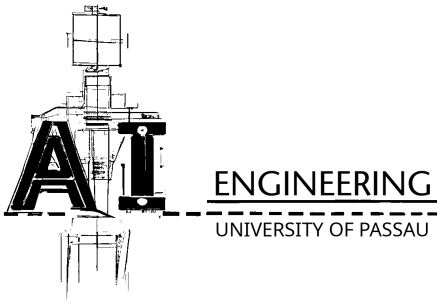
Deep Learning for Natural Language and Code

Exercise 8

Prof. Dr. Steffen Herbold
SoSe 2025
Due on 2025/07/10



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

One task that is executed in NLP is sentiment analysis. In this exercise, we will focus on pretraining (see Fig. 1) an encoder model for at least two tasks. For this exercise, you can use your own implementation of an encoder from last week's exercise (without the additional layer for classification) or use `Transformer Layers` from `Pytorch` to build one.

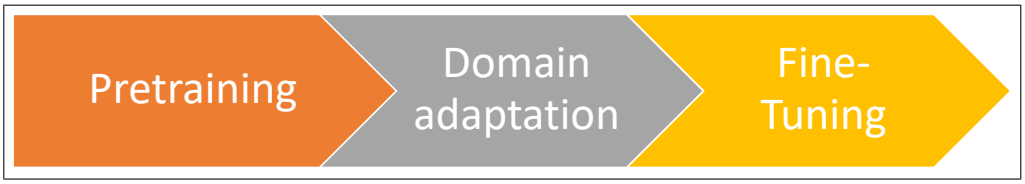


Figure 1: Image based on ULMFIT process by Howard J. and Ruder S. [2]

For this exercise, we are going to work with a dataset that consists of movies' reviews:

- `https://ai.stanford.edu/~amaas/data/sentiment/`

In StudIP, a Jupyter Notebook is provided as part of the exercise, be sure that you download it, before starting to solve this exercise.

Data Set Description

For this exercise, we are going to work with the *Large Movie Review Dataset* [3]. This dataset was built for binary sentiment classification. It is composed of 25,000 highly polar movie reviews for training and 25,000 for testing. Meaning that the middle values (i.e., scores of 5) are ignored.

Theoretical Questions - Part 1

1. This week's programming task is to implement an encoder model. In Fig. 1, we have three main steps when training a language model. Review these concepts and explain each one of them in your own words.
2. Now that these concepts are clear, take a look at the `Jupyter Notebook`, in particular at the code cells below the title *Theoretical Part1 - code related* and answer:
   - What is that block of code doing?
   - What is the meaning of `tok1`, `tok2`, `tok3`, and `tok4`?
   - What is the meaning of `maxlen_X`, `size_B`, and `max_pred_M`?
   - How are the previous bullets are related to possible tasks (e.g., Next Sentence Prediction (NSP), Masked Language Modeling (MLM))?

Programming Tasks (PT)

1. Build an encoder model with multiple encoder blocks (i.e., layers), see Fig. 2. *See the Restrictions and Tips section for more information.*

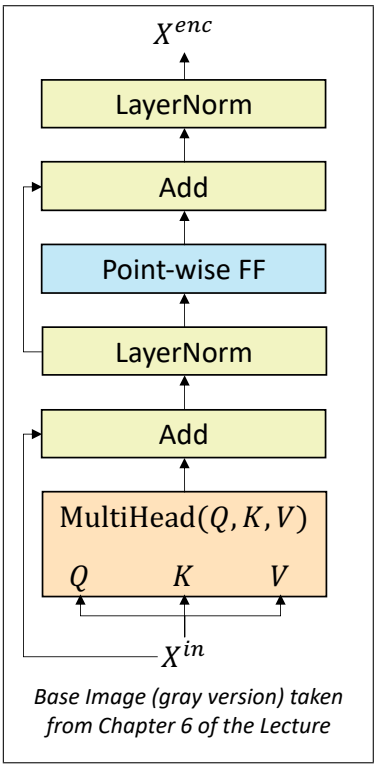


Figure 2: Figure based on the lecture slides - Chapter 6

2. Identify the tasks that are going to be used for pretraining your model.
3. Pre-process the reviews in order that you can use them as input for pretraining the model in more than one task. **Note:** you can get inspiration from BERT [1] and you can reuse the analyzed code from Theoretical Question Part 1. *See the Restrictions and Tips section for more information.*

4. Implement a training loop and pretrain your model with only a sample of the reviews to accomplish the previously defined tasks.

Restrictions and tips

- You are not allowed to use an existing pretrained model or a an existing encoder model. However, you are allowed to use `nn.TransformerEncoderLayer` from `Pytorch` (or use your encoder block implementation from last week's exercise). You are not allowed to use `nn.TransformerEncoder`.
- For the tokenization and embedding of the reviews, you can use existing implementations.
- *Memory limitations.* Do not load to memory all the reviews; start executing the exercise with only 100 reviews, and increase the number of instances as wanted.
- Note that you could use Google Colab to execute the exercises (`https://colab.research.google.com`) or Kaggle Kernel (`https://www.kaggle.com/code`).
- You can re-use code between exercises.

Theoretical Questions - Part 2

1. Consider Fig. 1. Give an example of a dataset that can be used in each step, and explain why (the purpose of using them).
2. Consider Fig. 1. Imagine that now you are not working with text anymore; now you are going to work with images. As the previous point, give examples of datasets that can be used in each step. In addition, give some hypotheses about what a model could learn in each step.
3. Review the following concepts/tasks.
   - Next Sentence Prediction (NSP)
   - Sequence classification
   - Masked Language Modeling (MLM)
   - Causal Language Modeling (CLM)

   In addition, imagine that you have the following text.

   > **The Ant and the Grasshopper.** The ant and the grasshopper were good friends. In the summer, the ant works hard to fill his storage with food. While the grasshopper was enjoying the fine weather and playing all day. When winter came, the ant was lying cozily in his home surrounded by the food he stored during the summer. While the grasshopper was in his home, hungry and freezing. He asked the ant for food and the ant gave him some. But it wasn't enough to last the entire winter. When he tried to ask the ant again, the latter replied: "I'm sorry my friend but my food is just enough for my family to last until the end of winter. If I give you more, we too will starve. We had the entire summer to prepare for the winter but you chose to play instead." **Author:** Esopo

   Give examples of inputs and outputs that could be used as training examples, at least three, for the aforementioned tasks (e.g., the whole text could be classified as a Fable. **Input**: Whole text, **output**: Class-Fable).

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

[3] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.