# Assignment 4: Structured Discriminative Training

CSCI 562

due: 11am, 2012 Nov 6

In this assignment, you will build a discriminatively-trained, supervised part-of-speech tagger. You may use any of the algorithms we discussed in class (perceptron, maximum-likelihood, large-margin), as long as your model is a linear or log-linear model with the features:

| feature | meaning | weight |
|---|---|---|
| $f_{t,w}$ | number of times word $w$ is tagged $t$ | $\lambda_{w,t}$ |
| $f_{t,t'}$ | number of times tag $t$ is followed by tag $t'$ | $\lambda_{t,t'}$ |
| $f_{\texttt{<s>},t}$ | number of times tag $t$ starts sentence | $\lambda_{\texttt{<s>},t}$ |
| $f_{t,\texttt{</s>}}$ | number of times tag $t$ ends sentence | $\lambda_{t,\texttt{</s>}}$ |

You may reuse code from previous assignments. You may also reuse someone else's code from previous assignments as long as you clearly acknowledge what code you reused, and from whom.

## Getting started

Download `hw4.tgz` from Blackboard. This package contains:

- This file
- `train.tags`, training data with gold tags
- `dev` and `dev.tags`, development data without and with gold tags
- `test` and `test.tags`, test data without and with gold tags
- `eval.py`, evaluation script
- `svector.py`, module for sparse vectors
- `bigfloat.py`, module for very small floating point numbers

The format of the `*.tags` files is `word1_tag1 word2_tag2 ...`

**Question 1** In `train.tags`, how many word types and how many tag types (that is, how many words/tags, not counting duplicates) are there? What word(s) have the most distinct tags? (You will use these data structures in the next part.)

## Implementation

Implement the Viterbi algorithm for the tagging model. Depending on what kind of model you use, the weight of a path should be either of:

$$s(\mathbf{w}, \mathbf{t}) = \sum_i \lambda_i f_i(\mathbf{w}, \mathbf{t})$$

$$P(\mathbf{t} \mid \mathbf{w}) = \frac{1}{Z(\mathbf{w})} \exp \sum_i \lambda_i f_i(\mathbf{w}, \mathbf{t})$$

Your tagger should output a tagging in the same format as `dev.tags`. **Note:** If a word $w$ has been seen before, you only need to consider those tags for $w$ that have been observed with $w$ in `train.tags`. But if $w$ has not been seen before, you must consider all possible tags.

Choose one of the three learning algorithms we discussed in class (perceptron, maximum-likelihood, large-margin) and implement it.

**Question 2** What algorithm did you implement? Briefly describe any decisions you had to make in its implementation.

**Question 3** How does this model handle unknown words differently from how a generative model would?

**Question 4** Train your model on `train.tags`. After each pass through the training data, report your accuracy on both the training data and `dev.tags`. Did you see any evidence of overfitting? (For comparison, a supervised generative model with naïve smoothing gets 77%.)

## Experiments

Now we will try to improve tagging accuracy. Your credit on the next two questions depends on the thoroughness and creativity of your investigation and the clarity of your description.

**Question 5** Try a different learning method. Or, experiment with varying the hyperparameters (regularization, learning rate) of your learning method. You're free to try something not discussed in class. Describe what you tried, and how it affected tagging accuracy and convergence time.

**Question 6** Try adding some new features to your model. Describe them and their effect on tagging accuracy.

**Question 7** What is the best accuracy you achieved on `dev`? Use this model to tag `test` and record your accuracy. Your credit on this question (10 points out of 120) depends on your score.

## What to hand in

- On paper: your answers to all questions above
- On Blackboard: all code you wrote for this assignment, and your outputs on `dev` and `test`.