Part of Speech Tagging with Discriminatively Re-ranked Hidden Markov Models

Brian Highfill
Stanford University
Department of Computer Science

***Abstract*- The task of part of speech tagging has been approached by various ways. Originally, constructed by way of hand-crafted rules for disambiguation, the majority of tagging is now accomplished by utilizing statistical machine learning methods. Two commonly applied statistical methods are hidden Markov models (HMM) and an extension of Markov chains combined with a maximum entropy classifier called maximum entropy Markov models (MEMM). This paper explores POS tagging by combining a standard HMM tagger separately with a maximum entropy classifier designed to re-rank the best sequence of tags produced by the HMM. Tested on the Brown corpus (Francis, 1979; Francis and Kucera, 1982), the discriminatively re-ranked tagger performed with an accuracy of 91.8%, slightly less than that of the HHM tagger alone.**

I. INTRODUCTION

A common task in natural language processing is part of speech (POS) tagging. Automatic part of speech tagging is used, for instance, to aid speech synthesis systems in pronunciation and information retrieval (IR) in the context of parsing and word sense disambiguation (Jurafsky, Martin, 2009).

Statistical POS tagging commonly involves using a corpus of sentences, in a particular language, which has been already tagged with part of speech information. There has also been research on unsupervised methods when a pre-tagged corpus is not available (Jurafsky, Martin, 2009). The machine learning methods used on supervised POS tagging use the statistics of word context to learn models of how part of speech is associated with words in a sentence. Once a model has been learned, some form of inference is required to select the best sequence of tags for a particular sentence. This is also known as decoding.

One common statistical method attempts to learn a hidden Markov model of POS attachment. HMM's are generative algorithms which treats part of speech tags as unseen "hidden" states and the words of a sentence as observations. The model assumes the tags form a Markov sequence such that each tag transitions to the next tag in a sentence with a probability only dependent on the previous tag. At each step of the generative process, a given tag "produces" a word from a probability distribution dependent only on that tag. The goal of a HMM POS tagging is to first learn these model parameters using a pre-tagged corpus. When one of these tagged "gold standard" training sets is available, the transition and observation probabilities can be estimated directly by counting methods. When used on its own, HMM POS tagging utilizes the Viterbi algorithm to generate the optimal sequence of tags for a given sentence.

Maximum entropy classification is another machine learning method used in POS tagging. MaxEnt models are discriminative and can be used to directly model the posterior probability of a POS tag given a word and its context in the sentence. The MaxEnt method is based on a number of features (in many cases, binary features) which indicate certain properties of context which are useful for making a tag decisions.

It is common to apply the supervised learning method of expectation-maximization (EM),

an iterative algorithm, to estimate the model parameters (Jurafsky, Martin, 2009). Once a MaxEnt model is learned, a slightly modified version of the Viterbi algorithm can be applied to a Markov chain of tags to select the best sequence of tags. Using a Markov chain of tags in this way together with a MaxEnt model for POS tagging is called maximum entropy Markov modeling (MEMM).

II. METHODS

A. Part of Speech Tagging

Given a sequence (sentence) of words w^n with n words, we seek the sequence of tags t^n of length n which has the largest posterior:

$$\hat{t}^n = \underset{t^n}{\operatorname{argmax}} P(t^n|w^n)$$

Using a hidden Markov models, or a MaxEnt model, we will be able to estimate this posterior. For our experiment, we built a HMM and MaxEnt model separately, then used a beam search with the HMM to produce the (approximately) top most likely sequences. Finally, we used our MaxEnt model to score each of these sequences and picked the top choice as the tagging for each sentence.

B. Hidden Markov Models

Using Bayes' rule, the posterior above can be rewritten as:

$$\hat{t}^n = \underset{t^n}{\operatorname{argmax}}\, P(w^n|t^n)P(t^n)$$

That is, as a product of a likelihood and prior respectively. Furthermore, making the (Markov) assumption that part of speech tags transition from one to another depending only on the last tag and that observation (word) probabilities only depend on the current tag, the above simplifies to:

$$\hat{t}^n = \underset{t^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Each of these parameters can be learned independently via counting:

$$P(t_i|t_{i-1}) = a_{i-1}\frac{C(t_{i-1}, t_i) + L}{C(t_{i-1})}$$

And

$$P(w_i|t_i) = b_i\frac{C(t_i, w_i) + L}{C(t_i)}$$

A small constant was added to each of the maximum likelihood estimates for smoothing (for our experiment, we used L = 0.001). Using a value of L much smaller than one (Laplace smoothing) results in better smoothing. This is because less probability mass will be "stolen" from non-zero counts. The constants in front of each equation above are chosen so that each distribution sum to one over each of the unconditioned variables. In practice, this is done by simply dividing each distribution by the post-smoothed sum.

Additionally, every HMM is also parameterized by a set of initial state (tag) probabilities. These were computed from counts of the first tag in each of the training sentences:

$$P(t_0) = \frac{C(t_0)}{Number\ of\ Sentences}$$

Dealing with words unseen in the training set is an issue that also face. To model them, we estimated the probability of an unknown word to be the fraction of words from the training sentences observed only once. Then for each training tag, we modified the above expression for $P(w_i|t_i)$ adding an unknown word token with a conditional probability as give above. Finally, each $P(w_i|t_i)$ is renormalized to a proper distribution.

C. MaxEnt Model

The MaxEnt model directly estimates the posterior probability:

$$\hat{t}^n = \underset{t^n}{\operatorname{argmax}}\, P(t^n|w^n)$$

We make the additional assumption that the posterior is conditionally dependent on the previous tag and the current word:

$$\hat{t}^n = \underset{t^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(t_i|w_i, t_{i-1})$$

Finally, given a set of features counts $f_i(t_i, w_i, t_{i-1})$ and a set of feature weights λ_i, we calculate the conditional likelihood as:

$$P(t_i|w_i, t_{i-1}, \lambda_i) = \frac{exp(\sum_i \lambda_i f_i(t_i, w_i, t_{i-1}))}{\sum_{t'} exp(\sum_i \lambda_i f_i(t', w_i, t_{i-1}))}$$

The features are usually hand-chosen, and can, in general, be dependent on more than just the previous tag and current word. In our experiment, we also looked at a few of the words ahead. We define the objective function (negative conditional log likelihood):

$$F(\vec{\lambda}) = - \sum_{\langle t_i, w_i, t_{i-1} \rangle \in \langle T, W \rangle} \log P(t_i|w_i, t_{i-1}, \lambda_i)$$

We can minimize this function over the weights using the expectation-maximization (EM) algorithm to learn the weighs.

D. Re-ranking Best HMM Tag Sequences using a Maximum Entropy Classifier

Normally, when using a HMM for POS tagging, one estimates the optimal sequence of tags using the Viterbi algorithm (Jurafsky, Martin, 2009). The Viterbi algorithm uses dynamic programming to search for the highest probability sequence posterior. The joint probability of any sequence of words with tags given is:

$$P(t^n, w^n) = P(w^n|t^n)P(t^n)$$
$$= P(w_0|t_0)P(t_0)\prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Thus the Viterbi algorithm finds the sequence of tags t^n which maximizes the above joint probability. One problem with relying on just the HMM and its Viterbi sequence as our estimated best tagging, is that the HMM may be wrong. Our HMM parameters learned may not be quite right, or more importantly, the HMM model may not be adequate for POS tagging; the strong independence assumptions that HMM's make may not be appropriate.

To help alleviate these problems, it may be useful to use another model (specifically the MaxEnt model) to assist in the HMM's decision of the best tagging for each new sentence. To this end, we forsake the Viterbi algorithm all together, but we can still use the joint probability calculation from above.

In a process called, discriminative re-ranking, we will use the HMM and a beam search to chose the top k POS-taggings for a sentence and utilize the MaxEnt model to choose the best tagging at the end.

The beam search creates a tree in a breadth first manner, at each expansion pruning all but the k highest ranked leaves. Here, the leaves of tree at step t are the k best length-t tags of the first t words in the sentence.

In the first step, $t = 1$ all r possible tags from the tagset are produced for the first word w_0 (figure 1).

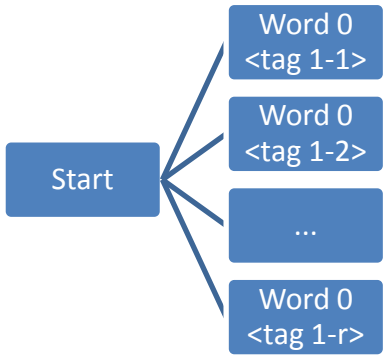

Figure 1: First step in beam search, generating all possible tags for the first word.

Next, each of the generated tag-word combinations is evaluated (scored) according to the joint probability. Only the top k are selected. Then,

we generate r tags stemming from each of first-word tagging (figure 2).

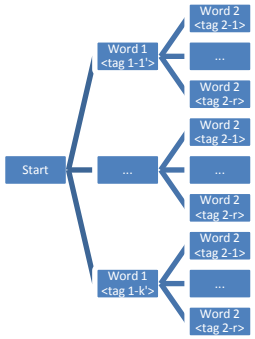

Figure 2: After selecting best k tags of word 1, producing all possible tags for word 2.

This process continues until we have only the top best k taggings of the entire sentence (figure 3).

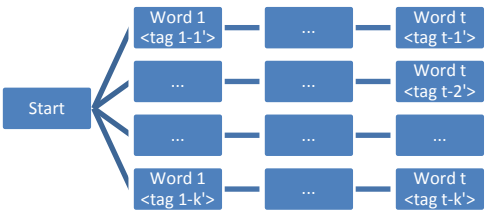

Figure 3: Final production and pruning, leaving only the top k sentences taggings

Beam search is a heuristic search algorithm, and thus we are not guaranteed the optimal tagging. This means that the Viterbi sequence of tags may not be among the k sequences returned by the beam search. We are willing to take this disadvantage in hopes that an even *better* sequence may be among these k sequences.

In the final step of discriminative re-ranking, well allow the MaxEnt classifier to re-score each of the k sequences to determine the best one.

III. Experiment

In this project we trained both a HMM and a MaxEnt classifier to re-rank the top POS tagging for each sentence of a test set using a beam search.

A. Corpus

Many English corpora exist complete with hand-tagged POS which can be used to train an automatic POS tagger. We used the ICAME Brown corpus dataset (ICAME-Browntag) licensed to the Stanford NLP group. This widely used data set contains the 57,066 POS tagged sentences of the original Brown corpus. The data set has 87 different POS tags; however, many words are ambiguous with respect to POS and have been tagged by multiple ways. We treat each of these "compound" tags as separate tags resulting in a total of 312 tags. Using the first 56,000 sentences as training, the training lexicon has size 52,279.

B. Baseline Model

As a baseline, we trained and used a simple most frequent class POS tagger which tags each word with the most frequent tag is occurred within the training set. The first 500 sentences of the Brown dataset were used for training and the final 1,066 sentences used for testing.

C. HMM Model

We trained a HMM model using support code from CS224n modified to directly estimate model parameters from tagged training data. The transition and observation probabilities were smoothed as discussed earlier by uniformly adding L=0.001 to each count and renormalizing. Unknown words were treated the same and estimated to be as frequent as words seen once in the training data. We used the first 56,000 sentences of the Brown corpus as training for the model, and the last 1,066 sentences to test on.

For inference, including tag sequence probabilities and Viterbi sequence calculation, we

interfaced our code with the common Java package JAHAMM.

D. MaxEnt Model

A maximum entropy model was trained using code developed for an earlier class project. The features were hand-selected and are shown in table 1 below.

Due to heavy computation, we chose to train the MaxEnt model on only the first 500 sentences of the Brown dataset.

Testing was performed on the same final 1,066 sentence as with the HMM model. Some of the features were chosen from intuition, like *current word, and previous tag.* These tend to be good baseline features. Other features were chosen to fix certain types of errors. By adding inflectional endings (i.e. *–ed, -s, -ing)* and derivational endings (i.e. *–ion, -al, -ive, and –ly)* we can improve recognition of some POS tags (Weischedel et al., 1993). Adding the morphological and orthographic features also aids recognition POS for unknown words. Some features were in light success from the work of other authors contributing to POS tagging using MaxEnt models. Toutanova et al. (2003) found that adding a feature detecting an uppercase word followed within 3 words by Co. or Inc. helped to increase recognition of certain proper nouns.

Table 1. MaxEnt Features

Feature
Current word
Previous tag
<Word AND previous Tag>
Word contains a numeral
Word ends with “ing”
Word end with “ness”
Word ends with “ity”
Word ends with “ed”
Word ends with “able”
Word ends with “s”
Word is capitalized
Word ends with “ion”
Word ends with “al”
Word ends with “ive”
Word ends with “ly”
Word is hyphenated
Word is all uppercase
Word is all uppercase with a numeral
Word is capitalized and a word ending in “Co.” or “Inc.” is found within 3 words ahead

E. MaxEnt Re-ranking using a HMM and Beamsearch

After training the HMM on the first 56,000 sentences and the MaxEnt model on the first 500, we put together the two models using a beam search (as described above) with widths varying between 1 and 20. Testing was done on the final 1,066 sentences of the dataset.

IV. Results

The results are tabulated in table 2 below. The HMM model, used by itself and tagging by means of the Viterbi algorithm, achieved the highest accuracy of 92.96%. Using the MaxEnt re-ranked HMM model with a beam width of k=1 resulted in an accuracy of 90.10%. When the beam width is k=1, this corresponds to a greedy search of the best tag sequence. In this case, the HMM works alone since at each word, the beam search simply selects the most probable sequence that can be created by adding only one tag. Since the resulting search ends

with only one choice, the MaxEnt model bears no influence on the final POS tag sequence.

When the beam width is increased, the MaxEnt model has more choices to re-rank. Thus, the larger the beam width (k), the more influence the MaxEnt model is able to assert on the final taggings. In the limit that the beam width were to be infinite, the search would be breadth first without any pruning, and the final set of sequences given to the MaxEnt model would be *every possible sequence*. Thus, when k=infinity, the HMM bears no influence on the final tagging.

We can think of the beam width as a parameter which lets us control how much influence each of the two models has on the final tagging decision. When k was increased to k=2, the accuracy of the tagger increased to 91.79%. At k=4, the accuracy reaches a peak of 91.80%. This may suggest that increasing the beam width further only allows the MaxEnt model to interfere destructively with the HMM model. It is interesting to note that although the accuracy on known words continued to decrease as k was increased, the accuracy on unknown words increased. This is partly due to the greater influence of the MaxEnt model which can take advantage of contextual clues, such as word morphological and orthographic features, such as endings. The HMM model treats every unknown word the same giving it less insight for proper POS tagging of unseen words.

Using the re-ranking by MaxEnt did not surpass the accuracy obtained by the Viterbi tagging of the HMM alone. This reinforces the fact that the Viterbi sequence is not guaranteed to be presented by the heuristic beam search. Furthermore, it is not clear that *had* the Viterbi sequence been presented to the MaxEnt by the beam search that it would have chosen *it* (or any sequence of better quality) over less optimal sequences. This may also have been the result of not training the MaxEnt model over the exact same sentences that the HMM was trained on. The MaxEnt model may have overfit the relatively small training set required due to complexity constraints.

Table 2. POS Tagger Results and accuracies

Experiment	Accuracy on known words	Accuracy on first unknown words
Baseline Most Frequent Class	73.41%	23.70%
HMM only; Viterbi	92.96%	32.76%
HMM with MaxEnt Re-ranking (beam width: k=1)	90.10%	23.80%
HMM with MaxEnt Re-ranking (beam width: k=2)	91.79%	28.59%
HMM with MaxEnt Re-ranking (beam width: k=3)	91.66%	30.72%
HMM with MaxEnt Re-ranking (beam width: k=4)	91.80%	37.23%
HMM with MaxEnt Re-ranking (beam width: k=5)	91.71%	39.17%
HMM with MaxEnt Re-ranking (beam width: k=10)	91.53%	45.78%
HMM with MaxEnt Re-ranking (beam width: k=20)	90.98%	47.91%

V. CONCLUSION

We presented a part of speech tagger which used a combination of a hidden Markov model and a maximum entropy classifier to re-rank tag sequences after applying a beam search to the HMM. The results of experiments showed little benefit to using the method over a pure hidden Markov model (not to mention other proven purely discriminative methods). This approach allowed us to implicitly control the amount of influence on sequence tagging due to each model. Perhaps with more careful design of features and the freedom to train on more data, this method could boost the performance to tagging accuracies greater than that resulting from the Viterbi algorithm run on an HMM in isolation.

VI. REFERENCES

[1] Daniel Jurafsky, James H. Martin, Speech and Language Processing, Upper Saddle River, New Jersy: Prentice-Hall, 2009.

[2] Kristina Toutanova, Christopher D. Manning, "*Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger*", *In Proceedings of HLT-NAACL,* pages 252-259*, 2003*.

[3] Ralph Weischedel, Richard Schwartz, Jeff Palmucci, Marie Meteer, Lance Ramshaw, "*Coping with Ambiguity and Unknown Words through Probabilistic Models*", *Computational Linguistics, Vo; 19, No. 2, 1993.*