

Multi-Tagging for Lexicalized-Grammar Parsing

James R. Curran
School of IT
University of Sydney
NSW 2006, Australia
james@it.usyd.edu.au

Stephen Clark
Computing Laboratory
Oxford University
Wolfson Building
Parks Road
Oxford, OX1 3QD, UK
sclark@comlab.ox.ac.uk

David Vadas
School of IT
University of Sydney
NSW 2006, Australia
dvadas1@it.usyd.edu.au

Abstract

With performance above 97% accuracy for newspaper text, part of speech (POS) tagging might be considered a solved problem. Previous studies have shown that allowing the parser to resolve POS tag ambiguity does not improve performance. However, for grammar formalisms which use more fine-grained grammatical categories, for example TAG and CCG, tagging accuracy is much lower. In fact, for these formalisms, premature ambiguity resolution makes parsing infeasible.

We describe a multi-tagging approach which maintains a suitable level of lexical category ambiguity for accurate and efficient CCG parsing. We extend this multi-tagging approach to the POS level to overcome errors introduced by automatically assigned POS tags. Although POS tagging accuracy seems high, maintaining some POS tag ambiguity in the language processing pipeline results in more accurate CCG supertagging.

1 Introduction

State-of-the-art part of speech (POS) tagging accuracy is now above 97% for newspaper text (Collins, 2002; Toutanova et al., 2003). One possible conclusion from the POS tagging literature is that accuracy is approaching the limit, and any remaining improvement is within the noise of the Penn Treebank training data (Ratnaparkhi, 1996; Toutanova et al., 2003).

So why should we continue to work on the POS tagging problem? Here we give two reasons. First, for lexicalized grammar formalisms such as TAG

and CCG, the tagging problem is much harder. Second, any errors in POS tagger output, even at 97% accuracy, can have a significant impact on components further down the language processing pipeline. In previous work we have shown that using automatically assigned, rather than gold standard, POS tags reduces the accuracy of our CCG parser by almost 2% in dependency F-score (Clark and Curran, 2004b).

CCG *supertagging* is much harder than POS tagging because the CCG tag set consists of fine-grained lexical categories, resulting in a larger tag set – over 400 CCG lexical categories compared with 45 Penn Treebank POS tags. In fact, using a state-of-the-art tagger as a front end to a CCG parser makes accurate parsing infeasible because of the high supertagging error rate.

Our solution is to use *multi-tagging*, in which a CCG supertagger can potentially assign more than one lexical category to a word. In this paper we significantly improve our earlier approach (Clark and Curran, 2004a) by adapting the forward-backward algorithm to a Maximum Entropy tagger, which is used to calculate a probability distribution over lexical categories for each word. This distribution is used to assign one or more categories to each word (Charniak et al., 1996). We report large increases in accuracy over single-tagging at only a small cost in increased ambiguity.

A further contribution of the paper is to also use multi-tagging for the POS tags, and to maintain some POS ambiguity in the language processing pipeline. In particular, since POS tags are important features for the supertagger, we investigate how supertagging accuracy can be improved by not prematurely committing to a POS tag decision. Our results first demonstrate that a surprising in-

crease in POS tagging accuracy can be achieved with only a tiny increase in ambiguity; and second that maintaining some POS ambiguity can significantly improve the accuracy of the supertagger.

The parser uses the CCG lexical categories to build syntactic structure, and the POS tags are used by the supertagger and parser as part of their statistical models. We show that using a multi-tagger for supertagging results in an effective pre-processor for CCG parsing, and that using a multi-tagger for POS tagging results in more accurate CCG supertagging.

2 Maximum Entropy Tagging

The tagger uses conditional probabilities of the form $P(y|x)$ where y is a tag and x is a local context containing y . The conditional probabilities have the following log-linear form:

$$P(y|x) = \frac{1}{Z(x)} e^{\sum_i \lambda_i f_i(x,y)} \quad (1)$$

where $Z(x)$ is a normalisation constant which ensures a proper probability distribution for each context x .

The feature functions $f_i(x,y)$ are binary-valued, returning either 0 or 1 depending on the tag y and the value of a particular contextual predicate given the context x . Contextual predicates identify elements of the context which might be useful for predicting the tag. For example, the following feature returns 1 if the current word is `the` and the tag is `DT`; otherwise it returns 0:

$$f_i(x,y) = \begin{cases} 1 & \text{if } \text{word}(x) = \text{the} \ \& \ y = \text{DT} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

`word(x) = the` is an example of a *contextual predicate*. The POS tagger uses the same contextual predicates as Ratnaparkhi (1996); the supertagger adds contextual predicates corresponding to POS tags and bigram combinations of POS tags (Curran and Clark, 2003).

Each feature f_i has an associated weight λ_i which is determined during training. The training process aims to maximise the entropy of the model subject to the constraints that the expectation of each feature according to the model matches the empirical expectation from the training data. This can be also thought of in terms of maximum likelihood estimation (MLE) for a log-linear model (Della Pietra et al., 1997). We use the L-BFGS op-

timisation algorithm (Nocedal and Wright, 1999; Malouf, 2002) to perform the estimation.

MLE has a tendency to overfit the training data. We adopt the standard approach of Chen and Rosenfeld (1999) by introducing a Gaussian prior term to the objective function which penalises feature weights with large absolute values. A parameter defined in terms of the standard deviation of the Gaussian determines the degree of smoothing.

The conditional probability of a sequence of tags, y_1, \dots, y_n , given a sentence, w_1, \dots, w_n , is defined as the product of the individual probabilities for each tag:

$$P(y_1, \dots, y_n | w_1, \dots, w_n) = \prod_{i=1}^n P(y_i | x_i) \quad (3)$$

where x_i is the context for word w_i . We use the standard approach of Viterbi decoding to find the highest probability sequence.

2.1 Multi-tagging

Multi-tagging — assigning one or more tags to a word — is used here in two ways: first, to retain ambiguity in the CCG lexical category sequence for the purpose of building parse structure; and second, to retain ambiguity in the POS tag sequence. We retain ambiguity in the lexical category sequence since a single-tagger is not accurate enough to serve as a front-end to a CCG parser, and we retain some POS ambiguity since POS tags are used as features in the statistical models of the supertagger and parser.

Charniak et al. (1996) investigated multi-POS tagging in the context of PCFG parsing. It was found that multi-tagging provides only a minor improvement in accuracy, with a significant loss in efficiency; hence it was concluded that, given the particular parser and tagger used, a single-tag POS tagger is preferable to a multi-tagger. More recently, Watson (2006) has revisited this question in the context of the RASP parser (Briscoe and Carroll, 2002) and found that, similar to Charniak et al. (1996), multi-tagging at the POS level results in a small increase in parsing accuracy but at some cost in efficiency.

For lexicalized grammars, such as CCG and TAG, the motivation for using a multi-tagger to assign the elementary structures (supertags) is more compelling. Since the set of supertags is typically much larger than a standard POS tag set, the tagging problem becomes much harder. In

fact, when using a state-of-the-art single-tagger, the per-word accuracy for CCG supertagging is so low (around 92%) that wide coverage, high accuracy parsing becomes infeasible (Clark, 2002; Clark and Curran, 2004a). Similar results have been found for a highly lexicalized HPSG grammar (Prins and van Noord, 2003), and also for TAG. As far as we are aware, the only approach to successfully integrate a TAG supertagger and parser is the Lightweight Dependency Analyser of Bangalore (2000). Hence, in order to perform effective full parsing with these lexicalized grammars, the tagger front-end must be a multi-tagger (given the current state-of-the-art).

The simplest approach to CCG supertagging is to assign all categories to a word which the word was seen with in the data. This leaves the parser the task of managing the very large parse space resulting from the high degree of lexical category ambiguity (Hockenmaier and Steedman, 2002; Hockenmaier, 2003). However, one of the original motivations for supertagging was to significantly reduce the syntactic ambiguity before full parsing begins (Bangalore and Joshi, 1999). Clark and Curran (2004a) found that performing CCG supertagging prior to parsing can significantly increase parsing efficiency with no loss in accuracy.

Our multi-tagging approach follows that of Clark and Curran (2004a) and Charniak et al. (1996): assign all categories to a word whose probabilities are within a factor, β , of the probability of the most probable category for that word:

$$\mathcal{C}_i = \{c \mid P(C_i = c|S) > \beta P(C_i = c_{\max}|S)\}$$

\mathcal{C}_i is the set of categories assigned to the i th word; C_i is the random variable corresponding to the category of the i th word; c_{\max} is the category with the highest probability of being the category of the i th word; and S is the sentence. One advantage of this adaptive approach is that, when the probability of the highest scoring category is much greater than the rest, no extra categories will be added.

Clark and Curran (2004a) propose a simple method for calculating $P(C_i = c|S)$: use the word and POS features in the local context to calculate the probability and ignore the previously assigned categories (the *history*). However, it is possible to incorporate the history in the calculation of the tag probabilities. A greedy approach is to use the locally highest probability history as a feature, which avoids any summing over alternative histories. Alternatively, there is a well-known

dynamic programming algorithm — the forward backward algorithm — which efficiently calculates $P(C_i = c|S)$ (Charniak et al., 1996).

The multitagger uses the following conditional probabilities:

$$P(y_i|w_{1,n}) = \sum_{y_{1,i-1}, y_{i+1,n}} P(y_i, y_{1,i-1}, y_{i+1,n}|w_{1,n})$$

where $x_{i,j} = x_i, \dots, x_j$. Here y_i is to be thought of as a fixed category, whereas y_j ($j \neq i$) varies over the possible categories for word j . In words, the probability of category y_i , given the sentence, is the sum of the probabilities of all sequences containing y_i . This sum is calculated efficiently using the forward-backward algorithm:

$$P(C_i = c|S) = \alpha_i(c)\beta_i(c) \quad (4)$$

where $\alpha_i(c)$ is the total probability of all the category sub-sequences that end at position i with category c ; and $\beta_i(c)$ is the total probability of all the category sub-sequences through to the end which start at position i with category c .

The standard description of the forward-backward algorithm, for example Manning and Schutze (1999), is usually given for an HMM-style tagger. However, it is straightforward to adapt the algorithm to the Maximum Entropy models used here. The forward-backward algorithm we use is similar to that for a Maximum Entropy Markov Model (Lafferty et al., 2001).

POS tags are very informative features for the supertagger, which suggests that using a multi-POS tagger may benefit the supertagger (and ultimately the parser). However, it is unclear whether multi-POS tagging will be useful in this context, since our single-tagger POS tagger is highly accurate: over 97% for WSJ text (Curran and Clark, 2003). In fact, in Clark and Curran (2004b) we report that using automatically assigned, as opposed to gold-standard, POS tags as features results in a 2% loss in parsing accuracy. This suggests that retaining some ambiguity in the POS sequence may be beneficial for supertagging and parsing accuracy. In Section 4 we show this is the case for supertagging.

3 CCG Supertagging and Parsing

Parsing using CCG can be viewed as a two-stage process: first assign lexical categories to the words in the sentence, and then combine the categories

The WSJ is a paper that I enjoy reading
 $\overline{NP/N}$ \overline{N} $\overline{(S[dcl]\backslash NP)/NP}$ $\overline{NP/N}$ \overline{N} $\overline{(NP\backslash NP)/(S[dcl]/NP)}$ \overline{NP} $\overline{(S[dcl]\backslash NP)/(S[ng]\backslash NP)}$ $\overline{(S[ng]\backslash NP)/NP}$

Figure 1: Example sentence with CCG lexical categories.

together using CCG’s combinatory rules.¹ We perform stage one using a supertagger.

The set of lexical categories used by the supertagger is obtained from CCGbank (Hockenmaier, 2003), a corpus of CCG normal-form derivations derived semi-automatically from the Penn Treebank. Following our earlier work, we apply a frequency cutoff to the training set, only using those categories which appear at least 10 times in sections 02-21, which results in a set of 425 categories. We have shown that the resulting set has very high coverage on unseen data (Clark and Curran, 2004a). Figure 1 gives an example sentence with the CCG lexical categories.

The parser is described in Clark and Curran (2004b). It takes POS tagged sentences as input with each word assigned a set of lexical categories. A packed chart is used to efficiently represent all the possible analyses for a sentence, and the CKY chart parsing algorithm described in Steedman (2000) is used to build the chart. A log-linear model is used to score the alternative analyses.

In Clark and Curran (2004a) we described a novel approach to integrating the supertagger and parser: start with a very restrictive supertagger setting, so that only a small number of lexical categories is assigned to each word, and only assign more categories if the parser cannot find a spanning analysis. This strategy results in an efficient and accurate parser, with speeds up to 35 sentences per second. Accurate supertagging at low levels of lexical category ambiguity is therefore particularly important when using this strategy.

We found in Clark and Curran (2004b) that a large drop in parsing accuracy occurs if automatically assigned POS tags are used throughout the parsing process, rather than gold standard POS tags (almost 2% F-score over labelled dependencies). This is due to the drop in accuracy of the supertagger (see Table 3) and also the fact that the log-linear parsing model uses POS tags as features. The large drop in parsing accuracy demonstrates that improving the performance of POS tag-

¹See Steedman (2000) for an introduction to CCG, and see Hockenmaier (2003) for an introduction to wide-coverage parsing using CCG.

TAGS/WORD	β	WORD ACC	SENT ACC
1.00	1	96.7	51.8
1.01	0.8125	97.1	55.4
1.05	0.2969	98.3	70.7
1.10	0.1172	99.0	80.9
1.20	0.0293	99.5	89.3
1.30	0.0111	99.6	91.7
1.40	0.0053	99.7	93.2
4.23	0	99.8	94.8

Table 1: POS tagging accuracy on Section 00 for different levels of ambiguity.

gers is still an important research problem. In this paper we aim to reduce the performance drop of the supertagger by maintaining some POS ambiguity through to the supertagging phase. Future work will investigate maintaining some POS ambiguity through to the parsing phase also.

4 Multi-tagging Experiments

We performed several sets of experiments for POS tagging and CCG supertagging to explore the trade-off between ambiguity and tagging accuracy. For both POS tagging and supertagging we varied the average number of tags assigned to each word, to see whether it is possible to significantly increase tagging accuracy with only a small increase in ambiguity. For CCG supertagging, we also compared multi-tagging approaches, with a fixed category ambiguity of 1.4 categories per word.

All of the experiments used Section 02-21 of CCGbank as training data, Section 00 as development data and Section 23 as final test data. We evaluate both per-word tag accuracy and sentence accuracy, which is the percentage of sentences for which every word is tagged correctly. For the multi-tagging results we consider the word to be tagged correctly if the correct tag appears in the set of tags assigned to the word.

4.1 Results

Table 1 shows the results for multi-POS tagging for different levels of ambiguity. The row corresponding to 1.01 tags per word shows that adding

METHOD	GOLD POS		AUTO POS	
	WORD	SENT	WORD	SENT
single	92.6	36.8	91.5	32.7
noseq	96.2	51.9	95.2	46.1
best hist	97.2	63.8	96.3	57.2
fwdbwd	97.9	72.1	96.9	64.8

Table 2: Supertagging accuracy on Section 00 using different approaches with multi-tagger ambiguity fixed at 1.4 categories per word.

TAGS/ WORD	β	GOLD POS		AUTO POS	
		WORD	SENT	WORD	SENT
1.0	1	92.6	36.8	91.5	32.7
1.2	0.1201	96.8	63.4	95.8	56.5
1.4	0.0337	97.9	72.1	96.9	64.8
1.6	0.0142	98.3	76.4	97.5	69.3
1.8	0.0074	98.4	78.3	97.7	71.0
2.0	0.0048	98.5	79.4	97.9	72.5
2.5	0.0019	98.7	80.6	98.1	74.3
3.0	0.0009	98.7	81.4	98.3	75.6
12.5	0	98.9	82.3	98.8	80.1

Table 3: Supertagging accuracy on Section 00 for different levels of ambiguity.

even a tiny amount of ambiguity (1 extra tag in every 100 words) gives a reasonable improvement, whilst adding 1 tag in 20 words, or approximately one extra tag per sentence on the WSJ, gives a significant boost of 1.6% word accuracy and almost 20% sentence accuracy.

The bottom row of Table 1 gives an upper bound on accuracy if the maximum ambiguity is allowed. This involves setting the β value to 0, so all feasible tags are assigned. Note that the performance gain is only 1.6% in sentence accuracy, compared with the previous row, at the cost of a large increase in ambiguity.

Our first set of CCG supertagging experiments compared the performance of several approaches. In Table 2 we give the accuracies when using gold standard POS tags, and also POS tags automatically assigned by our POS tagger described above. Since POS tags are important features for the supertagger maximum entropy model, erroneous tags have a significant impact on supertagging accuracy.

The *single* method is the single-tagger supertagger, which at 91.5% per-word accuracy is too inaccurate for use with the CCG parser. The remaining rows in the table give multi-tagger results for a cat-

egory ambiguity of 1.4 categories per word. The *noseq* method, which performs significantly better than *single*, does not take into account the previously assigned categories. The *best hist* method gains roughly another 1% in accuracy over *noseq* by taking the greedy approach of using only the two most probable previously assigned categories. Finally, the full forward-backward approach described in Section 2.1 gains roughly another 0.6% by considering all possible category histories. We see the largest jump in accuracy just by returning multiple categories. The other more modest gains come from producing progressively better models of the category sequence.

The final set of supertagging experiments in Table 3 demonstrates the trade-off between ambiguity and accuracy. Note that the ambiguity levels need to be much higher to produce similar performance to the POS tagger and that the upper bound case ($\beta = 0$) has a very high average ambiguity. This is to be expected given the much larger CCG tag set.

5 Tag uncertainty throughout the pipeline

Tables 2 and 3 show that supertagger accuracy when using gold-standard POS tags is typically 1% higher than when using automatically assigned POS tags. Clearly, correct POS tags are important features for the supertagger.

Errors made by the supertagger can multiply out when incorrect lexical categories are passed to the parser, so a 1% increase in lexical category error can become much more significant in the parser evaluation. For example, when using the dependency-based evaluation in Clark and Curran (2004b), getting the lexical category wrong for a ditransitive verb automatically leads to three dependencies in the output being incorrect.

We have shown that multi-tagging can significantly increase the accuracy of the POS tagger with only a small increase in ambiguity. What we would like to do is maintain some degree of POS tag ambiguity and pass multiple POS tags through to the supertagging stage (and eventually the parser). There are several ways to encode multiple POS tags as features. The simplest approach is to treat all of the POS tags as binary features, but this does not take into account the uncertainty in each of the alternative tags. What we need is a way of incorporating probability information into the Maximum Entropy supertagger.

6 Real-values in ME models

Maximum Entropy (ME) models, in the NLP literature, are typically defined with binary features, although they do allow real-valued features. The only constraint comes from the optimisation algorithm; for example, GIS only allows non-negative values. Real-valued features are commonly used with other machine learning algorithms.

Binary features suffer from certain limitations of the representation, which make them unsuitable for modelling some properties. For example, POS taggers have difficulty determining if capitalised, sentence initial words are proper nouns. A useful way to model this property is to determine the ratio of capitalised and non-capitalised instances of a particular word in a large corpus and use a real-valued feature which encodes this ratio (Vadas and Curran, 2005). The only way to include this feature in a binary representation is to discretize (or bin) the feature values. For this type of feature, choosing appropriate bins is difficult and it may be hard to find a discretization scheme that performs optimally.

Another problem with discretizing feature values is that it imposes artificial boundaries to define the bins. For the example above, we may choose the bins $0 \leq x < 1$ and $1 \leq x < 2$, which separate the values 0.99 and 1.01 even though they are close in value. At the same time, the model does not distinguish between 0.01 and 0.99 even though they are much further apart.

Further, if we have not seen cases for the bin $2 \leq x < 3$, then the discretized model has no evidence to determine the contribution of this feature. But for the real-valued model, evidence supporting $1 \leq x < 2$ and $3 \leq x < 4$ provides evidence for the missing bin. Thus the real-valued model generalises more effectively.

One issue that is not addressed here is the interaction between the Gaussian smoothing parameter and real-valued features. Using the same smoothing parameter for real-valued features with vastly different distributions is unlikely to be optimal. However, for these experiments we have used the same value for the smoothing parameter on all real-valued features. This is the same value we have used for the binary features.

7 Multi-POS Supertagging Experiments

We have experimented with four different approaches to passing multiple POS tags as features

through to the supertagger. For the later experiments, this required the existing binary-valued framework to be extended to support real values. The level of POS tag ambiguity was varied between 1.05 and 1.3 POS tags per word on average. These results are shown in Table 4.

The first approach is to treat the multiple POS tags as binary features (*bin*). This simply involves adding the multiple POS tags for each word in both the training and test data. Every assigned POS tag is treated as a separate feature and considered equally important regardless of its uncertainty. Here we see a minor increase in performance over the original supertagger at the lower levels of POS ambiguity. However, as the POS ambiguity is increased, the performance of the binary-valued features decreases and is eventually worse than the original supertagger. This is because at the lowest levels of ambiguity the extra POS tags can be treated as being of similar reliability. However, at higher levels of ambiguity many POS tags are added which are unreliable and should not be trusted equally.

The second approach (*split*) uses real-valued features to model some degree of uncertainty in the POS tags, dividing the POS tag probability mass evenly among the alternatives. This has the effect of giving smaller feature values to tags where many alternative tags have been assigned. This produces similar results to the binary-valued features, again performing best at low levels of ambiguity.

The third approach (*invrank*) is to use the inverse rank of each POS tag as a real-valued feature. The inverse rank is the reciprocal of the tag’s rank ordered by decreasing probability. This method assumes the POS tagger correctly orders the alternative tags, but does not rely on the probability assigned to each tag. Overall, *invrank* performs worse than *split*.

The final and best approach is to use the probabilities assigned to each alternative tag as real-valued features:

$$f_i(x, y) = \begin{cases} p(\text{POS}(x) = \text{NN}) & \text{if } y = \text{NP} \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

This model gives the best performance at 1.1 POS tags per-word average ambiguity. Note that, even when using the probabilities as features, only a small amount of additional POS ambiguity is required to significantly improve performance.

METHOD	POS AMB	WORD	SENT
orig	1.00	96.9	64.8
bin	1.05	97.3	67.7
	1.10	97.3	66.3
	1.20	97.0	63.5
	1.30	96.8	62.1
split	1.05	97.4	68.5
	1.10	97.4	67.9
	1.20	97.3	67.0
	1.30	97.2	65.1
prob	1.05	97.5	68.7
	1.10	97.5	69.1
	1.20	97.5	68.7
	1.30	97.5	68.7
invrnk	1.05	97.3	68.0
	1.10	97.4	68.0
	1.20	97.3	67.1
	1.30	97.3	67.1
gold	-	97.9	72.1

Table 4: Multi-POS supertagging on Section 00 with different levels of POS ambiguity and using different approaches to POS feature encoding.

Table 5 shows our best performance figures for the multi-POS supertagger, against the previously described method using both gold standard and automatically assigned POS tags.

Table 6 uses the Section 23 test data to demonstrate the improvement in supertagging when moving from single-tagging (*single*) to simple multi-tagging (*noseq*); from simple multi-tagging to the full forward-backward algorithm (*fwdbwd*); and finally when using the probabilities of multiply-assigned POS tags as features (MULTI-POS column). All of these multi-tagging experiments use an ambiguity level of 1.4 categories per word and the last result uses POS tag ambiguity of 1.1 tags per word.

8 Conclusion

The NLP community may consider POS tagging to be a solved problem. In this paper, we have suggested two reasons why this is not the case. First, tagging for lexicalized-grammar formalisms, such as CCG and TAG, is far from solved. Second, even modest improvements in POS tagging accuracy can have a large impact on the performance of downstream components in a language processing pipeline.

TAGS/ WORD	AUTO POS		MULTI POS		GOLD POS	
	WORD	SENT	WORD	SENT	WORD	SENT
1.0	91.5	32.7	91.9	34.3	92.6	36.8
1.2	95.8	56.5	96.3	59.2	96.8	63.4
1.4	96.9	64.8	97.5	67.0	97.9	72.1
1.6	97.5	69.3	97.9	73.3	98.3	76.4
1.8	97.7	71.0	98.2	76.1	98.4	78.3
2.0	97.9	72.5	98.4	77.4	98.5	79.4
2.5	98.1	74.3	98.5	78.7	98.7	80.6
3.0	98.3	75.6	98.6	79.7	98.7	81.4

Table 5: Best multi-POS supertagging accuracy on Section 00 using POS ambiguity of 1.1 and the probability real-valued features.

METHOD	AUTO POS	MULTI POS	GOLD POS
single	92.0	-	93.3
noseq	95.4	-	96.6
fwdbwd	97.1	97.7	98.2

Table 6: Final supertagging results on Section 23.

We have developed a novel approach to maintaining tag ambiguity in language processing pipelines which avoids premature ambiguity resolution. The tag ambiguity is maintained by using the forward-backward algorithm to calculate individual tag probabilities. These probabilities can then be used to select multiple tags and can also be encoded as real-valued features in subsequent statistical models.

With this new approach we have increased POS tagging accuracy significantly with only a tiny ambiguity penalty and also significantly improved on previous CCG supertagging results. Finally, using POS tag probabilities as real-valued features in the supertagging model, we demonstrated performance close to that obtained with gold-standard POS tags. This will significantly improve the robustness of the parser on unseen text.

In future work we will investigate maintaining tag ambiguity further down the language processing pipeline and exploiting the uncertainty from previous stages. In particular, we will incorporate real-valued POS tag and lexical category features in the statistical parsing model. Another possibility is to investigate whether similar techniques can improve other tagging tasks, such as Named Entity Recognition.

This work can be seen as part of the larger goal of maintaining ambiguity and exploiting un-

certainty throughout language processing systems (Roth and Yih, 2004), which is important for coping with the compounding of errors that is a significant problem in language processing pipelines.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful feedback. This work has been supported by the Australian Research Council under Discovery Project DP0453131.

References

- Srinivas Bangalore and Aravind Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.
- Srinivas Bangalore. 2000. A lightweight dependency analyser for partial parsing. *Natural Language Engineering*, 6(2):113–138.
- Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC Conference*, pages 1499–1504, Las Palmas, Gran Canaria.
- Eugene Charniak, Glenn Carroll, John Adcock, Anthony Cassandra, Yoshihiko Gotoh, Jeremy Katz, Michael Littman, and John McCann. 1996. Taggers for parsers. *Artificial Intelligence*, 85:45–57.
- Stanley Chen and Ronald Rosenfeld. 1999. A Gaussian prior for smoothing maximum entropy models. Technical report, Carnegie Mellon University, Pittsburgh, PA.
- Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING-04*, pages 282–288, Geneva, Switzerland.
- Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, pages 104–111, Barcelona, Spain.
- Stephen Clark. 2002. A supertagger for Combinatory Categorical Grammar. In *Proceedings of the TAG+ Workshop*, pages 19–24, Venice, Italy.
- Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the EMNLP Conference*, pages 1–8, Philadelphia, PA.
- James R. Curran and Stephen Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Meeting of the EACL*, pages 91–98, Budapest, Hungary.
- Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. 1997. Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 19(4):380–393.
- Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorical Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.
- Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorical Grammar*. Ph.D. thesis, University of Edinburgh.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williams College, MA.
- Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.
- Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.
- Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer, New York, USA.
- Robbert Prins and Gertjan van Noord. 2003. Reinforcing parser preferences through tagging. *Traitement Automatique des Langues*, 44(3):121–139.
- Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.
- D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.
- Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.
- Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the HLT/NAACL conference*, pages 252–259, Edmonton, Canada.
- David Vadas and James R. Curran. 2005. Tagging unknown words with raw text features. In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 32–39, Sydney, Australia.
- Rebecca Watson. 2006. Part-of-speech tagging models for parsing. In *Proceedings of the Computational Linguistics in the UK Conference (CLUK-06)*, Open University, Milton Keynes, UK.