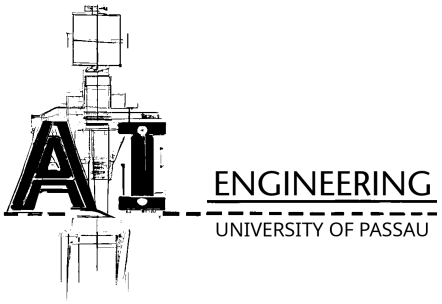
Deep Learning for Natural Language and Code

Exercise 6

Prof. Dr. Steffen Herbold
SoSe 2025
Due on 2025/06/26



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 developing a Vanilla self-attention mechanism of a transformer, to which we will append an additional layer to classify the reviews into positive or negative ones.

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

Data Set Description

For this exercise, we are going to work with the *Large Movie Review Dataset* [2]. 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.

Programming Tasks (PT)

1. Implement the self-attention mechanism presented in the lecture slides, Chapter 5.

- Tokenize your text. You may use external libraries to execute this part, including the `load_dataset` function from the `datasets` library to load the dataset [2] using `dataset = load_dataset("imdb", split="train")`.
- Use `gensim.downloader.load('glove-wiki-gigaword-100')` to load a *Glove* model to embed your texts (i.e., choose a compatible tokenization method for this embedding). In this way, you are able to represent X (see Fig. 1).
- Now, it is your turn to use X to implement the self-attention mechanism presented in Fig. 1. For this implementation, use *PyTorch* to implement the needed components (e.g., Dense layers).

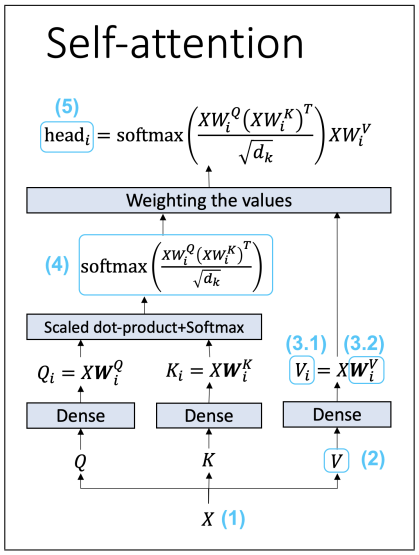


Figure 1: This image is taken from Chapter 5 (Lecture slides)

2. After implementing the attention mechanism, add an extra layer to the model to be able to classify reviews into positive or negative sentiments.
3. Implement a training loop and train your model with only a sample of the reviews.
4. Implement a function that receives as input one review and determines its sentiment.

Restrictions and tips

- *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 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.
- A great explanation of the attention mechanism: `https://www.youtube.com/watch?v=eMlx5fFNoYc`.

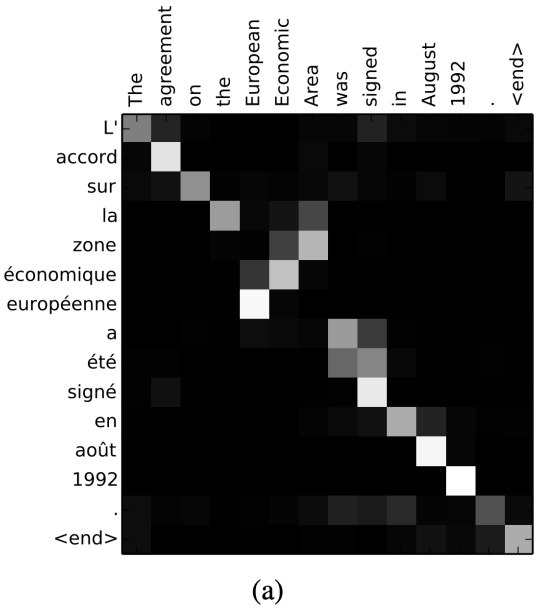
Theoretical Questions

1. Continuing with the analogy proposed in the lecture with a *search engine*, think of a particular engine, the engine of your preferences. What would be the keys (K), the queries (Q) and the values (V)?
2. Consider Fig. 1, identify which dimensions you had to define for the attention mechanisms. Now, define a specific number for:
    - Batch size.

- Sequence length (*number of tokens*).
- Dimensions for W_i^Q, W_i^K, W_i^V.

Now, identify the dimensions in Fig. 1 of: (1), (2), (3.1), (3.2), (4), and (5).

3. Explain, in your own words why, in (4) of Fig. 1, the dot product is divided by $\sqrt{d_k}$.
4. Once again, analyze Fig. 1 and your implementation, where do you have a bias added?
5. Imagine that you have the image presented below, and it represents the "attention weights" between the input sentence in English and the output sentence in French. These weights allow you to have a visual representation of what is happening in the model (the weights are represented in a gray-scale in which 1 (highest probability) is represented with white).



(a)

Figure 2: Image taken from [1].

Now, with the help of a translator if needed, try to understand the heat-map, in particular, understand why the focus (highest weight per word) is not in all cases on the diagonal of the matrix. In addition, think about which part of the self-attention mechanism in Fig. 1 could be used to generate (extract) this matrix.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] 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.