Data Intensive Computing

LAB – 2

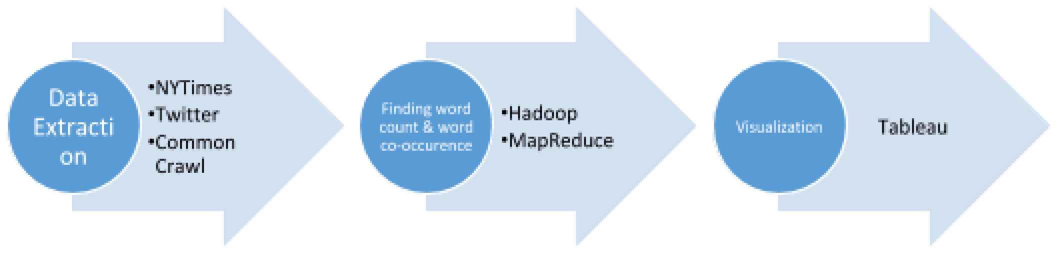
Vineel Kurma, vineelku, 50291799

Meghana Vadlapally, mvadlapa, 50298635

Project Exploration:

- In this project, we collected sports data from NYTimes, Twitter, Common crawl using APIs.
- Python has been used as a tool to collect data from the sources.
- We used VM to set up HDFS for applying big data analytical methods on the collected data.
- The obtained data has been given as an input to the MapReduce to find word count and word co-occurrence.
- The extracted word count and word co-occurrence files are visualized by creating word cloud using Tableau.

Flow chart:



1. **Data Extraction:** Contains sports data from NYTimes, Twitter, Common Crawl.

    Tool: Used python to extract data

    - **NYTimes code**: nytimescode.ipynb
    - **Twitter code**: used LAB-1's code
    - **Common Crawl**: CommonCrawlGetPy2.py, CommonCrawlEditPy3.py

SubTopics:
Used the FootBall, BasketBall, Cricket, NFL, GOLF as subtopics to extract the data.

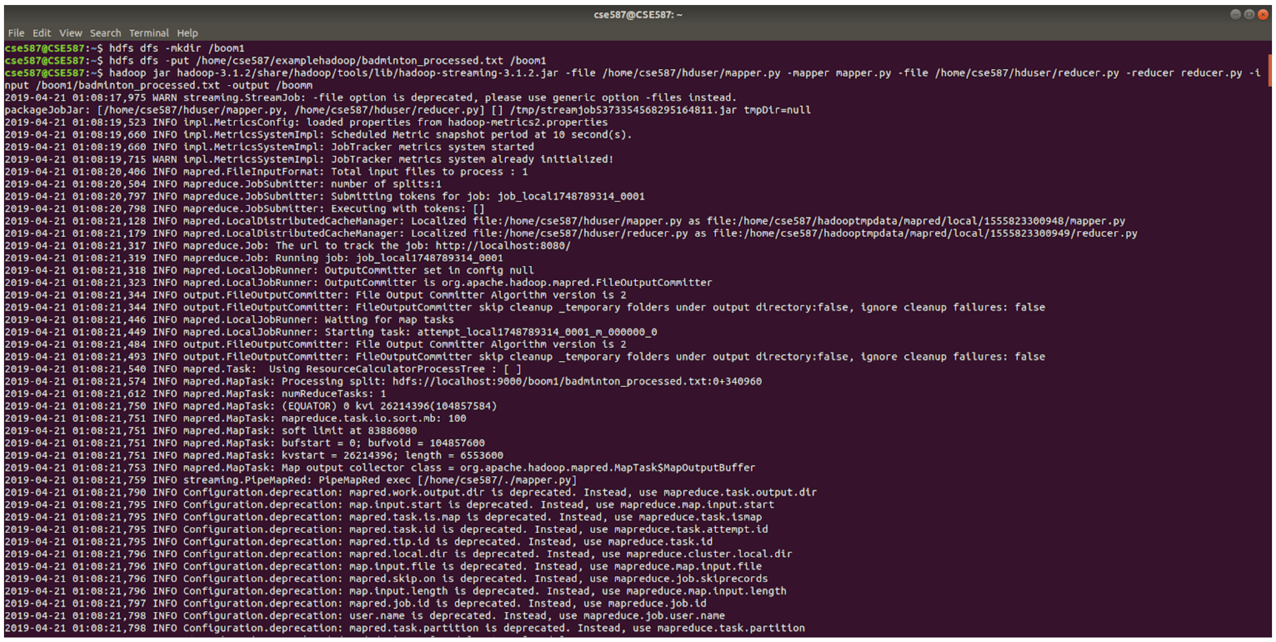
Twitter Data Collection:
For collecting the tweets used Tweepy library and used the twitter API credentials created from the previous lab, referenced the code from the GitHub.
link:https://github.com/shuzhanfan/Geo-tagged-streaming-tweets-collect.
After collecting the data, preprocessed by removing the duplicate tweets and used a combination of tweets cleaning library and regular expression to filter out emojis and non-ASCII characters. Overall we have collected around 60,000 total tweets in which 25K tweets are unique.

NewYork Times Data Collection:
For collecting the NewYork Times data, created an developer API and extracted the URLs and put them into a file. Then for each URL, used request module and extracted the raw HTML data. After fetching the HTML data, used the beautiful soup library and extracted the articles. In total, we have collected around 500 articles

Common Crawl Data Collection:
For collecting the Common Crawl Data, used the example project code as reference https://rushter.com/blog/python-fast-html-parser/. But since the example is in Python 2 and the library WARC is not supported in Python 3, created an python 2.7 environment in Anaconda and used the code to collect the data. To download the data used the https://index.commoncrawl.org/ link. From the link, we get a unique WARC file path and then filtered out the relevant sports links. Used the major sports website links and extracted the WARC files. Did a lot of hits and trails with various links as most of them have irrelevant data. Then used the Python 3 beautiful soup code used for NYTimes and extracted the data. We have collected around 450 links and articles.

2. **Word Count:** Word count of the obtained data has been done using the MapReduce model (Hadoop). In the programming used, we imported re library to extract only alphanumeric characters. Also, we are filtering the stop words we got from the NLTK library stop words and also we have manually added additional ~15 stop words.

Tool: We used a mapper.py, reducer.py files provided in cse587.



```
cse587@CSE587:~$ hdfs dfs -mkdir /boom1
cse587@CSE587:~$ hdfs dfs -put /home/cse587/examplehadoop/badminton_processed.txt /boom1
cse587@CSE587:~$ hadoop jar hadoop-3.1.2/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -file /home/cse587/hduser/mapper.py -mapper mapper.py -file /home/cse587/hduser/reducer.py -reducer reducer.py -i
nput /boom1/badminton_processed.txt -output /boomm
2019-04-21 01:08:17,975 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [/home/cse587/hduser/mapper.py, /home/cse587/hduser/reducer.py] [] /tmp/streamjob5373354568295164811.jar tmpDir=null
2019-04-21 01:08:19,523 INFO impl.MetricsConfig: loaded properties from hadoop-metrics2.properties
2019-04-21 01:08:19,660 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2019-04-21 01:08:19,660 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2019-04-21 01:08:19,715 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!
2019-04-21 01:08:20,406 INFO mapred.FileInputFormat: Total input files to process : 1
2019-04-21 01:08:20,504 INFO mapreduce.JobSubmitter: number of splits:1
2019-04-21 01:08:20,797 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1748789314_0001
2019-04-21 01:08:20,798 INFO mapreduce.JobSubmitter: Executing with tokens: []
2019-04-21 01:08:21,128 INFO mapred.LocalDistributedCacheManager: Localized file:/home/cse587/hduser/mapper.py as file:/home/cse587/hadooptmpdata/mapred/local/1555823300948/mapper.py
2019-04-21 01:08:21,179 INFO mapred.LocalDistributedCacheManager: Localized file:/home/cse587/hduser/reducer.py as file:/home/cse587/hadooptmpdata/mapred/local/1555823300949/reducer.py
2019-04-21 01:08:21,317 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2019-04-21 01:08:21,319 INFO mapreduce.Job: Running job: job_local1748789314_0001
2019-04-21 01:08:21,318 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2019-04-21 01:08:21,323 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter
2019-04-21 01:08:21,344 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2019-04-21 01:08:21,344 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2019-04-21 01:08:21,446 INFO mapred.LocalJobRunner: Waiting for map tasks
2019-04-21 01:08:21,449 INFO mapred.LocalJobRunner: Starting task: attempt_local1748789314_0001_m_000000_0
2019-04-21 01:08:21,484 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2019-04-21 01:08:21,493 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2019-04-21 01:08:21,540 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
2019-04-21 01:08:21,574 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/boom1/badminton_processed.txt:0+340960
2019-04-21 01:08:21,612 INFO mapred.MapTask: numReduceTasks: 1
2019-04-21 01:08:21,750 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2019-04-21 01:08:21,751 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2019-04-21 01:08:21,751 INFO mapred.MapTask: soft limit at 83886080
2019-04-21 01:08:21,751 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2019-04-21 01:08:21,751 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2019-04-21 01:08:21,753 INFO mapred.MapTask: Map output collector class = org.apache.hadoop.mapred.MapTask$MapOutputBuffer
2019-04-21 01:08:21,759 INFO streaming.PipeMapRed: PipeMapRed exec [/home/cse587/./mapper.py]
2019-04-21 01:08:21,790 INFO Configuration.deprecation: mapred.work.output.dir is deprecated. Instead, use mapreduce.task.output.dir
2019-04-21 01:08:21,795 INFO Configuration.deprecation: map.input.start is deprecated. Instead, use mapreduce.map.input.start
2019-04-21 01:08:21,795 INFO Configuration.deprecation: mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
2019-04-21 01:08:21,795 INFO Configuration.deprecation: mapred.task.id is deprecated. Instead, use mapreduce.task.attempt.id
2019-04-21 01:08:21,795 INFO Configuration.deprecation: mapred.tip.id is deprecated. Instead, use mapreduce.task.id
2019-04-21 01:08:21,796 INFO Configuration.deprecation: mapred.local.dir is deprecated. Instead, use mapreduce.cluster.local.dir
2019-04-21 01:08:21,796 INFO Configuration.deprecation: map.input.file is deprecated. Instead, use mapreduce.map.input.file
2019-04-21 01:08:21,796 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords
2019-04-21 01:08:21,796 INFO Configuration.deprecation: map.input.length is deprecated. Instead, use mapreduce.map.input.length
2019-04-21 01:08:21,797 INFO Configuration.deprecation: mapred.job.id is deprecated. Instead, use mapreduce.job.id
2019-04-21 01:08:21,798 INFO Configuration.deprecation: user.name is deprecated. Instead, use mapreduce.job.user.name
2019-04-21 01:08:21,798 INFO Configuration.deprecation: mapred.task.partition is deprecated. Instead, use mapreduce.task.partition
```

In the HDFS, a directory was made using

```
Hdfs dfs –mkdir /filename
```

The obtained data has been sent to the HDFS directory using

```
Hdfs dfs –put /home/cse587/examplehadoop/dataContainingFolder.txt /filename
```

The MapReduce tool was used on the uploaded file using

```
hadoop jar hadoop-3.1.2/share/hadoop/tools/lib/hadoop-streaming-3.1.2.jar -file
        /home/cse587/examplehadoop/mapper.py -mapper mapper.py -file
        /home/cse587/examplehadoop/reducer.py -reducer reducer.py -input
        /filename/dataContainingFolder.txt -output /outputfolder
```

Once the data has been processed with MapReduce the created file (A part-00000 file has been created containing word count) has been taken from HDFS to disk using

```
hdfs dfs -get /outputfolder/part-00000  /home/cse587/examplehadoop/results.txt
```

3. **Word co-occurrence:** To find the word co-occurrences of the collected data we used MapReduce

Tool: We used Python (Pairsmapper.py) programming language as a tool for the MapReduce implementation.

To Visualise the word co-occurrence using Tableau, we faced issues with separators. So we used the number 98765 as the separator in visualisation.

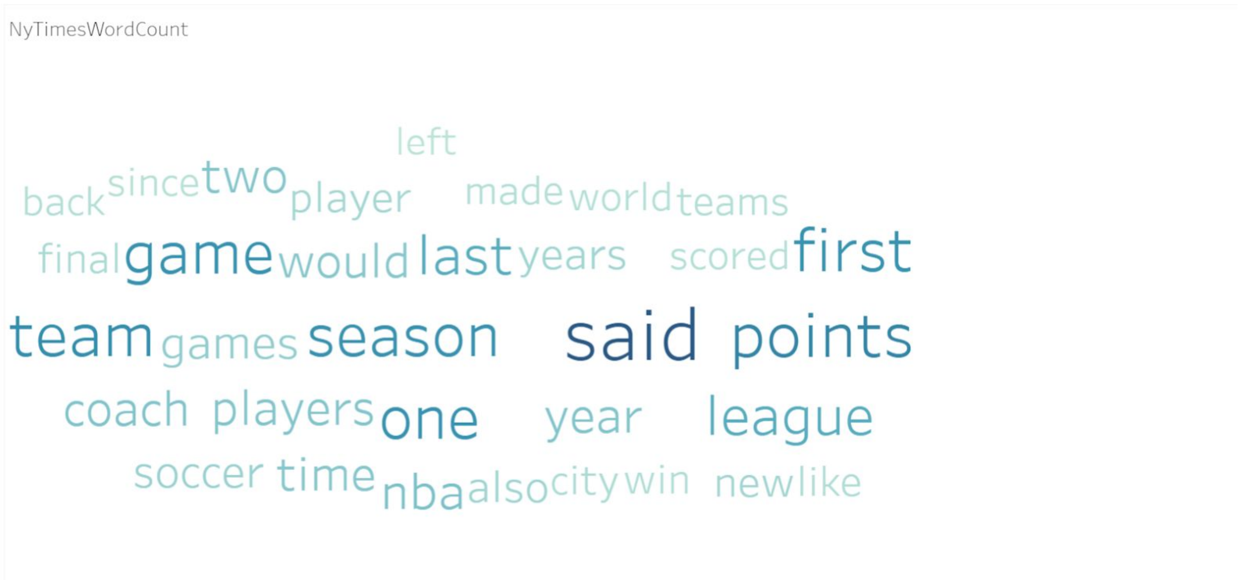
4. **Visualization:** To visualize the obtained word count and word co-occurrence, we used Tableau forming word cloud.

Link to our Tableau server: https://public.tableau.com/profile/vineel.kurma#!/vizhome/TwitterWordCoOccuranceAndWordCount/NyTimesWordCount

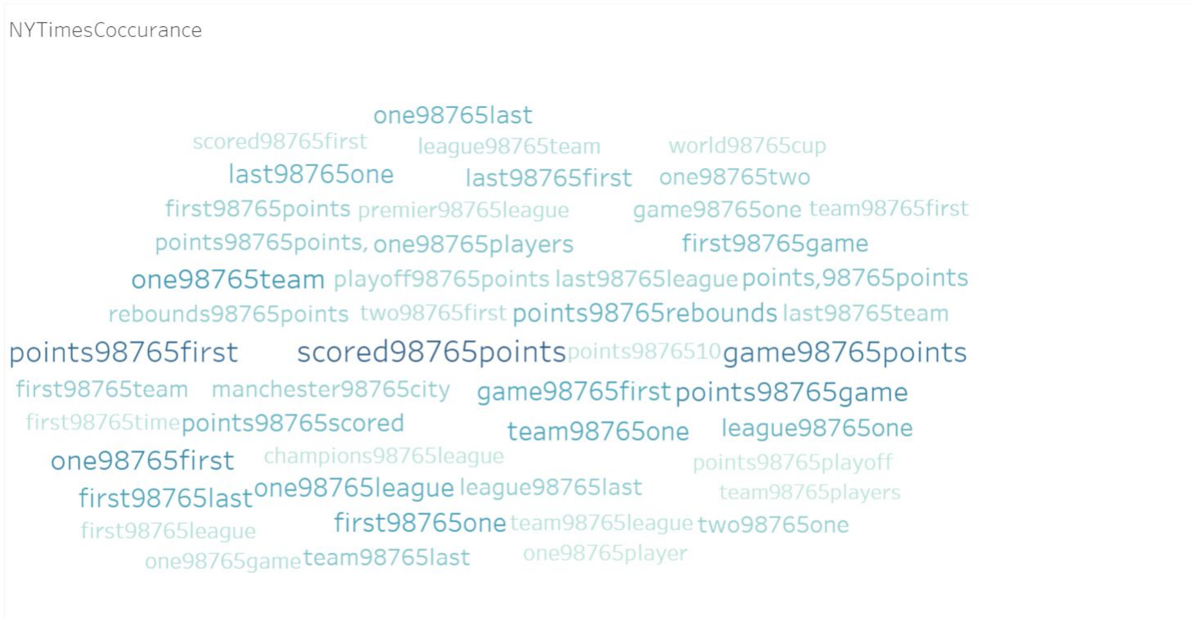
Below are the screenshots of the word clouds of NYTimes, Twitter, Common Crawl.

NYTimes:

Word count:

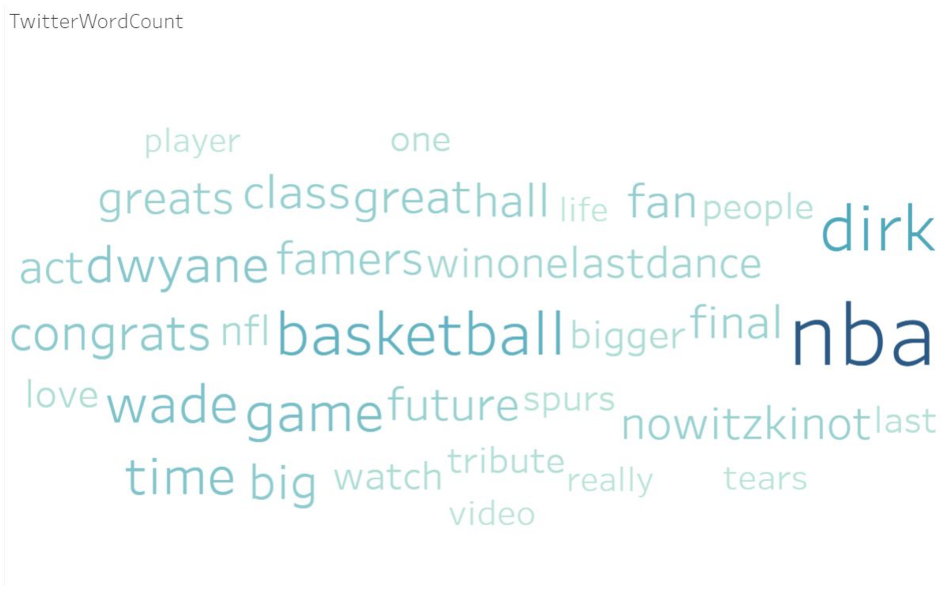


Word co-occurrence:

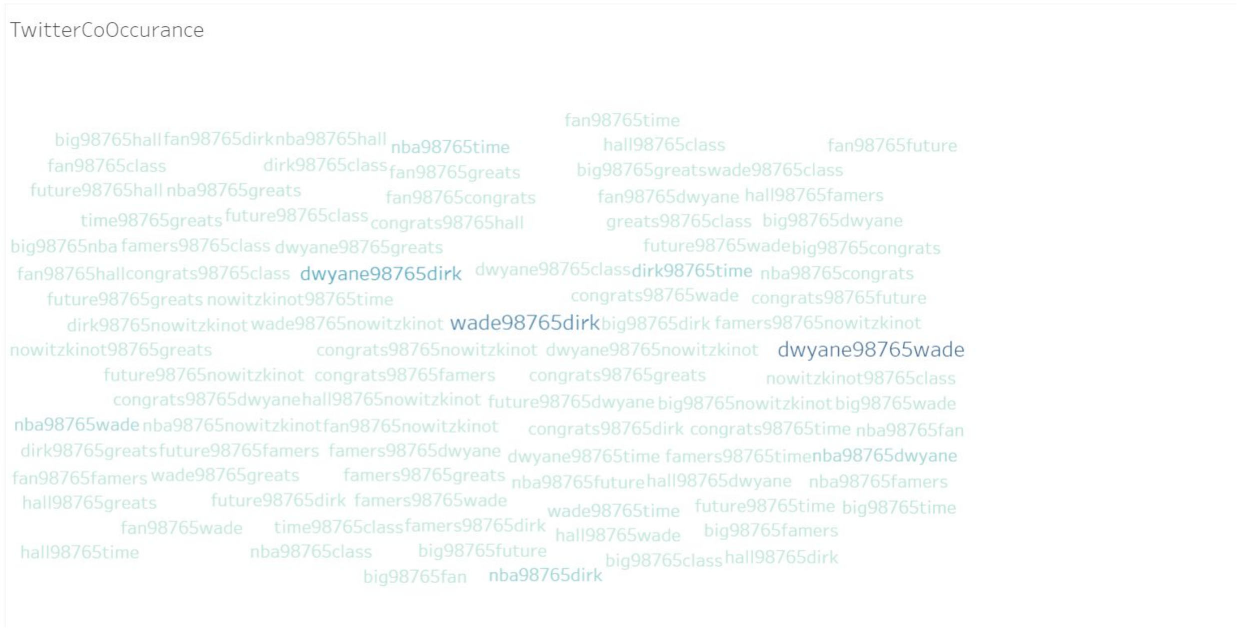


Twitter:

Word count:

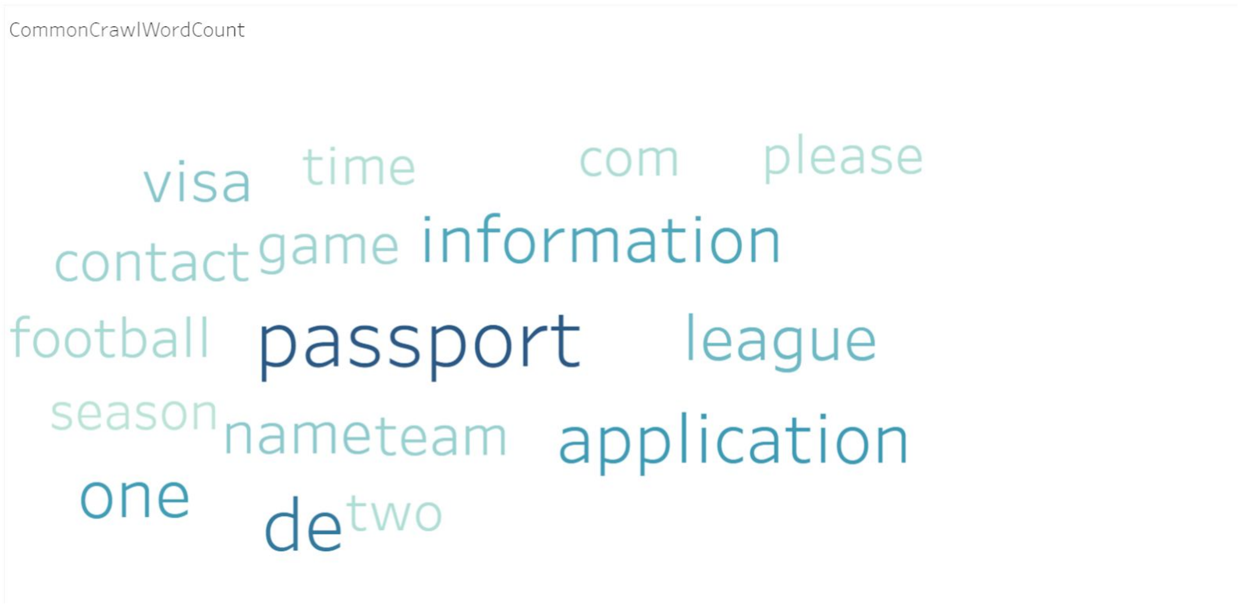


Word co-occurrence:

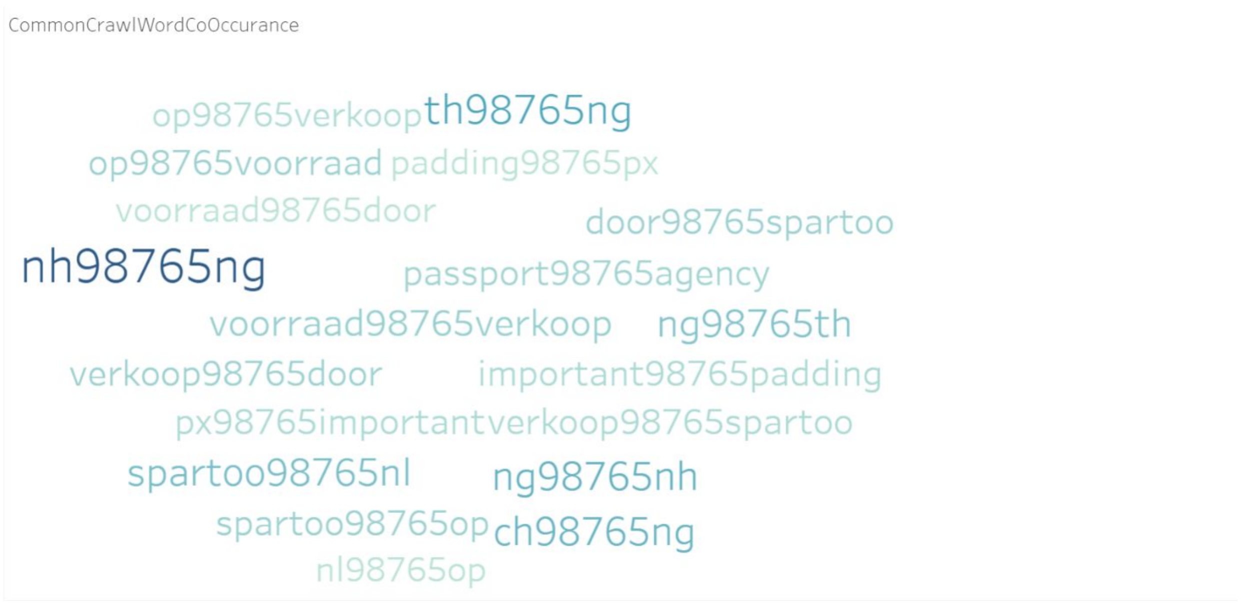


Common Crawl:

Word count:



Word co-occurrence:



Observations:

- The main issue observed in this project is there are a lot of irrelevant words which are not part of mainstream stop words from NLP libraries, hence the word cloud may not exactly reflect the keywords of our topic.
- Stopped doing stemming and lemmatization as we were getting undesired output with many invalid words.
- The problem with co-occurrence is that the pairs are duplicated I.E pair (a,b) was again observed as (b, a), to avoid that we need to store the pair so that we don't have to repeat the words.
- While working with common crawl, we have observed a lot of junk data, even after filtering with keywords. Due to lack of time, we could not proceed further with cleaning the data.