Deep Learning for Natural Language and Code

Exercise 3

Prof. Dr. Steffen Herbold
SoSe 2025
Due on 2025/05/22

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 tokenization. In this exercise, you will have the opportunity to build your own implementation of a BPE and a WordPiece tokenizers.

- `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. 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. Use the pre-processing pipeline implemented in Exercise 2 (without the stemming) in order to normalize the reviews. In an initial scenario, convert all the reviews to lowercase and remove punctuations. You can try to change this pre-processing later and understand the implications of it.
2. Implement a Byte-Pair Encoding (BPE) tokenizer.
   (a) As the base vocabulary, use of all the possible individual characters present in the reviews.
   (b) As the base vocabulary, use all the printable ASCII characters.

Use the normalized text in task #1 for training the tokenizer, and for each case, analyze the final obtained vocabulary and the capability of it for handling unknown words.

3. Implement a WordPiece tokenizer.
   (a) As the initial vocabulary, use every character from the corpus.
   (b) As the initial vocabulary, use every character from the corpus + the ASCII printable characters that are not present on it.

   Use the normalized text in task #1 for training the tokenizer, and for each case, analyze the final obtained vocabulary and the capability of handling unknown words.
4. Use BPE and WordPiece Tokenizer from HuggingFaces Library [1] and train them with the same data used for training your model. Compare the obtained vocabulary and analyze potential reasons for the differences.

Restrictions and tips

- The tasks must be solved using only Python Standard Library. You can use implemented data structures but for the text processing/tokenization you should use only the Python Standard Library.
- *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 from exercises.

Theoretical questions

1. Review the following concepts:
   - Word tokenization.
   - Character tokenization.

   What are some flaws/drawbacks of these two types of tokenization? Is sub-word tokenization related to these two types of tokenization? Why yes or why not?
2. In general terms, are there any differences in how "a token that represents non starting subword" is represented by BPE or WordPiece Tokenizers?
3. How are the *unknown words handled* by BPE or WordPiece tokenizers? (Assuming that the algorithms have seen all the possible individual characters)
4. How are the *unknown characters* handled by the BPE and WodPiece tokenizers?
5. For tokenizing a new text, once the algorithms has been trained, what is the process that should be followed by each algorithm? What are the differences between each tokenazation process[1]?

[1] At this point, with tokenazation we refer to the process of convert a new word/sentence into a sequence of tokens

References

[1] Hugging face- tokenizers. `https://huggingface.co/docs/tokenizers/api/models`.

[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.