Assignment 5

Evaluating Environmental Impact of Exam Portfolio

Date: 02/05/2024

Laura Givskov Rahbek

Description

This folder contains assignment 5 for Language Analytics. The objective of the assignment is to think critically about the effects and impact machine learning has on the environment, to write code that can extract approximate benchmarks showing this impact and to present and discuss the results in an understandable way. More specifically, the package `CodeCarbon`'s class `EmissionsTracker` is utilised to measure the approximate CO_2-equivalents (CO_2eq) in kg for assignments 1-4 in this repository. The resulting values are used to investigate the environmental impact of the different tasks performed in the four assignments.

The `assignment-5` folder contains:

- An `out` folder containing the .csv files with the emission data from each of the four assignments.
- A notebook `emissions.ipynb` with data wrangling and visualisations.
- A folder, `plots`, with the visualisations produced.

Usage and Reproducing of Analysis

All code used to handle the emmisions data as well as visualisations of these can be found in the notebook. If relevant, the packages needed to run the `emissions.ipynb` notebook can be installed by running `bash setup.sh` in the terminal. However, the dataframes and plots are visible in the notebook as is without needing to run it.

To track the emissions from the assignments, the `EmissionTracker` from `CoddeCarbon` was used, the code for this is embedded in the code for the four assignments. Below, a short example of the implementation is shown:

```python
def carbon_tracker(outpath):
    """ The function initalizes the carbon tracker """
    tracker = EmissionsTracker(project_name="Assignment-1",
                               output_dir=outpath)
    return tracker

def function(argument):
    """ Example function """
    tracker.start_task("Example task")
    some_output = some_operation(argument)
    tracker.stop_task()
    return some_output

def main():
    tracker = carbon_tracker(outpath)
    some_output = function(argument)
    tracker.stop()
```

For all scripts, the tracker is initialised with a project_name, reflecting the given assignment. Then depending on how it fit in the different scripts and functions, 'task' functionality was implemented, allowing for tracking of individual tasks. Each task was named according to the task at hand. When stopping the tracker in the `main()` function, all the different task trackings are saved to the outpath, in this case the`out` folder in assignment 5. The overall imapct and emissions for each assignment is inferred by summing the emissions from the individual tasks in each assignment.

Discussion

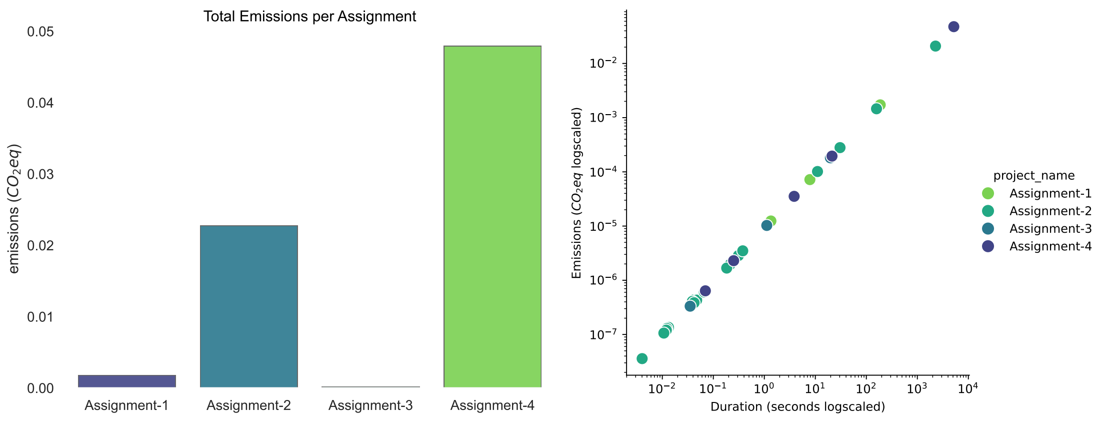
Before discussing the results of the environmental impact analysis, it is worth noting that these values are estimates and approximations to the actual emissions caused by the code run in these assignments. Nonetheless, these approximations are a useful representation and are used to base the following discussion of the cost of machine learning off of. Firstly, none of the code, analysis or machine learning conducted for this exam is of great importance, except for learning, which of course is why it has been run and made. I would not be able to argue that it is worth the substantial amount of CO_2eq to analyse the emotional differences during Game of Thrones seasons based only on the scripts. Considerations discussed are therefore concerned with the general use of machine learning, and the considerations when e.g. implementing grid search or using a much more expensive model that only does marginally better than a less expensive one.

Total Emissions for each Assignment

Assignment	Emissions (CO_2eq)	Duration (s)
Assignment 1	0.001807	196.275
Assignment 2	0.022769	2473.339
Assignment 3	0.000192	20.897
Assignment 4	0.047947	5208.031

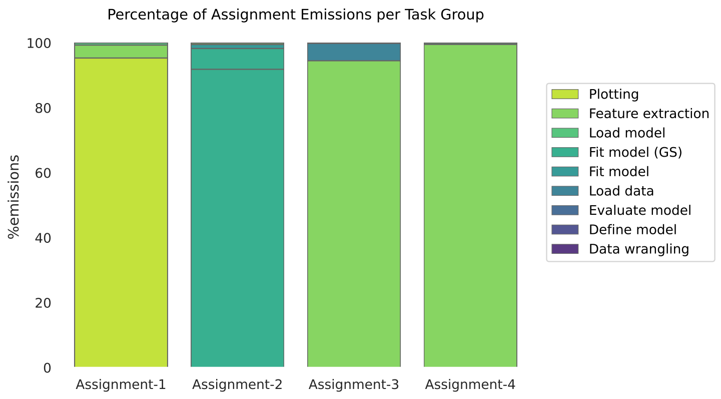
The above table showcases the total emissions (in CO_2eq) and total duration (in seconds) for each of the four assignments. Assignment 4 has the largest impact, followed by Assignment 2, Assignment 1 and lastly Assignment 3, as seen in the bargraph below. This pattern seem to be mirrored in the duration of the assignment. To investigate this apparent correlation between duration emissions, the scatterplot below was made. The scatterplot has emissions on the x-axis and the duration on the y-axis. Both axes have been logscaled to be able to see the points. Because of the large differences in the dataset, a plot with regular axes results in one point in the far right corner and the rest cluttered together. As we are interested in the correlation between the two variables, the difference in magnitude can be ignored for this. There seems to be an almost perfect correlation between seconds and CO_2eq. This is also reflected in the emissions_rate measure, which is almost identical for all tasks. The bar plot visualising each task's emission rate (CO_2eq/s) can be found here or in the `emissions.ipynb` notebook. The scale of the current analysis is quite small; it is not possible to conclude anything from it, but based purely on the correlations visualised in the plot below, it seems that (without needing `CarbonTracker`) the relative amount of emissions can be inferred from the duration of a run. And that if emissions should be limited, run time should be limited.

Total Emissions: per Assignment (left) and per Duration (right)



To take closer look at which tasks emitted the most within each assignment, the plot below was made. It visualises the emissions for each task-type in the assignments relative to the total emissions from the given assignment. Starting with the least impactful one: Assignment 3. In Assignment 3, the task with the largest impact is the loading and usage of the word-embedding model `glove-wiki-gigaword-50`. It is used to identify ten of the closest (most similar) words to a given keyword - but that is it. It is pretrained and used for a very specific task, resulting in a comparatively low total emission for the assignment. Assignment 1 is the second least impactful with the task of feature extraction, which consisted of the use of a `spaCy` model extracting and identifying six different types of tokens (parts of speech and named entity recognition). However, Assignments 1 and 3's impact is difficult to compare to the impact of both Assignment 2 and 4, as seen in the plot above visualising total emissions. The most impactful assignment is Assignment 4; where the primary task causing this level of emissions is the use of a pretrained large language model, handeling a large amount of text. The second most impactful assignment is Assignment 2; where a large amount of the emissions comes from the gridsearch performed on the Convolutional Neural Network Classifier from `scikit-learn`. As metioned in Assignment 2, the performance of the model resulting from the gridsearch was not improved in a way that justifies this amount of emissions.

Percentage of Total Emission per Assignment from Each Task Group



The task groups were manually assigned to declutter the plot and to be able to compare the individual tasks across assignments. The original task names with their emissions can be found in the `emissions.ipynb` *notebook.*

As touched upon above, for each of the assignments it is one task that is responssible for most of the emissions for that assignment. It is the use and training of large machine learning models that overwhelmingly costs the most. However, as seen on the plot below, the difference in emission rate (CO_2eq/s) is minimal between all the tasks. Thus, it is a combination of model structure and run time that should guide decisions when it comes to running analyses, with the goal of limiting emissions and environmental impact. To help lessen the impact cause by this portfolio some things could be implemented (or not implemented):

- In assignment 2, the assignment prompt called for the training of two different classifiers. In addition to this, I also implemented gridsearch for both. All four resulting classifiers trained for that assignment performed almost identically on the test data. If the purpose of these were not in the realm of teaching, I would argue that with an accuracy of 0.89 in a binary classification task, the `LogisiticRegression` model would suffice without needing to implement a convolutional neural network or gridsearch.
- The largest emitter was assignment 4, as mentioned the purpose of this (if not in a teaching context) cannot be justified when taking the amount of emissions into account. In any case like this, I would argue that a clear hypothesis or specific research question should be formulated before running such impactful code. Additionally, it would be a good idea to do a 'pilot run' first, to make sure that everything works as expected before running the entire code.

The same general ideas should be applied to assignment 1 and 3, but as their emissions were very low (especially compared to assignments 2 and 4) and the tasks they performed rather specific, no additional suggestions for these will be made.