Part-of-speech tagging

A basic problem in Natural Language Processing is part-of-speech tagging, in which the goal is to mark every word in a sentence with its part of speech (noun, verb, adjective, etc.). This is valuable for improving the performance of NLP systems for tasks such as machine translation.

Sometimes this is easy: a sentence like "Blueberries are blue" clearly consists of a noun, verb, and adjective, since each of these words has only one possible part of speech (e.g., "blueberries" is a noun and can't be a verb).

But in general, one has to look at all the words in a sentence to figure out the part of speech of any individual word. For example, consider the —- grammatically correct! —- sentence: "Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo." To figure out what it means, we can parse its parts of speech:

Buffalo	buffalo	Buffalo	buffalo	buffalo	buffalo	Buffalo	buffalo.
Adjective	Noun	Adjective	Noun	Verb	Verb	Adjective	Noun

(In other words: the buffalo living in Buffalo, NY that are buffaloed (intimidated) by buffalo living in Buffalo, NY buffalo (intimidate) buffalo living in Buffalo, NY.)
That's an extreme example, obviously. Here's a more mundane sentence:

Her	position	covers	a	number	of	daily	tasks	common	to	any	social	director.
DET	NOUN	VERB	DET	NOUN	ADP	ADJ	NOUN	ADJ	ADP	DET	ADJ	NOUN

where DET stands for a determiner, ADP is an adposition, ADJ is an adjective, and ADV is an adverb.1 Many of these words can be different parts of speech: "position" and "covers" can both be nouns or verbs, for example. The only way to resolve the ambiguity is to look at the surrounding words. Labeling parts of speech thus involves an understanding of the intended meaning of the words in the sentence, as well as the relationships between the words.

Fortunately, statistical models work amazingly well for NLP problems. Consider the Bayes net shown in Figure 2. This Bayes net has random variables S = {S1,...,SN} and W = {W1,...,WN}. The W 's represent observed words in a sentence. The S 's represent part of speech tags, so Si ∈ { VERB, NOUN, ...}. The arrows between W and S nodes model the relationship between a given observed word and the possible parts of speech it can take on, P (Wi | Si). (For example, these distributions can model the fact that the word "dog" is a fairly common noun but a very rare verb.) The arrows between S nodes model the probability that a word of one part of speech follows a word of another part of speech, P (Si+1 | Si). (For example, these arrows can model the fact that verbs are very likely to follow nouns, but are unlikely to follow adjectives.)

Data. To help you with this assignment, we've prepared a large corpus of labeled training and testing data. Each line consists of a sentence, and each word is followed by one of 12 part-of-speech tags: ADJ (adjective),ADV (adverb), ADP (adposition), CONJ (conjunction), DET (determiner), NOUN, NUM (number), PRON (pronoun), PRT (particle), VERB, X (foreign word), and . (punctuation mark).

What to do. Your goal in this part is to implement part-of-speech tagging in Python, using Bayes networks.

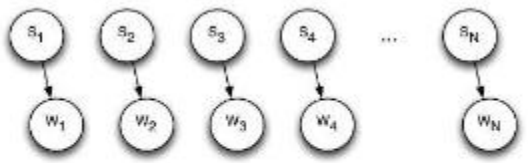

Figure 1: Simplified Model

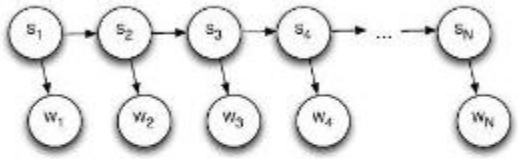

Figure 2: HMM

To get started, consider the simplified Bayes net in Figure 1. To perform part-of-speech tagging, we'll want to estimate the most-probable tag s*i for each word Wi,

$$s_i^* = \arg\max_{s_i} P(S_i = s_i \mid W)$$

Implement part-of-speech tagging using this simple model.

2. Now consider Figure 2, a richer Bayes net that incorporates dependencies between words. Implement Viterbi to find the maximum a posteriori (MAP) labeling for the sentence,

$$(s_1^*, \ldots, s_N^*) = \arg\max_{s_1,\ldots,s_N} P(S_i = s_i \mid W)$$

Your program should take as input a training filename and a testing filename. The program should use the training corpus to estimate parameters, and then display the output of Steps 1-2 on each sentence in the testing file. For the result generated by both approaches (Simple and HMM), as well as for the ground truth result, your program should output the logarithm of the joint probability P(S,W) for each solution it finds under each of the models in Figure 1 and 2. It should also display a running evaluation showing the percentage of words and whole sentences that have been labeled correctly so far. For example:

```
python3 ./label.py training_file testing_file

Learning model...
Loading test data...
Testing classifiers...
                  Simple    HMM    Magnus    ab    integro   seclorum   nascitur   ordo   .
 0.  Ground truth  -48.52  -64.33    noun    verb     adv       conj       noun     noun   .
  1.  Simplified   -47.29  -66.74    noun    noun     noun      adv        verb     noun   .
     2.  HMM       -47.48  -63.83    noun    verb     adj       conj       noun     verb   .

 ==> So far scored 1 sentences with 17 words.

                  Words correct:  Sentences correct:
 0.  Ground truth        100.00%            100.00 %
   1.  Simplified          2.85 %              0.00 %
          2.  HMM         71.43 %              0.00 %
```