Deep Learning in NLP

Homework 7

Solution to be sent to `waszczuk@phil.hhu.de` and `cwurm@phil.hhu.de` by 21.12.2020 (included, i.e. can be sent till Monday in the evening). It is allowed to do the homework in groups (max. 3 persons per group). Please specify in the email (or in an accompanying README) the authors of the solution.

The archive file with one Python file per exercise (`ex1.py`, `ex2.py`, ...) can be found on the course's website. The missing pieces in the code are marked with `TODO`. Please do not modify the docstring tests, nor the sample datasets refered from the docstrings, they will be used for evaluation. The solution sent by email should have the same form: it should be a zip archive file with the same number of Python files, having the same names: `ex1.py`, `ex2.py`, ...[1] In case the original code contains additional modules required to complete the homework (e.g. `utils.py`, `data.py`, etc.), please include them in your submission.

Evaluation

Unless specified otherwise, the following command will be used to check the solutions:

```
python -m doctest ex1.py
```

Similarly for the other exercises. The solutions which pass all the tests (both documentation tests and the manually designed tests, if any) will be considered as correct.[2]

Exercise 1

Implement the text classification *accuracy* function which, given a neural model and a list of encoded input/output pairs (where the output is the scalar index representing the target class), returns the accuracy of the model on the dataset. The accuracy is in this context defined as the percentage of the dataset pairs for which the model correctly

[1] In case you do not provide a solution for a particular exercise, no need to include it in the archive.

[2] However, we reserve the right to count solutions with significant issues in the code as merely „acceptable", even when all tests pass.

predicts the target class. See the doctests in `ex1.py` for a selection of properties that this function should satisfy.

Note: The *accuracy* function is required for Ex. 2. If you don't know to solve this exercise, just make the function return 0. Note that then you will not know how your model from Ex. 2 performs, though.

Exercise 2

This exercise is a follow-up of the language prediction homework. Your task is to (a) improve the baseline language prediction model and (b) train it on a real-scale name/language dataset.

You can find the training set in the `train.csv` file and the development set in the `dev.csv` file. Extraction and encoding are already taken care of (see `data.py`).[3]

You are free to choose an appropriate feature extraction / contextualisation method to improve the baseline model. What is important is that the improved model is able to reach a sufficiently high accuracy on the development set. Here are some possibilities of how the baseline model could be improved:

1. Contextualise the sequence of character embeddings $(x_i)_{i=1}^{n}$, where n is the length of the input name, using an LSTM. Then, take the last element of the resulting sequence as a representation of the entire input name (in place of the CBOW representation).

2. Apply a (1-dimensional) convolution on top of the input character embeddings. Then, use CBOW to transform the matrix of convoluted embeddings into a single vector of scores.

Hint: As a first step, implement the missing pieces (the training procedure) to be able to tell how well the baseline model performs on the dev set. Only then, try to modify the model itself to improve the performance.

Hint: If you have trouble training the model on the full training set, try to first perform the training on a small part of it (e.g. 10% of training name/language pairs) to make sure training actually works. Note that training in such scaled-down scenario will be significantly faster, so you should be able to perform development faster, too.

Hint: Once you scale up to the full training set, you may need to change the hyper-parameter values to make training successful. In particular, when the training set is larger, you might need to increase the size of the model (the number of its parameters), too. If training diverges, remember that you can also change the optimisation-related hyper-parameters (e.g., the learning rate).

Note: In your submission, please specify the final accuracy achieved on the dev set, as it might take us some time to execute all your code to check the results!

[3] However, you are allowed to implement additional pre-processing if you want. This should not be necessary, though.