# DAT470/DIT065
# Computational techniques for large-scale data

Assignment 3

**Deadline:** 2024-04-29 23:59

The file `/data/2024-DAT470-DIT065/twitter-2010_full.txt` contains a snapshot of the *social network* of Twitter from 2010 [1]. That is, it contains the description of a directed graph of *follows* relations: which users follow which other users. The data is formatted as follows, one line per user:

$$id_0\texttt{:}\quad id_1\texttt{,}\ id_2\texttt{,}\ \ldots$$

The *id*s are the Twitter IDs of users, unique integers that identify a user account (the user may change their handle but they cannot change their ID). The interpretation is that the user $id_0$ *follows* the users $id_1$, $id_2$, .... The *follows* relation is not reflexive, symmetric, nor transitive, so users do not follow themselves, and users need not follow back their followers. Not all users have followers, and not all users follow anyone.

The files `/data/2024-DAT470-DIT065/twitter-2010_`$x$`.txt` where $x$ can be `1k`, `10k`, `100k`, `1M`, or `10M` represent down-sampled versions of the dataset (a subgraph induced on a random subset of users). It will make sense to use these for debugging and perhaps using the `10M` variant for making the running time measurements, as the main dataset is rather large (approximately 13 GiB).

**Note.** Remember that you *cannot run any other commands* in a MRJob file. The file must only contain the class. This is because of limitations of the Hadoop architecture. Your measurements must be carried out in a *separate Python file*. You are provided a skeleton file for this purpose (`mrjob_twitter_measure.py`); you needn't return this file. You only need to return the MRJob class files.

Also, MRJob is not installed in the default Anaconda environment. You must specify the environment `mrjob`, e.g., by using the flag `-e mrjob` when running using the `run_job.sh` script.

In case you are interested, the original dataset[1] is stored in the files `/data/2024-DAT470-DIT065/twitter-2010.txt.gz` and `/data/2024-DAT470-DIT065/twitter-2010-ids.csv.gz`.

## Problem 1: Followed (12 pts)

(a) Describe a MapReduce algorithm determines the Twitter ID of the person who follows the largest amount of users, the number of users this person follows, the average number of users followed, and the number of accounts that do not follow anyone.

Pay attention to describing accurately the map and reduce operations that your algorithm makes use of. If you want, you can also illustrate your algorithm with a figure, but this is not necessary. (2 points)

(b) The file `mrjob_twitter_follows.py` contains the interface for a MRJob implementation that where you will need to fill in the blanks by implementing your algorithm. Implement the algorithm. (4 points)

[1] From `https://snap.stanford.edu/data/twitter-2010.html`.

(c) Measure the scalability of your algorithm on 1, 2, 4, ..., 32 cores. Plot the empirical speedup as the function of cores. Use the 10M dataset for scalability experiments. In addition to the plot, report the single-core runtime on the dataset. (3 points)

(d) Run your implementation (with a large number of cores) on the full dataset. Report the values (ID and number of accounts followed by the account that has follows most accounts, average number of accounts followed, and the number of accounts that follow no-one. (2 point)

(e) The account who follows most other accounts still exists and is active. Determine whose account it is. (1 point)

## Problem 2: Followers (12 pts)

We will now invert the relation from *follows* to *is followed by*.

(a) Describe a MapReduce algorithm determines the Twitter ID of the person who has the largest amount of followers, the number of followers this person has, the average number of followers, and the number of accounts that do not have any followers.

Pay attention to describing accurately the map and reduce operations that your algorithm makes use of. If you want, you can also illustrate your algorithm with a figure, but this is not necessary. (2 points)

(b) The file `mrjob_twitter_followers.py` contains the interface for a MRJob implementation that where you will need to fill in the blanks by implementing your algorithm. Implement the algorithm. (4 points)

(c) Measure the scalability of your algorithm on 1, 2, 4, ..., 32 cores. Plot the empirical speedup as the function of cores. Use the 10M dataset for scalability experiments. In addition to the plot, report the single-core runtime on the dataset. (3 points)

(d) Run your implementation (with a large number of cores) on the full dataset. Report the values (ID and number of followers by the account that has most followers, average number of followers, and the number of accounts that have no followers. (2 point)

(e) The account that the most followers, unfortunately, is defunct. Still, it should be relatively easy to find out who that was. Determine whose account it used to be. (1 point)

## Hints

- Remember that the code for measuring the execution time must be in a different file from your job class.
- Look up the documentation of MRJob, it contains useful examples.
- You may need more than one job step in Problem 2.

- You will need to use combiners to achieve reasonable performance.
- Remember that Markov and Bayes have a different directory structure, so you will need to call your script with appropriate parameters to find the files.
- Do not hog resources: if you only measure the running time with one core, do not request 32.
- Start experimenting with the small files first. You may even do this locally. Also, you can compute the baseline values yourself, e.g., with NumPy for the small files (it doesn't take that long).

## Returning your assignment

Return your assignment on Canvas. Your submission should consist of a report that answers all questions as PDF file (preferably typeset in LaTeX) called `assignment3.pdf`. In addition, you should provide the code you used in Problems 1 and 2 as `assignment3_problem1.py` and `assignment3_problem2.py`, respectively. The code must match the interfaces of `mrjob_twitter_follows.py` and `mrjob_twitter_follows.py`; the command line parameters must not be changed, and output must be correct. Use a separate file for making the runtime measurements; you do not need to return this file. Do *not* deviate from the requested filenames and do *not* produce the plots in these files; these files will be used for evaluating the quality of your implementations automatically.

## References

[1] Haewoon Kwak et al. "What is Twitter, a Social Network or a News Media?" In: *Proceedings of the 19th international conference on World wide web (WWW '10)*. 2010. DOI: https://doi.org/10.1145/1772690.1772751.