Exercise 4 - Sequence Classification using Encoder Transformers

Contextualized and Transformed

Deadlines

Deadline for Exercise 4: **Sunday, 16.11.2025, 23:59** CET (Zurich Time).

Deadline for the peer review: 24.11.2025, 23:59 CET (Zurich Time).

- Note, the peer reviews will be assigned one day after the exercise deadline. You will find instructions for the peer review process at the end of this document.

Learning goals

The goal of this exercise is to introduce you to the task of named entity recognition (NER) and applying transformer-based neural architectures to solve it. By completing this exercise, you should …

- … understand NER as a task and the IOB-format.
- …be able to use pre-trained transformer encoder models like BERT and fine-tune them on a specific task (using HuggingFace, Pytorch).
- … be able to use GLiNER to avoid costly fine-tuning.
- … Get a glimpse of real-world applications for NER.

Please keep in mind that you can always consult and use the exercise forum if you get stuck (note that we have a separate forum for the exercises).

Deliverables

Your solutions for each part should be submitted as self-contained iPython notebooks (ipynb files). The notebooks should contain your well-documented, EXECUTED, and EXECUTABLE code. That way, your reviewers can view and execute your code without potential dependency issues and/or installing new packages or versions of packages. We encourage you to solve the exercises on Google Colab/Kaggle and download the .ipynb files when everything is completed.

You will also have to write a short lab report at the end of each script. The lab report should be written in a markdown cell and contain a detailed description of the approaches you used to solve this exercise. You can discuss what worked and what didn't work, but in both cases, you please provide reasoning as to why something did or didn't work. Please also include the results. Inside each notebook, we have provided specific questions denoted with "📝❓" that should be addressed in the report.
Please hand in your code and your lab report. Hand in the following files and name them exactly in the following fashion:

- **ex04_ner_bert.ipynb**
- **ex04_ner_gliner.ipynb**

zip it and name the zip-folder *ex04_ml4nlp1.zip*.

Please note:

- Organise your code such that it is executed and executable.

- Executed code means that the cells must have already been run and the output must be visible to anyone checking your notebook without having to run the code again.
- Your assessors should be able to run your code. If it doesn't work, you can't get the maximum score.
- Please make sure that your answers to the specific questions outlined in the notebook and denoted with "📝❓" are clearly marked. If the assessors cannot easily find the answers they need, you can't get the maximum score. We recommend including these clearly in your lab report.
- Also, we highly recommend testing that your notebook runs as expected before submitting it. To do this, hit "Runtime" > "Restart runtime and run all", and make sure that there are no errors! Be sure to submit the version of the file with the executed outputs.

Data

For the first part of the exercise, you will work with a small part of the wikiann dataset, described in this paper. Use HuggingFace's datasets library to download.

The wikiann dataset contains over 170 languages. Choose one language to work with from that dataset. The following conditions need to hold for the language:

- The chosen language is not English.
- The dataset needs to contain at least 3000 train and 2000 test sentences in the chosen language.
- There must be a pretrained Hugging Face BERT/RoBERTa base/small model for the chosen language (an alternative solution is a multilingual model). (Check out this link for a list of all pretrained HuggingFace models.)

Using the datasets-library to extract two training sets, one containing 1,000 sentences and one containing 3,000 sentences.
Also, extract an evaluation set (in this exercise, we don't differentiate between development and test set) of 2,000 sentences.

NOTE: Check out this part of the documentation to only download the parts of the dataset you need.

For the second part, you will create your own dataset, in order to predict entities using GLiNER. The following conditions must hold true:

- Your dataset's length is at least 10 data points.
- Entities mentioned in the text should not be accompanied by their label, but rather be inferred by the context
    - Example of easy-"bad" point: *My address is Andreasstrasse 15, Zurich and my name is John.*
    - Example of harder-"good point": *I ordered the package to be delivered at Andreasstrasse 15, Zurich, and I added a small sticker, with "John" written on it.*
- Your labels and data points should be diverse.

PART 1 - Named Entity Recognition using BERT

Implement a named entity recognition system for your chosen language. Use HuggingFace's BertForTokenClassification-class and initialize it with a pretrained Hugging Face BERT-base model of your chosen language. This HuggingFace guide for fine-tuning serves as a good starting point.

Before passing the data to the model, you need to encode it using a HuggingFace tokenizer. Use the tokenizer corresponding to your BERT model. When provided with the right arguments, the tokenizer can also pad and truncate the input.

You can reduce the amount of code for this exercise by using the Trainer class explained at the bottom of the HuggingFace guide.

You will create 4 fine-tuned versions of the system:

1. Fine-tuned with 1,000 sentences
2. Fine-tuned with 3,000 sentences
3. Fine-tuned with 1,000 sentences and frozen transformer backbone weights
4. Fine-tuned with 3,000 sentences and frozen transformer backbone weights

Using your fine-tuned models, run inference on the evaluation set and evaluate the performance with f1-micro and f1-macro scores.

Please ensure to run your code on Google Colab or Kaggle with GPU as a hardware accelerator selected. Select the GPU at “Edit” → “Notebook Settings”.

NOTE: Loading several BERT-models at once into memory will lead to an out-of-memory error (at least as long as you don’t have more than ~40G memory available). To avoid this, load and fine-tune only one model at a time, and then delete the model from memory (or overwrite it). If you want to be extra cautious, you can save the model state before deletion. Specifically on Colab, the BERT model already uses up most of the GPU-memory, and with a large batch size, you may get an out-of-memory error. Reducing the batch size should solve the problem.

PART 2 - ZERO-shot NER and Anonymization with GLiNER

Implement a named entity recognition and anonymization system for your chosen language(s). Use this fine-tuned version of GLiNER (read the notebook and model card for more info). You will create your own dataset and define your own labels, and write a short report on the results. Part 2 is supposed to be more interactive and less time-consuming than part 1.

NOTE: You do not need to run inference with GPU for part 2.

Submission, Grading & Peer Review Guidelines:

As with the first exercises, peer reviews will be carried out on OLAT.

As soon as the deadline for handing in the exercise expires, you will have time to review the submissions of your peers. You need to do **1 review per person** to get the maximum points for this exercise.

Here are some additional rules:

- **All file submissions are anonymous (for peer review purposes): Do not write your name into the Python scripts.**
- **EVERY** member of each team submits on OLAT.
- Please submit a zip folder containing all the deliverables.

Grading coefficients:

- Part 1: 80%
- Part 2: 20%

Groups & Peer Reviews:

- You should work in the same group for all exercises. Each member should contribute equally!
- If you do not submit your peer review, you will receive a -0.25 point penalty on the final grade of the
- exercise.
- Please use full sentences when giving feedback. Please write the peer review in English.
- Provide constructive and helpful comments to your peers! If you criticise something, you should also be able to provide actionable and realistic suggestions for how something can be improved.
- Please answer all the pre-designed review questions of the peer review to the best of your ability.

Declaration of Usage of Generative AI

For every exercise, please append a short **"Declaration of Usage of Generative AI"** section at the end of your submission. In this declaration, briefly (a few sentences are sufficient!) describe **how** and to **what extent** you used Generative AI tools (e.g., ChatGPT, Copilot, Gemini, etc.) during the completion of this exercise.

Your statement should address the following points:

- **Source code and documentation:** How was Generative AI used to assist in developing, debugging, or documenting your code?
- **Written report:** Was Generative AI used to draft, revise, or post-edit the written components of your submission?
- **Learning support:** Did you use Generative AI as an assistant for concept clarification, idea exploration, or learning guidance?

Good luck working with transformers