Step 4: POS tagging

In this exercise we will implement a Markov POS tagger using the some of the outcomes of the preceding three steps.

Intermediate code due : Tuesday 23rd Feb. 2016
Final results (code and 4-page report) due : Tuesday 1st March 2016

1 The Tagger

Corpus: Use the training and test corpora which are given on 'blackboard'. The training corpus consists of sections 02-21 of the Wall Street Journal corpus (WSJ02-21.pos.gz) and the test corpus consists of section 23 of the Wall Street Journal corpus (WSJ23.pos.gz).

POS Tagger: This is a standard probabilistic POS tagger:

Language/Transition model: condition every POS on the POS tag of one preceding word (1st order Markov model on POS tags – bi-grams).

Lexical/Emission model: condition every word only on its own POS.

Tasks:

- Program the tagger in two steps:

  step 1: a training step in which the training corpus is used to estimate the probabilities for the models.

  step 2: a tagging step, using the Viterbi algorithm, in which the program receives a sentence as input and outputs the most probable POS tag sequence for the sentence (given the estimated models), that is:

$$argmax_{tags} P(tags \mid input\ sentence)$$

Advised strategy Implement step 2 above using dummy (not real) data; you can hard-code the training probabilities in the code. Use lecture slides for this, and check that you get the same numbers. Once the Viterbi algorithm is debugged using this dummy data, and you are sure it is working, use it for the real data. Submit this code as the intermediate code, at the very minimum (do more if you can). Write a few lines at the top of your code describing which parts are complete and debugged (e.g. Viterbi algorithm, reading in the WSJ data, building the lexical model, etc.)

- Test the tagger on the sentences of the test corpus and evaluate the output (compared with the 'correct' tagging given in the test corpus). Report accuracy of the tagger (see definition below).
- Smooth both the lexical and the language model of the tagger using the Good-Turing method (previous exercise). Report accuracy (compared with the 'correct' tagging given in the test corpus). For the language model, smooth only frequencies no greater than $k = 4$ (use the formulas of step 3).

  Smooth the lexical model separately for each tag (see extra slides in lecture 5 (46,47,48) explaining why). Consider as unknown words only words which do not appear at all in the training corpus. (That is, if a word w appears in the training corpus but not with the tag t, then $P(w|t)$ remains 0 after smoothing.) When you smooth in this way, it does not matter how many unknown words there are and you can assume 1 unknown word when calculating the Good-Turing smoothing (i.e., $n_0 = 1$).

  Because many POS tags have a very low frequency, it is best to smooth the lexical model $P(word \mid tag)$ only for events with frequencies of 0 and 1. In this case we only give half of the frequency of events of frequency 1 to events of frequency 0. The formula is for this smoothing is:

$$0^* = \frac{1}{2}\frac{n_1(t)}{n_0}$$

$$1^* = \frac{1}{2}$$

Because we assume that $n_0 = 1$ (the number of unknown words is 1) we get that for an unknown word u with tag T, $P(u|t) = \frac{1}{2}\frac{n_1(t)}{N(t)}$ which results in a higher probability for tags which have many words with frequency 1.

Definitions: The correctness of a tagger on a test corpus of length N words (sum of lengths of all sentences in the corpus) is given by:

$$Accuracy(tagger) \quad = \quad \frac{(number\ of\ words\ correctly\ tagged\ by\ tagger)}{N}$$

Here we only count words (that is, not the START and STOP symbols).

2 Instructions for the experiments

- Consider "./." or "============" as the end of a sentence. An empty line does not indicate the end of a sentence.
- Ignore, both in the training and the test corpus, all pairs X/Y where Y does not begin with an alphanumeric character (that is, all POS tags, such as punctuation, which do not begin with an alphanumeric character can be thrown out of the corpora).
- You may ignore all sentences in the *test* file longer than 15 words. Don't do this for the training corpus, so as not to lose valuable statistics.

3 To submit

- The program (incl source files), use the following command line:

  Trains the model, tests it on a given test file, save the predictions and reports the accuracy, with and without smoothing
  ```
  ./tagger -smoothing [yes|no] -train-set [path] -test-set [path]
  -test-set-predicted [path]
  ```
- The test corpus tagged by your tagger (with and without smoothing).
- The accuracy of your tagger on the test corpus (with and without smoothing).

- Final report for all steps (see separate instructions on Blackboard).