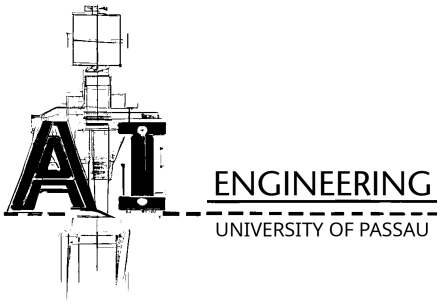
Deep Learning for Natural Language and Code

Exercise 2

Prof. Dr. Steffen Herbold
SoSe 2025
Due on 2025/05/15



General information for all exercises (read carefully!)

Within the "Deep Learning for Natural Language and Code Exercise," you will execute different tasks that relate 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 developing models that are able to classify reviews in different sentiments/emotions. For this, we assume that you have an initial implementation of the BOW from exercise # 1. For this exercise, we are going to work with a dataset that consists of movie reviews:

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

In StudIP, a Jupyter Notebook is provided as a template for this exercise. In the template, a series of sections are provided in order to help you organize and structure the code. Additionally, some blocks of codes are also provided to facilitate some exercise tasks.

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. However, for this exercise, those are not going to be taken into account.

Programming Tasks (PT)

1. Build a pipeline for cleaning the raw reviews. In this pipeline you should be able to activate or deactivate the following options:
   (a) Tokenize based on spaces and punctuation or only spaces.
   (b) Convert everything to lower case or keep the capitalization as it is.

(c) Remove punctuation.
(d) Remove the terms that appear more often than a certain percentage. You should be able to vary this percentage (i.e., it should be a parameter).
(e) Replace the numbers with a token (i.e., a special token `<NUM>`).

2. Build different representations of the data:
   (a) Bag of Words.
   (b) Binary vector representation using presence/absence of words/tokens.
3. Create a logistic regression model:
   (a) Define a binary logistic regression architecture with Pytorch to classify the reviews into positive and negative ones.
   (b) Train a logistic regression model and test it.
   (c) Imagine that you now have 4 different sentiments: i) Wonderful: reviews having scores of 9 and 10; ii) Pleased: reviews with scores of 8 and 7; iii) Disgusted: reviews with scores of 4 and 3; and iv) Annoyed: reviews with scores of 2 and 1. What should you change in the model to be able to classify the reviews into the four classes mentioned? Implement the changes.
4. Build a feedforward neural network that classifies the reviews into positives and negatives:
   (a) Define the architecture. It must have at least two hidden layers, and it is not allowed to use convolution, recurrent or transformers layers.
   (b) Define the training loop.
   (c) Define the validation loop.
   (d) Train the model with the different representations requested in PT#2.
   (e) Test the model with the respective representation.

Restrictions and tips

- The tasks must be solved using only the Pytorch packages `torch.nn` and `torch.optim`. It is also allowed to use other packages/libraries for loading the data, but not for preprocessing them.
- Throughout the exercises we will work mainly with PyTorch. Please refer to the following series of tutorials if you need an introduction to the library:
  - `https://pytorch.org/tutorials/`
- Note that you could use Google Colab for executing the exercises (`https://colab.research.google.com`).
- You can re-use code from previous exercises.

Theoretical questions

1. What is a stop-word?
2. Consider the following scenarios for the PT#1:
   - Initial scenario: you perform: a) tokenising by spaces only, b) leaving the capitalisation as it is, c) not removing punctuation, d) not removing anything, and e) not

replacing the number with a token.

- The same scenario as before, but now you for e) you replace the numbers with a token.
- The same scenario as the last bullet but now for b) you lower all the cases.

What will happen to the size of the vocabulary in relation to the initial scenario when each of the scenarios presented is carried out? What could be an advantage in each scenario?

3. What happens if each review is not represented by the same number of items? For example, review-1: [1,2,54,70], review-n: [40, 1, 30, 25, 15, 10]? What should you do to use this representation for the models proposed in the PTs?

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.