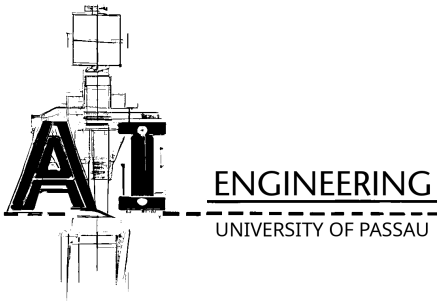
Deep Learning for Natural Language and Code

Exercise 5

Prof. Dr. Steffen Herbold
SoSe 2025
Due on 2023/06/12



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 is executed in NLP is sentiment analysis. In this exercise, we will focus on developing models that are able to classify reviews in different sentiments/emotions. For this exercise, we are going to work with a dataset that consists of movies' reviews:

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

Data Set Description

For this exercise, we are going to work with the *Large Movie Review Dataset* [1]. 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. The dataset also contains unlabeled data. If you have memory restrictions please, follow the tips given in the *Restrictions and tips* section.

Programming Tasks (PT)

1. Implement an RNN model with pyTorch.
    - Pre-process the reviews by transforming the textual data into a word-embedding representation, either by loading the pretrained embeddings from the previous exercise (CBOW or Skip-Gram model), or by using randomly initialized embeddings and learning them during RNN training. Consider experimenting with both options and comparing performance.
    - Define the architecture of your RNN. **For this, you cannot use the the nn.rnn layer of pyTorch**.

- The model should be able to classify the reviews into positive or negative the reviews.
- Try training with different input sequence lengths. *(If you have memory restrictions, please refer to the restrictions and tip section)*
- Evaluate the model performance with the test data. *(If you have memory restrictions, please refer to the restrictions and tip section)*

Restrictions and tips

- For the RNN implementation the nn.rnn layer from pyTorch can not be used.
- *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 for executing the exercises (`https://colab.research.google.com`) or Kaggle Kernel (`https://www.kaggle.com/code`).
- *Memory limitations.* You can download a dataset directly to Google Colab. There are multiples tutorial in the the web for doing that. For example *https://niruhan.medium.com/downloading-a-dataset-and-displaying-an-image-in-google-colab-5370f20b236d*.
- You can re-use code between exercises.

Theoretical Questions

1. What are the main differences between a vanilla RNN and a LSTM?
2. What are the drawbacks of a vanilla LSTM?
3. What are the differences between uni vs. bi-directional RNNs/LSTMs?
4. Why is the bi-directional LSTM model much slower than the LSTM model? Why does it require more time for training?

References

[1] 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.