A Maximum Entropy Model for Part-Of-Speech Tagging

Adwait Ratnaparkhi
University of Pennsylvania
Dept. of Computer and Information Science
adwait@gradient.cis.upenn.edu

Abstract

This paper presents a statistical model which trains from a corpus annotated with Part-Of-Speech tags and assigns them to previously unseen text with state-of-the-art accuracy(96.6%). The model can be classified as a *Maximum Entropy* model and simultaneously uses many contextual "features" to predict the POS tag. Furthermore, this paper demonstrates the use of specialized features to model difficult tagging decisions, discusses the corpus consistency problems discovered during the implementation of these features, and proposes a training strategy that mitigates these problems.

Introduction

Many natural language tasks require the accurate assignment of Part-Of-Speech (POS) tags to previously unseen text. Due to the availability of large corpora which have been manually annotated with POS information, many taggers use annotated text to "learn" either probability distributions or rules and use them to automatically assign POS tags to unseen text.

The experiments in this paper were conducted on the Wall Street Journal corpus from the Penn Treebank project(Marcus et al., 1994), although the model can train from any large corpus annotated with POS tags. Since most realistic natural language applications must process words that were never seen before in training data, all experiments in this paper are conducted on test data that include unknown words.

Several recent papers(Brill, 1994, Magerman, 1995) have reported 96.5% tagging accuracy on the Wall St. Journal corpus. The experiments in this paper test the hypothesis that *better use of context will improve the accuracy*. A Maximum Entropy model is well-suited for such experiments since it combines diverse forms of contextual information in a principled manner, and does not impose any distributional assumptions on the training data. Previous uses of this model include language modeling(Lau et al., 1993), machine translation(Berger et al., 1996), prepositional phrase attachment(Ratnaparkhi et al., 1994), and word morphology(Della Pietra et al., 1995). This paper briefly describes the maximum entropy and maximum likelihood properties of the model, features used for POS tagging, and the experiments on the Penn Treebank Wall St. Journal corpus. It then discusses the consistency problems discovered during an attempt to use specialized features on the word context. Lastly, the results in this paper are compared to those from previous work on POS tagging.

The Probability Model

The probability model is defined over $\mathcal{H} \times \mathcal{T}$, where $\mathcal{H}$ is the set of possible word and tag contexts, or "histories", and $\mathcal{T}$ is the set of allowable tags. The model's probability of a history h together with a tag t is defined as:

$$p(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)} \qquad (1)$$

where π is a normalization constant, $\{\mu, \alpha_1, \ldots, \alpha_k\}$ are the positive model parameters and $\{f_1, \ldots, f_k\}$ are known as "features", where $f_j(h,t) \in \{0,1\}$. Note that each parameter α_j corresponds to a feature f_j.

Given a sequence of words $\{w_1, \ldots, w_n\}$ and tags $\{t_1, \ldots t_n\}$ as training data, define h_i as the history available when predicting t_i. The parameters $\{\mu, \alpha_1, \ldots, \alpha_k\}$ are then chosen to maximize the likelihood of the training data using p:

$$L(p) = \prod_{i=1}^{n} p(h_i, t_i) = \prod_{i=1}^{n} \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h_i,t_i)}$$

This model also can be interpreted under the Maximum Entropy formalism, in which the goal is to maximize the entropy of a distribution subject to certain constraints. Here, the entropy of the distribution p is defined as:

$$H(p) = -\sum_{h\in\mathcal{H}, t\in\mathcal{T}} p(h,t)\log p(h,t)$$

and the constraints are given by:

$$Ef_j = \tilde{E}f_j, \quad 1 \leq j \leq k \tag{2}$$

where the model's feature expectation is

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

and the observed feature expectation is

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

and where $\tilde{p}(h_i,t_i)$ denotes the observed probability of (h_i,t_i) in the training data. Thus the constraints force the model to match its feature expectations with those observed in the training data. In practice, $\mathcal{H}$ is very large and the model's expectation Ef_j cannot be computed directly, so the following approximation(Lau et al., 1993) is used:

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

where $\tilde{p}(h_i)$ is the observed probability of the history h_i in the training set.

It can be shown (Darroch and Ratcliff, 1972) that if p has the form (1) and satisfies the k constraints (2), it uniquely maximizes the entropy $H(p)$ over distributions that satisfy (2), and uniquely maximizes the likelihood $L(p)$ over distributions of the form (1). The model parameters for the distribution p are obtained via *Generalized Iterative Scaling*(Darroch and Ratcliff, 1972).

Features for POS Tagging

The joint probability of a history h and tag t is determined by those parameters whose corresponding features are active, i.e., those α_j such that $f_j(h,t) = 1$. A feature, given (h,t), may activate on any word or tag in the history h, and must encode any information that might help predict t, such as the spelling of the current word, or the identity of the previous two tags. The specific word and tag context available to a feature is given in the following definition of a history h_i:

$$h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$$

For example,

$$f_j(h_i,t_i) = \begin{cases} 1 & \text{if suffix}(w_i) = \text{“}\texttt{ing}\text{”} \ \& \ t_i = \texttt{VBG} \\ 0 & \text{otherwise} \end{cases}$$

If the above feature exists in the feature set of the model, its corresponding model parameter will contribute towards the joint probability $p(h_i,t_i)$ when w_i ends with "`ing`" and when t_i =`VBG`[1]. Thus a model parameter α_j effectively serves as a "weight" for a certain contextual predictor, in this case the suffix "`ing`", towards the probability of observing a certain tag, in this case a `VBG`.

The model generates the space of features by scanning each pair (h_i,t_i) in the training data with the feature "templates" given in Table 1. Given h_i as the current history, a feature always asks some yes/no question about h_i, and furthermore constrains t_i to be a certain tag. The instantiations for the variables X, Y, and T in Table 1 are obtained automatically from the training data.

The generation of features for tagging unknown words relies on the hypothesized distinction that "rare" words[2] in the training set are similar to unknown words in test data, with respect to how their spellings help predict their tags. The rare word features in Table 1, which look at the word spellings, will apply to both rare words and unknown words in test data.

For example, Table 2 contains an excerpt from training data while Table 3 contains the features generated while scanning (h_3,t_3), in which the current word is `about`, and Table 4 contains features generated while scanning (h_4,t_4), in which the current word, `well-heeled`, occurs 3 times in training data and is therefore classified as "rare".

The behavior of a feature that occurs very sparsely in the training set is often difficult to predict, since its statistics may not be reliable. Therefore, the model uses the heuristic that any feature

[1]`VBG` is the Treebank POS tag for Verb Gerund.

[2]A "rare" word here denotes a word which occurs less than 5 times in the training set. The count of 5 was chosen by subjective inspection of words in the training data.

Condition	Features	
w_i is not rare	$w_i = X$	$\& \ t_i = T$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$	$\& \ t_i = T$
	X is suffix of w_i, $\|X\| \leq 4$	$\& \ t_i = T$
	w_i contains number	$\& \ t_i = T$
	w_i contains uppercase character	$\& \ t_i = T$
	w_i contains hyphen	$\& \ t_i = T$
$\forall \ w_i$	$t_{i-1} = X$	$\& \ t_i = T$
	$t_{i-2}t_{i-1} = XY$	$\& \ t_i = T$
	$w_{i-1} = X$	$\& \ t_i = T$
	$w_{i-2} = X$	$\& \ t_i = T$
	$w_{i+1} = X$	$\& \ t_i = T$
	$w_{i+2} = X$	$\& \ t_i = T$

Table 1: Features on the current history h_i

Word:	`the`	`stories`	`about`	`well-heeled`	`communities`	`and`	`developers`
Tag:	`DT`	`NNS`	`IN`	`JJ`	`NNS`	`CC`	`NNS`
Position:	1	2	3	4	5	6	7

Table 2: Sample Data

$w_i =$ `about`	$\& \ t_i =$ `IN`
$w_{i-1} =$ `stories`	$\& \ t_i =$ `IN`
$w_{i-2} =$ `the`	$\& \ t_i =$ `IN`
$w_{i+1} =$ `well-heeled`	$\& \ t_i =$ `IN`
$w_{i+2} =$ `communities`	$\& \ t_i =$ `IN`
$t_{i-1} =$ `NNS`	$\& \ t_i =$ `IN`
$t_{i-2}t_{i-1} =$ `DT NNS`	$\& \ t_i =$ `IN`

Table 3: Features Generated From h_3 (for tagging `about`) from Table 2

$w_{i-1} =$ `about`	$\& \ t_i =$ `JJ`
$w_{i-2} =$ `stories`	$\& \ t_i =$ `JJ`
$w_{i+1} =$ `communities`	$\& \ t_i =$ `JJ`
$w_{i+2} =$ `and`	$\& \ t_i =$ `JJ`
$t_{i-1} =$ `IN`	$\& \ t_i =$ `JJ`
$t_{i-2}t_{i-1} =$ `NNS IN`	$\& \ t_i =$ `JJ`
prefix(w_i)=`w`	$\& \ t_i =$ `JJ`
prefix(w_i)=`we`	$\& \ t_i =$ `JJ`
prefix(w_i)=`wel`	$\& \ t_i =$ `JJ`
prefix(w_i)=`well`	$\& \ t_i =$ `JJ`
suffix(w_i)=`d`	$\& \ t_i =$ `JJ`
suffix(w_i)=`ed`	$\& \ t_i =$ `JJ`
suffix(w_i)=`led`	$\& \ t_i =$ `JJ`
suffix(w_i)=`eled`	$\& \ t_i =$ `JJ`
w_i contains hyphen	$\& \ t_i =$ `JJ`

Table 4: Features Generated From h_4 (for tagging `well-heeled`) from Table 2

which occurs less than 10 times in the data is unreliable, and ignores features whose counts are less than 10.[3] While there are many smoothing algorithms which use techniques more rigorous than a simple count cutoff, they have not yet been investigated in conjunction with this tagger.

Testing the Model

The test corpus is tagged one sentence at a time. The testing procedure requires a search to enumerate the candidate tag sequences for the sentence, and the tag sequence with the highest probability is chosen as the answer.

Search Algorithm

The search algorithm, essentially a "beam search", uses the conditional tag probability

$$p(t|h) = \frac{p(h,t)}{\sum_{t' \in T} p(h,t')}$$

and maintains, as it sees a new word, the N highest probability tag sequence candidates up to that point in the sentence. Given a sentence $\{w_1 \dots w_n\}$, a tag sequence candidate $\{t_1 \dots t_n\}$ has conditional probability:

$$P(t_1 \dots t_n | w_1 \dots w_n) = \prod_{i=1}^{n} p(t_i|h_i)$$

In addition the search procedure optionally consults a *Tag Dictionary*, which, for each known word, lists the tags that it has appeared with in the training set. If the Tag Dictionary is in effect, the search procedure, for known words, generates only tags given by the dictionary entry, while for unknown words, generates all tags in the tag set. Without the Tag Dictionary, the search procedure generates all tags in the tag set for every word.

Let $W = \{w_1 \dots w_n\}$ be a test sentence, and let s_{ij} be the jth highest probability tag sequence up to and including word w_i. The search is described below:

1. Generate tags for w_1, find top N, set s_{1j}, $1 \leq j \leq N$, accordingly.
2. Initialize $i = 2$
   (a) Initialize $j = 1$
   (b) Generate tags for w_i, given $s_{(i-1)j}$ as previous tag context, and append each tag to $s_{(i-1)j}$ to make a new sequence
   (c) $j = j + 1$, Repeat from (b) if $j \leq N$
3. Find N highest probability sequences generated by above loop, and set s_{ij}, $1 \leq j \leq N$, accordingly.
4. $i = i + 1$, Repeat from (a) if $i \leq n$
5. Return highest probability sequence, s_{n1}

Experiments

In order to conduct tagging experiments, the Wall St. Journal data has been split into three contiguous sections, as shown in Table 5. The feature set and search algorithm were tested and debugged only on the Training and Development sets, and the official test result on the unseen Test set is presented in the conclusion of the paper. The performances of the "baseline" model on the Development Set, both with and without the Tag Dictionary, are shown in Table 6.

All experiments use a beam size of $N = 5$; further increasing the beam size does not significantly increase performance on the Development Set but adversely affects the speed of the tagger. Even though use of the Tag Dictionary gave an apparently insignificant (.12%) improvement in accuracy, it is used in further experiments since it significantly reduces the number of hypotheses and thus speeds up the tagger.

The running time of the parameter estimation algorithm is $O(NTA)$, where N is the training set size, T is the number of allowable tags, and A is the average number of features that are active for a given event (h,t). The running time of the search procedure on a sentence of length N is $O(NTAB)$, where T, A are defined above, and B is the beam size. In practice, the model for the experiment shown in Table 6 requires approximately 24 hours to train, and 1 hour to test[4] on an IBM RS/6000 Model 380 with 256MB of RAM.

Specialized Features and Consistency

The Maximum Entropy model allows arbitrary binary-valued features on the context, so it can use additional specialized, i.e., word-specific, features

[3] Except for features that look only at the current word, i.e., features of the form $w_i =$`<word>` and $t_i =$`<TAG>`. The count of 10 was chosen by inspection of Training and Development data.

[4] The search procedure has not been optimized and the author expects it to run 3 to 5 times faster after optimizations.

DataSet	Sentences	Words	Unknown Words
Training	40000	962687	-
Development	8000	192826	6107
Test	5485	133805	3546

Table 5: WSJ Data Sizes

	Total Word Accuracy	Unknown Word Accuracy	Sentence Accuracy
Tag Dictionary	96.43%	86.23%	47.55%
No Tag Dictionary	96.31%	86.28%	47.38%

Table 6: Baseline Performance on Development Set

Word	Correct Tag	Model's Tag	Frequency
about	RB	IN	393
that	DT	IN	389
more	RBR	JJR	221
up	IN	RB	187
that	WDT	IN	184
as	RB	IN	176
up	IN	RP	176
more	JJR	RBR	175
that	IN	WDT	159
about	IN	RB	144
that	IN	DT	127
out	RP	IN	126
that	DT	WDT	123
much	JJ	RB	118
yen	NN	NNS	117
chief	NN	JJ	116
up	RP	IN	114
ago	IN	RB	112
much	RB	JJ	111
out	IN	RP	109

Table 7: Top Tagging Mistakes on Training Set for Baseline Model

to correctly tag the "residue" that the baseline features cannot model. Since such features typically occur infrequently, the training set consistency must be good enough to yield reliable statistics. Otherwise the specialized features will model noise and perform poorly on test data.

Such features can be designed for those words which are especially problematic for the model. The top errors of the model (over the training set) are shown in Table 7; clearly, the model has trouble with the words `that` and `about`, among others. As hypothesized in the introduction, better features on the context surrounding `that` and `about` should correct the tagging mistakes for these two words, assuming that the tagging errors are due to an impoverished feature set, and not inconsistent data.

Specialized features for a given word are constructed by conjoining certain features in the baseline model with a question about the word itself. The features which ask about previous tags and surrounding words now additionally ask about the identity of the current word, e.g., a specialized feature for the word `about` in Table 3 could be:

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{if } w_i = \texttt{about} \;\&\; t_{i-2}t_{i-1} = \texttt{DT NNS} \\ & \&\; t_i = \texttt{IN} \\ 0 & \text{otherwise} \end{cases}$$

Table 8 shows the results of an experiment in which specialized features are constructed for "difficult" words, and are added to the baseline feature set. Here, "difficult" words are those that are mistagged a certain way at least 50 times when the training set is tagged with the baseline model. Using the set of 29 difficult words, the model performs at 96.49% accuracy on the Development Set, an insignificant improvement from the baseline accuracy of 96.43%. Table 9 shows the change in error rates on the Development Set for the frequently occurring "difficult" words. For most words, the specialized model yields little or no improvement, and for some, i.e., `more` and `about`, the specialized model performs worse.

The lack of improvement implies that either the feature set is still impoverished, or that the training data is inconsistent. A simple consistency test is to graph the POS tag assignments for a given word as a function of the article in which it occurs. Consistently tagged words should have roughly the same tag distribution as the article numbers vary. Figure 1 represents each POS tag with a unique integer and graphs the POS annotation of `about` in the training set as a function of the article# (the points are "scattered" to show density). As seen in figure 1, `about` is usually annotated with tag#1, which denotes `IN` (preposition), or tag#9, which denotes `RB` (adverb), and the observed probability of either choice depends heavily on the current article#. Upon further examination[5], the tagging distribution for `about` changes *precisely* when the annotator changes. Figure 2, which again uses integers to denote POS tags, shows the tag distribution of `about` as a function of annotator, and implies that the tagging errors for this word are due mostly to inconsistent data. The words `ago`, `chief`, `down`, `executive`, `off`, `out`, `up` and `yen` also exhibit similar bias.

Thus specialized features may be less effective for those words affected by inter-annotator bias. A simple solution to eliminate inter-annotator inconsistency is to train and test the model on data that has been created by the *same* annotator. The results of such an experiment[6] are shown in Table 10. The total accuracy is higher, implying that the singly-annotated training and test sets are more consistent, and the improvement due to the specialized features is higher than before (.1%) but still modest, implying that either the features need further improvement or that *intra*-annotator inconsistencies exist in the corpus.

Comparison With Previous Work

Most of the recent corpus-based POS taggers in the literature are either statistically based, and use Markov Model(Weischedel et al., 1993, Merialdo, 1994) or Statistical Decision Tree(Jelinek et al., 1994, Magerman, 1995)(SDT) techniques, or are primarily rule based, such as Brill's Transformation Based Learner(Brill, 1994)(TBL). The Maximum Entropy (MaxEnt) tagger presented in this paper combines the advantages of all these methods. It uses a rich feature representation, like TBL and SDT, and generates a tag probability distribution for each word, like Decision Tree and Markov Model techniques.

---

[5]The mapping from article to annotator is in the file `doc/wsj.wht` on the Treebank CDROM.

[6]The single-annotator training data was obtained by extracting those articles tagged by "maryann" in the Treebank v.5 CDROM. This training data does not overlap with the Development and Test set used in the paper. The single-annotator Development Set is the portion of the Development Set which has also been annotated by "maryann". The word vocabulary and tag dictionary are the same as in the baseline experiment.

Number of "Difficult" Words	Development Set Performance
29	96.49%

Table 8: Performance of Baseline Model with Specialized Features

Word	# Baseline Model Errors	# Specialized Model Errors
that	246	207
up	186	169
about	110	120
out	104	97
more	88	89
down	81	84
off	73	78
as	50	38
much	47	40
chief	46	47
in	39	39
executive	37	33
most	23	34
ago	22	18
yen	18	17

Table 9: Errors on Development Set with Baseline and Specialized Models

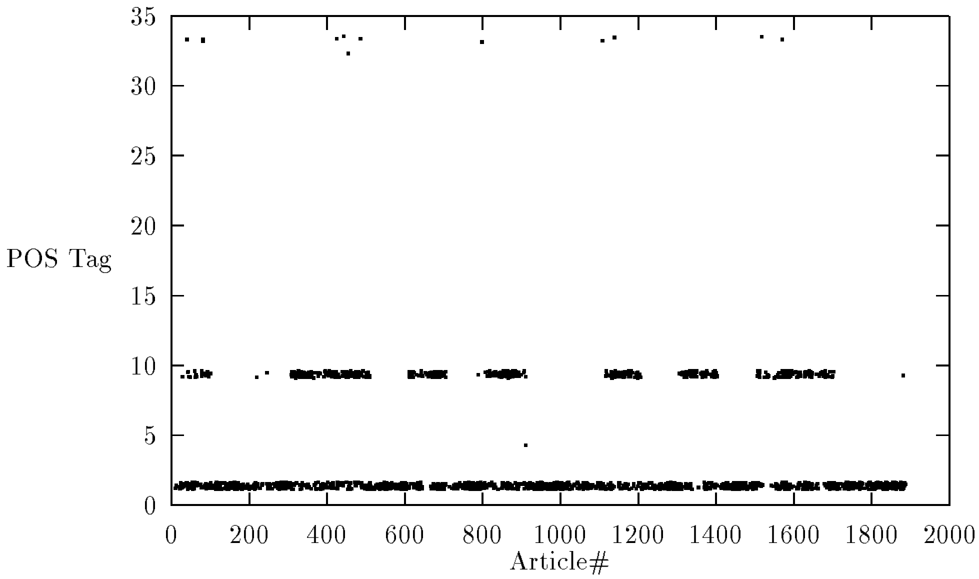

Figure 1: Distribution of Tags for the word "about" vs. Article#

Training Size(words)	Test Size(words)	Baseline	Specialized
571190	44478	97.04%	97.13%

Table 10: Performance of Baseline & Specialized Model When Tested on Consistent Subset of Development Set

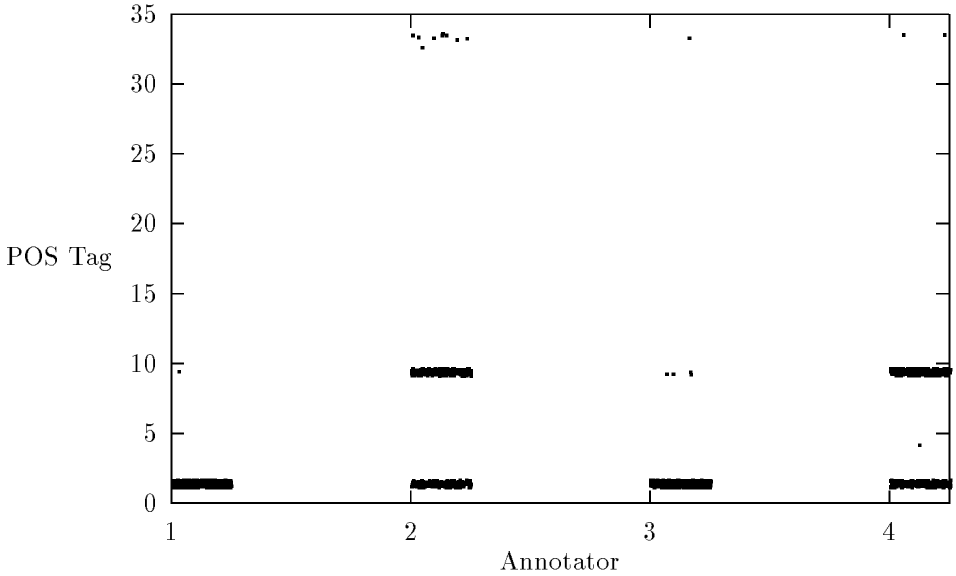

Figure 2: Distribution of Tags for the word "about" vs. Annotator

(Weischedel et al., 1993) provide the results from a battery of "tri-tag" Markov Model experiments, in which the probability $P(W,T)$ of observing a word sequence $W = \{w_1, w_2, \ldots, w_n\}$ together with a tag sequence $T = \{t_1, t_2, \ldots, t_n\}$ is given by:

$$P(T|W)P(W) = p(t_1)p(t_2|t_1) \times \prod_{i=3}^{n} p(t_i|t_{i-1}t_{i-2}) \left(\prod_{1}^{n} p(w_i|t_i) \right)$$

Furthermore, $p(w_i|t_i)$ for unknown words is computed by the following heuristic, which uses a set of 35 pre-determined endings:

$$\begin{aligned} p(w_i|t_i) &= p(\text{unknownword}|t_i) \times \\ & \quad p(\text{capitalfeature}|t_i) \times \\ & \quad p(\text{endings, hypenation}|t_i) \end{aligned}$$

This approximation works as well as the MaxEnt model, giving 85% unknown word accuracy(Weischedel et al., 1993) on the Wall St. Journal, but cannot be generalized to handle more diverse information sources. Multiplying together all the probabilities becomes less convincing of an approximation as the information sources become less independent. In contrast, the MaxEnt model combines diverse and non-local information sources without making any independence assumptions.

A POS tagger is one component in the SDT based statistical parsing system described in (Jelinek et al., 1994, Magerman, 1995). The total word accuracy on Wall St. Journal data, 96.5%(Magerman, 1995), is similar to that presented in this paper. However, the aforementioned SDT techniques require word classes(Brown et al., 1992) to help prevent data fragmentation, and a sophisticated smoothing algorithm to mitigate the effects of any fragmentation that occurs. Unlike SDT, the MaxEnt training procedure does not recursively split the data, and hence does not suffer from unreliable counts due to data fragmentation. As a result, no word classes are required and a trivial count cutoff suffices as a smoothing procedure in order to achieve roughly the same level of accuracy.

TBL is a non-statistical approach to POS tagging which also uses a rich feature representation, and performs at a total word accuracy of 96.5% and an unknown word accuracy of 85%.(Brill, 1994). The TBL representation of the surrounding word context is almost the same[7] and the TBL representation of unknown words is a superset[8] of the unknown word representation in this paper. However, since TBL is non-statistical, it does not provide probability distributions and

[7](Brill, 1994) looks at words ± 3 away from the current, whereas the feature set in this paper uses a window of ± 2.

[8](Brill, 1994) uses prefix/suffix additions and deletions, which are not used in this paper.

unlike MaxEnt, cannot be used as a probabilistic component in a larger model. MaxEnt can provide a probability for each tagging decision, which can be used in the probability calculation of any structure that is predicted over the POS tags, such as noun phrases, or entire parse trees, as in (Jelinek et al., 1994, Magerman, 1995).

Thus MaxEnt has at least one advantage over each of the reviewed POS tagging techniques. It is better able to use diverse information than Markov Models, requires less supporting techniques than SDT, and unlike TBL, can be used in a probabilistic framework. However, the POS tagging accuracy on the Penn Wall St. Journal corpus is roughly the same for all these modelling techniques. The convergence of the accuracy rate implies that either all these techniques are missing the right predictors in their representation to get the "residue", or more likely, that any corpus based algorithm on the Penn Treebank Wall St. Journal corpus will not perform much higher than 96.5% due to consistency problems.

Conclusion

The Maximum Entropy model is an extremely flexible technique for linguistic modelling, since it can use a virtually unrestricted and rich feature set in the framework of a probability model. The implementation in this paper is a state-of-the-art POS tagger, as evidenced by the 96.6% accuracy on the unseen Test set, shown in Table 11.

The model with specialized features does not perform much better than the baseline model, and further discovery or refinement of word-based features is difficult given the inconsistencies in the training data. A model trained and tested on data from a single annotator performs at .5% higher accuracy than the baseline model and should produce more consistent input for applications that require tagged text.

References

[ARP, 1994] ARPA. 1994. *Proceedings of the Human Language Technology Workshop.*

[Berger et al., 1996] Adam Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

[Brill, 1994] Eric Brill. 1994. Some Advances in Transformation-Based Part of Speech Tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, volume 1, pages 722–727.

[Brown et al., 1992] Peter F Brown, Vincent DellaPietra, Peter V deSouza, Jennifer C Lai, and Robert L Mercer. 1992. Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4).

[Darroch and Ratcliff, 1972] J. N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Linear Models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

[Della Pietra et al., 1995] Steven Della Pietra, Vincent Della Pietra, and John Lafferty. 1995. Inducing Features of Random Fields. Technical Report CMU-CS95-144, School of Computer Science, Carnegie-Mellon University.

[Jelinek et al., 1994] F Jelinek, J Lafferty, D Magerman, R Mercer, A Ratnaparkhi, and S Roukos. 1994. Decision Tree Parsing using a Hidden Derivational Model. In *Proceedings of the Human Language Technology Workshop* (ARP, 1994), pages 272–277.

[Lau et al., 1993] Ray Lau, Ronald Rosenfeld, and Salim Roukos. 1993. Adaptive Language Modeling Using The Maximum Entropy Principle. In *Proceedings of the Human Language Technology Workshop*, pages 108–113. ARPA.

[Magerman, 1995] David M. Magerman. 1995. Statistical Decision-Tree Models for Parsing. In *Proceedings of the 33rd Annual Meeting of the ACL.*

[Marcus et al., 1994] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[Merialdo, 1994] Bernard Merialdo. 1994. Tagging English Text with a Probabilistic Model. *Computational Linguistics*, 20(2):155–172.

[Ratnaparkhi et al., 1994] Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. In *Proceedings of the Human Language Technology Workshop* (ARP, 1994), pages 250–255.

Total Word Accuracy	Unknown Word Accuracy	Sentence Accuracy
96.63%	85.56%	47.51%

Table 11: Performance of Specialized Model on Unseen Test Data

[Weischedel et al., 1993] Ralph Weischedel, Marie Meteer, Richard Schwartz, Lance Ramshaw, and Jeff Palmucci. 1993. Coping With Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19(2):359–382.