CS 6320: Natural Language Processing

Homework 2: Hidden Markov Models (50 points)

Due date: 09/27/2019 4pm

For this homework, you will develop a hidden Markov Model for part-of-speech (POS) tagging, using a modified Brown corpus as training data. To simplify the problem, we will use a tagset which is composed of 11 tags. i.e. Noun, Pronoun, Verb, Adjective, Adverb, Conjunction, Preposition, Determiner, Number, Punctuation and Other.

The corpus for this assignment can be downloaded from this link:

http://www.hlt.utdallas.edu/~takshak/brown_modified_pos.zip

Your goal is to design and implement a class 'Tagger' that is able to tag the following sentences:

1. The Secretariat is expected to race tomorrow .
2. People continue to enquire the reason for the race for outer space .

The class should support the following methods:

1. `load_corpus(path)`: This function loads the corpus at the given path and returns it as a list of POS-tagged sentences. Each line in the files should be treated as a separate sentence, where sentences consist of sequences of whitespace-separated strings of the form "token/POS". Your function should return a list of lists, with individual entries being 2-tuples of the form (token, POS).

   For example, consider the input sentence shown below:

   ```
   B./NOUN J./NOUN Connell/NOUN is/VERB the/DETERMINER present/ADJECTIVE treasurer/
   NOUN and/CONJUNCTION manager/NOUN ./PUNCT
   ```

   Your function should return the output as: `[('B.', 'NOUN'), ('J.', 'NOUN'), ('Connell', 'NOUN'), ('is', 'VERB'), ('the', 'DET') ... ]`

2. `initialize_probabilities(sentences)`: This function takes as input the list of sentences generated by `load_corpus()` and initializes all variables needed by the tagger. In particular, if $\{t_1, t_2, ..., t_n\}$ denotes the set of tags and $\{w_1, w_2, ..., w_m\}$ denotes the set of tokens found in the input sentences, you should compute:

   - The initial tag probabilities $\pi(t_i)$ for $1 \leq i \leq n$, where $\pi(t_i)$ is the probability that a sentence begins with tag t_i.
   - The transition probabilities $a(t_i \rightarrow t_j)$ for $1 \leq i,\ j \leq n$, where $a(t_i \rightarrow t_j)$ is the probability that tag t_j occurs after tag t_i.
   - The emission probabilities $b(t_i \rightarrow w_j)$ for $1 \leq i,\ j \leq m$, where $b(t_i \rightarrow w_j)$ is the probability that token w_j is generated, given tag t_i.

   Note that you have to use add-one smoothing to ensure your tagger does well with new inputs.

3. `viterbi_decode(sentence)`: This method returns the most likely tag sequence for the two sentences using the Viterbi algorithm.

   For example, consider the input sentence shown below:

   sentence = "People race tomorrow . "

   Your function should return the output as: `['NOUN', 'VERB', 'NOUN', 'PUNCTUATION']`

A sample Python skeleton file is attached as a reference to this homework. If you are using a programming language other than Python, please use the same class definition for that language. Essentially, we should be able to import your class as a package and use these methods directly.

To submit: Submit a single zip file that contains the following:

1. All your source code (do not include the corpus files).
2. A README file giving clear instructions on how to run the program.
3. A report containing the tags output by the method `viterbi_decode(sentence)` for the two sentences.