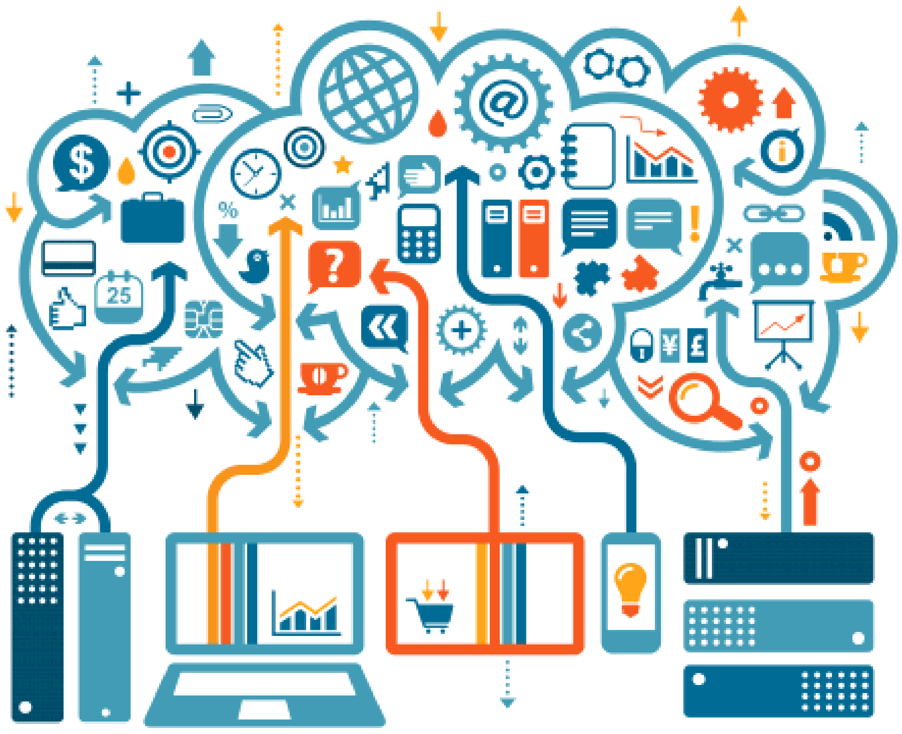
TDT4305: Project Phase 1
Exploratory Analysis of Twitter Dataset

Fredrik Bakken and Tor Arne Hagen

March 19, 2018



NTNU
Norwegian University of
Science and Technology

Task 1: Load RDD and Explore

Source Code and Results

- **Source code:** task_1.py (Github)
- **Results:** /data/results/result_1.tsv (Github)

Questions to be Answered

a) *How many tweets are there?*

Total number of tweets: **2715066**.

Solved by using the Spark function *.count()* on the entire RDD data set.

b) *How many distinct users (username) are there?*

Number of distinct usernames: **499822**.

Solved by using the Spark functions *.map()*, *.distinct()*, and *.count()* on the USERNAME row of the data set.

c) *How many distinct countries (country_name) are there?*

Number of distinct country names: **70**.

Solved by using the Spark functions *.map()*, *.distinct()*, and *.count()* on the COUNTRY_NAME row of the data set.

d) *How many distinct places (place_name) are there?*

Number of distinct place names: **23121**.

Solved by using the Spark functions *.map()*, *.distinct()*, and *.count()* on the PLACE_NAME row of the data set.

e) *In how many languages users post tweets?*

   Number of distinct languages: **46**.

   Solved by using the Spark functions *.map()*, *.distinct()*, and *.count()* on the LANGUAGE row of the data set.

f) *What is the minimum latitude?*

   Minimum latitude: **-54.87555556**.

   Solved by using the Spark function *.min()* on the LATITUDE row of the data set.

g) *What is the minimum longitude?*

   Minimum longitude: **-159.83019441**.

   Solved by using the Spark function *.min()* on the LONGITUDE row of the data set.

h) *What is the maximum latitude?*

   Maximum latitude: **69.83186826**.

   Solved by using the Spark function *.max()* on the LATITUDE row of the data set.

i) *What is the maximum longitude?*

   Maximum longitude: **153.03508445**.

   Solved by using the Spark function *.max()* on the LONGITUDE row of the data set.

j) *What is the average length of a tweet text in terms of characters?*

Average number of characters in each tweet: **87.2014098368**.

Solved by using the Spark functions *.map()* and *.mean()* on the length of the TWEET_TEXT row of the data set.

k) *What is the average length of a tweet text in terms of words?*

Average number of words in each tweet: **12.2284228081**.

Solved by using the Spark functions *.map()* and *.mean()* on the split (by empty space) length of the TWEET_TEXT row of the data set.

Task 2: Tweet Counts per Country

Source Code and Results

- **Source code:** task_2.py (Github)
- **Results:** /data/results/result_2.tsv (Github)

Questions to be Answered

a) *Find the total number of tweets posted from each country and sort them in descending order of tweet counts. For countries with equal number of tweets, sorting must be in alphabetical order.*

To solve task 2, the tweets are first mapped by the origin country before they are counted. The output items are then sorted by the built-in Python method for sorting elements, where the tweet count is sorted in descending order and the countries with the same amount of tweets are sorted in alphabetic order.

Task 3: Geographical Centroids per Country

Source Code and Results

- **Source code:** task_3.py (Github)
- **Results:** /data/results/result_3.tsv (Github)

Questions to be Answered

a) *Write a code (named "task_3") that outputs in a TSV file the latitude and longitude of the centroids and the names of the countries, in the form of <country_name>tab<latitude>tab<longitude>.*

The final code solution in task 3 uses a method which creates an extreme run-time of more 30min. Eventhough the execution time is extremely slow because of so many Spark tasks, it still gives the correct result.

A more efficient solution would take advantage of the aggregation method.

b) *Visualize the results in CartoDB.*

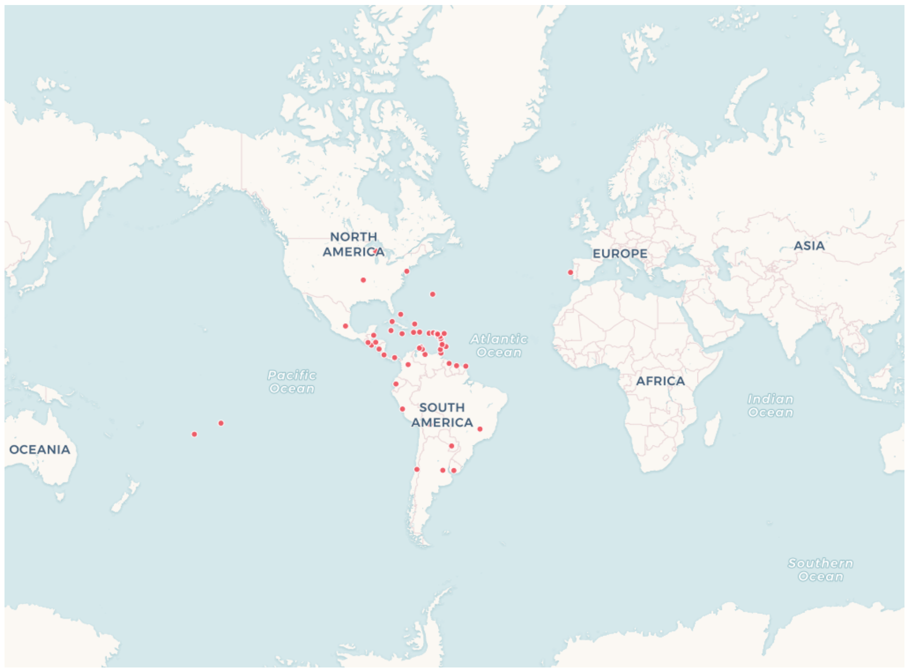


Figure 1: Result Visualization in CartoDB.

Task 4: Most Active Hours per Country

Source Code and Results

- **Source code:** task_4.py (Github)
- **Results:** /data/results/result_4.tsv (Github)

Questions to be Answered

a) *Calculate local time for each tweet (UTC time + timezone offset) and find the 1-hour interval with maximum number of tweets for each country in the form of <country_name>tab<begining_hour>tab<tweet_count>3. Times should be rounded down to the hour, in 24 hour scale, so that a tweet posted between [13:00,14:00) should be interpreted as posted at 13. Write a code (named "task_4") that writes the results in a TSV file and name it "result_4.tsv".*

   In task 4, one starts by mapping the tweets by COUNTRY_NAME, UTC_TIME, and TIMEZONE_OFFSET, where the two time rows are added together. Then the total time is transformed to actual hour-based time using the datetime library. Next, tweets for each hour for every country is counted and compared to find the hour where most tweets are posted for each country.

Task 5: Tweet Counts per City

Source Code and Results

- **Source code:** task_5.py (Github)
- **Results:** /data/results/result_5.tsv (Github)

Questions to be Answered

a) *Find the number of tweets from each city in US (place_type = "city" and country_code = "US") in the form of <place_name>tab<tweet_count>. Write a code (named "task_5" that writes the results in a TSV file named "result_5.tsv" in descending order of tweet counts. For cities with equal number of tweets, sorting must be in alphabetical order.*

   In task 5, one first filters the tweets by the COUNTRY_CODE 'US' and PLACE_TYPE 'city'. Then, each PLACE_NAME is mapped with a value, before they are added together and sorted in descending order.

Task 6: Frequent Words in a Country

Source Code and Results

- **Source code:** task_6.py (Github)
- **Results:** /data/results/result_6.tsv (Github)

Questions to be Answered

a) *Find the 10 most frequent words (in lowercase) and their frequencies from the US, excluding the words shorter than 2 characters (length <2) and the words from the stop words file. Write a code (named "task_6") that writes the results in a TSV file named "result_6.tsv" in the form of <word>tab<frequency>.*

To solve this task, one first reads the stop words from the stop_words.tsv file and appends all the words to a list. Thereafter, processing of the tweets start by filteringer on the COUNTRY_CODE 'US'. All words TWEET_TEXTs are then set to lowercase and split to lists, before stop words and short words are filtered out. The words that are left are now mapped with (word, 1), to represent the count for each word. ReduceByKey is then used to count the total instances of each word, before sorting the words by their count in descending order. The 10 most used words are then taken out of the result and appended to result_6.tsv.

Task 7: Frequent Words per City

Source Code and Results

- **Source code:** task_7.py (Github)
- **Results:** /data/results/result_7.tsv (Github)

Questions to be Answered

a) *Find the 5 cities in the US with the highest number of tweets (place_type = "city" and country_code = "US", ordered by their tweet counts/alphabetical). For these 5 cities, find the 10 most frequent words ordered by their frequency, ignoring the stop words from the file and excluding words shorted than 2 characters (length <2). Write a code (named "task_7") that writes the results in a TSV file named "result_7.tsv" in the form of <place_name>*

tab<word1>tab<frequency1>tab<word2>tab<frequency2>... <word10> tab<frequency10>.

In task 7 one first finds the top 5 cities in the US by filtering on COUNTRY_NAME and PLACE_TYPE, before mapping each place with a value. Then reducebykey and sorting in descending order is used to find the top 5 cities. Afterwards, a loop is initiated for loop through each of the cities to find each of their top 10 words as was done in task 6.

Task 8: Explore using Spark SQL and Dataset API

Source Code and Results

- **Source code:** task_8.py (Github)
- **Results:** /data/results/result_8.tsv (Github)

Questions to be Answered

a) *Number of tweets?*

Total number of tweets: **2715066**.

Solved by using the Spark SQL function *.count()*.

b) *Number of distinct users (username)?*

Number of distinct usernames: **499822**.

Solved by using the Spark SQL functions *.count()* and *.distinct()*.

c) *Number of distinct countries (country name)?*

Number of distinct country names: **70**.

Solved by using the Spark SQL functions *.count()* and *.distinct()*.

d) *Number of distinct places (place name)?*

Number of distinct place names: **23121**.

Solved by using the Spark SQL functions *.count()* and *.distinct()*.

e) *Number of distinct languages users post tweets?*

Number of distinct languages: **46**.

Solved by using the Spark SQL functions *.count()* and *.distinct()*.

f) *Minimum values of latitude and longitude?*

Minimum latitude: **-54.87555556**.
Minimum longitude: **-159.83019441**.

Solved by using the Spark SQL function *.min()*.

g) *Maximum values of latitude and longitude?*

Maximum latitude: **69.83186826**.
Maximum longitude: **153.03508445**.

Solved by using the Spark SQL function *.max()*.