DL4NLP 2022 — Exercise 8

Jonas Belouadi, Steffen Eger
Natural Language Learning Group (NLLG),
University of Bielefeld,
Summer Semester 2022

To prepare for the tutorial, you can already install the dependencies listed in `{lstm\_ner,transformer\_ner}/requirements.txt`. If you run into problems read the hints in the exercises.

Task 1 (10min): RNN Extensions

(a) What is the benefit of bidirectional RNNs over unidirectional RNNs? Explain in up to two sentences and give an example.

(b) What is the benefit of adding "output connections" to RNNs? Explain in up to two sentences and give an example.

Task 2 (20min): Named-entity recognition with LSTMs

According to Wikipedia[1], Named-entity recognition (NER) is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

Recurrent Neural Networks like LSTMs are a perfect fit for this cause. However, training RNNs from scratch for this is very resource-intensive which is why will fall back to pre-trained models in this exercise. We will use Nils Reimers BiLSTM-CNN-CRF implementation[2].

(a) Pre-trained networks are available at `https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf/blob/master/docs/Pretrained_Models.md`. Which model would you choose for our task at hand? Download your choice and set the `modelPath` variable in `lstm_ner/run_lstm.py` to the path to your model. The sentences we want to annotate are located in `input.txt`.

(b) The model expects its input to be tokenized into words. Use `nltk.sent_tokenize` and `nltk.word_tokenize`[3] to split the raw text into a list of list with words. Complete the code in `YOUR CODE HERE`. After that run the code, does the output make sense to you?

Hint LSTMs are seldomly used nowadays. Accordingly the code base for this task very old. It was tested with Python versions up to 3.6. If you run into problems with newer versions consider downgrading. Pyenv[4] is a tool which can help you do that easily.

[1] `https://en.wikipedia.org/wiki/Named-entity_recognition`
[2] `https://github.com/UKPLab/emnlp2017-bilstm-cnn-crf`
[3] `https://www.nltk.org/api/nltk.tokenize.html`
[4] `https://github.com/pyenv/pyenv`

Task 3 (15min): Named-entity recognition with Transformers

For this task we return to the state-of-the-art and approach NER with transformers. While transformers are much faster than their LSTM counterparts, they are usually also much larger. Because of this we are going to use DistilBERT for this task. DistilBERT is a Bert-derived transformer encoder which uses a technique called knowledge distillation to reduce model parameters. Compared to Bert, DistilBERT has 40% less parameters, runs 60% faster while preserving over 95% of Bert 's performance. Since DistilBERT is pre-trained we just have to fine-tune it for our problem which is feasible in the time frame of this exercise.

(a) The provided code in `transformer_ner/run_distilbert.py` is complete except for the training loop. Run the code as is and see how well the vanilla DistilBERT model performs on this task. How would you explain the results?

(b) The only code missing for fine-tuning is the following function call:

```
model.fit(
    tf_train_dataset,
    validation_data=tf_eval_dataset,
    epochs=int(training_args.num_train_epochs),
    steps_per_epoch=train_batches_per_epoch,
    validation_steps=eval_batches_per_epoch,
)
```

Analyze the code in `transformer_ner/run_distilbert.py` and put the above snippet where it belongs, then run the code again. How do your results change?

Hint The code is adjusted for training on a CPU. If you have the privilege of owning a CUDA-compatible GPU you can increase `max_train_samples` and `max_eval_samples` to your liking.