CMPT 825: Natural Language Processing **Spring 2008**

Lecture 8 — Jan 23, 2008

Lecturer: Anoop Sarkar *Scribe: Anton Venema*

8.1 A Second-Order Hidden Markov Model for Part-of-Speech Tagging

This paper was written by Scott M. Thede and Mary P. Harper. In it, they describe a new approach for using HMMs in part-of-speech tagging. Using existing standards for contextual information, they add more detailed lexical information and tag prediction for unknown words using suffix data. There are three key components to their model:

1. A standard contextual trigram tagger
   $P(t_i|t_{i-1}, t_{i-2})$
2. A bigram lexical tagger
   $P(w_i|t_i, t_{i-1})$
3. An bigram unknown-word
   $P(s_i|t_i, t_{i-1})$

The notation used for the probability descriptions was somewhat confusing. I believe the descriptions above say the same thing with more standard notation. The lexical and unknown-word taggers were at first going to be trigram taggers, but the results were unsatisfactory. Since examining the current and previous two tags proved to be detrimental, only the current and previous one tag were considered in the final results. The Jelinek-Mercer method of smoothing (Jelinek and Mercer, 1980) was used to help reduce sparseness in the data The three equations below represent the contextual, lexical, and unknown data probability distributions.

$$\begin{aligned}
\hat{P} &= k_3 \cdot \frac{N_3}{C_2} + (1-k_3) \cdot \left(k_2 \cdot \frac{N_2}{C_1} \cdot (1-k_2) \cdot \frac{N_1}{C_0} \right) && (8.1)\\
& k_2 = \frac{log(N_2+1)+1}{log(N_2+1)+2},\ k_3 = \frac{log(N_3+1)+1}{log(N_3+1)+2}\\
& N_1 = count(t_i),\ N_2 = count(t_{i-1}t_i),\ N_3 = count(t_{i-2}t_{i-1}t_i)\\
& C_0 = count(),\ C_1 = count(t_{i-1}),\ C_2 = count(t_{i-2}t_{i-1})
\end{aligned}$$

$$\begin{aligned}
\hat{P} &= \left(\frac{log(N_3+1)+1}{log(N_3+1)+2} \right) \cdot \frac{N_3}{C_2} + \left(\frac{1}{log(N_3+1)+2} \right) \cdot \frac{N_2}{C_1} && (8.2)\\
& N_2 = count(w_i|t_{i-1}),\ N_3 = count(w_i|t_{i-2}t_{i-1})\\
& C_1 = count(t_{i-1}),\ C_2 = count(t_{i-2}t_{i-1})
\end{aligned}$$

$$\begin{aligned}
\hat{P}_i &= f(N_i)\hat{c}_i(s_i) + (1-f(N_k))\hat{P}_i(s_i-1) \mid 1 < k \leq 4 && (8.3)\\
&= \hat{c}_1 \mid k = 1
\end{aligned}$$

The unknown data probability calculations are recursive, with $\hat{P}_i = \hat{c}_1$ providing the base case. (The subscripts have been rewritten to improve readability.) This method gives more weight to longer and more frequently appearing suffixes. An alternative smoothing method was proposed that would use word class information (see Taoukermann and Radev, 1996).

The results indicated an improvement over standard bigram/trigram taggers and over HMMs using only second-order lexical probabilities. The comparisons to other researchers was a bit weak, since the alternatives available for closed lexicon used different training and test data. In spite of this, the results show a high improvement over alternative training methods.

I would be interested to see if these results hold up in languages other than English - specifically a language which does not have a high degree of suffixation. I would also be interested to see if any improvements would result from a model that doesn't discriminate based on suffixation only, but prefixation as well.

8.2 Tagging English Text with a Probabilistic Model

This paper was written by Bernard Merialdo. He describes two methods for measuring the quality of a tagging procedure:

1. At the sentence level, the percentage correctly tagged, evaluated with Viterbi tagging
2. At the word level, the percentage correctly tagged, evaluated with Maximum Likelihood (ML) tagging

It's useful to note that sentence-level performance will always be lower than word-level, since the former is dependent on the latter. Merialdo goes on to describe what he calls a triclass model, commonly known as a trigram model:

$$p(W,T) = \prod_{i=1}^{n} p(w_i|t_i) \cdot p(t_i|t_{i-2}t_{i-1})$$

Two training methods are described. The first method is relative frequency (RF) training, which uses tagged text to count observations and generate frequencies for tag sequences and word/tag pairs (supervised).

$$p(W,T) = \prod_{i=1}^{n} \frac{N(w_i,t_i)}{N(t_i)} \cdot \frac{N(t_{i-2},t_{i-1},t_i)}{N(t_{i-2},t_{i-1})}$$

Deleted interpolation (Jelinek-Mercer, 1980) is then applied to smooth for unseen data. ($|V(t_i)|$ is the number of words with tag t_i.)

$$p(W,T) = \prod_{i=1}^{n} \left(\frac{\lambda \cdot N(w_i,t_i)}{N(t_i)} + \frac{1-\lambda}{|V(t_i)|} \right) \cdot \left(\frac{\lambda \cdot N(t_{i-2},t_{i-1},t_i)}{N(t_{i-2},t_{i-1})} + \frac{1-\lambda}{N_T} \right)$$

The second method is maximum likelihood (ML) training, which does not require tagged text (unsupervised). Merialdo used the Forward-Backward algorithm to maximize the probability of the training text, using the same training data as the relative frequency training, but without looking at the tags. An RF-trained model was used to initialize the probabilities.

The results indicated that RF training was quite accurate (95.4%) with about 2000 training sentences. Using 100 times as much training data yielded only a 1.6% improvement. ML training improved the RF-trained models when very few (between 0 and 5000) training sentences were used for the initial model. It showed its most significant improvements with 0 to 100 sentences used initially. After 3 iterations, it degraded all performance except in the 0-sentence initial-model case.

Merialdo briefly mentioned the idea of constraining ML training to hinder accuracy degradation. The *tw-constraint* keeps the probability of a given word/tag pair constant if it occurs frequently (i.e. in the top 1000 words). The *t-constraint* keeps the probability of a tag constant. Unfortunately, details on the implementation of this method were not described, other than to mention its complexity and adverse effect on the overall running time of the algorithms.

The paper was a good comparison of RF versus ML training. The take-away message is that there is no data like well-tagged data, so use it as much as possible. If using ML training, test it on a data set after each iteration to ensure you are not making your model worse.

8.3 Does Baum-Welch Re-estimation Help Taggers?

This paper was written by David Elworthy. It yields results that are strikingly consistent with those demonstrated by Bernard Merialdo (1994), who concluded that the best way to get a high-performance model was to use as much tagged data as possible, and only use Baum-Welch re-estimation (using the Forward-Backward algorithm) if the amount of tagged data available is minimal. He additionally concluded that even when Baum-Welch re-estimation improves model accuracy, the number of iterations used should be kept minimal, since multiple iterations tend to degrade the performance in the long run.

Elworthy had the goal in mind not of refuting Merialdo's results, but providing more detailed information as to why he could draw the conclusions he did. His end results were almost identical. When using HMMs, he recommends using as much tagged text as possible to train the model. If the test data is similar to the training data, then BW re-estimation should be used minimally, if at all. For less robust starting conditions (i.e. limited corpora), use as much pre-conditioned information as possible, and then use BW re-estimation minimally. If no training data is available, use BW re-estimation, but only up to a point where it ceases to be beneficial (i.e. iteration x_i shows no improvement over iteration x_{i-1}). It is important to note that the effectiveness of BW re-estimation even at this point is highly dependent on good initial lexical (emission) and transition probabilities.

While the information presented by Elworthy is more detailed than that provided by Merialdo, one has to question its necessity. Very little new information was presented, and no new conclusions were drawn. In fact, Elworthy concludes his paper by restating Merialdo's conclusions. The new information that Elworthy gives is the classification of models into one of three categories - initial, early, and classical - based on their responsiveness to BW re-estimation. Even so, this seems to merely be giving a name to what Merialdo previously reported.

- Classical - rising accuracy on each iteration
- Initial maximum - highest accuracy at outset, degradation at each iteration
- Early maximum - rising accuracy on first few iterations, then degradation

Classical patterns can be observed with no initial training data. Models exhibiting early maximum patterns can be observed when their initial probability distributions were obtained through a supervised learning process on a relatively small ($< 100,000$ sentences) amount of training data. Initial maximum patterns can be seen for models attained in much the same way, but with a larger amount of training data.