. The next step would be to construct two loops.

The outer loop will iterate over each word in the array and the inner loop will iterate over the “neighbours” of the current word. The number of iterations for the inner loop is dictated by the size of our “window” to capture neighbours of the current word. At the bottom of each iteration in the inner loop, we will emit a WordPair object (consisting of the current word on the left and the neighbour word on the right) as the key, and a count of one as the value.

Stripes:

Implementing the stripes approach to co-occurrence is equally straightforward. The approach is the same, but all of the “neighbor” words are collected in a HashMap with the neighbor word as the key and an integer count as the value. When all of the values have been collected for a given word (the bottom of the outer loop), the word and the hashmap are emitted.

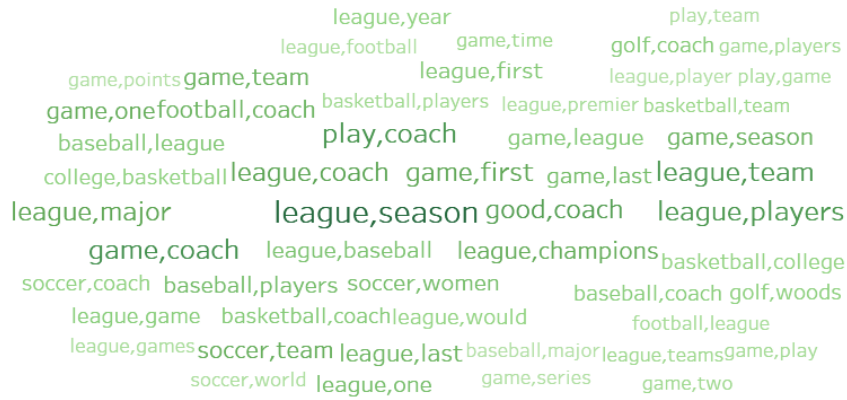
Common Crawl Word Count :



Common Crawl Co-Occurrence :

NYTimes Word Count :

NYTimes Cooccurrence :



Twitter Co-Occurrence :

Twitter Word Count :

Data Visualisation :

We used the Tableau Software to generate the WordCloud. It is really helpful to see the most common answer in a survey qualitative data.Big Data visualization allows a comprehensible visual representation of data patterns, enabling users to more easily glean insights from data. Big Data visualization involves the presentation of data of almost any type in a graphical format that makes it easy to understand and interpret.

Hadoop Commands :

start-dfs.sh //starting namenodes and datanodes.

Setting up the single node cluster :

sudo rm -r /tmp/* //cleaning everything in tmp directory

hdfs namenode -format

hdfs dfs -ls / //check whether there is any directory in hdfs

hdfs dfs -mkdir /test //if there is no directory create one named test

hdfs dfs -put /home/cse587/examplehadoop/exam.txt /test
//putting the exam.txt from local directory into the test directory inside hdfs

cd hadoop-3.1.2/bin

hadoop jar /home/cse587/hadoop-3.1.2/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.2.jar wordcount /test/exam.txt /test/output

//Now you should have successfully run wordcount on exam.txt file

hdfs dfs -ls /test/output //You should see the output file as part-r0000