Language Models
Workshop 2
Sessions 4 and 6

A link to the materials needed to solve the tasks from this list will be provided on the course website. In each task, you should use a language model for the Polish (or English) language, from the Hugging Face website. Two relatively small models are suggested: PapuGaPT, or GPT-2, but you are not limited to these. There will be sample programs on the lecture page that you can freely use and modify.

Task 1. (**5+6p**) This task is quite open-ended: you need to 'play the role' of a researcher and examine how well a chosen language model (approximately 100M-1B in size) handles calculations of relatively simple arithmetic expressions in a few-shot learning scenario.

What constitutes an arithmetic expression, how complex it is, how prompts are constructed, and how examples are chosen is up to you; to earn 5 points, only two requirements must be met:

1. You must find a scenario that shows that the model's results in the given task are significantly better than random.
2. The findings from your research should be thorough and documented in code/notebook form.

Additional points are awarded at the discretion of the instructors to recognize students who put exceptional effort into this task or achieved particularly interesting results. Up to 5 points are included in the maximum.

An interesting question is whether any bias of the model can be observed in this task.

Task 2. (**6p**) In this task, you need to modify the token generation procedure in such a way that:

1. It generates exactly one word **from a given set of words** (this word may consist of more than one token), and
2. It is sufficiently efficient (capable of generating many continuations of a prefix in a reasonably short time).

Naturally, this may require generating a non-zero number of redundant tokens. **Optional part:**. Then use this procedure to solve the Riddles task[1] from the AI Olympiad in the following way: create a prompt consisting of a riddle and some connector, then generate (several times?) possible one-word continuations of such a prompt (these continuations should, of course, come from the set of possible answers). Provide the accuracy of this solution (for more than one model).

Note: This task will have a continuation, where points will be awarded for the achieved results.

Task 3. (**6p**) In this task, we will work on disambiguating text. We assume that the text with variants is given as follows:

smart—smarty—small students—elephants—houses from—farm—firm—forum—fermi—fram—free
many—yemen—mania—man—mine—mean—money countries—cantors—counters—centaurs—contours—centers
study—stud—steady—suited—studio— at—taiyo—tye—tae—tay—ate—to—auto—yeti
our—aria—euro university—universe—universal

Tokenization is performed using the split method, and (for convenience) the correct word is always in the first position (but you must not use this fact in any way).

Write a program that finds the correct word variant. Your program should use language models and some non-greedy optimization algorithm. This could be, for example, beam search, but other methods are also allowed.

Task 4. (**6p**) In this task, we will generate sentences in natural language that meet an additional property: **all words should start with the same letter**. The generation should be prompted using randomly chosen word (is up to you how to design this process). Your generations should end with a period (or another punctuation mark ending the sentence).

Your program should:

[1] `https://github.com/OlimpiadaAI/I-OlimpiadaAI/blob/main/first_stage/riddles/zagadki.ipynb`

1. Use both top-k and top-p sampling.
2. Modify the token probability distribution.
3. Generate p̈roperẗexts, meaning words consisting of letters, with spaces as separators, and properly formatted punctuation (e.g., commas should be attached to the preceding word).
4. Avoid repetitions.
5. Generate multiple variants and select the best one according to a criterion defined by you.