Computational Linguistics
Assignment 2 (2021-11-05)

Winter Semester 2021/22 – Prof. Dr. Alexander Koller

Part-of-speech tagging with HMMs

Implement a bigram part-of-speech (POS) tagger based on Hidden Markov
Models from scratch. Using NLTK is disallowed, except for the modules
explicitly listed below. For this, you will need to develop and/or utilize the
following modules:

1. Corpus reader and writer (10 points)
2. Training procedure (30 points)
3. Viterbi tagging, including unknown word handling (50 points)
4. Evaluation (10 points)

The task is mostly very straightforward, but each step requires careful de-
sign. Thus, we suggest you proceed in the following way.

Viterbi algorithm. First, implement the Viterbi algorithm for finding the
optimal state (tag) sequence given the sequence of observations (words). We
suggest you test your implementation on a small example for which you
know the correct tag sequence, such as the *Eisner's Ice Cream* HMM from
the lecture.

Make sure your Viterbi algorithm runs properly on the example before you
proceed to the next step. Submit the best state sequence x that your Viterbi
implementation finds for $y = 3, 1, 3$ and its joint probability $P(x, y)$.

There are plenty of other detailed illustrations for the Viterbi algorithm on
the Web from which you can take example HMMs. Please resist the temp-
tation to copy Python code from those websites; that would be plagiarism.

Training. Second, learn the parameters of your HMM from data, i.e. the
initial, transition, and emission probabilities. Implement a maximum like-
lihood training procedure for supervised learning of HMMs.

You can get a corpus at `http://www.coli.uni-saarland.de/~koller/`
`materials/anlp/de-utb.zip`. It contains a training set, a test set, and

an evaluation set. The training set (`de-train.tt`) and the evaluation set (`de-eval.tt`) are written in the commonly used CoNLL format. They are text files with two colums; the first column contains the words, the POS tags are in the second column, and empty lines delimit sentences. The test set (`de-test.t`) is a copy of the evaluation set with tags stripped, as you should tag the test set using your tagger and then compare your results with the gold-standard ones in the evaluation set. The corpus uses the 12-tag universal POS tagset by Petrov et al. (2012). Feel free to use the NLTK module `nltk.corpus.reader` (and submodules) for reading the corpus.

Evaluation. Once you have trained a model, evaluate it on the unseen data from the test set. Run the Viterbi algorithm with each of your models, and output a tagged corpus in the two-column CoNLL format (`*.tt`). We will provide an evaluation script on Classroom. Run it on the output of your tagger and the evaluation set and report your results.

Note that your tagger will initially fail to produce output for sentences that contain words you haven't seen in training. If you have such a word w appear at sentence position t, you will have $b_j(w) = 0$ for all states/tags j, and therefore $V_t(j) = 0$ for all j. Adapt your tagger by implementing the following crude approach to unknown words. Whenever you get $V_t(j) = 0$ for all j because of an unknown word w at position t, pretend that $b_j(w) = 1$ for all j. This will basically set $V_t(j) = \max_i V_{t-1}(i) \cdot a_{ij}$, and allow you to interpolate the missing POS tag based on the transition probabilities alone.

Extra credit. The task is challenging as it stands. However, feel free to go further for extra credit, e.g. by doing one of the following: implement better unknown word handling, use a trigram tagger, plot a learning curve for your tagger (accuracy as a function of training data size), plot a speed vs. sentence length curve.

Please submit your code, instructions for running your tagger and tagging output(s). Document any additional data you submit. With this, you will have implemented your first POS tagger! Well done! ☺