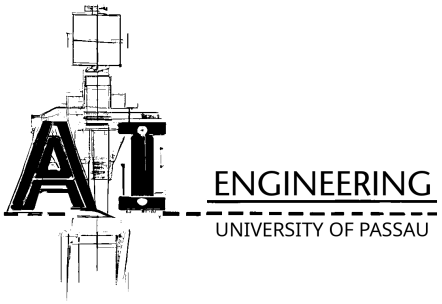
Deep Learning for Natural Language and Code

Exercise 9

Prof. Dr. Steffen Herbold
SoSe 2025
Due on 2025/07/17



General information for all exercises (read carefully!)

Within the "Deep Learning for Natural Language and Code Exercise," you will execute different tasks related to various NLP concepts. The main goal of these exercises is to teach you how to develop approaches. Once you have gained this knowledge, you will have the opportunity to use your own solution and compare it with existing solutions from popular libraries. This means that these exercises are not just about knowing how to use the libraries.

Problem Description

Now it is time to work with language models for *code*. In this exercise, we will learn about different large language models that are available on *Hugging Face*, and then use `PolyCoder` to generate code.

Programming Tasks (PT)

1. Review the website: Generating text with transformers[1]. In particular, review the following:
   - The architectures of the different models: `CodeParrot`, `InCoder`, `CodeGen`, `PolyCoder`.
   - How to import the different models with Python.
   - How to generate code.
   - How to evaluate the model.
2. Once you have reviewed how the *Hugging Face* models work, select one of the `PolyCoder` models and create a function that receives a snippet of code and completes it. *Note:* base your selection on the computational resources available to you. You can also use *Google Colab*.
3. Read the Documentation for *Metric: code_eval*[2]. Implement at least three test cases with a known snippet, i) one that fails, ii) one that passes, and iii) one that has a partial match.

[1] `https://huggingface.co/spaces/codeparrot/code-generation-models`
[2] `https://huggingface.co/spaces/evaluate-metric/code_eval`

Restrictions and tips

- For selecting the model, start with the smallest model, then restart the kernel and change the selected model.
- Note that you could use *Google Colab* to execute the exercises (`https://colab.research.google.com`) or *Kaggle Kernel* (`https://www.kaggle.com/code`).

Theoretical Questions

1. Once you have coded your script for generating code, answer the following questions:
   - What is `num_beams`? (Parameter of the `generate` method)
   - What is `max_length`? (Parameter of the `generate` method)
   - What is `num_return_sequences`? (Parameter of the `generate` method)
   - What is `temperature`? (Parameter of the `generate` method)
   - Why can `num_return_sequences` not be higher than `num_beams`? (Parameter of the `generate` method)
2. Once you have understood and implemented test functions as explained in *Metric: code_eval*, answer the following questions:
   - What does it mean to have a *Full Match*, *No Match*, and *Partial Match*?
   - What does the `k` parameter of the compute function do?
   - What does the @ mean in the metric used for evaluation?
3. As a review, list the different large language models that you have seen in the lecture, and for each one of them, list:
   (a) The dataset(s) (corpus(es)) that were used for pre-training the model.
   (b) The tasks used for pre-training the model.