Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. Alan Akbik

Berlin, 19.04.2024
Lehrstuhl Maschinelles Lernen

Introduction to Natural Langauge Processing

Exercise 1

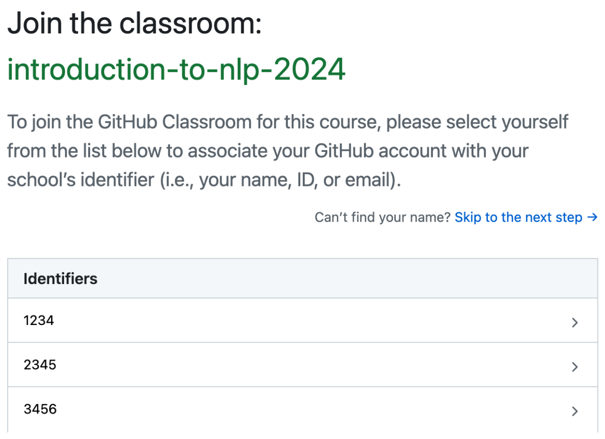
Submission: Hand your homework until 29.04.2024 (Monday), 23:59 via Github Classroom. Each exercise is worth a total of **10 points** and you need **80%** (of total points over all exercises) to be admitted to the final exam. Sometimes, there are optional tasks that will give you extra points. Exercises in Github Classroom should be completed and submitted individually. However, we encourage you to work on the problems in teams. Please note the information available in Moodle: `https://hu.berlin/nlp24-moodle`.

Task 0 (Setup) **0 points**

To complete exercises in this lecture, you are required to code in Python. We will be using Python version **3.9** throughout our lecture. We recommend to use *conda* to manage your Python environments. It is installed on university servers, but you can find installation guidelines here if you are working on your local machine.

(a) Enroll in the current exercise: `https://classroom.github.com/a/IpwD52NO`. You need to set a personal access token to clone this private repository (see this documentation.)

(b) Select your student ID from the list to link it with your Github Account. If your student ID is not listed, continue without selecting your student ID and let us know your student ID together with your Github username via Moodle or Mail.



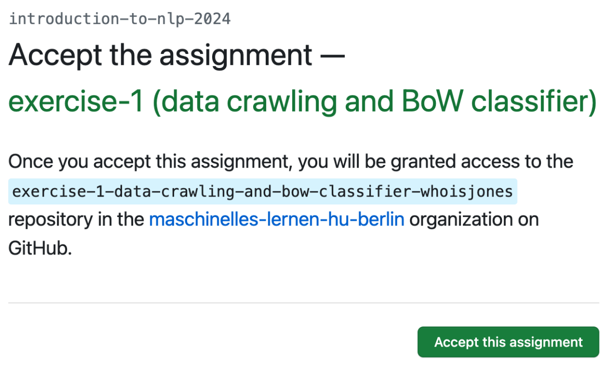
Join the classroom:
introduction-to-nlp-2024

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? Skip to the next step →

Identifiers
1234 ›
2345 ›
3456 ›

(c) Accept the assignment. It will create a private repository for you.

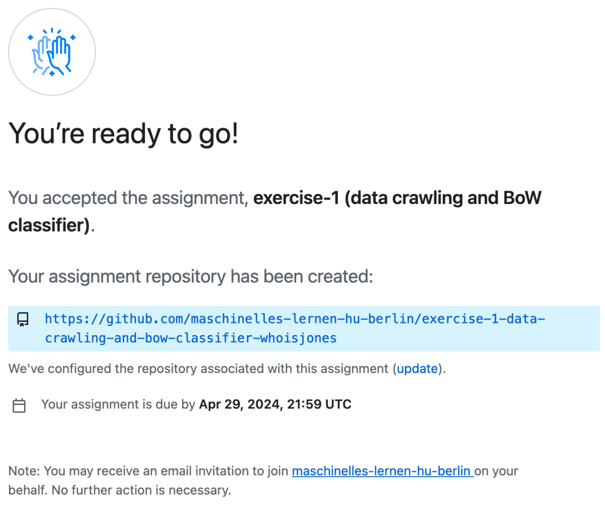


introduction-to-nlp-2024

Accept the assignment —
exercise-1 (data crawling and BoW classifier)

Once you accept this assignment, you will be granted access to the `exercise-1-data-crawling-and-bow-classifier-whoisjones` repository in the maschinelles-lernen-hu-berlin organization on GitHub.

(d) Navigate to your created repository.



You're ready to go!

You accepted the assignment, **exercise-1 (data crawling and BoW classifier)**.

Your assignment repository has been created:

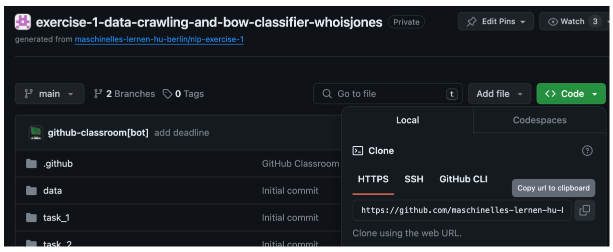
https://github.com/maschinelles-lernen-hu-berlin/exercise-1-data-crawling-and-bow-classifier-whoisjones

We've configured the repository associated with this assignment (update).

Your assignment is due by **Apr 29, 2024, 21:59 UTC**

Note: You may receive an email invitation to join maschinelles-lernen-hu-berlin on your behalf. No further action is necessary.

(e) Clone your repository with `git clone`.



exercise-1-data-crawling-and-bow-classifier-whoisjones Private
generated from maschinelles-lernen-hu-berlin/nlp-exercise-1

github-classroom[bot] add deadline

.github GitHub Classroom
data Initial commit
task_1 Initial commit
task_2 Initial commit

https://github.com/maschinelles-lernen-hu-l

Clone using the web URL.

(f) Install a conda environment for the lecture:

```
conda create -n nlpcourse python=3.9
conda activate nlpcourse
```

(g) Install the required dependencies from the cloned repository:

```
pip install -r requirements.txt
```

(h) Verify installation:

```
conda list flair
```

should give you:

```
# packages in environment at ~/.conda/envs/nlpcourse:
#
# Name      Version      Build   Channel
flair       0.13.1       pypi_0  pypi
```

(i) We provide you with two example scripts in the `example` folder to try out flair and fundus. You can execute with your environment set up like:

```
python examples/flair_example.py

python examples/fundus_example.py
```

Task 1 (Data Crawling and Sentiment Analysis) — $2 + 2 + 1 = 5$ points

In this task, you are required to crawl headlines from online publishers and analyze their sentiment in order to answer the question whether certain newspapers tend to write rather positive or negative headlines. To do so, you will use fundus for crawling which offers a wide online publishers to choose from. Once you obtained the data, you should analyze the sentiment of each headline with a pre-trained sentiment classification model from flair.
You are given one main file `run_task_1.py` and corresponding folder `task_1` with two files: `flair_model.py` and `fundus_crawler.py`. You are required to complete code snippets / functions in each of the files. Your code must be working by running it from command line with `python run_task_1.py`.

(a) Implement `crawl_headlines()` in `task_1/fundus_crawler.py` to crawl 10 headlines for two publishers of your choice. The fundus documentation lists all available publishers and how to crawl headlines for any of them. Hint: Try to store publisher information and list of headlines in a dictionary.

(b) Implement `predict_labels()` in `task_1/flair_model.py` which annotates each headline with its sentiment. You do not need to train a model for this task - you can simply load a pre-trained sentiment model with flair and predict the sentiment for each headline. Hint: You need to convert strings into flair Sentence objects (see slides).

(c) `print_statistics()` in `run_task_1.py`: This function should print out information about the headline sentiments per publisher, i.e. how many headlines are positive or negative per publisher.

Task 2 (Bag-of-words Classifier) — $1 + 2 + 2 = 5$ points

In this task, you will implement a simple bag-of-words classifier in PyTorch. We provide you with a training script (`run_task_2.py`) which do not need to adjust. Further, we provide you with three files in `task_2` folder: `bow_model.py`, `data.py`, and `evaluation.py`. Your task is again to complete code snippets / functions inside these files to make the entire code work.

(a) Complete the bag-of-words model inside `bow_model.py`. The constructor and forward method are missing. The constructor requires a linear layer which input dimension is equal to the vocabulary's length and the output dimension is equal to the number of labels that we want to predict. The forward method takes a `bow_vector` as input and needs to pass it through this previously defined linear layer. Finally, the `log-softmax` operation needs to be applied to the output of previous operation. See PyTorch documentation (linear layer and log-softmax) or check the corresponding exercise slides for reference.

(b) Implement the `get_large_training_data()` method in `task_2/data.py`. This function should read-in data from a given path (the `data` folder contains two .txt files - one for training and one for validation) and return a list of tuples in format *(label, sentence)*. The *sentence* needs to be split into words (Hint: You can get a list of words by splitting on whitespaces). You need to set the `dataset_size` variable to ”large“ to use the `get_large_training_data()` method. For debugging, you can set this parameter to ”small“ for a small example dataset.

```
if __name__ == "__main__":
    dataset_size = "large" # Use "small" for a small dataset
    main(dataset_size)
```

(c) Implement the `evaluate_final_model()` in `task_2/evaluation.py`: This function should return the accuracy metric on your test dataset. Hint: Start by setting the `dataset_size` variable to ”small“ in `run_task_2.py` for debugging.

The accuracy metric is defined as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

This means, for each example, you have to compare the model prediction with its true label. Once all predictions have been compared to their true labels, you should return the accuracy metric.

Note: You may need to consider out-of-vocabulary words (words that are not observed in the training dataset). To handle these words, you may want to adjust `make_word_dictionary()` and `make_bow_vector()` in `task_2/bow_model.py`.

Task 3 (Optional: Add a publisher to fundus) 5 points

fundus supports accurate and high-quality data crawling of online publishers. To do so, fundus uses publisher-specific rules to accurately extract information such as headlines, author or main text.

(a) Add a new publisher to fundus. See this tutorial for reference.