Choose from one of the following three options (compute requirements are least for 1 and greatest for 3). For the first two options, your write-up should include a description of your implementation and results reported on the dev and test sets with a comparison to some baseline. For the last option, you should also include a separate text file with your generated responses, so that we can do a human evaluation on a subset of the results.

1) Automatic punctuation prediction (word-based tagging)

The task is to predict the punctuation associated with a lower-cased transcript of conversational speech. For each word in a sequence, predict the following punctuation using the labels: no punctuation (0), comma (1), incomplete sentence (2), question mark (3), and period (4). There are very few exclamation points, so they have been merged with period. The incomplete sentence indicator is needed because people often interrupt themselves or others mid-sentence. The data is based on the Switchboard corpus; a tsv file with lower cased words and labels is available in the assignment folder on canvas, and a division into train/dev/test is in csv files. (The division is based on train/dev/test having conversations {2*,3*}/{44-49}/{40,41} respectively.) In reporting results, you should compute a confusion matrix and compute the macro-F1 score. For a baseline, you can use the result from BertPunc that the grader has uploaded to the HW3 directory. The uploaded file includes no-punctuation in the macro average, but you should exclude it, so the macro P=.345, R=.515, and F1=.413. (Note that BertPUNC does not predict incomplete sentences and probably wasn't designed for such casual speech, so you might be able to beat it.)

You can implement a tagging model using either a CRF, a moving window CNN, an RNN or a transformer to generate a sequence of hidden vectors that are fed into a feedforward layer that predicts the label with a softmax output. Note that if you use a pre-trained transformer that uses word pieces, then you will need a mechanism to get a one-vector per word sequence. You can either take the last word piece or do a mean pooling of the word pieces.

2) Named entity detection in Twitter data (W-NUT 2016 shared task Links to an external site.)
The task is to recognize 10 entity types in Twitter data using a tagging model with BIO (or BIOES) labels for each entity type. If you use a BIO model, you would have 21 labels (2 for each entity type plus 1 for O). The data is available here Links to an external site.. In reporting results, you should use the ConLL evaluation script provided on the data page. Baseline results for a CRF are reported, so you can use that as your baseline to compare to.

You can use all the same options as in (1). A CRF baseline is provided for this task. If you choose to use that baseline, then try adding some new features to the model.

3) Chatbot text generation associated with the PersonaChat dataset Links to an external site.

The task is to generate responses in a dialog based on previous turns and knowledge given about the bot's supposed "persona" (e.g. I'm an artist, I have four children, etc.). The basic idea is that you use a neural sequence model to encode the conversation history and the bot's persona, then you use a language model to predict a sequence of words that will make up the response. The response is done when you predict the end-of-turn "word". The data is available from ParlAI Links to an external site.. There are a variety of automatic metrics that are used to evaluate text generation. Three scores were proposed for this data. We'll just use one of theirs: the perplexity of your language model on the human responses. For a baseline, you can use the seq2seq model result in the paper Links to an external site.. In addition, **we'll do a human evaluation of your results as a class assignment if you submit your generated samples by May 13 noon**. The write-up can be submitted May 14.

You can use a sequence-to-sequence model or a language model that uses the conversation history and bot persona as the start of the sequence, but it will need to be a generative model for which you can compute perplexity.