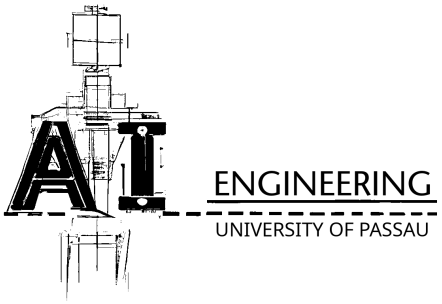
Deep Learning for Natural Language and Code

Exercise 7

Prof. Dr. Steffen Herbold
SoSe 2025
Due on 2025/07/03



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 an encoder block[1] of a vanilla transformer (see Fig. 1) to classify the reviews into positive or negative ones.

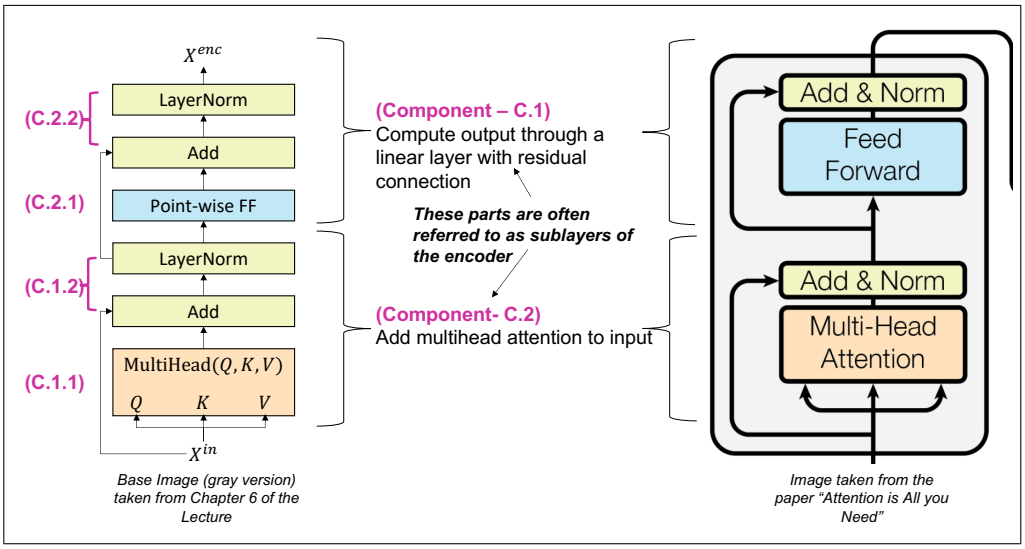


Figure 1: Encoder block. On the left, the encoder block is presented as it is presented in the lecture slides. The equivalent diagram from the paper "Attention is all you need" [4] is presented on the right.

[1] In this exercise, we are using the notation introduced in the lecture.

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* [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.

Programming Tasks (PT)

For building the encoder, we need to build the two components of the encoder block (see Fig. 1).

C.1. A transformation of the inputs based on the *multi-head attention*.
C.2. An additional transformation based on a small *feed-forward* network

Both transform the input through residual connections (C.1.2 & C.2.2)

1. For implementing **C.1**, we will start from the implementation of the *self-attention* mechanism in the previous exercise, see Fig. 2 (a).

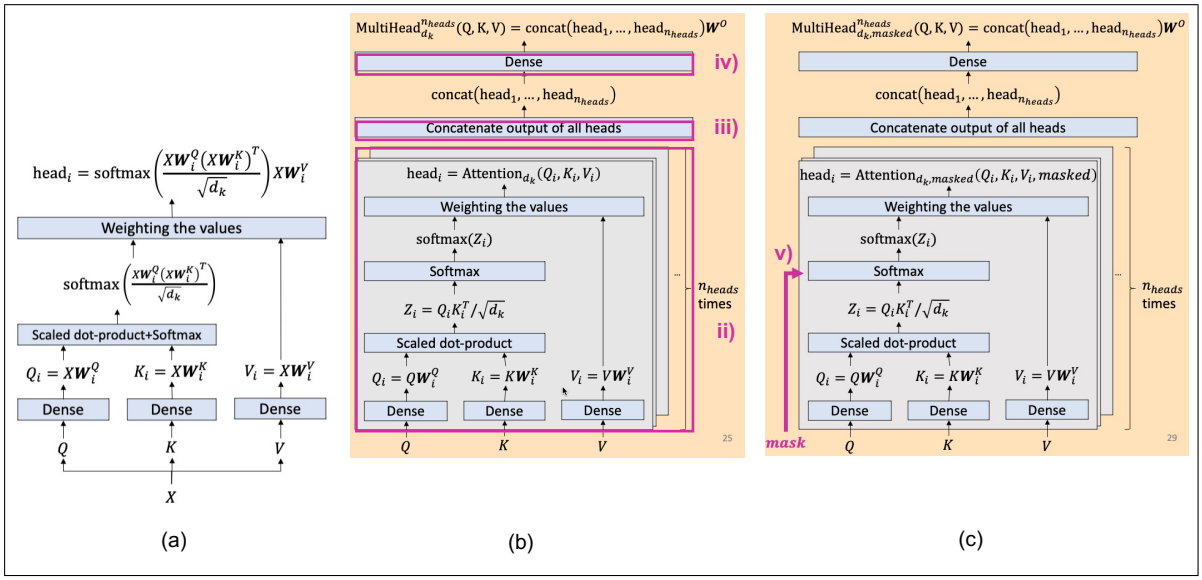


Figure 2: Self-attention mechanism to Masked multi-head attention. *(a)* Self-attention mechanism. *(b)* Multi-head attention without a mask. *(c)* Masked multi-head attention (C.1.1). *These images were taken from Chapter 5 (lecture slides).*

C.1.1 In Exercise #6, we developed a model that used a self-attention mechanism.

i) Be sure to remove the extra layer added after the self-attention mechanism in Exercise #6

ii) As depicted in Fig. 2 (b) , we have to modify the self-attention mechanism in order to have n_{heads}
iii) Their output (the output of the n_{heads}) should be concatenated.
iv) The concatenated output should be passed as input to a dense layer.
v) Finally, we have to add an option of masking the *scaled dot-product* (see Fig. 2 (c))

C.1.2 After implementing the *multi-head attention*, we have to implement a *residual connection* [2] followed by a *normalization layer*, see Fig. 1.

2. The **C.2** is composed of a *point-wise-feed-forward network* (C.2.1) followed by a *residual connection* [2] and a *normalization layer* (C.2.2). See Fig.1 and Fig. 3.

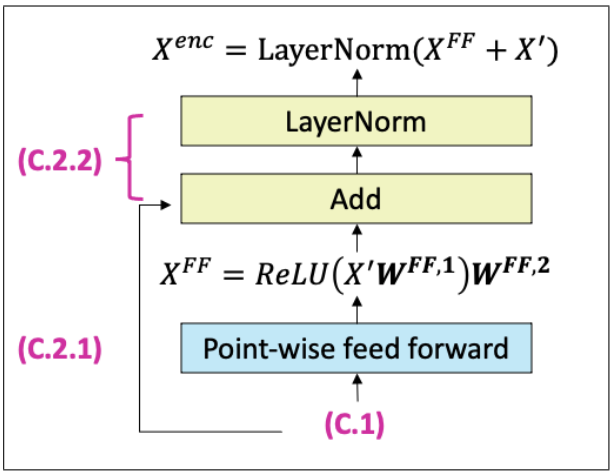


Figure 3: Second component of the encoder block. Image based on the diagram presented on Chapter 6 (lecture slides).

C.2.1 We need to implement C.2.1, which receives as input the output of C.1 into a *point-wise-feed forward network* ($X' = LayerNorm(MultiHead(X^{in}, X^{in}, X^{in}) + X^{in})$). This small *feed-forward network* is composed of two linear transformations with a *ReLU* [1] activation in between.

C.2.2 After implementing C.2.1, we have to implement a *residual connection* [2] and a *normalization layer* following the same architecture of C.1.2, but instead of receiving the original X, it is going to receive the output of C.1 (X').

3. Once we have implemented the encoder block, you need to think if an extra layer(s) to classify the reviews into reviews with positive or negative sentiments is needed. In case that an additional layer(s) is/are needed, implement it/them. *Tip: read Chapter 6 - The prediction layer (lecture slides).*
4. Tokenize and embed the text using the same approach used in Exercise #6 in order to create X.
5. Implement a training loop and train your model with only a sample of the reviews. In this exercise, you can simply deactivate the *mask* in the *multi-head attention*; we will use masking in the following exercise.
6. 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 to execute 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 multiple tutorial on 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. In this exercise, we added "something" to our self-attention mechanism that allows us to learn how a word relates to others in different ways. For example, look at the following reviews' pieces presented in Fig. 4.

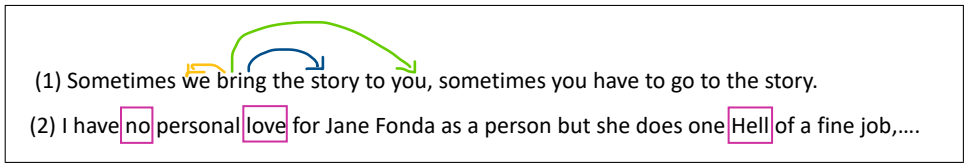


Figure 4: Two example pieces of reviews extracted from the dataset [3].

   - What kind of relation do the yellow, blue, and green arrows represent between the word "bring" and the other words in the first example (e.g., you + have = subject and a modal verb that expresses a command that the subject (you) has to execute)?
   - Do the different types of relations between "no," "love," "hell," and the other words in the sentence matter when classifying the second review into positive or negative review?
   - What is that "something" that allows us to identify different types of relations between words?

2. Consider the encoder block architecture in Fig. 5. In this figure the mathematical equations are presented for each component previously discussed.

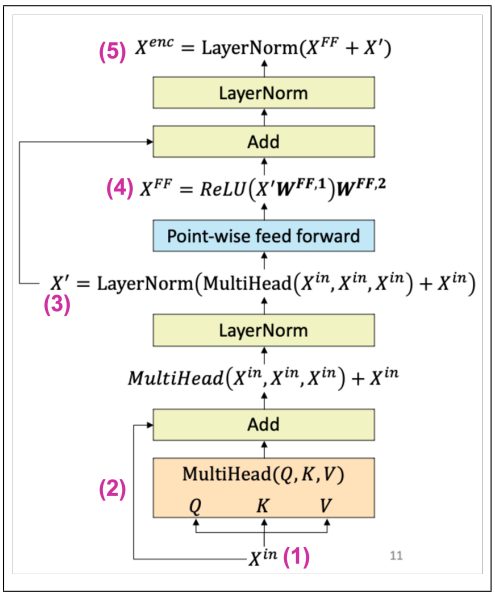


Figure 5: Encoder block. Image taken from Chapter 6 (lecture slides).

Now, define a specific number for:

- Batch size.
- Sequence length (*number of tokens*).
- Embedding size.
- Dimensions for W_i^Q, W_i^K, W_i^V, W^0, $W^{FF,1}$, $W^{FF,2}$.
- Number of heads n_{heads}

Now, identify:

- The number of total parameters in the encoder block.
- The dimensions in Fig. 5 of: (1), (2), (3), and (4).
- In which components/subcomponents (layers/sublayers) the dimensions stayed the same (i.e., the dimensions do not change before and after the (sub)component is applied)?

3. Explain, in your own words:
   - Why do we use *residual connections* (C.1.2 & C.2.2)?
   - Why is C.2.1 called the *point-wise-feed-forward* network and not only *feed-forward network*?
   - Post layer normalization and pre layer normalization (in the context of a transformer model).

References

[1] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[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.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.