Arizona State University		Homework#3	POS tagging
CSE 476 Introduction to NLP		Due date:	February 28, 2022
Spring 2022		**Name:**	
Instructor: Eduardo Blanco			

Instructions
Submit by pushing to your repository:

- A pdf file with the answers to Q1 and the written part of Q2. For Q2, write a brief summary of what you did (in particular, discuss your results and your strategies to deal with unseen words). Include answers to the extra credit questions if you choose to do them.
- An updated version of `pos_tagger.py` with your implementation for Q2. Please write all your code in this file.
- We will run your code on unseen sentences (we have a *real* test set), so do not hardcode anything.
- A good implementation runs in around 2 minutes. Your implementation has to run in less than 10 minutes. The vast majority of running time will be predicting. Training the model (i.e., estimating probabilities) is fast. When you are developing and debugging your code, test with only a few sentences instead of the whole test file.
- There are 4 major milestones to implement an HMM part-of-speech tagger. Your submission will be graded accordingly:
  - Calculate the prior and likelihood probabilities (`create_model`).
  - Create and fill the Viterbi Matrix correctly (`predict_tags`).
  - Obtain the optimal sequence of tags after filling the Viterbi Matrix (`predict_tags`).
  - Your strategies to deal with unknown words, the overall results, and the report.

Question 1 [20pt]
Go to `https://explosion.ai/demos/displacy`, uncheck *Merge Phrases*, and run spaCy with the following sentences (do not change anything, copy and paste the sentences below (one at a time) without changing anything. Do not change punctuation, upper and lower case, etc.):

1. `Move left to the wall on the right side of the sink where the towel holder is.`
2. `The LAS & LDWS does not operate until the system detects white (yellow) lane lines on either the left or right.`
3. `Place it on the middle shelf.`
4. `Place it on the middle grey shelf.`
5. `Sentences with tipos are harder.`
6. `state or describe exactly the nature, scope, or meaning of.`
7. `U.S. school district to allow students back in classrooms`
8. `Fourth-Largest U.S. School District to Allow Students Back in Classrooms`
9. `I want comprar un dog.`
10. `Encantado de conocerte.`
11. `Voici votre carte d'embarquement.`
12. `I want comprar un dog.`

Comment on the POS tagging errors you see. The following questions may help you, but I expect you to dig a bit deeper. You cannot prove anything, but you can certainly make hypotheses about what may be the causes of errors:

- Do you see any tokenization issues?
- Do you think that *left* is usually a verb, an adjective or an adverb in the training corpus?
- Do you think the training corpus includes the tokens *LAS* and *LDWS*?
- Is it reasonable to assume that words unseen during training are proper nouns?
- Does the system include a language detector?
- Do you think letter case affects the system?

Note: You may run spaCy from Python if you prefer, but you do not have to.

Question 2 [80pt]
Implement a bigram HMM POS tagger. During training, you have to calculate two kinds of probabilities (store them in two large dictionaries):

- $P(t_i|t_{i-1})$ (similar to the bigram LM you have already implemented, use Laplace smoothing). These are the prior probabilities.
- $P(w_i|t_i)$, the probability of observing word w_i given POS tag t_i. You don't need to smooth this probability distribution. These are the likelihood probabilities.

Use Viterbi decoding during testing (you have to update `token.tag` with the right tag).
Your can run the starter code with the following command:

```
$ python3 code/pos_tagger.py data/train data/heldout --mode always_NN
** Testing the model with the training instances (boring, this is just a sanity check)
[always_NN  ] Accuracy [35000 sentences]:  12.90 [not that useful, mostly a sanity check]

** Testing the model with the test instances (interesting, these are the numbres that matter)
[always_NN:11] Accuracy [13928 sentences]:  12.84


$ python3 code/pos_tagger.py data/train data/heldout --mode majority
** Testing the model with the training instances (boring, this is just a sanity check)
[majority   ] Accuracy [35000 sentences]:  12.90 [not that useful, mostly a sanity check]

** Testing the model with the test instances (interesting, these are the numbres that matter)
[majority:11] Accuracy [13928 sentences]:  12.84


$ python3 code/pos_tagger.py data/train data/heldout --mode hmm
** Testing the model with the training instances (boring, this is just a sanity check)
[hmm        ] Accuracy [35000 sentences]:  12.90 [not that useful, mostly a sanity check]

** Testing the model with the test instances (interesting, these are the numbres that matter)
[hmm:11] Accuracy [13928 sentences]:  12.84
```

There are three part-of-speech taggers. The first one always predicts `NN` and is already implemented. The second one will be the *majority tag per word* baseline and you have to implement it. The third one will be the HMM tagger and you have to implement it.

A few hints:

- You only need to modify methods `create_model(sentences)` and `predict_tags(sentences, model)`. Don't change any other code.
- When calculating $P(t_i|t_{i-1})$, don't cross sentence boundaries. Also, add `<s>` at the beginning of each sentence.

- Unknown words. You will find words w in the test set that you have never seen before, so you don't know $P(w|t_i)$. You can estimate $P(t_i|w)$ using several methods:
  - Uniform distribution (guess randomly).
  - Prior probabilities $P(t_i)$ from training. If you have to guess, you are more likely to guess right if you choose the most likely tag. Even better: choose the most likely tag for unseen words.
  - Use morphology to assign more probability to more likely tags for that word. This is simple and effective, and it will make a big difference. For example, unseen words that ...
    * start with an upper case character and are not the first token are likely to be proper nouns (related question: are proper nouns more likely to be unseen?);
    * end in *-ed* are most likely a verb in past tense, a verb past participle or an adjective;
    * unseen words that are a bunch of numbers are `CD` (related question: are you likely to see all possible combinations of digits in training?);
    * and many others.

    It is worth going over the list of POS tags and try to define some rules that intuitively make sense for some tag (and only some tags, for example, it is most likely useless to define rules for unseen prepositions).

  Once you have $P(t_i|w)$, you can get $P(w|t_i)$ using Bayes:

$$P(w|t_i) = \frac{P(t_i|w) \times P(w)}{P(t_i)}$$

  One option to estimate $P(w)$ is to consider all unknown words the same, and assign a very low probability to them. You could also use Good-Turing discounting.
  I recommend that you start thinking about morphology only after you have Viterbi decoding working.
- Your should get around 95% after implementing Viterbi and a bit higher (97–98%) after you implement a few rules for unseen words.
- Grading: 20 points for the majority baseline, 60 points for the HMM tagger, and 20 point for your report discussing your results and the rules you use for unseen words.

Extra Credit You can get extra credit by:

- Calculating (and discussing) accuracy for seen and unseen words. Which ones are harder? How good are you predicting tags for unseen words?
- Calculating (and discussing) accuracy per tag (or clustered tags, e.g., `NOUN`, `VERB`, `ADJECTIVES`, `OTHER`. Which tags are easy and difficult to predict? Are prepositions easy to tag?
- A few students asked me why we bother assigning POS tags to words. We will see more interesting applications later in the class, but you can already implement a relatively interesting application: capitalizing titles.
  You can get a decent set of rules to capitalize titles from here: `http://www.quickanddirtytips.com/education/grammar/capitalizing-titles`. Implementing these rules should be easy once you have a working POS tagger. The input should be a sentence that has already been tokenized (e.g., *A novel method to predict the weather*), and the output should be the same sentence with the proper capitalization (e.g., *A Novel Method to Predict the Weather*).
  You are responsible to create your own test cases. Determining whether "Prepositions are only capitalized if they are used adjectivally or adverbially" is harder than it looks. We will not be too picky testing your implementation, but do discuss when it fails. Also, there are several styles to capitalize title (e.g., some guidelines state that tokens with more than 4 characters should always be capitalized). Explain briefly the rules you decided to implement.

- Scoring at the top of the class on unseen text (to make sure you don't overfit). The top 20 submissions will get up to 10 extra points (top 10: 10 points, submissions 11 through 20: 5 points). The easiest way to score high is to do well with unseen words. You can also use a trigram model, but that takes more time for a small improvement.