Practical Natural Language Processing
Instructor: Rasika Bhalerao
Assignment 4
Due October 5 (at the beginning of class at 2pm)

Please submit your solutions using NYU Classes.

Part 1: HMM
Rasika and Zhihao are talking. We want to know who said each word in the conversation. Consider a HMM with two states, Rasika and Zhihao, and a start and end state.

Here are the emission probabilities:
In state Rasika, the HMM can emit “hi” (with 0.6 probability) or “hey” (0.4 probability).
In state Zhihao, the HMM can emit “hello” (0.9 probability) or “hey” (0.1 probability).

Here are the transition probabilities:

- From the start state, the HMM goes to state Zhihao with 1.0 probability (i.e., always).
- From state Zhihao, the HMM can remain in state Zhihao (0.5 probability), go to state Rasika (0.3 probability), or go to state end (0.2 probability).
- From state Rasika, the HMM can remain in state Rasika (0.5 probability), go to state Zhihao (0.3 probability), or go to state end (0.2 probability).

The conversation is “hello hey hi.” (You can ignore punctuation.)

a) Use the Viterbi algorithm to decode the sequence (find the most likely state sequence). What is the probability of emitting this sentence from this state sequence? Please show your work.
b) Is there another state sequence which also generates “hello hey hi?” If so, what is it? What is the *total probability* of emitting this sentence?

Part 2: POS Tagging using NLTK in Python
For this part, we will explore POS tagging using nltk (https://www.nltk.org/). Nltk is a popular and powerful Python platform called Natural Language Toolkit. You can install nltk using `pip` or the instructions on their website.

Import nltk using `import nltk`.

You will need to download the necessary corpora. The easiest way to do this is to run `nltk.download()` and in the GUI that pops up, select `all`, and click `Download`. It may take a couple minutes to download.

Once the download is complete, you can import functions that split text into sentences and split sentences into words:

```
from nltk.tokenize import word_tokenize, sent_tokenize
```

You can split text into sentences and sentences into words like this:

```
sentences = sent_tokenize("Fish sleep. Cats sleep more.")
words = word_tokenize("Fish sleep.")
```

`sentences` will have an array where each element is a sentence, and `words` will have an array where each element is a token.

You can output the POS tags for each sentence in `sentences` like this:

```
for i, sentence in enumerate(sentences):
    print(f'Sentence {i}: {sentence}')

    words = nltk.word_tokenize(sentence)
    tags = nltk.pos_tag(words)

    for word, tag in tags:
        print(f'{word}: {tag}')

    print()
```

a) Come up with your own sentence that is tagged incorrectly by the POS tagger. Show the tags output by the POS tagger, and explain what the correct tags should be.

b) `nltk.corpus.brown.tagged_words()` will return a list of tuples: the POS tagged Brown Corpus. Take the words and re-predict their POS tags using `nltk.pos_tag()`. Make a confusion matrix to analyze these POS tag predictions and plot the confusion matrix as a heatmap (with axis labels). Discuss in a sentence or two which tags are predicted correctly, and which tags are being mistaken for other tags. Find an example of an incorrectly predicted tag from the corpus and use it to support this short discussion. Hints:

   1. You will need to set the `tagset` parameter to the same value for the `tagged_words` function and the `pos_tag` function.
   2. Use Scikit-learn to make the confusion matrix: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
   3. Use Seaborn's heatmap: https://seaborn.pydata.org/generated/seaborn.heatmap.html