Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network

Kristina Toutanova
Computer Science Dept.
Stanford University
Stanford, CA 94305-9040
kristina@cs.stanford.edu

Dan Klein
Computer Science Dept.
Stanford University
Stanford, CA 94305-9040
klein@cs.stanford.edu

Christopher D. Manning
Computer Science Dept.
Stanford University
Stanford, CA 94305-9040
manning@stanford.edu

Yoram Singer
School of Computer Science
The Hebrew University
Jerusalem 91904, Israel
singer@cs.huji.ac.il

Abstract

We present a new part-of-speech tagger that demonstrates the following ideas: (i) explicit use of both preceding and following tag contexts via a dependency network representation, (ii) broad use of lexical features, including jointly conditioning on multiple consecutive words, (iii) effective use of priors in conditional loglinear models, and (iv) fine-grained modeling of unknown word features. Using these ideas together, the resulting tagger gives a 97.24% accuracy on the Penn Treebank WSJ, an error reduction of 4.4% on the best previous single automatically learned tagging result.

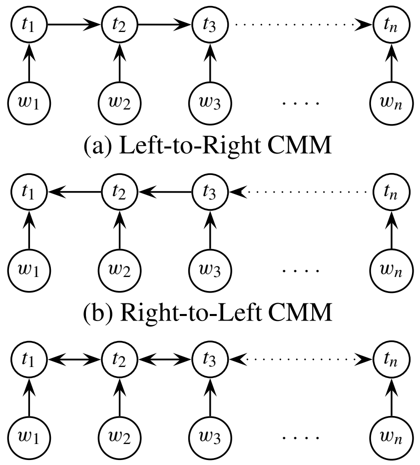
1 Introduction

Almost all approaches to sequence problems such as part-of-speech tagging take a unidirectional approach to conditioning inference along the sequence. Regardless of whether one is using HMMs, maximum entropy conditional sequence models, or other techniques like decision trees, most systems work in one direction through the sequence (normally left to right, but occasionally right to left, e.g., Church (1988)). There are a few exceptions, such as Brill's transformation-based learning (Brill, 1995), but most of the best known and most successful approaches of recent years have been unidirectional.

Most sequence models can be seen as chaining together the scores or decisions from successive local models to form a global model for an entire sequence. Clearly the identity of a tag is correlated with both past and future tags' identities. However, in the unidirectional (causal) case, only one direction of influence is explicitly considered at each local point. For example, in a left-to-right first-order HMM, the current tag t_0 is predicted based on the previous tag t_{-1} (and the current word).[1] The backward interaction between t_0 and the next tag t_{+1} shows up implicitly later, when t_{+1} is generated in turn. While unidirectional models are therefore able to capture both directions of influence, there are good reasons for suspecting that it would be advantageous to make information from both directions explicitly available for conditioning at each local point in the model: (i) because of smoothing and interactions with other modeled features, terms like $P(t_0|t_{+1}, \ldots)$ might give a sharp estimate of t_0 even when terms like $P(t_{+1}|t_0, \ldots)$ do not, and (ii) jointly considering the left and right context together might be especially revealing. In this paper we exploit this idea, using dependency networks, with a series of local conditional loglinear (aka maximum entropy or multiclass logistic regression) models as one way of providing efficient bidirectional inference.

Secondly, while all taggers use lexical information, and, indeed, it is well-known that lexical probabilities are much more revealing than tag sequence probabilities (Charniak et al., 1993), most taggers make quite limited use of lexical probabilities (compared with, for example, the bilexical probabilities commonly used in current statistical parsers). While modern taggers may be more principled than the classic CLAWS tagger (Marshall, 1987), they are in some respects inferior in their use of lexical information: CLAWS, through its IDIOMTAG module, categorically captured many important, correct taggings of frequent idiomatic word sequences. In this work, we incorporate appropriate multiword feature templates so that such facts can be learned and used automatically by

[1] Rather than subscripting all variables with a position index, we use a hopefully clearer relative notation, where t_0 denotes the current position and t_{-n} and t_{+n} are left and right context tags, and similarly for words.



(a) Left-to-Right CMM

(b) Right-to-Left CMM

(c) Bidirectional Dependency Network

Figure 1: Dependency networks: (a) the (standard) left-to-right first-order CMM, (b) the (reversed) right-to-left CMM, and (c) the bidirectional dependency network.

the model.

Having expressive templates leads to a large number of features, but we show that by suitable use of a prior (i.e., *regularization*) in the conditional loglinear model – something not used by previous maximum entropy taggers – many such features can be added with an overall positive effect on the model. Indeed, as for the voted perceptron of Collins (2002), we can get performance gains by reducing the support threshold for features to be included in the model. Combining all these ideas, together with a few additional handcrafted unknown word features, gives us a part-of-speech tagger with a per-position tag accuracy of 97.24%, and a whole-sentence correct rate of 56.34% on Penn Treebank WSJ data. This is the best automatically learned part-of-speech tagging result known to us, representing an error reduction of 4.4% on the model presented in Collins (2002), using the same data splits, and a larger error reduction of 12.1% from the more similar best previous loglinear model in Toutanova and Manning (2000).

2 Bidirectional Dependency Networks

When building probabilistic models for tag sequences, we often decompose the global probability of sequences using a directed graphical model (e.g., an HMM (Brants, 2000) or a conditional Markov model (CMM) (Ratnaparkhi, 1996)). In such models, the probability assigned to a tagged sequence of words $x = \langle t, w \rangle$ is the product of a sequence of local portions of the graphical model, one from each time slice. For example, in the left-to-right CMM shown in figure 1(a),

$$P(t, w) = \prod_i P(t_i | t_{i-1}, w_i)$$

That is, the replicated structure is a local model $P(t_0 | t_{-1}, w_0)$.[2] Of course, if there are too many conditioned quantities, these local models may have to be estimated in some sophisticated way; it is typical in tagging to populate these models with little maximum entropy models. For example, we might populate a model for $P(t_0 | t_{-1}, w_0)$ with a maxent model of the form:

$$P_\lambda(t_0 | t_{-1}, w_0) = \frac{\exp(\lambda_{\langle t_0, t_{-1} \rangle} + \lambda_{\langle t_0, w_0 \rangle})}{\sum_{t'_0} \exp(\lambda_{\langle t'_0, t_{-1} \rangle} + \lambda_{\langle t'_0, w_0 \rangle})}$$

In this case, the w_0 and t_{-1} can have joint *effects* on t_0, but there are not joint *features* involving all three variables (though there could have been such features). We say that this model uses the *feature templates* $\langle t_0, t_{-1} \rangle$ (previous tag features) and $\langle t_0, w_0 \rangle$ (current word features).

Clearly, *both* the preceding tag t_{-1} and following tag t_{+1} carry useful information about a current tag t_0. Unidirectional models do not ignore this influence; in the case of a left-to-right CMM, the influence of t_{-1} on t_0 is explicit in the $P(t_0 | t_{-1}, w_0)$ local model, while the influence of t_{+1} on t_0 is implicit in the local model at the next position (via $P(t_{+1} | t_0, w_{+1})$). The situation is reversed for the right-to-left CMM in figure 1(b).

From a seat-of-the-pants machine learning perspective, when building a classifier to label the tag at a certain position, the obvious thing to do is to explicitly include in the local model all predictive features, no matter on which side of the target position they lie. There are two good formal reasons to expect that a model explicitly conditioning on both sides at *each* position, like figure 1(c) could be advantageous. First, because of smoothing effects and interaction with other conditioning features (like the words), left-to-right factors like $P(t_0 | t_{-1}, w_0)$ do not always suffice when t_0 is implicitly needed to determine t_{-1}. For example, consider a case of observation bias (Klein and Manning, 2002) for a first-order left-to-right CMM. The word *to* has only one tag (TO) in the PTB tag set. The TO tag is often preceded by nouns, but rarely by modals (MD). In a sequence *will to fight*, that trend indicates that *will* should be a noun rather than a modal verb. However, that effect is completely lost in a CMM like (a): $P(t_{will} | will, \langle start \rangle)$ prefers the modal tagging, and $P(\text{TO} | to, t_{will})$ is roughly 1 regardless of t_{will}. While the model has an arrow between the two tag positions, that path of influence is severed.[3] The same problem exists in the other direction. If we use the symmetric right-

[2]Throughout this paper we assume that enough boundary symbols always exist that we can ignore the differences which would otherwise exist at the initial and final few positions.

[3]Despite use of names like "label bias" (Lafferty et al., 2001) or "observation bias", these effects are really just unwanted explaining-away effects (Cowell et al., 1999, 19), where two nodes which are not actually in causal competition have been modeled as if they were.

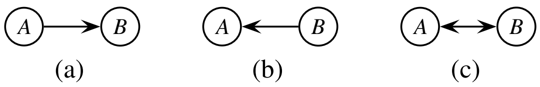


Figure 2: Simple dependency nets: (a) the Bayes' net for $P(A)P(B|A)$, (b) the Bayes' net for $P(A|B)P(B)$, (c) a bidirectional net with models of $P(A|B)$ and $P(B|A)$, which is not a Bayes' net.

to-left model, *fight* will receive its more common noun tagging by symmetric reasoning. However, the bidirectional model (c) discussed in the next section makes both directions available for conditioning at all locations, using replicated models of $P(t_0|t_{-1}, t_{+1}, w_0)$, and will be able to get this example correct.[4]

2.1 Semantics of Dependency Networks

While the structures in figure 1(a) and (b) are well-understood graphical models with well-known semantics, figure 1(c) is not a standard Bayes' net, precisely because the graph has cycles. Rather, it is a more general *dependency network* (Heckerman et al., 2000). Each node represents a random variable along with a local conditional probability model of that variable, conditioned on the source variables of all incoming arcs. In this sense, the semantics are the same as for standard Bayes' nets. However, because the graph is cyclic, the net does not correspond to a proper factorization of a large joint probability estimate into local conditional factors. Consider the two-node cases shown in figure 2. Formally, for the net in (a), we can write $P(a, b) = P(a)P(b|a)$. For (b) we write $P(a, b) = P(b)P(a|b)$. However, in (c), the nodes A and B carry the information $P(a|b)$ and $P(b|a)$ respectively. The chain rule doesn't allow us to reconstruct $P(a, b)$ by multiplying these two quantities. Under appropriate conditions, we could *reconstruct* $P(a, b)$ from these quantities using Gibbs sampling, and, in general, that is the best we can do. However, while reconstructing the joint probabilities from these local conditional probabilities may be difficult, estimating the local probabilities themselves is no harder than it is for acyclic models: we take observations of the local environments and use any maximum likelihood estimation method we desire. In our experiments, we used local maxent models, but if the event space allowed, (smoothed) relative counts would do.

[4]The effect of indirect influence being weaker than direct influence is more pronounced for conditionally structured models, but is potentially an issue even with a simple HMM. The probabilistic models for basic left-to-right and right-to-left HMMs with emissions on their states can be shown to be equivalent using Bayes' rule on the transitions, provided start and end symbols are modeled. However, this equivalence is violated in practice by the addition of smoothing.

```
function bestScore()
   return bestScoreSub(n + 2, ⟨end, end, end⟩);

function bestScoreSub(i + 1, ⟨t_{i−1}, t_i, t_{i+1}⟩)
   % memoization
   if (cached(i + 1, ⟨t_{i−1}, t_i, t_{i+1}⟩))
      return cache(i + 1, ⟨t_{i−1}, t_i, t_{i+1}⟩);
   % left boundary case
   if (i = −1)
      if (⟨t_{i−1}, t_i, t_{i+1}⟩ == ⟨start, start, start⟩)
         return 1;
      else
         return 0;
   % recursive case
   return max_{t_{i−2}} bestScoreSub(i, ⟨t_{i−2}, t_{i−1}, t_i⟩)×
                     P(t_i|t_{i−1}, t_{i+1}, w_i);
```

Figure 3: Pseudocode for polynomial inference for the first-order bidirectional CMM (memoized version).

2.2 Inference for Linear Dependency Networks

Cyclic or not, we can view the product of local probabilities from a dependency network as a score:

$$score(x) = \prod_i P(x_i|Pa(x_i))$$

where $Pa(x_i)$ are the nodes with arcs to the node x_i. In the case of an acyclic model, this score will be the joint probability of the event x, $P(x)$. In the general case, it will not be. However, we can still ask for the event, in this case the tag sequence, with the highest score. For dependency networks like those in figure 1, an adaptation of the Viterbi algorithm can be used to find the maximizing sequence in polynomial time. Figure 3 gives pseudocode for the concrete case of the network in figure 1(d); the general case is similar, and is in fact just a max-plus version of standard inference algorithms for Bayes' nets (Cowell et al., 1999, 97). In essence, there is no difference between inference on this network and a second-order left-to-right CMM or HMM. The only difference is that, when the Markov window is at a position i, rather than receiving the score for $P(t_i|t_{i-1}, t_{i-2}, w_i)$, one receives the score for $P(t_{i-1}|t_i, t_{i-2}, w_{i-1})$.

There are some foundational issues worth mentioning. As discussed previously, the maximum scoring sequence need not be the sequence with maximum likelihood according to the model. There is therefore a worry with these models about a kind of "collusion" where the model locks onto conditionally consistent but jointly unlikely sequences. Consider the two-node network in figure 2(c). If we have the following distribution of observations (in the form ab) $\langle 11, 11, 11, 12, 21, 33\rangle$, then clearly the most likely state of the network is 11. However, the score of 11 is $P(a = 1|b = 1)P(b = 1|a = 1) = 3/4 \times 3/4 = 9/16$, while the score of 33 is 1. An additional related problem is that the training set loss (sum of negative logarithms of the sequence scores) does not bound the training set error (0/1 loss on sequences) from

Data Set	Sect'ns	Sent.	Tokens	Unkn
Training	0–18	38,219	912,344	0
Develop	19–21	5,527	131,768	4,467
Test	22–24	5,462	129,654	3,649

Table 1: Data set splits used.

above. Consider the following training set, for the same network, with each entire data point considered as a label: $\langle 11, 22 \rangle$. The relative-frequency model assigns loss 0 to both training examples, but cannot do better than 50% error in regenerating the training data labels. These issues are further discussed in Heckerman et al. (2000). Preliminary work of ours suggests that practical use of dependency networks is not in general immune to these theoretical concerns: a dependency network can choose a sequence model that is bidirectionally very consistent but does not match the data very well. However, this problem does not appear to have prevented the networks from performing well on the tagging problem, probably because features linking tags and observations are generally much sharper discriminators than tag sequence features.

It is useful to contrast this framework with the conditional random fields of Lafferty et al. (2001). The CRF approach uses similar local features, but rather than chaining together local models, they construct a single, globally normalized model. The principal advantage of the dependency network approach is that advantageous bidirectional effects can be obtained without the extremely expensive global training required for CRFs.

To summarize, we draw a dependency network in which each node has as neighbors all the other nodes that we would like to have influence it directly. Each node's neighborhood is then considered in isolation and a local model is trained to maximize the conditional likelihood over the training data of that node. At test time, the sequence with the highest product of local conditional scores is calculated and returned. We can always find the exact maximizing sequence, but only in the case of an acyclic net is it guaranteed to be the maximum likelihood sequence.

3 Experiments

The part of speech tagged data used in our experiments is the Wall Street Journal data from Penn Treebank III (Marcus et al., 1994). We extracted tagged sentences from the parse trees.[5] We split the data into training, development, and test sets as in (Collins, 2002). Table 1 lists characteristics of the three splits.[6] Except where indicated for the model BEST, all results are on the development set.

One innovation in our reporting of results is that we present whole-sentence accuracy numbers as well as the traditional per-tag accuracy measure (over all tokens, even unambiguous ones). This is the quantity that most sequence models attempt to maximize (and has been motivated over doing per-state optimization as being more useful for subsequent linguistic processing: one wants to find a coherent sentence interpretation). Further, while some tag errors matter much more than others, to a first cut getting a single tag wrong in many of the more common ways (e.g., proper noun vs. common noun; noun vs. verb) would lead to errors in a subsequent processor such as an information extraction system or a parser that would greatly degrade results for the entire sentence. Finally, the fact that the measure has much more dynamic range has some appeal when reporting tagging results.

The per-state models in this paper are log-linear models, building upon the models in (Ratnaparkhi, 1996) and (Toutanova and Manning, 2000), though some models are in fact strictly simpler. The features in the models are defined using templates; there are different templates for rare words aimed at learning the correct tags for unknown words.[7] We present the results of three classes of experiments: experiments with directionality, experiments with lexicalization, and experiments with smoothing.

3.1 Experiments with Directionality

In this section, we report experiments using log-linear CMMs to populate nets with various structures, exploring the relative value of neighboring words' tags. Table 2 lists the discussed networks. All networks have the same vertical feature templates: $\langle t_0, w_0 \rangle$ features for known words and various $\langle t_0, \sigma(w_{1n}) \rangle$ word signature features for all words, known or not, including spelling and capitalization features (see section 3.3).

Just this vertical conditioning gives an accuracy of 93.69% (denoted as "Baseline" in table 2).[8] Condition-

[5] Note that these tags (and sentences) are *not* identical to those obtained from the tagged/pos directories of the same disk: hundreds of tags in the RB/RP/IN set were changed to be more consistent in the parsed/mrg version. Maybe we were the last to discover this, but we've never seen it in print.

[6] Tagger results are only comparable when tested not only on the same data and tag set, but with the same amount of training data. Brants (2000) illustrates very clearly how tagging performance increases as training set size grows, largely because the percentage of unknown words decreases while system performance on them increases (they become increasingly restricted as to word class).

[7] Except where otherwise stated, a count cutoff of 2 was used for common word features and 35 for rare word features (templates need a support set strictly greater in size than the cutoff before they are included in the model).

[8] Charniak et al. (1993) noted that such a simple model got 90.25%, but this was with no unknown word model beyond a prior distribution over tags. Abney et al. (1999) raise this baseline to 92.34%, and with our sophisticated unknown word model, it gets even higher. The large number of unambiguous tokens and ones with very skewed distributions make the base-

Model	Feature Templates†	Features	Sentence Accuracy	Token Accuracy	Unkn. Word Accuracy
Baseline	$\emptyset$	56,805	26.74%	93.69%	82.61%
L	$\langle t_0, t_{-1}\rangle$	27,474	41.89%	95.79%	85.49%
R	$\langle t_0, t_{+1}\rangle$	27,648	36.31%	95.14%	85.65%
L+L$_2$	$\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{-2}\rangle$	32,935	44.04%	96.05%	85.92%
R+R$_2$	$\langle t_0, t_{+1}\rangle$, $\langle t_0, t_{+2}\rangle$	33,423	37.20%	95.25%	84.49%
L+R	$\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{+1}\rangle$	32,610	49.50%	96.57%	87.15%
LL	$\langle t_0, t_{-1}, t_{-2}\rangle$	45,532	44.60%	96.10%	86.48%
RR	$\langle t_0, t_{+1}, t_{+2}\rangle$	45,446	38.41%	95.40%	85.58%
LR	$\langle t_0, t_{-1}, t_{+1}\rangle$	45,478	49.30%	96.55%	87.26%
L+LL+LLL	$\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{-1}, t_{-2}\rangle$, $\langle t_0, t_{-1}, t_{-2}, t_{-3}\rangle$	118,752	45.14%	96.20%	86.52%
R+LR+LLR	$\langle t_0, t_{+1}\rangle$, $\langle t_0, t_{-1}, t_{+1}\rangle$, $\langle t_0, t_{-1}, t_{-2}, t_{+1}\rangle$	115,790	51.69%	96.77%	87.91%
L+LL+LR+RR+R	$\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{-1}, t_{-2}\rangle$, $\langle t_0, t_{-1}, t_{+1}\rangle$, $\langle t_0, t_{+1}\rangle$, $\langle t_0, t_{+1}, t_{+2}\rangle$	81,049	53.23%	96.92%	87.91%

Table 2: Tagging accuracy on the development set with different sequence feature templates. †All models include the same vertical word-tag features ($\langle t_0, w_0\rangle$ and various $\langle t_0, \sigma(w_{1n})\rangle$), though the baseline uses a lower cutoff for these features.

Model	Feature Templates	Support Cutoff	Features	Sentence Accuracy	Token Accuracy	Unknown Accuracy
BASELINE	$\langle t_0, w_0\rangle$	2	6,501	1.63%	60.16%	82.98%
	$\langle t_0, w_0\rangle$	0	56,805	26.74%	93.69%	82.61%
3W	$\langle t_0, w_0\rangle$, $\langle t_0, w_{-1}\rangle$, $\langle t_0, w_{+1}\rangle$	2	239,767	48.27%	96.57%	86.78%
3W+TAGS	tag sequences, $\langle t_0, w_0\rangle$, $\langle t_0, w_{-1}\rangle$, $\langle t_0, w_{+1}\rangle$	2	263,160	53.83%	97.02%	88.05%
BEST	see text	2	460,552	55.31%	97.15%	88.61%

Table 3: Tagging accuracy with different lexical feature templates on the development set.

Model	Feature Templates	Support Cutoff	Features	Sentence Accuracy	Token Accuracy	Unknown Accuracy
BEST	see text	2	460,552	56.34%	97.24%	89.04%

Table 4: Final tagging accuracy for the best model on the test set.

ing on the previous tag as well (model L, $\langle t_0, t_{-1}\rangle$ features) gives 95.79%. The reverse, model R, using the next tag instead, is slightly inferior at 95.14%. Model L+R, using both tags simultaneously (but with only the individual-direction features) gives a much better accuracy of 96.57%. Since this model has roughly twice as many tag-tag features, the fact that it outperforms the unidirectional models is not by itself compelling evidence for using bidirectional networks. However, it also outperforms model L+L$_2$ which adds the $\langle t_0, t_{-2}\rangle$ second-previous word features instead of next word features, which gives only 96.05% (and R+R$_2$ gives 95.25%). We conclude that, if one wishes to condition on two neighboring nodes (using two sets of 2-tag features), the symmetric bidirectional model is superior.

High-performance taggers typically also include joint three-tag counts in some way, either as tag trigrams (Brants, 2000) or tag-triple features (Ratnaparkhi, 1996, Toutanova and Manning, 2000). Models LL, RR, and CR use only the vertical features and a single set of tag-triple features: the left tags (t_{-2}, t_{-1} and t_0), right tags (t_0, t_{+1}, t_{+2}), or centered tags (t_{-1}, t_0, t_{+1}) respectively. Again, with roughly equivalent feature sets, the left context is better than the right, and the centered context is better than either unidirectional context.

line for this task high, while substantial annotator noise creates an unknown upper bound on the task.

3.2 Lexicalization

Lexicalization has been a key factor in the advance of statistical parsing models, but has been less exploited for tagging. Words surrounding the current word have been occasionally used in taggers, such as (Ratnaparkhi, 1996), Brill's transformation based tagger (Brill, 1995), and the HMM model of Lee et al. (2000), but nevertheless, the only lexicalization consistently included in tagging models is the dependence of the part of speech tag of a word on the word itself.

In maximum entropy models, joint features which look at surrounding words and their tags, as well as joint features of the current word and surrounding words are in principle straightforward additions, but have not been incorporated into previous models. We have found these features to be very useful. We explore here lexicalization both alone and in combination with preceding and following tag histories.

Table 3 shows the development set accuracy of several models with various lexical features. All models use the same rare word features as the models in Table 2. The first two rows show a baseline model using the current word only. The count cutoff for this feature was 0 in the first model and 2 for the model in the second row. As there are no tag sequence features in these models, the accuracy drops significantly if a higher cutoff is used (from a per tag accuracy of about 93.7% to only 60.2%).

The third row shows a model where a tag is decided solely by the three words centered at the tag position (3W). As far as we are aware, models of this sort have not been explored previously, but its accuracy is surprisingly high: despite having no sequence model at all, it is more accurate than a model which uses standard tag fourgram HMM features ($\langle t_0, w_0\rangle$, $\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{-1}, t_{-2}\rangle$, $\langle t_0, t_{-1}, t_{-2}, t_{-3}\rangle$, shown in Table 2, model L+LL+LLL).

The fourth and fifth rows show models with bidirectional tagging features. The fourth model (3W+TAGS) uses the same tag sequence features as the last model in Table 2 ($\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{-1}, t_{-2}\rangle$, $\langle t_0, t_{-1}, t_{+1}\rangle$, $\langle t_0, t_{+1}\rangle$, $\langle t_0, t_{+1}, t_{+2}\rangle$) and current, previous, and next word. The last model has in addition the feature templates $\langle t_0, w_0, t_{-1}\rangle$, $\langle t_0, w_0, t_{+1}\rangle$, $\langle t_0, w_{-1}, w_0\rangle$, and $\langle t_0, w_0, w_{+1}\rangle$, and includes the improvements in unknown word modeling discussed in section 3.3.[9] We call this model BEST. BEST has a token accuracy on the final test set of 97.24% and a sentence accuracy of 56.34% (see Table 4). A 95% confidence interval for the accuracy (using a binomial model) is (97.15%, 97.33%).

In order to understand the gains from using right context tags and more lexicalization, let us look at an example of an error that the enriched models learn not to make. An interesting example of a common tagging error of the simpler models which could be corrected by a deterministic fixup rule of the kind used in the IDIOMTAG module of (Marshall, 1987) is the expression *as X as* (often, *as far as*). This should be tagged *as/RB X/{RB,JJ} as/IN* in the Penn Treebank. A model using only current word and two left tags (model L+L_2 in Table 2), made 87 errors on this expression, tagging it *as/IN X as/IN* – since the tag sequence probabilities do not give strong reasons to disprefer the most common tagging of *as* (it is tagged as IN over 80% of the time). However, the model 3W+TAGS, which uses two right tags and the two surrounding words in addition, made only 8 errors of this kind, and model BEST made only 6 errors.

3.3 Unknown word features

Most of the models presented here use a set of unknown word features basically inherited from (Ratnaparkhi, 1996), which include using character n-gram prefixes and suffixes (for n up to 4), and detectors for a few other prominent features of words, such as capitalization, hyphens, and numbers. Doing error analysis on unknown words on a simple tagging model (with $\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{-1}, t_{-2}\rangle$, and $\langle w_0, t_0\rangle$ features) suggested several additional specialized features that can usefully improve performance. By far the most significant is a crude company name detector which marks capitalized words followed within 3 words by a company name suffix like *Co.* or *Inc.* This suggests that further gains could be made by incorporating a good named entity recognizer as a preprocessor to the tagger (reversing the most common order of processing in pipelined systems!), and is a good example of something that can only be done when using a conditional model. Minor gains come from a few additional features: an allcaps feature, and a conjunction feature of words that are capitalized and have a digit and a dash in them (such words are normally common nouns, such as *CFC-12* or *F/A-18*). We also found it advantageous to use prefixes and suffixes of length up to 10. Together with the larger templates, these features contribute to our unknown word accuracies being higher than those of previously reported taggers.

[9]Thede and Harper (1999) use $\langle t_{-1}, t_0, w_0\rangle$ templates in their "full-second order" HMM, achieving an accuracy of 96.86%. Here we can add the opposite tiling and other features.

Smoothed	Features	Sentence Accuracy	Token Acc.	Unk. W. Acc.
yes	45,532	44.60%	96.10%	86.48%
no	45,532	42.81%	95.88%	83.08%
yes	292,649	54.88%	97.10%	88.20%
no	292,649	48.85%	96.54%	85.20%

Table 5: Accuracy with and without quadratic regularization.

3.4 Smoothing

With so many features in the model, overtraining is a distinct possibility when using pure maximum likelihood estimation. We avoid this by using a Gaussian prior (aka quadratic regularization or quadratic penalization) which resists high feature weights unless they produce great score gain. The regularized objective F is:

$$F(\boldsymbol{\lambda}) = \sum\nolimits_i \log(P_{\boldsymbol{\lambda}}(t_i|\mathbf{w}, \mathbf{t})) + \sum\nolimits_{j=1}^{n} \frac{\lambda_j^2}{2\sigma^2}$$

Since we use a conjugate-gradient procedure to maximize the data likelihood, the addition of a penalty term is easily incorporated. Both the total size of the penalty and the partial derivatives with repsect to each λ_j are trivial to compute; these are added to the log-likelihood and log-likelihood derivatives, and the penalized optimization procedes without further modification.

We have not extensively experimented with the value of σ^2 – which can even be set differently for different parameters or parameter classes. All the results in this paper use a constant $\sigma^2 = 0.5$, so that the denominator disappears in the above expression. Experiments on a simple model with σ made an order of magnitude higher or lower both resulted in worse performance than with $\sigma^2 = 0.5$.

Our experiments show that quadratic regularization is very effective in improving the generalization performance of tagging models, mostly by increasing the number of features which could usefully be incorporated. The

Tagger	Support cutoff	Accuracy
Collins (2002)	0	96.60%
	5	96.72%
Model 3W+TAGS variant	1	96.97%
	5	96.93%

Table 6: Effect of changing common word feature cutoffs (features with support $\leq$ cutoff are excluded from the model).

number of features used in our complex models – in the several hundreds of thousands, is extremely high in comparison with the data set size and the number of features used in other machine learning domains. We describe two sets of experiments aimed at comparing models with and without regularization. One is for a simple model with a relatively small number of features, and the other is for a model with a large number of features.

The usefulness of priors in maximum entropy models is not new to this work: Gaussian prior smoothing is advocated in Chen and Rosenfeld (2000), and used in all the stochastic LFG work (Johnson et al., 1999). However, until recently, its role and importance have not been widely understood. For example, Zhang and Oles (2001) attribute the perceived limited success of logistic regression for text categorization to a lack of use of regularization. At any rate, regularized conditional loglinear models have not previously been applied to the problem of producing a high quality part-of-speech tagger: Ratnaparkhi (1996), Toutanova and Manning (2000), and Collins (2002) all present unregularized models. Indeed, the result of Collins (2002) that including low support features helps a voted perceptron model but harms a maximum entropy model is undone once the weights of the maximum entropy model are regularized.

Table 5 shows results on the development set from two pairs of experiments. The first pair of models use common word templates $\langle t_0, w_0 \rangle$, $\langle t_0, t_{-1}, t_{-2} \rangle$ and the same rare word templates as used in the models in table 2. The second pair of models use the same features as model BEST with a higher frequency cutoff of 5 for common word features.

For the first pair of models, the error reduction from smoothing is 5.3% overall and 20.1% on unknown words. For the second pair of models, the error reduction is even bigger: 16.2% overall after convergence and 5.8% if looking at the best accuracy achieved by the unsmoothed model (by stopping training after 75 iterations; see below). The especially large reduction in unknown word error reflects the fact that, because penalties are effectively stronger for rare features than frequent ones, the presence of penalties increases the degree to which more general cross-word signature features (which apply to unknown words) are used, relative to word-specific sparse features (which do not apply to unknown words).

Secondly, use of regularization allows us to incorporate features with low support into the model while improving

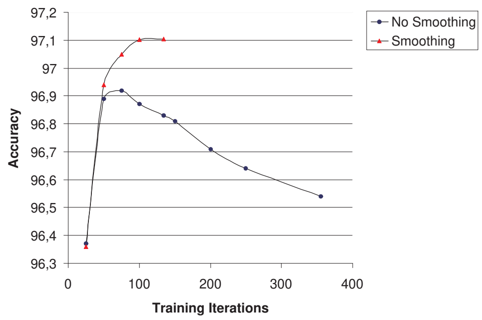


Figure 4: Accuracy by training iterations, with and without quadratic regularization.

performance. Whereas Ratnaparkhi (1996) used feature support cutoffs and early stopping to stop overfitting of the model, and Collins (2002) contends that including low support features harms a maximum entropy model, our results show that low support features are useful in a regularized maximum entropy model. Table 6 contrasts our results with those from Collins (2002). Since the models are not the same, the exact numbers are incomparable, but the difference in direction is important: in the regularized model, performance improves with the inclusion of low support features.

Finally, in addition to being significantly more accurate, smoothed models train much faster than unsmoothed ones, and do not benefit from early stopping. For example, the first smoothed model in Table 5 required 80 conjugate gradient iterations to converge (somewhat arbitrarily defined as a maximum difference of 10^{-4} in feature weights between iterations), while its corresponding unsmoothed model required 335 iterations, thus training was roughly 4 times slower.[10] The second pair of models required 134 and 370 iterations respectively. As might be expected, unsmoothed models reach their highest generalization capacity long before convergence and accuracy on an unseen test set drops considerably with further iterations. This is not the case for smoothed models, as their test set accuracy increases almost monotonically with training iterations.[11] Figure 4 shows a graph of training iterations versus accuracy for the second pair of models on the development set.

4 Conclusion

We have shown how broad feature use, when combined with appropriate model regularization, produces a superior level of tagger performance. While experience sug-

[10] On a 2GHz PC, this is still an important difference: our largest models require about 25 minutes *per iteration* to train.

[11] In practice one notices some wiggling in the curve, but the trend remains upward even beyond our chosen convergence point.

gests that the final accuracy number presented here could be slightly improved upon by classifier combination, it is worth noting that not only is this tagger better than any previous single tagger, but it also appears to outperform Brill and Wu (1998), the best-known combination tagger (they report an accuracy of 97.16% over the same WSJ data, but using a larger training set, which should favor them).

While part-of-speech tagging is now a fairly well-worn road, and our ability to win performance increases in this domain is starting to be limited by the rate of errors and inconsistencies in the Penn Treebank training data, this work also has broader implications. Across the many NLP problems which involve sequence models over sparse multinomial distributions, it suggests that feature-rich models with extensive lexicalization, bidirectional inference, and effective regularization will be key elements in producing state-of-the-art results.

Acknowledgements

This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program, by the National Science Foundation under Grant No. IIS-0085896, and by an IBM Faculty Partnership Award.

References

Steven Abney, Robert E. Schapire, and Yoram Singer. 1999. Boosting applied to tagging and PP attachment. In *EMNLP/VLC 1999*, pages 38–45.

Thorsten Brants. 2000. TnT – a statistical part-of-speech tagger. In *ANLP 6*, pages 224–231.

Eric Brill and Jun Wu. 1998. Classifier combination for improved lexical disambiguation. In *ACL 36/COLING 17*, pages 191–195.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Eugene Charniak, Curtis Hendrickson, Neil Jacobson, and Mike Perkowitz. 1993. Equations for part-of-speech tagging. In *AAAI 11*, pages 784–789.

Stanley F. Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for maximum entropy models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

Kenneth W. Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *ANLP 2*, pages 136–143.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*.

Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. 1999. *Probabilistic Networks and Expert Systems*. Springer-Verlag, New York.

David Heckerman, David Maxwell Chickering, Christopher Meek, Robert Rounthwaite, and Carl Myers Kadie. 2000. Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, 1(1):49–75.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *ACL 37*, pages 535–541.

Dan Klein and Christopher D. Manning. 2002. Conditional structure versus conditional estimation in NLP models. In *EMNLP 2002*, pages 9–16.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML-2001*, pages 282–289.

Sang-Zoo Lee, Jun ichi Tsujii, and Hae-Chang Rim. 2000. Part-of-speech tagging based on Hidden Markov Model assuming joint independence. In *ACL 38*, pages 263–169.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Ian Marshall. 1987. Tag selection using probabilistic methods. In Roger Garside, Geoffrey Sampson, and Geoffrey Leech, editors, *The Computational analysis of English: a corpus-based approach*, pages 42–65. Longman, London.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP 1*, pages 133–142.

Scott M. Thede and Mary P. Harper. 1999. Second-order hidden Markov model for part-of-speech tagging. In *ACL 37*, pages 175–182.

Kristina Toutanova and Christopher Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP/VLC 1999*, pages 63–71.

Tong Zhang and Frank J. Oles. 2001. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31.