TCSS 456 Assignment 2

NOTE: Be sure to adhere to the University's **Policy on Academic Integrity** as discussed in class. Programming assignments are to be written individually and submitted programs must be the result of your own efforts. Any suspicion of academic integrity violation will be dealt with accordingly

Assignment Details:

Consider the following sentences written in Klingon. For each sentence, the part of speech of each "word" has been given (for ease of translation, some prefixes/suffixes have been treated as words), along with a translation. Using these training sentences, we're going to build a Hidden Markov Model (HMM) to predict the part of speech of an unknown sentence using the Viterbi algorithm.

N	**PRO**	**V**	**N**	**PRO**
pa'Daq	ghah	taH	tera'ngan	'e
room (inside)	he	is	human	of

The human is in the room

V	**N**	**V**	**N**
ja'chuqmeH	rojHom	neH	tera'ngan
in order to parley	truce	want	human

The enemy commander wants a truce in order to parley

N	**V**	**N**	**CONJ**	**N**	**V**	**N**
tera'ngan	qIp	puq	'eg	puq	qIp	tera'ngan
human	bit	child	and	child	bit	child

The child bit the human, and the human bit the child

Step 1: Creating the Emission probability table (`emission.java` or `emission.py`)

Create a Emission probability table by computing the frequencies of each part of speech in the table below for all POS tags. We'll use a smoothing factor of 0.1 (as discussed in class) to make sure that no event is impossible; add this number to all of your observations. Sample table values of two parts of speech have been shown.

Probability(word|tag) = Count(word,tag) / Count(tag)

	NOUN	VERB	CONJ	PRO
'e			0.1	1.1
'eg			1.1	0.1
ghaH			0.1	1.1
ja'chuqmeH			0.1	0.1
legh			0.1	0.1
neH			0.1	0.1
pa'Daq			0.1	0.1
puq			0.1	0.1
qIp			0.1	0.1
rojHom			0.1	0.1
taH			0.1	0.1
tera'ngan			0.1	0.1
yaS			0.1	0.1

Step 2: Creating the Transition probability table (`transition.java` or `transition.py`)

Generate a transition probability table by calculating the transition frequencies from one POS tag to another. Now, for each part of speech, total the number of times it transitioned to each other part of speech. Again, use a smoothing factor of 0.1. After you've done this, compute the start and transition probabilities. Sample table values of transition for two parts of speech have been shown.

$$\text{Probability}(tag_i | tag_{i-1}) = \text{Count}(tag_{i-1}, tag_i) \;/\; \text{Count}(tag_{i-1})$$

	NOUN	VERB	CONJ	PRO
START				
N			1.1	2.1
V			0.1	0.1
CONJ			0.1	0.1
PRO			0.1	0.1

Step 3: Viterbi Decoding (`decoding.java` or `decoding.py`)

Now consider the following test sentence:

"*tera'ngan legh yaS*"

Use the Viterbi algorithm (in slides), to generate the most likely sequence of POS tags for the given test sentence.

Data File:

The training examples from above are already in the given text file. Each line is a new sentence and each sentence consist of tokens separated by space. Each token is in the following format: `word/POS`.

- Klingon_Train.txt

Read in the text file to create the transition and emission probability table.

Python users: You can use NLTK to break the `word/POS` tokens into tuples.

NOTE: ‘ is considered to be part of the Klingon word. No need to remove it or consider it a separate punctuation.

Submission Guidelines:

Submit your files on Canvas using the Programming Assignment 2 submission Link. **You will submit a zip file containing:**

- Turn in your 3 models: emission, transition, and decoding
- `Readme.txt`: your Readme should consist of the following
    - values from the Emission table
    - values from the Transition table
    - Tagged output of the given test sentence in the format:

      ```
      tera’ngan/POS legh/POS yaS/POS
      ```

    - (For fun) What do you think this sentence means? What word is the subject of the sentence?