# Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger

Kristina Toutanova
Dept of Computer Science
Gates Bldg 4A, 353 Serra Mall
Stanford, CA 94305–9040, USA
kristina@cs.stanford.edu

Christopher D. Manning
Depts of Computer Science and Linguistics
Gates Bldg 4A, 353 Serra Mall
Stanford, CA 94305–9040, USA
manning@cs.stanford.edu

**Abstract**

This paper presents results for a maximum-entropy-based part of speech tagger, which achieves superior performance principally by enriching the information sources used for tagging. In particular, we get improved results by incorporating these features: (i) more extensive treatment of capitalization for unknown words; (ii) features for the disambiguation of the tense forms of verbs; (iii) features for disambiguating particles from prepositions and adverbs. The best resulting accuracy for the tagger on the Penn Treebank is 96.86% overall, and 86.91% on previously unseen words.

## Introduction[1]

There are now numerous systems for automatic assignment of parts of speech ("tagging"), employing many different machine learning methods. Among recent top performing methods are Hidden Markov Models (Brants 2000), maximum entropy approaches (Ratnaparkhi 1996), and transformation-based learning (Brill 1994). An overview of these and other approaches can be found in Manning and Schütze (1999, ch. 10). However, all these methods use largely the same information sources for tagging, and often almost the same features as well, and as a consequence they also offer very similar levels of performance. This stands in contrast to the (manually-built) EngCG tagger, which achieves better performance by using lexical and contextual information sources and generalizations beyond those available to such statistical taggers, as Samuelsson and Voutilainen (1997) demonstrate.

This paper explores the notion that automatically built tagger performance can be further improved by expanding the knowledge sources available to the tagger. We pay special attention to unknown words, because the markedly lower accuracy on unknown word tagging means that this is an area where significant performance gains seem possible.

We adopt a maximum entropy approach because it allows the inclusion of diverse sources of information without causing fragmentation and without necessarily assuming independence between the predictors. A maximum entropy approach has been applied to part-of-speech tagging before (Ratnaparkhi 1996), but the approach's ability to incorporate non-local and non-HMM-tagger-type evidence has not been fully explored. This paper describes the models that we developed and the experiments we performed to evaluate them.

## 1 The Baseline Maximum Entropy Model

We started with a maximum entropy based tagger that uses features very similar to the ones proposed in Ratnaparkhi (1996). The tagger learns a loglinear conditional probability model from tagged text, using a maximum entropy method.

The model assigns a probability for every tag $t$ in the set $T$ of possible tags given a word and its context $h$, which is usually defined as the sequence of several words and tags preceding the word. This model can be used for estimating the probability of a tag sequence $t_1 \ldots t_n$ given a sentence $w_1 \ldots w_n$:

$$p(t_1 \ldots t_n \mid w_1 \ldots w_n) = \prod_{i=1}^{n} p(t_i \mid t_1 \ldots t_{i-1}, w_1 \ldots w_n) \approx \prod_{i=1}^{n} p(t_i \mid h_i)$$

As usual, tagging is the process of assigning the maximum likelihood tag sequence to a string of words.

The idea of maximum entropy modeling is to choose the probability distribution $p$ that has the highest entropy out of those distributions

[1] We thank Dan Klein and Michael Saunders for useful discussions, and the anonymous reviewers for many helpful comments.

that satisfy a certain set of constraints. The constraints restrict the model to behave in accordance with a set of statistics collected from the training data. The statistics are expressed as the expected values of appropriate functions defined on the contexts *h* and tags *t*. In particular, the constraints demand that the expectations of the features for the model match the empirical expectations of the features over the training data.

For example, if we want to constrain the model to tag *make* as a verb or noun with the same frequency as the empirical model induced by the training data, we define the features:

$$f_1(h,t) = 1 \quad \text{iff} \quad w_i = \textit{make} \text{ and } t = \text{NN}$$
$$f_2(h,t) = 1 \quad \text{iff} \quad w_i = \textit{make} \text{ and } t = \text{VB}$$

Some commonly used statistics for part of speech tagging are: how often a certain word was tagged in a certain way; how often two tags appeared in sequence or how often three tags appeared in sequence. These look a lot like the statistics a Markov Model would use. However, in the maximum entropy framework it is possible to easily define and incorporate much more complex statistics, not restricted to *n*-gram sequences.

The constraints in our model are that the expectations of these features according to the joint distribution $p$ are equal to the expectations of the features in the empirical (training data) distribution $\tilde{p}$: $E_{p(h,t)} f_i(h,t) = E_{\tilde{p}(h,t)} f_i(h,t)$. Having defined a set of constraints that our model should accord with, we proceed to find the model satisfying the constraints that maximizes the conditional entropy of $p$. The intuition is that such a model assumes nothing apart from that it should satisfy the given constraints.

Following Berger et al. (1996), we approximate $p(h,t)$, the joint distribution of contexts and tags, by the product of $\tilde{p}(h)$, the empirical distribution of histories *h*, and the conditional distribution $p(t \mid h)$: $p(h,t) \approx \tilde{p}(h) \cdot p(t \mid h)$. Then for the example above, our constraints would be the following, for $j \in \{1,2\}$:

$$\sum_{h \in H, t \in T} \tilde{p}(h,t) f_j(h,t) = \sum_{h \in H, t \in T} \tilde{p}(h) p(t \mid h) f_j(h,t)$$

This approximation is used to enable efficient computation. The expectation for a feature $f$ is:

$$E\ f = \sum_{h \in H, t \in T} \tilde{p}(h) p(t \mid h) f(h,t)$$

where $H$ is the space of possible contexts *h* when predicting a part of speech tag *t*. Since the contexts contain sequences of words and tags and other information, the space *H* is huge. But using this approximation, we can instead sum just over the smaller space of observed contexts *X* in the training sample, because the empirical prior $\tilde{p}(h)$ is zero for unseen contexts *h*:

$$E\ f = \sum_{h \in X,\ t \in T} \tilde{p}(h) p(t \mid h) f(h,t) \qquad (1)$$

The model that is a solution to this constrained optimization task is an exponential (or equivalently, loglinear) model with the parametric form:

$$p(t \mid h) = \frac{\prod_{j=1,\ldots,K} e^{\lambda_j f_j(h,t)}}{\sum_{t' \in T} \prod_{j=1,\ldots,K} e^{\lambda_j f_j(h,t')}}$$

where the denominator is a normalizing term (sometimes referred to as the partition function). The parameters •$_j$ correspond to weights for the features $f_j$.

We will not discuss in detail the characteristics of the model or the parameter estimation procedure used – Improved Iterative Scaling. For a more extensive discussion of maximum entropy methods, see Berger et al. (1996) and Jelinek (1997). However, we note that our parameter estimation algorithm directly uses equation (1). Ratnaparkhi (1996: 134) suggests use of an approximation summing over the training data, which does not sum over possible tags:

$$E\ f_j \approx \sum_{i=1}^{n} \tilde{p}(h_i) p(t_i \mid h_i) f_j(h_i, t_i)$$

However, we believe this passage is in error: such an estimate is ineffective in the iterative scaling algorithm. Further, we note that expectations of the form (1) appear in Ratnaparkhi (1998: 12).

### 1.1 Features in the Baseline Model

In our baseline model, the context available when predicting the part of speech tag of a word $w_i$ in a sentence of words $\{w_1 \ldots w_n\}$ with tags $\{t_1 \ldots t_n\}$ is $\{t_{i-1}\ t_{i-2}\ w_i\ w_{i+1}\}$. The features that define the constraints on the model are obtained by instantiation of feature templates as in Ratnaparkhi (1996). Special feature templates exist for rare words in the training data, to increase the model's prediction capacity for unknown words.

The actual feature templates for this model are shown in the next table. They are a subset of the features used in Ratnaparkhi (1996).

| No. | Feature Type | Template |
|---|---|---|
| 1. | General | $w_i$=$X$ & $t_i$ =$T$ |
| 2. | General | $t_{i-1}$=$T_1$ & $t_i$=$T$ |
| 3. | General | $t_{i-1}$=$T_1$ & $t_{i-2}$=$T_2$ & $t_i$=$T$ |
| 4. | General | $w_{i+1}$=$X$ & $t_i$ =$T$ |
| 5. | Rare | Suffix of $w_i$ =$S$, $\|S\|<5$ & $t_i$=$T$ |
| 6. | Rare | Prefix of $w_i$ =$P$, $1<\|P\|<5$ & $t_i$=$T$ |
| 7. | Rare | $w_i$ contains a number & $t_i$=$T$ |
| 8. | Rare | $w_i$ contains an uppercase character & $t_i$=$T$ |
| 9. | Rare | $w_i$ contains a hyphen & $t_i$=$T$ |

**Table 1** Baseline Model Features

General feature templates can be instantiated by arbitrary contexts, whereas rare feature templates are instantiated only by histories where the current word $w_i$ is rare. Rare words are defined to be words that appear less than a certain number of times in the training data (here, the value 7 was used).

In order to be able to throw out features that would give misleading statistics due to sparseness or noise in the data, we use two different cutoff values for general and rare feature templates (in this implementation, 5 and 45 respectively). As seen in Table 1 the features are conjunctions of a boolean function on the history *h* and a boolean function on the tag *t*. Features whose first conjuncts are true for more than the corresponding threshold number of histories in the training data are included in the model.

The feature templates in Ratnaparkhi (1996) that were left out were the ones that look at the previous word, the word two positions before the current, and the word two positions after the current. These features are of the same form as template 4 in Table 1, but they look at words in different positions.

Our motivation for leaving these features out was the results from some experiments on successively adding feature templates. Adding template 4 to a model that incorporated the general feature templates 1 to 3 only and the rare feature templates 5–8 significantly increased the accuracy on the development set – from 96.0% to 96.52%. The addition of a feature template that looked at the preceding word and the current tag to the resulting model slightly reduced the accuracy.

## 1.2 Testing and Performance

The model was trained and tested on the part-of-speech tagged WSJ section of the Penn Treebank. The data was divided into contiguous parts: sections 0–20 were used for training, sections 21–22 as a development test set, and sections 23–24 as a final test set. The data set sizes are shown below together with numbers of unknown words.

| Data Set | Tokens | Unknown |
|---|---|---|
| Training | 1,061,768 | |
| Development | 116,206 | 3271 (2.81%) |
| Test | 111,221 | 2879 (2.59%) |

**Table 2** Data Sizes

The testing procedure uses a beam search to find the tag sequence with maximal probability given a sentence. In our experiments we used a beam of size 5. Increasing the beam size did not result in improved accuracy.

The preceding tags for the word at the beginning of the sentence are regarded as having the pseudo-tag NA. In this way, the information that a word is the first word in a sentence is available to the tagger. We do not have a special end-of-sentence symbol.

We used a tag dictionary for known words in testing. This was built from tags found in the training data but augmented so as to capture a few basic systematic tag ambiguities that are found in English. Namely, for regular verbs the *-ed* form can be either a VBD or a VBN and similarly the stem form can be either a VBP or VB. Hence for words that had occurred with only one of these tags in the training data the other was also included as possible for assignment.

The results on the test set for the Baseline model are shown in Table 3.

| Model | Overall Accuracy | Unknown Word Accuracy |
|---|---|---|
| Baseline | 96.72% | 84.5% |
| Ratnaparkhi (1996) | 96.63% | 85.56% |

**Table 3** Baseline model performance

This table also shows the results reported in Ratnaparkhi (1996: 142) for convenience. The accuracy figure for our model is higher overall but lower for unknown words. This may stem from the differences between the two models' feature templates, thresholds, and approximations of the expected values for the features, as discussed in the beginning of the section, or may just reflect differences in the choice of training and test sets (which are not precisely specified in Ratnaparkhi (1996)).

The differences are not great enough to justify any definite statement about the different use of feature templates or other particularities of the model estimation. One conclusion that we can draw is that at present the additional word features used in Ratnaparkhi (1996) – looking at words more than one position away from the current – do not appear to be helping the overall performance of the models.

### 1.3 Discussion of Problematic Cases

A large number of words, including many of the most common words, can have more than one syntactic category. This introduces a lot of ambiguities that the tagger has to resolve. Some of the ambiguities are easier for taggers to resolve and others are harder.

Some of the most significant confusions that the Baseline model made on the test set can be seen in Table 5. The row labels in Table 5 signify the correct tags, and the column labels signify the assigned tags. For example, the number 244 in the (NN, JJ) position is the number of words that were NNs but were incorrectly assigned the JJ category. These particular confusions, shown in the table, account for a large percentage of the total error (2652/3651 = 72.64%). Table 6 shows part of the Baseline model's confusion matrix for just unknown words.

Table 4 shows the Baseline model's overall assignment accuracies for different parts of speech. For example, the accuracy on nouns is greater than the accuracy on adjectives. The accuracy on NNPS (plural proper nouns) is a surprisingly low 41.1%.

| **Tag** | **Accuracy** | **Tag** | **Accuracy** |
|---|---|---|---|
| IN | 97.3% | JJ | 93.0% |
| NN | 96.5% | RB | 92.2% |
| NNP | 96.2% | VBN | 90.4% |
| VBD | 95.2% | RP | 41.5% |
| VB | 94.0% | NNPS | 41.1% |
| VBP | 93.4% | | |

**Table 4** Accuracy of assignments for different parts of speech for the Baseline model.

Tagger errors are of various types. Some are the result of inconsistency in labeling in the training data (Ratnaparkhi 1996), which usually reflects a lack of linguistic clarity or determination of the correct part of speech in context. For instance, the status of various noun premodifiers (whether *chief* or *maximum* is NN or JJ, or whether a word in *-ing* is acting as a JJ or VBG) is of this type. Some, such as errors between NN/NNP/NNPS/NNS largely reflect difficulties with unknown words. But other cases, such as VBN/VBD and VB/VBP/NN, represent systematic tag ambiguity patterns in English, for which the right answer is invariably clear in context, and for which there are in general good structural contextual clues that one should be able to use to disambiguate. Finally, in another class of cases, of which the most prominent is probably the RP/IN/RB ambiguity of words like *up*, *out*, and *on*, the linguistic distinctions, while having a sound empirical basis (e.g., see Baker (1995: 198–201), are quite subtle, and often require semantic intuitions. There are not good syntactic cues for the correct tag (and furthermore, human taggers not infrequently make errors). Within this classification, the greatest hopes for tagging improvement appear to come from minimizing errors in the second and third classes of this classification.

In the following sections we discuss how we include additional knowledge sources to help in the assignment of tags to forms of verbs, capitalized unknown words, particle words, and in the overall accuracy of part of speech assignments.

## 2 Improving the Unknown Words Model

The accuracy of the baseline model is markedly lower for unknown words than for previously seen ones. This is also the case for all other taggers, and reflects the importance of lexical information to taggers: in the best accuracy figures published for corpus-based taggers, known word accuracy is around 97%, whereas unknown word accuracy is around 85%.

In following experiments, we examined ways of using additional features to improve the accuracy of tagging unknown words. As previously discussed in Mikheev (1999), it is possible to improve the accuracy on capitalized words that might be proper nouns or the first word in a sentence, etc.

| | JJ | NN | NNP | NNPS | RB | RP | IN | VB | VBD | VBN | VBP | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| JJ | 0 | **177** | **56** | 0 | **61** | 2 | 5 | 10 | 15 | **108** | 0 | 488 |
| NN | **244** | 0 | **103** | 0 | 12 | 1 | 1 | 29 | 5 | 6 | 19 | 525 |
| NNP | **107** | **106** | 0 | **132** | 5 | 0 | 7 | 5 | 1 | 2 | 0 | 427 |
| NNPS | 1 | 0 | **110** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 142 |
| RB | **72** | 21 | 7 | 0 | 0 | 16 | **138** | 1 | 0 | 0 | 0 | 295 |
| RP | 0 | 0 | 0 | 0 | **39** | 0 | **65** | 0 | 0 | 0 | 0 | 104 |
| IN | 11 | 0 | 1 | 0 | **169** | **103** | 0 | 1 | 0 | 0 | 0 | 323 |
| VB | 17 | **64** | 9 | 0 | 2 | 0 | 1 | 0 | 4 | 7 | **85** | 189 |
| VBD | 10 | 5 | 3 | 0 | 0 | 0 | 0 | 3 | 0 | **143** | 2 | 166 |
| VBN | **101** | 3 | 3 | 0 | 0 | 0 | 0 | 3 | **108** | 0 | 1 | 221 |
| VBP | 5 | 34 | 3 | 1 | 1 | 0 | 2 | **49** | 6 | 3 | 0 | 104 |
| Total | 626 | 536 | 348 | 144 | 317 | 122 | 279 | 102 | 140 | 269 | 108 | 3651 |

**Table 5** Confusion matrix of the Baseline model showing top confusion pairs overall

| | JJ | NN | NNP | NNS | NNPS | VBN | Total |
|---|---|---|---|---|---|---|---|
| JJ | 0 | 55 | 25 | 1 | 0 | 10 | 107 |
| NN | 55 | 0 | 26 | 5 | 0 | 2 | 98 |
| NNP | 20 | 41 | 0 | 5 | 4 | 0 | 87 |
| NNPS | 0 | 0 | 10 | 11 | 0 | 0 | 23 |
| NNS | 1 | 3 | 6 | 0 | 1 | 0 | 15 |
| VBN | 12 | 1 | 1 | 0 | 0 | 0 | 20 |
| Total | 109 | 121 | 98 | 33 | 7 | 19 | 448 |

**Table 6** Confusion matrix of the Baseline model for unknown words showing top confusion pairs

| | **Baseline** | **Model 1 Capitalization** | **Model 2 Verb forms** | **Model 3 Particles** |
|---|---|---|---|---|
| Accuracy Test Set | 96.72% | 96.76% | 96.83% | 96.86% |
| Unknown Words Accuracy Test Set | 84.50% | 86.76% | 86.87% | 86.91% |
| Accuracy Development Set | 96.53% | 96.55% | 96.58% | 96.62% |
| Unknown Words Accuracy Development Set | 85.48% | 86.03% | 86.03% | 86.06% |

**Table 7** Accuracies of all models on the test and development sets

| | **Baseline** | **Model 1 Capitalization** | **Model 2 Verb Forms** | **Model 3 Particles** |
|---|---|---|---|---|
| 1. Current word | 15,832 | 15,832 | 15,837 | 15,927 |
| 2. Previous tag | 1,424 | 1,424 | 1,424 | 1,424 |
| 3. Previous two tags | 16,124 | 16,124 | 16,124 | 16,124 |
| 4. Next word | 80,075 | 80,075 | 80,075 | 80,075 |
| 5. Suffixes | 3,361 | 3,361 | 3,361 | 3,387 |
| 6. Prefixes | 5,311 | 0 | 0 | 0 |
| 7. Contains uppercase character | 34 | 34 | 34 | 34 |
| 8. Contains number | 7 | 7 | 7 | 7 |
| 9. Contains hyphen | 20 | 20 | 20 | 20 |
| 10. Capitalized and mid. sentence | 0 | 33 | 33 | 33 |
| 11. All letters uppercase | 0 | 30 | 30 | 30 |
| 12. VBP\|VB feature | 0 | 0 | 2 | 2 |
| 13. VBD\|VBN feature | 0 | 0 | 3 | 3 |
| 14. Particles, type 1 | 0 | 0 | 0 | 9 |
| 15. Particles, type 2 | 0 | 0 | 0 | 2,178 |
| Total | 122,188 | 116,940 | 116,960 | 118,944 |

**Table 8** Number of features of different types

For example, the error on the proper noun category (NNP) accounts for a significantly larger percent of the total error for unknown words than for known words. In the Baseline model, of the unknown word error 41.3% is due to words being NNP and assigned to some other category, or being of other category and assigned NNP. The percentage of the same type of error for known words is 16.2%.

The incorporation of the following two feature schemas greatly improved NNP accuracy:

(1) A feature that looks at whether all the letters of a word are uppercase. The feature that looked at capitalization before (cf. Table 1, feature No. 8) is activated when the word contains an uppercase character. This turns out to be a notable distinction because, for example, in titles in the WSJ data all words are in all uppercase, and the distribution of tags for these words is different from the overall distribution for words that contain an uppercase character.

(2) A feature that is activated when the word contains an uppercase character and it is not at the start of a sentence. These word tokens also have a different tag distribution from the distribution for all tokens that contain an uppercase character.

Conversely, empirically it was found that the prefix features for rare words were having a net negative effect on accuracy. We do not at present have a good explanation for this phenomenon.

The addition of the features (1) and (2) and the removal of the prefix features considerably improved the accuracy on unknown words and the overall accuracy. The results on the test set after adding these features are shown below:

| **Overall Accuracy** | **Unknown Word Accuracy** |
|---|---|
| 96.76% | 86.76% |

**Table 9** Accuracy when adding capitalization features and removing prefix features.

Unknown word error is reduced by 15% as compared to the Baseline model.

It is important to note that (2) is composed of information already 'known' to the tagger in some sense. This feature can be viewed as the conjunction of two features, one of which is already in the baseline model, and the other of which is the negation of a feature existing in the baseline model – since for words at the beginning of a sentence, the preceding tag is the pseudo-tag NA, and there is a feature looking at the preceding tag. Even though our maximum entropy model does not require independence among the predictors, it provides for free only a simple combination of feature weights, and additional 'interaction terms' are needed to model non-additive interactions (in log-space terms) between features.

## 3 Features for Disambiguating Verb Forms

Two of the most significant sources of classifier errors are the VBN/VBD ambiguity and the VBP/VB ambiguity. As seen in Table 5, VBN/VBD confusions account for 6.9% of the total word error. The VBP/VB confusions are a smaller 3.7% of the errors. In many cases it is easy for people (and for taggers) to determine the correct form. For example, if there is a *to* infinitive or a modal directly preceding the VB/VBP ambiguous word, the form is certainly non-finite. But often the modal can be several positions away from the current position – still obvious to a human, but out of sight for the baseline model.

To help resolve a VB/VBP ambiguity in such cases, we can add a feature that looks at the preceding several words (we have chosen 8 as a threshold), but not across another verb, and activates if there is a *to* there, a modal verb, or a form of *do, let, make,* or *help* (verbs that frequently take a bare infinitive complement).

Rather than having a separate feature look at each preceding position, we define one feature that looks at the chosen number of positions to the left. This both increases the scope of the available history for the tagger and provides a better statistic because it avoids fragmentation.

We added a similar feature for resolving VBD/VBN confusions. It activates if there is a *have* or *be* auxiliary form in the preceding several positions (again the value 8 is used in the implementation).

The form of these two feature templates was motivated by the structural rules of English and not induced from the training data, but it should be possible to look for "predictors" for certain parts of speech in the preceding words in the sentence by, for example, computing association strengths.

The addition of the two feature schemas helped reduce the VB/VBP and VBD/VBN confusions. Below is the performance on the test set of the resulting model when features for disambiguating verb forms are added to the model of Section 2. The number of VB/VBP confusions was reduced by 23.1% as compared to the baseline. The number of VBD/VBN confusions was reduced by 12.3%.

| Overall Accuracy | Unknown Word Accuracy |
|---|---|
| 96.83% | 86.87% |

**Table 10** Accuracy of the extended model

## 4 Features for Particle Disambiguation

As discussed in section 1.3 above, the task of determining RB/RP/IN tags for words like *down*, *out*, *up* is difficult and in particular examples, there are often no good local syntactic indicators. For instance, in (2), we find the exact same sequence of parts of speech, but (2a) is a particle use of *on*, while (2b) is a prepositional use. Consequently, the accuracy on the rarer RP (particles) category is as low as 41.5% for the Baseline model (cf. Table 4).

(2) a. Kim took on the monster.
    b. Kim sat on the monster.

We tried to improve the tagger's capability to resolve these ambiguities through adding information on verbs' preferences to take specific words as particles, or adverbs, or prepositions. There are verbs that take particles more than others, and particular words like *out* are much more likely to be used as a particle in the context of some verb than other words ambiguous between these tags.

We added two different feature templates to capture this information, consisting as usual of a predicate on the history *h*, and a condition on the tag *t*. The first predicate is true if the current word is often used as a particle, and if there is a verb at most 3 positions to the left, which is "known" to have a good chance of taking the current word as a particle. The verb-particle pairs that are known by the system to be very common were collected through analysis of the training data in a preprocessing stage.

The second feature template has the form: The last verb is *v* and the current word is *w* and *w* has been tagged as a particle and the current tag is *t*. The last verb is the pseudo-symbol NA if there is no verb in the previous three positions.

These features were some help in reducing the RB/IN/RP confusions. The accuracy on the RP category rose to 44.3%. Although the overall confusions in this class were reduced, some of the errors were increased, for example, the number of INs classified as RBs rose slightly. There seems to be still considerable room to improve these results, though the attainable accuracy is limited by the accuracy with which these distinctions are marked in the Penn Treebank (on a quick informal study, this accuracy seems to be around 85%). The next table shows the final performance on the test set.

| Overall Accuracy | Unknown Word Accuracy |
|---|---|
| 96.86% | 86.91% |

**Table 11** Accuracy of the final model

For ease of comparison, the accuracies of all models on the test and development sets are shown in Table 7. We note that accuracy is lower on the development set. This presumably corresponds with Charniak's (2000: 136) observation that Section 23 of the Penn Treebank is easier than some others. Table 8 shows the different number of feature templates of each kind that have been instantiated for the different models as well as the total number of features each model has. It can be seen that the features which help disambiguate verb forms, which look at capitalization and the first of the feature templates for particles are a very small number as compared to the features of the other kinds. The improvement in classification accuracy therefore comes at the price of adding very few parameters to the maximum entropy model and does not result in increased model complexity.

## Conclusion

Even when the accuracy figures for corpus-based part-of-speech taggers start to look extremely similar, it is still possible to move performance levels up. The work presented in this paper explored just a few information sources in addition to the ones usually used for tagging. While progress is slow, because each new feature applies only to a limited range of cases, nevertheless the improvement in accuracy as compared to previous results is noticeable, particularly for the individual decisions on which we focused.

The potential of maximum entropy methods has not previously been fully exploited for the task of assignment of parts of speech. We incorporated into a maximum entropy-based tagger more linguistically sophisticated features, which are non-local and do not look just at particular positions in the text. We also added features that model the interactions of previously employed predictors. All of these changes led to modest increases in tagging accuracy.

This paper has thus presented some initial experiments in improving tagger accuracy through using additional information sources. In the future we hope to explore automatically discovering information sources that can be profitably incorporated into maximum entropy part-of-speech prediction.

## References


Baker, C. L. 1995. *English Syntax.* Cambridge, MA: MIT Press, 2nd edition.

Berger, Adam L., Della Pietra, Stephen A., and Della Pietra, Vincent J. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics* 22: 39–71.

Brants, Thorsten. 2000. TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP 2000),* Seattle, WA, pp. 224–231.

Brill, Eric. 1994. Some Advances in Transformation-Based Part of Speech Tagging. *Proceedings of AAAI*, Vol. 1, pp. 722–727.

Charniak, Eugene. 2000. A Maximum-Entropy-Inspired Parser. *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 132–139.

Jelinek, Frederick. 1997. *Statistical Methods for Speech Recognition*. Cambridge, MA: MIT Press.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Mikheev, Andrei. 1999. Periods, Capitalized Words, etc. Ms., University of Edinburgh. Available at: http://www.ltg.ed.ac.uk/~mikheev/papers.html

Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, University of Pennsylvania, pp. 133–142.

Ratnaparkhi, Adwait. 1998. Maximum Entropy Models for Natural Language Ambiguity Resolution. PhD Thesis, University of Pennsylvania.

Samuelsson, Christer and Atro Voutilainen. 1997. Comparing a Linguistic and a Stochastic Tagger. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pp. 246–253.