Deep Learning in NLP

Practical Session P10

Introduction

We continue to work on the implementation of the LSTM-based POS tagger for UD. Recall the description of the task and the last-week exercises.

Preparation

Download the current version of the code from the website (or from github). It contains new code fragments we will work on today and an English UD dataset.

Exercise 5

Train the implemented model (based on word embeddings and a linear scoring layer) on the English UD train set and determine the resulting accuracy on the dev set.

5a

Complete the implementation of the `total_loss` function in `main.py` which calculates the cross-entropy loss over the given batch of sentences. The CrossEntropyLoss usage example should help with this exercise.

5b

Use the training function from `neural/training.py` in order to train the model. It is a variant of the `train` function that we implemented before, abstracted so as to work with generic PyTorch models. Place the training-related code at the end of `main.py`.

Use the following hyperparameters:

- embedding size = 10
- learning rate = 0.01
- number of epochs = 25

Determine the accuracy of the trained model on the develoment set (already loaded as a dataset in `main.py`).

5c (optional)

Tag the sentence *have a break* using the trained tagger. Is there any chance our current tagger could tag it correctly, given the present training set?

Exercise 6

Add LSTM on top of the input word embeddings in order to obtain contextualized word representations, before they are scored with the linear layer.

Before doing this exercise, read the example of the PyTorch LSTM usage (just the example, no need to read about the LSTM hyperparameters just yet).

6a

Add the LSTM submodule in the initialization method of the `PosTagger`. This additionally requires:

- Adding an additional parameter: size of the hidden layer.
- Updating the input size of the linear scoring layer.

6b

In the `forward` method of the `PosTagger`, contextualize the input embeddings using LSTM. The output of LSTM – hidden layer – should be fed as input to the linear scoring layer.

6c

Repeat the training of the model after setting the size of the hidden layer to 10. Otherwise, keep the same hyperparameter values as in **5b**. Determine the accuracy of the trained tagger on the develoment set. Does it benefit from the LSTM layer?

6d (optional)

Experiment with different LSTM hyperparameters:

- bidirectional LSTM (by default, PyTorch LSTM is unidirectional)
- number of layers (by default, one LSTM layer is used)

Is any of these crucial to get good POS tagging accuracy?