1. Repeat exercise 9 of Part 3 (text classification with MLPs), now using a bi-directional stacked RNN (with GRU or LSTM cells) and a self-attention MLP (slide 16), all implemented (by you) in Keras/TensorFlow or PyTorch.[1] Tune the hyper-parameters (e.g., number of stacked RNNs, number of hidden layers in the self-attention MLP, dropout probability) on the development subset of your dataset. Monitor the performance of the RNN on the development subset during training to decide how many epochs to use. You may optionally add an extra RNN layer to produce word embeddings from characters (e.g., as on slide 19), concatenating each resulting character-based word embedding with the corresponding pre-trained word embedding (e.g., obtained with Word2Vec). You may optionally add a pre-trained language model (e.g., ELMo, slides 34–36) as an extra layer to obtain context-sensitive word embeddings.[2] Include experimental results of a baseline majority classifier, as well as experimental results of your best probabilistic classifier from exercise 15 of Part 2 and your MLP classifier from exercise 9 of Part 3 (if you chose that exercise), now treated as baselines. Include in your report:

- Curves showing the loss on training and development data as a function of epochs.
- Precision, recall, F1, precision-recall AUC scores for each class and classifier, separately for the training, development, and test subsets, as in exercise 15 of Part 2.
- Macro-averaged precision, recall, F1, precision-recall AUC scores (averaging the corresponding scores of the previous bullet over the classes), for each classifier, separately for the training, development, and test subsets, as in exercise 15 of Part 2.
- A short description of the methods and datasets you used, including statistics about the datasets (e.g., average document length, number of training/dev/test documents, vocabulary size) and a description of the preprocessing steps that you performed.