Roskilde University

NiNE

Green Hackathon 2023

GSA - Green Sentiment Analysis

This assignment is part of the **Green Hackathon 2023** with focus on Green Sentiment Analysis.
The assignment is developed by
Associate Prof. Maja H Kirkeby, Roskilde University and Nine A/S.
and approved by the judges:
Prof. Kerstin I. Eder, University of Bristol and
Associate Prof. João Saraiva, Minho University, Portugal.

The Green Hackathon 2023 is sponsored by Nine A/S, Energy Cluster Denmark, and IT Branchen (Danish ICT Industry Association).

In connection with an increasing need for monitoring and control of data on the Internet, we need an easy and quick way to assess whether a text is positively or negatively charged.

We, therefore, need an energy-efficient solution that can take texts in different formats, analyze them, and provide a categorisation: 0 for negative, and 1 for positive.

To solve the task, we will use a data set with approx. 30,000 movie reviews and sentiments as training data. The data comes in different formats and with slightly varying content (not just plain text).

This Hackathon is part of a research project on Energy Consumption of Software. Submitted solutions are published as **open source** under the GNU General Public License v3.0 or later. The aim of this Hackathon is to learn from you and your experiences. After the event, we will first assess the output of your program with respect to the above functional specification. We will then measure the energy consumed by your program. The winning submission is the one that consumes the lowest energy and meets the functional specification. In addition, we will (1) send each of you a small questionnaire to gain insights into your line of thought and experiences, and (2) analyse all the individual parts of your solutions. We aim to publish a paper with your solutions and experiences.

All data and descriptions can be found at
https://github.com/SustainableSoftware/Green-Hackathon-2023.git

Assignment description

To solve the task, the team must do the following:

1) Develop a *sentiment analysis* that can assess whether a film review is positively (represented with 1) or negatively (represented with 0) charged. Training the model can be done using the 30,000 film reviews and sentiments that we provide, referred to as *training data*. Testing of the system takes place with another 10,000 film reviews, which are not handed out; this set will be referred to as *test data*. There is also a small extra set of data, which are only meant for checking that your setup of the evaluation-script works correctly before submission; this set will be referred to as *development data,* see more about this later.

2) The solution you create is supposed to run on two Raspberry Pis: RP1 and RP2, that emulate a client and a server. Thus, you will need to develop both a *client program,* which will run on RP1, and a *server program,* which will run on RP2. On RP1, your *client program* must load and prepare the test data consisting of 10,000 movie reviews with a unique ID, and send the reviews to RP2. On RP2, your *server program* must receive and analyze the reviews, and categorize whether they are positive 1 or negative 0. RP2 must send the prediction back to RP1, which saves the prediction together with the review ID; this pair must be saved into an output file on RP1, named `result.csv`. The output file must be a two-column csv file with `id` as first column and `sentiment` as the next column; the first line in the file should be the column name and the values should be separated by a comma. For example, here is an output file with two review predictions:

```
id,sentiment
18,1
23,0
```

3) Your programs running on RP1 and RP2 will communicate via **ethernet** and **use the static IP addresses** provided in `Raspberry-specs.doc`. RP1 and RP2 are two Raspberry Pi Model 4 B (4GB) with a standard OS installation (with the latest updates), please see `Raspberry-specs.doc` for more details.

4) A submitted solution is considered *functionally* sufficiently accurate if it gives a minimum of 80% correct answers in each of the categories. We will use the program `evaluate_solution.py` for evaluating this criterion and measuring the energy. The program `evaluate_solution.py` is expected to run on the client, RP1, and it is expected to initiate the execution of your client program, which then loads the data, obtain the classification for the data, and save each pair of ID number and classification into a new line in the file `result.csv`.
We will evaluate your solution by executing the command
`python3 evaluate_solution.py` on RP1.
Since the server and client programs are connected, it is important that you, in your final solution,

include a description of how we start the server program on RP2 and in which order this should happen.
Therefore, it is important that you update lines 1-19 in `evaluate_solution.py` with

- a call_command that it initiates your client program.

See comments inside `evaluate_solution.py` for more information.
Please include your updated version of `evaluate_solution.py` in your final solution. **We require** that you put your (1) *client program*, (2) the updated `evaluation_solution.py`, and (3) your output file `result.csv` in the same folder. Make sure you **include instructions** on where to store these files on the RP1 and where to install the server program on RP2 on in your final solution.

5) The movie review data will be provided in three csv files:
a) Training data `movie_training.csv` with 30,000 reviews and categorizations; this can be used when developing your sentiment analysis,
b) Development data `movie_dev.csv` with 500 reviews and ID numbers; this is similar in format to test data. This is the file that RP1 is supposed to load data from,
c) Development checklist data `movie_dev_checklist.csv` with the same 500 reviews and ID numbers as `movie_dev.csv`, and in addition, their correct sentiment. This file can be given as input to `evaluate_solution.py` to be able to ensure that the evaluation program works as intended. Note that the correctness scores found by this small set may very well differ from the correctness score during the energy evaluation of your solution.
When your solution is evaluated, we will use a pair of data files similar to `movie_dev_checklist.csv` and `movie_dev.csv`, but with with 10'000 reviews instead of 500. The data files we will use during evaluation are called `movie_test_checklist.csv` and `movie_test.csv`.

6) The winner is the functional solution with the lowest collective energy consumption of RP1 and RP2; the energy consumed during the time when `evaluate_solution.py` is executing. An *energy battle* may occur; an energy battle is when the two or more functional solutions consume the same amount of energy. In case of an energy battle, the solution with the highest total number of correct answers will win.

7) For your solution, you may use any type of **freely** available software, this includes programming language, compilers, libraries, packages, etc. Together with your solution, you'll need to
a) **an `RP1` folder** to be downloaded to RP1 containing all files needed for RP1
b) **an `RP2` folder** to be downloaded to RP2 containing all files needed for RP2
c) **provide two bash scripts** (one for RP1 and one for RP2) that installs all required programs and packages on the Raspberry Pis, creates executables (if needed). See https://youtu.be/2quic2W5RbA?t=276 for an introduction to creating installer scripts for Raspberry Pi, and https://www.youtube.com/watch?v=jURrRJt3UWI for a short introduction to bash scripts.
d) **provide a readme file** with instructions on
i)how to download the solutions and where to store it,
ii)how to run the bash scripts,

iii) how to start the server program and the client program, and in which order to do so.

And, in addition,

iv) a list with the names of all team members

v) a short 3-5 line description of your program and

vi) a list of the required software with version numbers.

e) **Include a** *GNU General Public License v3.0 or later* in your solution. See more and find the license text here: https://choosealicense.com/licenses/gpl-3.0/.

NB: If we cannot run your solution due to missing software or if your program does not execute correctly after following your instructions, your program will not be considered functional.

Submission instructions

You need to upload your solution before the deadline.

The **deadline** is: **21:00 CEST, Sunday, the 14th of May 2023.**

To submit your solution, you or your team should

A. create a **private** GitHub repository (include a list of all team members' names in the readme file)

B. Invite MHKirkeby as admin in your repository (and only your team mates)

C. Upload your solution into the github repository before the deadline; the last push commit before deadline is the one that counts.

Please carry out step A and step B as soon as possible.

Please be aware that we will copy the content of your private repository to our public repository, afterwards; contact Maja H Kirkeby via majaht@ruc.dk before 1st of June 2023 if you prefer that we remove your names from the readme file before publishing.

Make sure that your uploaded solution contains all the specified parts in the above.

If any issues occur with the submission, feel free to contact Maja H Kirkeby via email: majaht@ruc.dk

Winner announcement

We will send all participants an invitation to the winner announcement.

The **winner announcement** will be 10:00 CEST, Thursday, 1st of June 2023.

All data and descriptions can be found at

https://github.com/SustainableSoftware/Green-Hackathon-2023.git