1 Part-of-Speech Tagging

Bags-of-words and n-grams are very simple language models. Part-of-speech (POS) models are the next step in terms of complexity. These models assume that each token in a sequence of natural language is paired with one of a relatively small set of POS tags.

1.1 HMMs and CRFs

We'll compare two methods for inferring POS tag sequences.

Hidden Markov Model HMMs can be used as a generative model of any system of discrete, linear symbols, such as language. An HMM usually takes a sequence of observed output symbols, like tokens of language, and infers the underlying states that 'produced' each symbol, which correspond to the POS tags. It models the full joint distribution of labels and observations. This means it can also infer the most likely tokens (output), given a sequence of tags, but this is not a very useful task when dealing with language.

Conditional Random Fields CRFs are a more recently developed alternative to the HMM approach. CRFs are discriminative models—they only model the conditional probability of labels, e.g., POS tags, given a sequence of tokens. CRFs do not perform exact inference; they are basically a combination of logistic regressions whose parameters are learned via Maximum Likelihood estimation, via an optimization method such as L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno).

HMMs can be trained easily and quickly using dynamic programming. Multiple iterations are unnecessary, and a convergence criterion does not need to be specified. However, they are not as accurate as CRFs, because they are not optimized for the label (POS tag) inference task. HMMs are not easily extended to handle multiple features, while that is trivial for CRFs.

There is a trade-off between training time and accuracy. CRFs take considerably longer to train, but they are more accurate.

The results for a series of comparisons between Mallet's[1] CRF and HMM implementations are show below in Table 1.

Model	Corpus	Training accuracy	Testing accuracy	OOV-only	Time (seconds)
HMM	Atis	0.888	0.865	0.180	2.987
CRF	Atis	0.999	0.928	0.273	84.971
CRF+	Atis	0.999	0.940	0.440	84.570
HMM	WSJ	0.862	0.785	0.379	51.304
CRF	WSJ	0.951	0.752	0.431	2795.950
CRF+	WSJ	0.992	0.862	0.626	5608.262

Table 1: 'CRF+' refers to the CRF with additional word-property features; 'OOV-only' refers to accuracy on out-of-vocabulary items in the held-out test data. ATIS refers to a corpus of conversations related to airline reservation; each figure in those rows is the average over a 10-fold train/test split, training on 80% of the data and testing on the remaining 20%. WSJ refers to the Wall Street Journal; the models were trained on section 00 and tested on section 01.

The CRF almost unilaterally beats the HMM in both the very constrained corpus (ATIS) and a more diverse corpus (WSJ). While the results are not state-of-the-art, they clearly indicate the big differences between the HMM and CRF. The WSJ HMM happens to beat the CRF in testing

[1] MALLET: A Machine Learning for Language Toolkit. `http://mallet.cs.umass.edu`

data accuracy, but CRF performance in the current configuration varies markedly. Over the 10-fold evaluation of the ATIS data, the CRF test accuracy ranges between 90.6% and 94.2%, nearly four percentage points. A more robust CRF implementation would likely run more iterations and have less variable results.

As we might expect, test accuracy among only out-of-vocabulary (OOV) items is much worse than among in-vocabulary tokens. The CRF is a consistent winner in this task, demonstrating an advantage of the discriminative model. The HMM struggles with OOV tokens, since it models the joint distribution of both labels and tokens, and it has never seen these tokens before. The CRF results leave much room for improvement, but the CRF is much better tuned to solving the problem of inferring POS tags for OOV tokens.

The HMM is not quite as prone to over-fitting as the CRF. The CRF has astounding test accuracy results, particularly on the smaller data set. Therefore, the difference between training and testing performance, for the CRF, is much more pronounced. I don't think this has much to say in terms of advantages or disadvantages of one model over the other, but just shows how crucial it is to have a test dataset.

The largest difference between the HMM and CRF models is the training time required. Due to the complex optimizations that the CRF has to perform in inferring its parameters, it takes between 20 and 100 times longer to train. Whether or not this is a major consideration depends on the application. I imagine that in most cases, the increase in accuracy is well worth a couple orders of magnitude more compute time, even if the increased accuracy is disproportionately small compared to the increased training time. Like testing accuracy, this too has a lot of variability, in part due to other jobs running on the same machine, or differences in processing power between each machine.

Adding binary morphological features significantly increases accuracy. The CRF+ model above differed from the CRF model only in terms of the additional features, but it performed remarkably better. The features I added were:

GERUND Ends in 'ing'

PLURAL Ends in 's'

SHORT Less than four characters

UPPER Starts with an uppercase character

These features mark a lot of false positives as PLURAL or GERUND, but they still help increase accuracy, particularly on OOV items.

1.2 Reversed Model

My extension to the established model was to run the same experiment, but after reversing each sentence, as in the first part of the first homework. Because the HMM is a directed model and the CRF is a form of a Markov Random Field (MRF), which is not directed, we might expect to see differences between the 'forward' HMM and the 'backward' HMM, but no differences in the CRF performance.

Model	Corpus	Training accuracy	Testing accuracy	OOV-only	Time (seconds)
HMM	Atis	0.901	0.882	0.197	5.594
CRF	Atis	0.998	0.929	0.313	85.081
CRF+	Atis	0.999	0.942	0.462	83.374
HMM	WSJ	0.863	0.785	0.383	87.367
CRF	WSJ	0.996	0.811	0.480	6377.220
CRF+	WSJ	0.994	0.874	0.620	5782.205

Table 2: Exactly the same figures as before, but all after reversing each input sentence string.

However, the results are practically indistinguishable from those in the first table. This doesn't say so much about the directionality, I think; we saw the same effect in the first homework's experiment where we got identical results for both forward and backward models, as long as we left out the sentence boundaries.

1.3 Instructions

Again, all code, with full SBT project, is available at:

```
git clone https://github.com/chbrown/nlp.git
```

Some commands used to run the code can be found in 'hw02.README'.