IWS 12-13. HMM & POS TAGGING.

Exercises

5.1 Find one tagging error in each of the following sentences that are tagged with the Penn Treebank tagset:

1. I/PRP need/VBP a/DT flight/NN from/IN Atlanta/NN
2. Does/VBZ this/DT flight/NN serve/VB dinner/NNS
3. I/PRP have/VB a/DT friend/NN living/VBG in/IN Denver/NNP
4. Can/VBP you/PRP list/VB the/DT nonstop/JJ afternoon/NN flights/NNS

5.2 Use the Penn Treebank tagset to tag each word in the following sentences from Damon Runyon's short stories. You may ignore punctuation. Some of these are quite difficult; do your best.

1. It is a nice night.
2. This crap game is over a garage in Fifty-second Street...
3. ...Nobody ever takes the newspapers she sells ...
4. He is a tall, skinny guy with a long, sad, mean-looking kisser, and a mournful voice.
5. ...I am sitting in Mindy's restaurant putting on the gefillte fish, which is a dish I am very fond of, ...
6. When a guy and a doll get to taking peeks back and forth at each other, why there you are indeed.

5.3 Now compare your tags from the previous exercise with one or two friend's answers. On which words did you disagree the most? Why?

5.4 Now tag the sentences in Exercise 5.2; use the more detailed Brown tagset in Fig. 5.7.

5.5 Implement the TBL algorithm in Fig. 5.21. Create a small number of templates and train the tagger on any POS-tagged training set you can find.

5.6 Implement the "most likely tag" baseline. Find a POS-tagged training set, and use it to compute for each word the tag that maximizes $p(t|w)$. You will need to implement a simple tokenizer to deal with sentence boundaries. Start by assuming that all unknown words are NN and compute your error rate on known and unknown words. Now write at least five rules to do a better job of tagging unknown words, and show the difference in error rates.

5.7 Recall that the Church (1988) tagger is not an HMM tagger since it incorporates the probability of the tag given the word:

$$P(\text{tag}|\text{word}) * P(\text{tag}|\text{previous } n \text{ tags}) \tag{5.59}$$

rather than using the likelihood of the word given the tag, as an HMM tagger does:

$$P(\text{word}|\text{tag}) * P(\text{tag}|\text{previous } n \text{ tags}) \tag{5.60}$$

Interestingly, this use of a kind of "reverse likelihood" has proven to be useful in the modern log-linear approach to machine translation (see page 903). As a gedanken-experiment, construct a sentence, a set of tag transition probabilities, and a set of lexical tag probabilities that demonstrate a way in which the HMM tagger can produce a better answer than the Church tagger, and create another example in which the Church tagger is better.

5.8 Build a bigram HMM tagger. You will need a part-of-speech-tagged corpus. First split the corpus into a training set and test set. From the labeled training set, train the transition and observation probabilities of the HMM tagger directly on the hand-tagged data. Then implement the Viterbi algorithm from this chapter and Chapter 6 so that you can decode (label) an arbitrary test sentence. Now run your algorithm on the test set. Report its error rate and compare its performance to the most frequent tag baseline.

5.9 Do an error analysis of your tagger. Build a confusion matrix and investigate the most frequent errors. Propose some features for improving the performance of your tagger on these errors.

5.10 Compute a bigram grammar on a large corpus and re-estimate the spelling correction probabilities shown in Fig. 5.25 given the correct sequence ... *was called a "stellar and versatile* **acress** *whose combination of sass and glamour has defined her..."*. Does a bigram grammar prefer the correct word *actress*?

5.11 Read Norvig (2007) and implement one of the extensions he suggests to his Python noisy channel spellchecker.