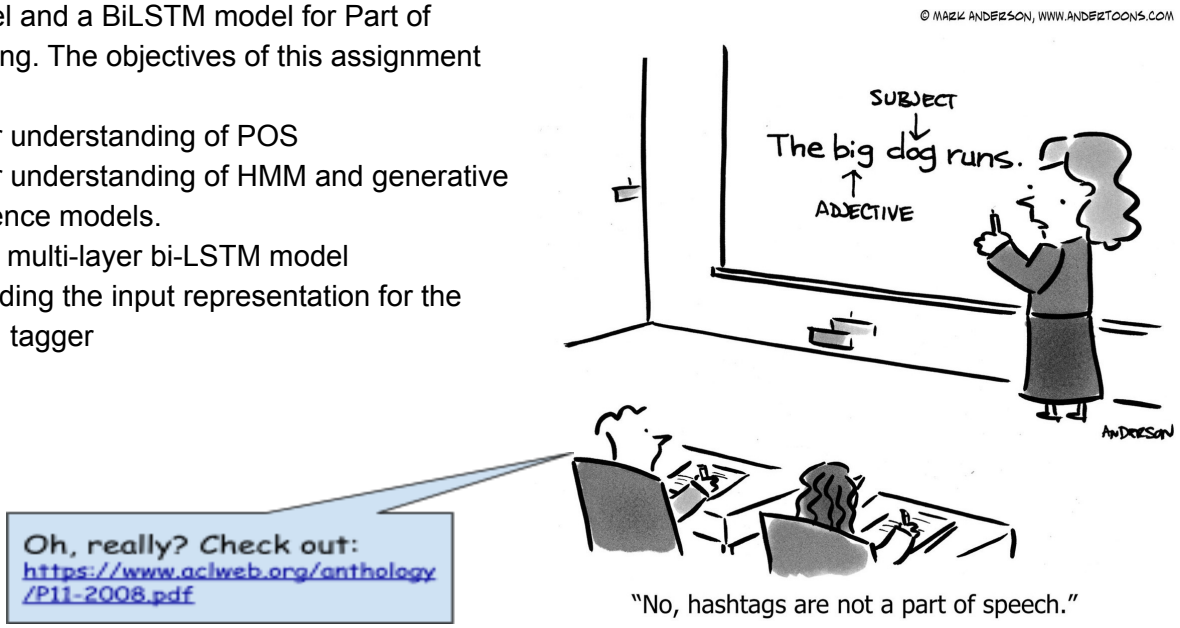
Assignment 4 - Part of Speech (POS) Tagging

In this assignment you will implement a Hidden Markov Model and a BiLSTM model for Part of Speech tagging. The objectives of this assignment are:

1. Better understanding of POS
2. Better understanding of HMM and generative sequence models.
3. Using multi-layer bi-LSTM model
4. Extending the input representation for the LSTM tagger



Submission deadline: 23:59, Jan, 11, 2022

Data and documentation

Extracted tokens and POS tags are taken from the English Web Treebank via the Universal Dependencies Project (http://universaldependencies.org/).
Tagset overviews:
* Universal: http://people.cs.georgetown.edu/nschneid/p/UPOS-English.pdf
* Penn (full new-style tagset): https://spacy.io/docs/usage/pos-tagging#pos-tagging-english
* Penn (examples): http://surdeanu.info/mihai/teaching/ista555-fall13/readings/PennTreebankTagset.html

Data for training and testing: training and development datasets.

Pretrained GloVe embedding vectors are available here: https://www.dropbox.com/s/qxak38ybjom696y/glove.6B.100d.txt?dl=0 (to be used in `load_pretrained(path, vocab=None)`, see API below). You can assume that this is the file we will be using, and there is no need to submit it as part of the tar.gz file.

API and requirements:

A code skeleton with the API, documentation and hints can be found here: tagger.py
You can use all standard modules and packages (not requiring special installation) and the modules imported in the API files (like torch).

Tips and clarifications:

The expected pipeline for the RNN model:

1. Initializing model with all that is needed, including dimensions, training data and pretrained embeddings. It is assumed that preprocessing functions will be called from the **`initialize_rnn_model`** function. This stage returns a dictionary object `model_d`.
2. Training the RNN model: this is done given the output of the initialization `model_d` and a list of annotated sentences.
3. Use the trained model (again, using `model_d`) to tag a new sentence (the sentence is given as a list of words). This is done via the **`tag_sentence(sentence, model)`** function.
4. Evaluation with `count_correct()` can be called. Note that this is a general function.

In testing the LSTM models, we will be calling (this is for the LSTM, the tester included other tests, ofcourse):

```
model = initialize_rnn_model(model_params)
train_rnn(model, train_data)
tag_sentence(sentence, model_to_use)
```

We will be initializing the **`model_params`** before we initialize the model. If you are using extra key/value pairs make sure they have default values that are independent of the params specified in the minimal setting.
Note that this sequence is a 'stand-alone' block and the only input it requires is the `model_params`. Once this dictionary is provided, the sequence should be executed smoothly. You should assume that at least in one setting we will use your `get_best_performing_model_params()` to initialize the model, only changing the paths to the training data and to the pretrained embeddings to load our locally stored files.

Submission guidelines:

1. You should submit one **tar.gz** file with all relevant code files (at least tagger.py, supporting the specified API)
2. You should use Python 3.8 for the coding part.

Integrity

As always - you could (and should) consult each other, but don’t share code.