Chapter 10

Part-of-Speech Tagging

10.1 Problem

Let's turn to a new task: Given a sentence, assign a part-of-speech (POS) tag (noun, verb, etc.) to each word. Many words have a unique POS tag, but some words are ambiguous: for example, *short* can be an adjective (*short vowel*), a noun (direct a *short*), an adverb (to throw a ball *short*) or a verb (to *short* an appliance). Figuring out which POS is the correct one depends on the context, including the POS tags of the neighboring words.

This is a sequence labeling task, just like the first version of the typo correction problem we looked at, which involved only character substitutions and no insertions or deletions. Other examples of sequence labeling tasks are:

- Word segmentation: Given a representation of a sentence without any word boundaries, reconstruct the word boundaries. In some languages, like Chinese, words are written without any spaces in between them. (Indeed, it can be difficult to settle on the definition of a "word" in such languages.) This situation also occurs with any spoken language.

- Word sense disambiguation: Given a dictionary which gives one or more *word senses* to each word (for example, a *bank* can either be a financial institution or the sloped ground next to a river), and given a sentence, guess the sense of each word in the sentence.

- Named entity detection: Given a sentence, identify all the proper names (Notre Dame, Apple, etc.) and classify them as persons, organizations, places, etc. The typical way to set this up as a sequence-labeling problem is called *BIO tagging*. Each word is labeled *B* (beginning) if it is the first word in a named entity, *I* (inside) if it is a subsequent word in a named entity, and *O* (outside) otherwise. Other encodings are possible as well.

10.2 Hidden Markov Models

Previously, we talked about Hidden Markov Models in the context of typo correction. Now the reason for the choice of variable names **t** and **w** emerges – part-of-speech tagging is the classic application of HMMs in NLP, and **w** stands for "words" and **t** stands for "tags." The same methods we discussed previously can be used to do supervised part-of-speech tagging fairly well.

What about unsupervised part-of-speech tagging? Given just plain text, can an HMM automatically figure out what the parts of speech are? With some caveats. . .kind of.

The procedure at a high level would look like this:

1. Choose a set of possible tags.
2. Start with an initial model (estimates for the $p(t' \mid t)$ and $p(w \mid t)$).
3. E step: For each sentence in the trianing data, use the Forward-Backward algorithm to compute a distribution over possible taggings; or (for hard EM) use the Viterbi algorithm to find the best tagging.
4. M step: Count up the tag bigrams and tag-word pairs, and restimate $p(t' \mid t)$ and $p(w \mid t)$.
5. Go to 3.

The first problem is with step 1. You can choose a tag set like $\{\text{N}, \text{V}\}$, but there's absolutely no way for EM to know that you want N to be the nouns and V to be the verbs. So there's no point in giving the tags names/meanings; just number them, and try to assign names/meanings afterwards.

A second, related, problem is with step 2: if you initialize the model to uniform probabilities, then the model will be perfectly symmetric between the tags. So EM will get stuck, not doing anything. (This didn't happen to us before, because we assumed a pre-trained language model that breaks the symmetry.) Fortunately, the solution is very simple: just add some random noise to the initial probabilities.

The third problem, also related to the first, is that there's nothing here that tells EM to look for *parts of speech*; it could go off and look for something else instead, like words about finance versus words about politics. Whatever helps the model give the highest probability to the observed sentences is what EM will try to find, and that might or might not be parts of speech.

I'm not aware of a satisfying solution to this problem. There's a restricted version of the task, commonly known as the Merialdo task, that solves all three problems by assuming the availability of a dictionary that says, for each word, what all the possible tags for that word are. Under this restriction, EM is able to do something resembling POS tagging.

Yet another problem is overfitting. If you choose the tag set to be too large (say, one million), then EM could simply give each word *token* its own tag, which would allow the model to give the observed sentences a very high probability. (Why?) Even if you choose a tag set with a reasonable size (45), though, this problem can crop up in subtle ways.

10.3 More Models

For supervised POS tagging, HMMs were replaced a long time ago by *conditional random fields* (Lafferty, McCallum, and Pereira, 2001). We'll defer the details of CRFs until later (when we talk about named entity recognition), but the essential idea is that a CRF is a finite automaton whose weights are not required to be probabilities; they're just nonnegative numbers. The weights are learned to maximize the weight of observed tag sequences and minimize the weight of other tag sequences.

More recently, CRFs have been improved by attaching them to neural networks: a recurrent neural network runs over an input string and computes a sequence of vectors, and the weights of the CRF are partly computed from these vectors. Again, we will try to look at details when we talk about named entity recognition.

Bibliography

Lafferty, John, Andrew McCallum, and Fernando Pereira (2001). “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *ICML*, pp. 282–289.