CS-481/585 — Artificial Intelligence: Natural Language Understanding

Programming Assignment: Part-of-Speech Tagging

You are to experiment with and enhance a statistical part-of-speech tagger that uses the Viterbi algorithm. A unigram POS tagger is supplied (written in Java), which tags each word according to the single most probable tag for that word. (If you prefer, you may start from scratch in any reasonable programming language, but make sure to CLEARLY document your code.) Files of text with part-of-speech tags (in an SGML format) are also supplied for training and testing (in the dat/ directory).

Please see the readme.txt file and the instructions document about how to hand-in programming assignments for more information.

You may work as individuals or in pairs. If you work in a pair, you must implement one enhancement (see below). If you are taking CS-585 (the graduate course) you must implement one extra enhancement. Hence a pair of students taking CS 585 must implement two enhancements. Extra credit will be given for correctly implementing and evaluating extra enhancements (provided the base functionality is correct!).

Problems:

1. Evaluate the accuracy of unigram part-of-speech tagging on the given test files.
    - Train on small-train.xml and test on text_1.xml. What is the accuracy?
    - Train on small-train.xml and test on text_2.xml. What is the accuracy?
    - Can you explain the discrepancy in accuracy between them?
    - Examine the output of the tagger and see if any patterns emerge regarding where the tagger tends to make more errors.
    - What conclusions can you draw? What do you think might improve the tagger's performance?
2. Modify POSTag.java to tag sentences using a bigram tagger, i.e., so that the probability of a tag depends on the previous tag as well. You will need to add a Hashmap to store Tag-Tag probabilities, using plus-one (or plus-*k*) smoothing. Document all your changes.
3. Evaluate your bigram tagger on both test files. How did performance change?
4. Optional enhancements (at least one required for pairs or CS-585 students):
    - Extend the context to give a trigram tagger.
    - Implement a better smoothing scheme (such as Good-Turing) for the probabilities.
    - Do something intelligent in the model with capitalized words or morphology (words ending in -ing or in -s) to enhance the probabilistic model. Talk to the professor first, though, since this may be non-obvious...
    - Implement a transformational POS tagger using the statistical tagger as the initial tagging system.

    Make sure to document your work both in the code, and in an accompanying document explaining your changes. If you have another (comparatively simple) idea to enhance your tagger, that's great - but talk to the professor first...
5. Evaluate your chosen enhancement(s) in the same way you evaluated the base system.
    - Is the accuracy improved?
    - Does your chosen enhancement reduce any of the types of errors that you previously noted for the base tagger? If so, which and why?
    - How do you think you might improve the tagger even more?