Tutorial 09: Natural Language Processing (21+5P)

In this exercise we will focus on natural language processing (NLP), a branch of AI that incorporates the computational analysis of written language. Similarly to exercise sheet 07 we will do so considering the use case of emotion recognition. The database of interest will be **ISEAR** containing self-reports of participants in a psychological experiment. The individuals tell about occasions in which they felt a certain emotion (anger, fear, disgust, guilt, joy, sadness, shame). The data set is single-label, i.e. every text has exactly one of the emotions associated with it. Our goal here is to predict the emotion given the text.
Many state-of-the-art NLP-approaches are based on pre-trained embeddings, the most popular ones being BERT and its variants (with some very creative names like camemBERT), which are however too resource extensive for this tutorial. For this reason we will look into two other pre-trained word embeddings **word2vec** and **Emotional Word Embeddings (EWE)**.
Please submit your code and results by 14 June 14:15 to manuel.milling@informatik.uni-augsburg.de AND maurice.gerczuk@informatik.uni-augsburg.de.

1 Download Data and Embeddings (1P)

First, download and unzip the data `09_dl_tut_data.zip` the ISEAR data as `.tsv` files in the directory `data/isear/`, as well as the embedding dictionaries of word2vec and EWE as `.vec` in the directory `data/embeddings/`.

2 Load Embeddings (4P)

First, have a look at the `.vec` files. Then, load the data from the word2vec embedding file `word2vec-40k-wiki-news-300d.vec` and create:

- A numpy array `w2v_emb_matrix` of shape `(num_words + 2, embedding_dim)` containing the embedding matrix represented by the file. You need to add two special embeddings at the beginning of the matrix:
  - Row 0 should be zeros. This embedding will be used to pad sequences for batch processing.
  - Row 1 should be the mean of all rows. This embedding will be used for words that are not in the vocabulary (out-of-vocabulary, OOV)
- A dictionary `w2v_word2idx` mapping all words in the file to their index in the embedding matrix.

3 Load data (11P)

Load the three data partitions from the `data/isear/` directory and prepare them for training. Follow the steps below:

- Load the raw sentences, which will end up as the input `X` for our network, as well as the raw string labels from the partition files. Sentences and labels are separated by a tab `'\t'`.
- Unify the sentences such that the following conditions are fulfilled:
  - All punctuation is removed.
  - All white space except for spaces are removed.
  - Leading and trailing spaces are removed.
  - Everything is written in lower case.
- Tokenise the sentences into words (currently separated by a space character) and replace the words with their corresponding index in the embedding dictionary (see above for the OOV case). Finally, make each sentence to be of the same length of `128` by cutting off longer sentences and by padding shorter sentences (see above).
- Convert the string-encoded labels to integers.

The final shapes of `X` and `y` should be `(num_sentences, sentence_length)` and `(num_sentences,)` (the labels can, alternatively, be one-hot encoded).

4 Model Initialisation and Training (5P)

Initialise a model with the following layers:

- An Embedding Layer that converts the word indices to the 300-dimensional embedding vectors of the embedding matrix. Padded inputs should be masked and the layer should be excluded from training.
- A bidirectional LSTM layer with 64 cells and a dropout of 0.5.
- Another bidirectional LSTM layer with 64 cells and a dropout of 0.5.
- A dense layer with 7 neurons and softmax activation.

Note: Be careful with how your LSTM handles the sequential form of the data.
Compile your model with categorical cross-entropy, Adam optimiser (lr=0.005) and monitor the accuracy.
Train your model for 5 epochs with a batch size of 32, monitor the training on the validation data and evaluate the final model on the test data. Results should be an accuracy of $> 50\%$ for validation and test data.

5 Bonus: EWE Embeddings (2P)

Repeat the whole process with the EWE word embeddings `data/embeddings/ewe-40k-300d.vec`, which are specifically pre-trained for emotion related tasks. Use the same hyperparameters as before.
Note: Be careful not to retain the word-to-idx-mapping from the word2vec embeddings.

6 Custom Embeddings (3P)

Repeat the training process again. This time, however, randomly initialise the embeddings and make them trainable. Use the same hyperparameters as before.