

Homework 8:

Neural Networks

Philipp Wicke, PhD
Computerlinguistische Anwendungen

Due: Wednesday July 20, 2022, 14:00

In this homework you will implement sentiment prediction using a long short-term memory RNN, and a convolutional neural network.

If you have not yet installed Keras, you need to do so using the commands:

```
pip3 install --user keras
pip3 install --user tensorflow
```

You can check your progress using unit tests:

```
python3 -m unittest -v hw08_neural_networks/test_neural_networks.py
```

Exercise 1: Creating the vocabulary [2 points]

Complete the function `create_dictionary(texts, vocab_size)` in the file `get_data.py`. It takes a list of word lists, and returns a dictionary mapping the most frequent words to an integer id. The dictionary must also contain a special token (the module variable `UNKNOWN_TOKEN`) with the id 0.

Exercise 2: Mapping texts to ids [2 points]

Complete the function `to_ids(words, dictionary)` in the file `get_data.py`. It takes a word list and a dictionary, and returns a list where every word is mapped to its id. Words which are not in the dictionary are mapped to the id of the special unknown Token (0).

Exercise 3: Training the RNN [6 points]

Now, we will train a bidirectional RNN model, and evaluate it using development data. Make yourself familiar with how the data is read in (`get_data.nltk_data(...)`). Then, complete the function `lstm.build_and_evaluate_model(...)` following the steps below.

1. The data we obtain from `nltk_data(...)` consists of lists of different length. Use the Keras function `pad_sequences(...)` to obtain a numpy array with `MAX_LEN`

columns (longer sequences are cut off, shorter ones are padded). You also have to convert the lists with the labels to numpy arrays.

2. Add the necessary layers to the model. Use the default settings if not specified otherwise.
 - For the embedding layer, use an embedding size of 50.
 - Use a bidirectional LSTM with 25 units (for each direction).
 - Predict the probability for the positive class by predicting 1 value using a dense layer and the sigmoid activation.
3. Compile the model using the binary crossentropy loss (this corresponds to the log-likelihood) and the 'adam' optimizer. Also specify that the model should use accuracy as its metric.
4. Fit the model to the training data. Pass the module variables `BATCH_SIZE` and `EPOCHS` as hyper-parameters. Also provide the development data, in order to monitor training progress.

Exercise 4: Training the CNN [6 points]

An alternative architecture is the CNN. Complete the function `cnn.build_and_evaluate_model(...)` following the steps below (the main differences to the lstm model are marked in **bold**).

1. Again, use the Keras function `pad_sequences(...)` to bring the data into the required format as before. You also have to convert the lists with the labels to numpy arrays.
2. Add the necessary layers to the model. Use the default settings if not specified otherwise.
 - For the embedding layer, use an embedding size of 50.
 - **Use a sequential CNN with an output size of 25 filters that model 3-grams. (In other words, the convolution is 1-dimensional with 25 filters, and the length of the convolution window is 3.) Use the Rectified Linear Unit as the non-linear activation function.**
 - **Get a vector summarizing the whole sentence using max-pooling over the entire sequence.**
 - Predict the probability for the positive class by predicting 1 value using a dense layer and the sigmoid activation.
3. Compile the model using the binary crossentropy loss (this corresponds to the log-likelihood) and the 'adam' optimizer. Also specify that the model should use accuracy as its metric.

4. Fit the model to the training data. Pass the module variables `BATCH_SIZE` and `EPOCHS` as hyper-parameters. Also provide the development data, in order to monitor training progress.