NTNU – Trondheim
Norwegian University of
Science and Technology

TDT4305 Big Data Architecture

Spark Programming Exercise

Andreas Drivernes
Stein-Otto Svorstøl

April 26, 2016

Chapter 1

Exploratory Analysis of Foursquare Dataset

1.1 Task 1-3

1. Use `SparkContext.textFile` to read in checkins and countries to RDDs.
2. Use the `map` functions in `task1/mappers.py` to manipulate each row of the data input. Remove first row, split on tab and convert to object (`record_to_object`), calculate local time (`calculate_local_time`), assign a city the naive way (`find_nearest_city_and_country`, iterating through the entire list of countries for each checkin).
3. Persist the result for further usage.

1.2 Task 4

1. `map` the wanted key (`user_id`, `checkin_id`, etc.)
2. Use `distinct` to remove all duplicates.
3. Use `count` to get the length of the RDD.

This gave the following results:

(a) How many unique users are represented in the dataset?
Number of distinct user IDs : 256307

(b) How many times did they check-in in total?
Number of total checkins: 19265256

(c) How many check-in sessions are there in the dataset?
Number of distinct session IDs : 6338302

(d) How many countries are represented in the dataset?
Number of distinct country : 77

(e) How many cities are represented in the dataset?
Number of distinct cities : 413

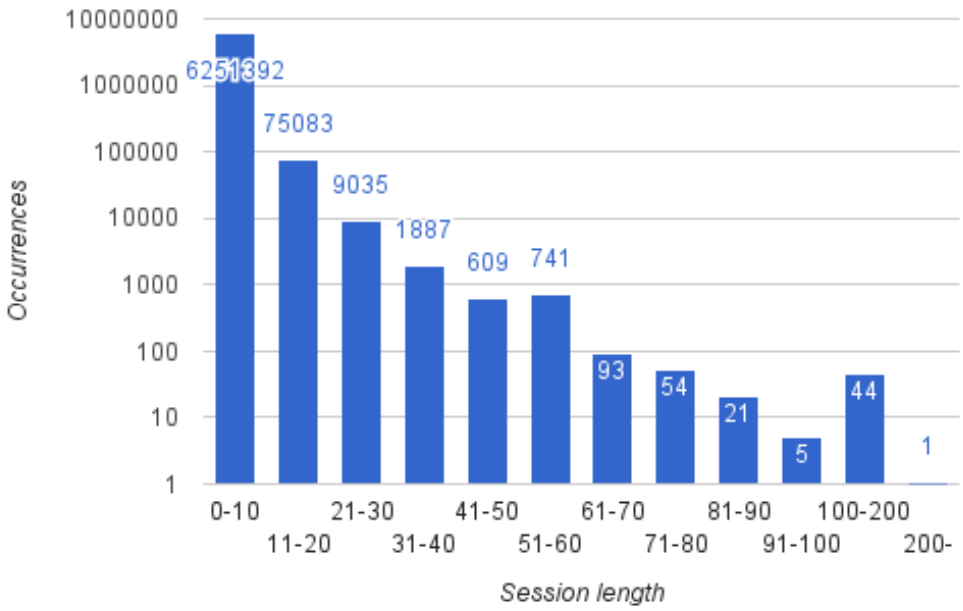

Figure 1.1: Histogram showing the number of occurrences of grouped session lengths. Logarithmic scale.

1.3 Task 5

1. `map` `session_id` as key, and 1 as value.
2. Use `reduceByKey(lambda a, b: a + b)` to accumulate the number of checkins per session.
3. `map` the session lengths as key, and 1 as value.
4. Use `reduceByKey` again to find lengths of sessions as number of checkins.
5. `saveAsTextFile` to write the result to disk.

We used Google Spreadsheets to produce a histogram with occurrences of session lengths as seen in figure 1.1.

1.4 Task 6-7

A screenshot from CartoDB can be seen in figure 1.2. The map can be further explored here.

1. `map` `session_id` as key, and the rest as value.
2. `groupByKey` to get an iterator per `session_id`.

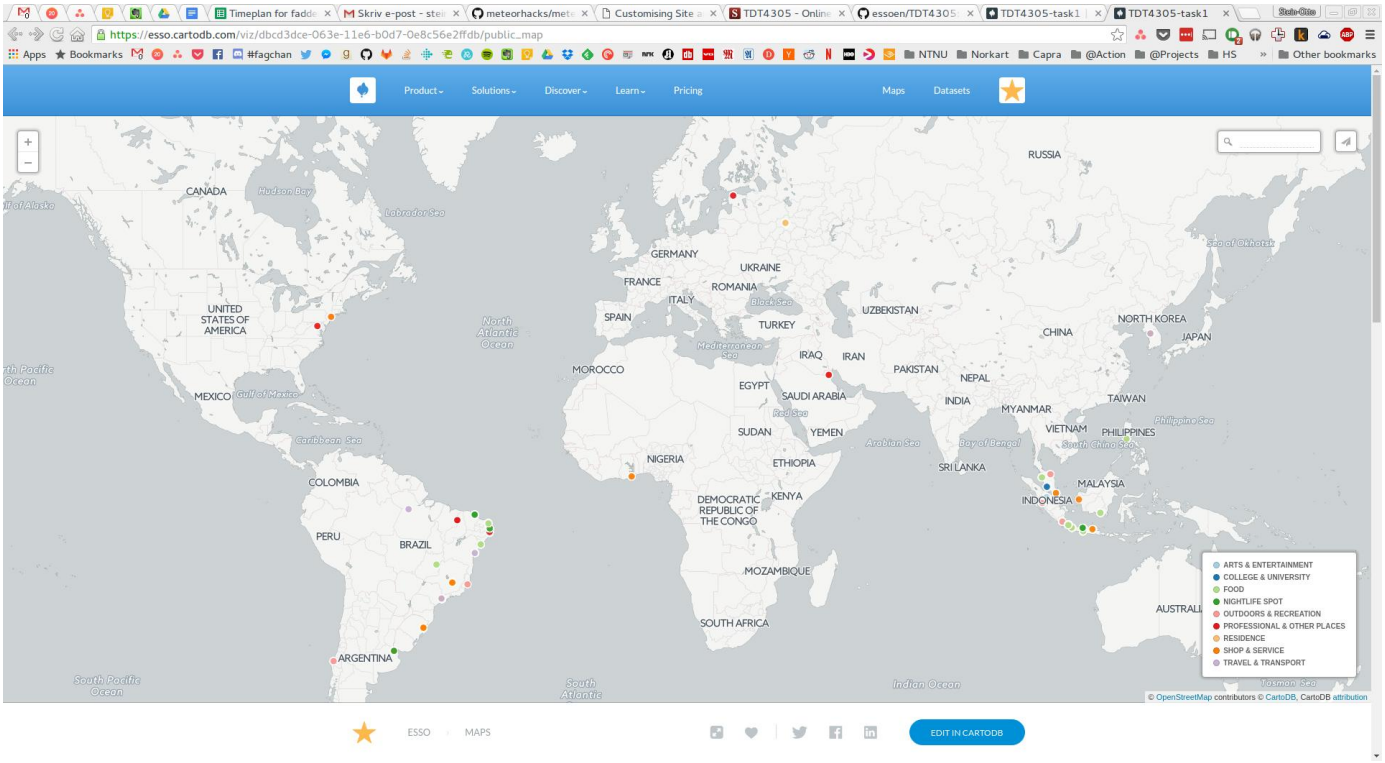

Figure 1.2: Screenshot of the map produced with CartoDB.

3. `mapValues(lambda o: list(o)` to get a list per `session_id`.
4. `filter` out every list with length less than 4.
5. `map` in the session distance (`calculate_session_distance`) as a third tuple element.
6. `filter` out sessions with distance smaller than 50.
7. Use `takeOrdered(100, key=lambda o: -len(o[1])` to return a list with the 100 longest sessions.
8. Open a result file, and write tab separated values to disk again.

Chapter 2

Sentiment Analysis on Twitter

To solve this exercise we developed the following algorithm:

1. Load the dataset
2. Use `filter` in Spark to remove all the tweets that are not in english, and are not American cities. That is we removed tweets that had `lang` unequal to "en", `country_code` unequal to "US" and `place_type` unequal to "city".
3. With the `map`-method in Spark we created a new dataset on the limited one. They new dataset consist of `(key, value)`-pairs, where they key is a tuple `(city_name, weekday)`. The value is the tweet text.
4. A new `map`-call calculates the sentiment for each tweet. We did not do it in the last step, as doing it as a seperate step saved us 2 seconds.
5. Now we simply used `reduceByKey` in Spark to combine the elements that had the same key, that is those with the same city and weekday. We just added the sentiment, that is the value, for the tuples.
6. Now the dataset had one entry for each city and weekday, and we we combined it all to one single dataset, the used `saveAsTextFile` in Spark to write to file.