Deep Learning for NLP 2022
Homework 4

Due on June 13, 2021 at 23:59

TECHNISCHE UNIVERSITÄT DARMSTADT

Version of May 31, 2022

Overall: 10 points.
This homework involves training more complex neural networks which can take some time. **Plan accordingly for the extended training time!**

Please note that this task requires a different conda environment with an older version of Tensorflow and Keras. Setup a new conda environment with the requirements file in the homework archive and **use it for this task only**. We apologize for the inconvenience.

Submission Guidelines

This homework comes with a zip archive containing the data and code templates. Write your code only in the `rnn.py` file. Then upload a zip file with the following folder structure:

```
hw4_group###.zip
├── hw4.pdf
└── rnn.py
```

- `hw4.pdf` should answer the questions **and** report your results.
- **Do NOT** include the subfolders in your submission (especially not the embeddings and your saved model). You can expect the embeddings to be available at `./embeddings/glove.6B.50d.txt`, the data in the `./data/` folder, and the utils at `./util/utils.py` (where "`.`" is the root folder of your zip).
- `###` is your group number (e.g., the file name could be `hw4_group2.zip`).
- Please make sure that your code prints **all** relevant results for **all** subtasks when the tutors run `rnn.py`.[a]

Make sure that the results you report are reproducible and that your code is runnable in the **alternate conda environment**. If you are aware that your network never stops training, please be honest and add a short statement saying so. Thank you!

[a] If you think that it makes your code more readable/structured, you can also submit other `.py` files whose functions are called in `rnn.py`. But the tutors will only execute `rnn.py`.

1 Sequence Tagging with RNNs (10P)

In this task, you will implement LSTM and Bi-LSTM architectures with Keras to perform part-of-speech tagging (a sequence tagging task).

Several Keras code snippets are provided for this home exercise. There is not much code to write in this exercise, but training RNNs is a computationally intensive task! **Plan accordingly for the extended training time.**

Data We use a subset of the data from the CoNLL-2003 shared task on Named Entity Recognition (provided in the zip). It is pre-partioned into a training, development and test set.

The dataset consists of pre-tokenized sentences where every token is annotated with a part-of-speech tag, a syntactic chunk tag and a named entity tag. In this home exercise, we only use the part-of-speech tag.

1.1 Download Pretrained Embeddings (0P)

Download the pretrained, uncased GloVe embeddings with 6B tokens (`glove.6B.zip`) from Stanford: `https://nlp.stanford.edu/projects/glove/`. For performance reasons, we will only use the 50-dimensional embeddings (`glove.6B.50d.txt`).

1.2 LSTM and Bi-LSTM Model (2P)

Complete the network architecture by adding a respective LSTM and Bi-LSTM layer in the `build_model` method. Since the task is sequence tagging, your LSTM/Bi-LSTM should output a sequence. After the LSTM/Bi-LSTM, add a dense layer with n units to every sequence output, where n is the number of classes in the sequence tagging task plus 1.

Apply dropout before and after the LSTM layer.

Print the model summary in your code. **Copy-paste** or screenshot and paste the model summaries of both the LSTM and the Bi-LSTM into your PDF.

Hint:

- `https://keras.io/layers/recurrent/`

1.3 Intermediate Results (2P)

Set the batch size to 10, dropout to 0.5, the number of hidden units to 100, and train your LSTM and Bi-LSTM on the training dataset for 10 epochs. **Print** and **report** (in your PDF) the final `val_categorical_accuracy` on the development set. **Print** and **report** the macro F1 score you achieve on the test set at this point in time.

1.4 F1 Model Checkpointer (3P)

a) The current implementation stores the model which produced the best `val_categorical_accuracy` on the development set. State in one sentence why this is a problematic approach and why a model checkpointer relying on macro F1 would be better suited. (1P)

b) Complete the `on_epoch_end(...)` method in the skeleton of the `F1ScoreModelCheckpointer` class for using the macro F1 score as the indicator for the best model. Save the best model in each epoch using the method `model.save`. Retrain the model using the parameters from 1.3 and **print** and **report** your new macro F1 score on the test set. (2P)

   Hint: You may use `sklearn.metrics.f1_score(...)`.

1.5 Hyperparameter Optimization (3P)

Experiment with the following hyperparameters: `hidden_units`, `dropout`, `batch_size`.

You may employ random hyperparameter optimization as usual. However, given the duration it takes to train a single configuration, it is fine to manually select a few promising hyperparameter configurations in this home exercise.

Report your three best hyperparameter sets ("best" according to the development set). Also **report** the macro F1 score of these three parameter sets on the test set.