Faster parsing and supertagging model estimation

Jonathan K. Kummerfeld[a] **James R. Curran**[a] **Jessika Roesner**[b]
School of Information Technologies[a] Department of Computer Science[b]
University of Sydney University of Texas at Austin
NSW 2006, Australia Austin, TX, USA
{jkum0593,james}@it.usyd.edu.au jessi@mail.utexas.edu

Abstract

Parsers are often the bottleneck for data acquisition, processing text too slowly to be widely applied. One way to improve the efficiency of parsers is to construct more confident statistical models. More training data would enable the use of more sophisticated features and also provide more evidence for current features, but gold standard annotated data is limited and expensive to produce.

We demonstrate faster methods for training a supertagger using hundreds of millions of automatically annotated words, constructing statistical models that further constrain the number of derivations the parser must consider. By introducing new features and using an automatically annotated corpus we are able to double parsing speed on Wikipedia and the Wall Street Journal, and gain accuracy slightly when parsing Section 00 of the Wall Street Journal.

1 Introduction

Many systems in NLP for tasks such as Question–Answering rely on large volumes of data. Parsers are a useful means of extracting extra information about text, providing the syntactic structure of sentences. However, when they are the bottleneck in the data acquisition phase of a system simple solutions are to use less data, or not use a parser at all. If we can improve the speed of parsers this will be unnecessary.

For lexicalised grammars such as Combinatory Categorial Grammar (CCG) (Steedman, 2000) the step in which words are labelled with lexical categories has great influence on parsing speed and accuracy. In these formalisms, the labels chosen constrain the set of possible derivations so much that the process of choosing them, *supertagging*, is described as 'almost parsing' (Joshi and Bangalore, 1994). If the supertagger is more accurate it can further constrain the set of possible derivations by supplying fewer categories, leaving the parser with less to do.

One means of improving the supertagger's statistical model of language is to provide more evidence, in this case, more annotated text. However, creating a significant amount of extra gold standard annotated text is not feasible. An alternative approach is 'semi-supervised training', in which a small set of annotated data and a much larger set of unannotated data is used. Training a system directly on its own output, 'self-training', is not normally effective (Clark et al., 2003), but recently McClosky et al. (2006) demonstrated that parser output can be made useful for retraining by the application of a reranker.

To enable the use of more training data we have parallelised the C&C parser's supertagger training process and implemented perceptron–based algorithms for parameter estimation. In the process of this work we also modified the C&C parser's use of a particular CCG rule, based on observations of its behaviour. Our unlabeled training data was part of the English section of Wikipedia, consisting of 47 million sentences. We used the C&C parser to label the data with supertags, producing training data that could then be used to retrain its supertagger. The reasoning behind the use of this data is that the supertagger will provide categories that the parser is most likely to use in a spanning analysis.

Models trained on WSJ and Wikipedia data parsed sentences up to twice as fast, without decreasing accuracy. And the perceptron–based algorithms enabled the use of much larger data sets, without loss in parsing speed or accuracy.

2 Background

Parsing is the process of analysing a set of tokens and extracting syntactic structure. In the context of natural language we are faced with several challenges, including ambiguous sentences that are context sensitive and a grammar that is unknown and constantly changing.

Two main classes of grammars have been used to try to understand and model natural language, phrasal and lexicalised grammars. Phrasal grammars generally define a small set of labels that capture the syntactic behaviour of a word in a sentence, such as noun and adverb, and then use a large set of rules to construct a phrase structure tree in which the leaves are the words and the internal nodes are applications of rules. Lexicalised grammars provide a much larger set of categories for lexical items and only a few rules. The categories provide a more detailed description of a word's purpose in a sentence, while the rules are simple descriptions of how pairs of categories can combine to form the parse tree.

We use the lexicalised grammar formalism Combinatory Categorial Grammar (CCG) (Steedman, 2000). In CCG, there are two types of categories, *atomic*, which are one of S, N, NP and PP, and *complex*, which contain two parts, an argument and a result, denoted by either 'Result / Argument' or 'Result \ Argument', where the slashes indicate whether the Argument is expected to lie to the right or left respectively, and the result and argument are categories themselves. To form a derivation for English sentences these categories are combined according to seven rules, forward and backward application, forward and backward composition, backward crossed composition, type raising and coordination.

Figure 1 presents two example CCG derivations. In both examples, the line directly beneath the words contains the category that was assigned to each word, NP for 'I', (S\NP)/NP for 'ate' and so on. The lines that follow show a series of rule applications, building up the parse tree.

The lines with a $>$ sign at the end indicate forward application, which occurs when a complex category is of the form 'Result / Argument' and its argument is the same as the category to its right. The lines with a $<$ sign at the end are instances of backward application, which works in the same way, but in the opposite direction.

Note in particular the change of tag for 'with' in the two examples and its affect on the subsequent rule applications. The decision made by the supertagger effectively decides which analysis will be found, or if both are provided the parser must consider more possible derivations.

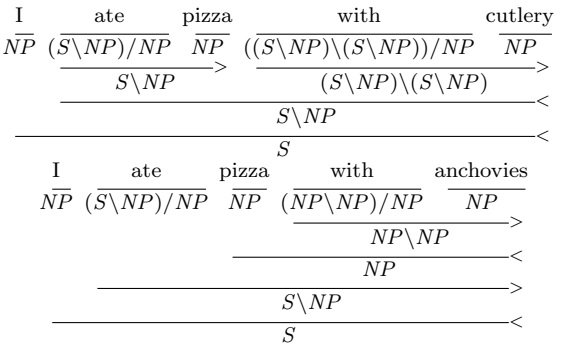

Figure 1: Two CCG derivations with PP ambiguity.

The CCG parser and associated supertagger we have used is the C&C parser (Clark and Curran, 2003; Clark and Curran, 2007b). The supertagger applies categories to words using the forward backward algorithm, and the parser forms a derivation by applying the Cocke–Younger–Kasami (CKY) chart parsing algorithm (Younger, 1967; Kasami, 1967) and dynamic programming.

2.1 Supertagging

Supertags were first proposed by Joshi and Bangalore (1994) for Lexicalized Tree-Adjoining Grammar (LTAG). Like POS tags, supertags are assigned to each word in the sentence prior to parsing, but supertags contain much more detailed syntactic information. This leads to tag sets that are up to two orders of magnitude larger. The first supertaggers gave each word a single tag based only on the POS tags in the local context and had an accuracy below 90% (Chandrasekar and Bangalore, 1997b). While this is not accurate enough for incorporation into a wide–coverage parser, it was enough to be useful in an information retrieval system (Chandrasekar and Bangalore, 1997a), attaining an F-score of 92% for filtering out irrelevant documents. Accuracy was improved by the use of multitaggers (Chen et al., 1999), but as more tags are supplied the parsing efficiency decreases (Chen et al., 2002), demon-

strating that lexical ambiguity is an important factor in parsing complexity (Sarkar et al., 2000).

Supertagging was first applied to CCG by Clark (2002). Rather than defining a fixed number of tags to be produced per word, the CCG supertagger includes all tags with probabilities within some factor, β, of the most probable tag. Also, during parsing the β value starts high, and if a derivation is not found it is progressively decreased. This provides similar speed benefits to a single tagger, but without a loss in coverage. Previous attempts to expand the feature set used by the CCG supertagger were unsuccessful because of data sparseness issues (Cooper, 2007).

Perhaps the closest previous work was by (Sarkar, 2007), who incorporated a supertagger with a full LTAG parser, and demonstrated improved efficiency through the use of training data annotated by the parser. This led to higher performance than entirely supervised training methods.

2.2 Semi-supervised training

One of the first demonstrations of semi-supervised training in NLP was the use of a 'co-training' method by Yarowsky (1995), who achieved 96% accuracy on a word sense disambiguation task. A similar method was subsequently applied to statistical parsing by Sarkar (2001), leading to a 9% increase in F-score for an LTAG parser.

Co-training relies upon two independent views of the data to construct models that can inform each other. Another method of semi-supervised training is to apply a re-ranker to the output of a system to generate new training data. By applying a re-ranker to the output of a parser McClosky et al. (2006) were able to improve on the best result for Wall Street Journal parsing by 0.8%, but with no significant change in efficiency.

2.3 Perceptron Algorithms

The perceptron is an online classification method that was proposed by Rosenblatt (1958). However, the algorithm only converges for linearly separable datasets. Recently Freund and Schapire (1999) developed the Averaged Perceptron (AP), which stores all weight vectors during training and combines them in a weighted majority vote to create the final weight vector. This variation led to performance competitive with modern techniques, such as Support Vector Machines, on a handwritten digit classification task.

Another recent variation is the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003), which adjusts the weight vector only enough to cause the current instance to be correctly classified with a specified margin. This method generally has a lower relative error than the standard perceptron but makes more updates.

Collins (2002) showed that applying these methods to tasks in NLP produced better performance than maximum entropy models. Specifically, using a voted perceptron and trigram features for training, a Viterbi based system achieved an F-score of 93.53% for NP Chunking and an error rate of 2.93% for POS tagging, compared to 92.65% and 3.28% respectively for a similar system trained with a maximum entropy model.

Collins and Roark (2004) applied these methods to parsing, using an incremental beam search parser. The parser performed similarly to another based on a generative model, with an F-score of 87.8% for data with gold standard POS tags, and 86.6% for tags generated by a tagger. Similar methods were recently applied to the C&C parser (Clark and Curran, 2007a), leading to similar performance to a log-linear model, but with much lower system requirements. Zhang et al. (2009) used the averaged perceptron algorithm to train an HPSG supertagger, with similar improvements to training time as described here.

3 Implementation

The parser uses the CKY algorithm to construct the 'chart', an efficient representation of all possible analyses for a sentence. The most probable derivation is found using the Viterbi algorithm and probabilities are calculated based on a conditional log-linear model.

The supertagger uses a maximum entropy based model to assign a set of possible lexical categories to each word in the sentence. The main aspect of the tagging process relevant to this work is the role of beta levels.

If the supertagger assigns only one category to each word its accuracy is too low to be effectively incorporated into a parser. By multitagging we can make the supertagger more accurate, but

at the cost of speed as the parser must consider larger sets of possible categories. The beta levels define cutoffs for multitagging based on the probabilities from the maximum entropy model. If the parser is unable to form a spanning analysis the beta level is decreased and the supertagger is rerun to retrieve larger sets of supertags.

These levels have a large influence on parsing accuracy and speed. Accuracy varies because the set of possible derivations increases as more tags are supplied, leading the parser to choose different derivations at different levels. Speed varies as the time spent attempting to form a parse increases as more tags are supplied. Also, for sentences that are not parsed at the first level each attempt at another level requires more time.

The initial feature set used for tagging included unigrams of POS tags and words and bigrams of POS tags, all in a five word window surrounding the word being tagged. The weights for these features were estimated on a single CPU using either Generalised Iterative Scaling (GIS) (Darroch and Ratcliff, 1972) or the Broyden-Fletcher-Goldfarb-Shanno method (BFGS) (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). Here we consider two other algorithms, a parallelised form of the process, and a range of extra features.

3.1 Averaged Perceptron

The standard multi-class perceptron maintains a matrix of weights, containing a row for each attribute and a column for each class. When all attributes are binary valued the class is assigned by ignoring all rows for attributes that do not occur and determining which column has the greatest sum. During training the class that corresponds to the column with the greatest sum is compared to the true class and if it is correct no change is made. If the predicted class is incorrect the weights are updated by subtracting 1.0 from all weights for the predicted class and adding 1.0 to all weights for the true class. The averaged perceptron follows the same algorithm, but returns the average of the weight matrix over the course of training, rather than its final state.

3.2 Margin Infused Relaxed Algorithm

MIRA also follows the standard multi-class perceptron algorithm, but applies a different update method. The intention is to make the smallest change to the weights such that the correct class is produced by a given margin. We use a slight variation of the update function defined by Crammer and Singer (2003), expressed as follows:

$$\min\left(\max, \frac{\text{margin}+\sum_f p_w - t_w}{|\text{features}|(1+\frac{1}{n_{above}})}\right)$$

where $margin$ is the absolute difference that will be created between the true classification and those that previously ranked above it, the sum is over all features, p_w and t_w are the weights associated with the feature f for the predicted and true classes respectively, $|features|$ is the number of active features, and n_{above} is the number of categories that had higher sums than the correct category. The constant max is introduced to prevent a single event causing extremely large changes to the model.

We have also applied shuffling between iterations of the algorithm to prevents the model from overfitting to the particular order of training instances.

3.3 Parallelisation

To enable the use of more data and features we increased the amount of accessible RAM and processing power by parallelising the supertagger training using the Message Passing Interface (MPI) and the MapReduce library MRMPI[1].

The first stages of supertagging are feature extraction and aggregation. Extraction was parallelised by dividing the data amongst a set of computers and having each one extract the features in its set. Aggregation is necessary to determine overall frequencies for features, and to reorder the features to maximise efficiency. For the aggregation process we used the MRMPI library.

For weight estimation using maximum entropy methods the main calculations are sums of weights across all training instances. The parallel versions of GIS and BFGS differ in three main ways. First, the data is divided between a set of computers. Second, sums are calculated across all computers to determine necessary changes to weights. And third, after each update the changes are distributed to all nodes.

[1] http://www.sandia.gov/ sjplimp/mapreduce.html

The perceptron methods adjust the weights based on each training instance individually and so the parallelisation above was not applicable. The training instances are still distributed across a cluster of computers, but only one computer is working at a time, adjusting the weights based on each of its instances and then passing the weights to the next node. This saves time by removing the cost of loading the training instances from disk when there are too many to fit in RAM.

3.4 Blocking Excess Backward Composition

In the process of debugging the parser, we investigated the number of times particular pairs of categories were combined. We were surprised to discover that a very large number of backward compositions were being performed in the chart, even though backward composition rarely occurred in the parser output (or in the gold standard itself).

Backward composition is normally used for non-constituent coordination between pairs of type-raised categories, but the parser was also using it for combining non-type-raised and type-raised categories. This is an instance where the Eisner (1996) normal form constraints have failed to stop non-normal form derivations, because Eisner's constraints were not designed to work with type-raising. We added a constraint that only allows backward composition to occur if both children are type-raised.

4 Methodology

4.1 Data

Evaluation has been performed using Section 00 of CCGBank, a translation of the Penn Treebank to CCG (Hockenmaier, 2003). Sections 02-21 were used as training data and are simply referred to as WSJ in the following section. The raw Wikipedia data was tokenised using Punkt (Kiss and Strunk, 2006) and the NLTK tokeniser (Bird et al., 2009), and parsed using the C&C parser and models version 1.02[2]. The WSJ sentences had an average length of 23.5 words and a variance of 122.0 while the Wikipedia sentences had an average length of 21.7 words and a variance of 151.0.

[2]http://svn.ask.it.usyd.edu.au/trac/candc

4.2 Evaluation

For a given beta level the number of categories assigned to each word by the supertagger will vary greatly between models. This presents a problem because as described in Section 3 the number of categories assigned has a large influence on parsing speed and accuracy. To fairly compare the models presented here we have tuned all five beta levels on the test set to ensure all models assign the same number of categories per word on average. When testing on Wikipedia text we have used the same beta levels as for WSJ and included the ambiguity this leads to in the tables of results.

F-scores are calculated based on comparisons with gold standard labelled dependencies for Section 00. Category accuracies (Cat.) are for the first beta level only, and are the percentage of words in the sentence that were assigned a tagset that includes the correct category. Category accuracy for Wikipedia was measured over three hundred Wikipedia sentences that were hand–annotated with supertags and grammatical relations, containing 6696 word–category pairs (Clark et al., 2009).

Statistical significance testing was used to determine if changes in performance were meaningful. The test applied reports whether two sets of responses are drawn from the same distribution, where scores of 0.05 and lower are considered significant (Chinchor, 1992).

To measure parsing speed we used ten thousand unseen WSJ sentences from 1988 and ten thousand unseen Wikipedia sentences. The WSJ set was chosen as it is similar to the CCGBank WSJ evaluation set, but much larger and so the per sentence speed should be more accurate. The Wikipedia set is used as the two domains contain different writing styles, meaning the use of Wikipedia based self-training data should lead to particular improvement in speed on that form of text. The datasets only contain sentences of at least six and at most two hundred and fifty tokens.

As the amount of training data scales up, so too does the time it takes to train models. To demonstrate the benefits of perceptron based techniques we measured the amount of time models take to train. These measurements were performed using a 3GHz Intel Core 2 Duo CPU, and 4Gb of RAM.

	Accuracy (%)		Speed	
	WSJ		WSJ	Wiki
Parser	Cat.	F	(sent / sec)	
C&C 1.02	96.07	83.22	31.7	30.8
Modified	96.07	83.41	47.8	45.8

Table 1: The effect of introducing extra constraints on the use of backward composition on speed and accuracy. The supertagging model was constructed using BFGS and sections 02–21 of the WSJ.

5 Results

In this section we present four sets of experiments. First, the change in backward composition handling is evaluated, comparing the speed and accuracy of the standard model before and after the change. Second we consider the benefits of larger data sets, training models using the same algorithm but a range of training sets. Next we compare the new estimation algorithms described above with GIS and BFGS. Finally we explore the impact of introducing extra features.

5.1 Modified Backward Composition

The influence of the change to backward composition handling is shown in Table 1. A clear speed increase of more than 45% is achieved, and a statistically significant increase in F-score occurred.

5.2 Training Data Type and Volume

To investigate the effectiveness of semi-supervised training we constructed a series of models using the GIS algorithm and a selection of datasets. In Table 5.2 we can see that the use of Wikipedia data labelled by the parser causes a clear improvement in parsing speed on Wikipedia. We also observe that when the WSJ accounts for less than 10% of the training set, parsing speed on the WSJ decreases.

It is interesting to compare the baseline model and the models trained on Wikipedia. The model trained on forty thousand Wikipedia sentences only, approximately the same amount of text as in section 02-21 of the WSJ, has much lower supertagging accuracy, but much higher parsing speed. This makes sense as the text the model is trained on is not the true derivation, but rather the derivation that the parser chose. This means that the supertagging model is trained to produce the set of tags that the parser is most likely to combine into its final analysis. As a result, the set assigned at the first beta level is less accurate, but more likely to form a spanning analysis.

	Accuracy (%)			Amb.	Speed	
	WSJ		Wiki	Wiki	WSJ	Wiki
Data	Cat.	F	Cat.		(sent / sec)	
WSJ						
0k	**96.32**	83.82	95.34	1.32	51.7	46.8
Wiki						
40k	93.90	79.83	94.79	1.26	48.1	**61.3**
400k	95.07	81.75	95.71	1.27	46.9	**61.3**
2000k	95.54	82.57	**95.80**	1.28	45.0	57.3
WSJ + Wiki						
40k	96.31	**83.90**	95.37	1.29	**54.3**	58.9
400k	96.22	83.69	95.68	1.28	50.6	59.7
2000k	96.27	83.70	95.73	1.28	47.9	59.7

Table 2: The effect of self-training on supertagging accuracy and parsing F-score. Numbers in the 'Data' column indicate how much Wikipedia text was used.

The decreases in F-score when training on only Wikipedia are statistically significant, while the changes when training on a combination of the WSJ and Wikipedia are not. Interestingly, the models trained on only Wikipedia are also slower when parsing the WSJ than the baseline, and the models trained on a mixture of data are progressively slower as more Wikipedia data is used.

5.3 Algorithm Comparison

Using larger datasets for training can take a prohibitive amount of time for the GIS and BFGS algorithms. However, any time benefits provided by other algorithms need to be balanced with their influence on accuracy. Table 3 shows the results of experiments investigating this trade-off.

It is clear from the training speed column that the perceptron based algorithms, AP and MIRA, train approximately two orders of magnitude faster than GIS and BFGS.

It is also interesting to note the change in the average number of categories assigned for Wikipedia sentences. As expected, the ambiguity level is decreasing as more Wikipedia text is used, but at the same time the tagging accuracy remains fairly constant or improves slightly. This indicates that the automatically labelled data is useful in adapting the supertagger to the Wikipedia domain.

Importantly, the changes in F-score between

	Accuracy (%)			Amb.	Speed		
	WSJ		Wiki	Wiki	Train	WSJ	Wiki
Data	Cat.	F	Cat.		(sec)	(sent / sec)	
WSJ							
GIS	**96.32**	83.82	95.34	1.32	7,200	51.7	46.8
BFGS	96.29	83.73	95.33	1.31	6,300	52.1	48.5
AP	95.65	83.74	94.49	1.35	76	59.2	57.1
MIRA	96.19	83.69	95.19	1.33	96	50.6	47.9
WSJ + 40k Wiki							
GIS	96.31	**83.90**	95.37	1.29	14,000	54.3	58.9
BFGS	96.14	83.86	95.24	1.29	13,000	52.1	60.7
AP	95.68	83.79	94.61	1.28	160	**62.8**	**69.7**
MIRA	96.18	83.77	95.28	1.30	200	54.0	58.6
WSJ + 400k Wiki							
GIS	96.22	83.69	95.68	1.28	*	50.6	59.7
AP	95.77	83.56	95.16	1.27	950	57.8	69.4
MIRA	96.19	83.41	95.58	1.28	1,200	52.3	61.4
WSJ + 2,000k Wiki							
GIS	96.27	83.70	**95.73**	1.28	*	47.9	59.7
MIRA	96.22	83.52	95.62	1.28	*	52.0	59.3

Table 3: Comparison of model estimation algorithms. The models missing times were trained on a different computer with more RAM and are provided for accuracy comparison.

models in each section are not statistically significant. This indicates that the perceptron based algorithms are just as effective as GIS and BFGS.

5.4 Feature Extension

The final set of experiments involved the exploration of extra features. Using the MIRA training method we were able to quickly construct a large set of models, as shown in Table 4.

The standard features used by the supertagger are unigrams of words and unigrams and bigrams of POS tags in a five word window. We considered expansions of this set to include bigrams of words and trigrams of POS tags, and all of the features extended to consider a seven word window, which are indicated by the word 'far'.

The results in the first section of the table, training on the WSJ only, are unsurprising. With such a small amount of data these features are too rare to have a significant impact, and it is likely that they lead the model to over-fit. The best result in this section does not produce a statistically significant improvement over the baseline. However, in the second and third sections of the table the differences between the best models and the baseline are statistically significant. Also, the model with

	Accuracy (%)			Speed	
	WSJ		Wiki	WSJ	Wiki
Features	Cat.	F	Cat.	(sent / sec)	
WSJ					
All	96.25	83.69	95.12	45.1	42.8
- far tags	96.13	83.68	95.15	46.4	42.9
- bitags	96.15	83.84	95.24	45.2	42.1
- far bitags	96.22	83.83	95.24	45.3	43.2
- tritags	96.23	83.79	**95.34**	45.2	42.6
- far tritags	96.22	83.86	95.31	45.5	43.2
- far words	96.28	83.83	95.27	46.2	43.1
- biwords	96.22	83.81	95.19	45.9	45.4
- far biwords	96.26	83.89	95.19	45.5	43.7
- triwords	**96.31**	83.80	95.16	48.0	46.0
- far triwords	96.25	**83.91**	95.24	46.2	43.6
Baseline	96.19	83.69	95.19	**50.6**	**47.9**
WSJ + 40k Wiki					
All	96.29	84.00	95.45	48.7	55.9
- far tags	96.20	83.96	95.45	48.1	53.6
- bitags	96.15	83.84	95.24	45.2	42.3
- far bitags	96.28	**84.17**	95.33	48.3	55.8
- tritags	96.25	83.88	95.47	48.2	54.5
- far tritags	96.34	83.85	**95.49**	49.7	54.9
- far words	96.32	84.04	95.47	48.1	55.7
- biwords	96.31	84.04	95.31	50.5	57.4
- far biwords	96.35	84.10	95.42	49.2	55.0
- triwords	**96.39**	**84.17**	95.42	50.0	57.8
- far triwords	96.32	84.00	95.47	49.6	55.5
Baseline	96.18	83.77	95.28	**54.0**	**58.6**
WSJ + 400k Wiki					
All	96.42	83.80	95.82	48.2	57.4
- far tags	96.38	83.72	95.79	48.3	57.3
- bitags	96.34	83.79	95.85	42.0	56.9
- far bitags	96.34	83.85	95.79	49.4	57.8
- tritags	96.38	83.81	**95.91**	49.2	57.9
- far tritags	96.39	83.94	**95.91**	50.2	56.8
- far words	**96.46**	83.73	**95.91**	48.8	57.4
- biwords	96.35	83.74	95.74	50.0	58.3
- far biwords	96.40	**83.97**	95.82	49.7	57.8
- triwords	96.37	83.96	95.74	49.7	58.8
- far triwords	96.40	83.86	95.83	49.9	58.4
Baseline	96.19	83.41	95.58	**52.3**	**61.4**

Table 4: Subtractive analysis of various feature sets. In each section the category accuracy values that are lower than those for 'All' have been underlined as removing these features decreases accuracy. The bold values are the best in each column for each section. The baseline model uses the default feature set for the C&C parser.

the best result in the table produces a statistically significant improvement in recall over the models in Table 3 constructed using the same data.

6 Future Work

A wide range of directions exist for extension of this work. The most direct extensions would be to perform experiments using more of Wikipedia, particularly for the feature exploration.

As well as the simple extensions of current features that are described here, extra data may enable the use of more complex features. For example, a feature to encode the presence of one attribute and the absence of another.

Now that we have a range of different algorithms for model estimation it may be possible to perform co-training style experiments. Even a simpler method, such as using the set of weights found by one algorithm as the initial weights for another, may lead to improved results. Additionally, the current system takes the weights produced by the perceptron algorithms, normalises them and treats them as a probability distribution in the same way as the weights from GIS and BFGS are treated. It would be interesting to explore the possibility of multi-tagging with a perceptron instead. The perceptron based algorithms can also be adjusted at run time, making it feasible to learn continuously.

Here we have presented results for training on automatically labelled Wikipedia text, but we could perform the same experiments on effectively any corpus. It would be interesting to explore the ability of the system to adapt to new domains through semi-supervised training.

7 Conclusion

This work has shown that semi-supervised supertagger training can boost parsing speed considerably and demonstrated that perceptron based algorithms can effectively estimate supertagger model parameters. To achieve this we adjusted the C&C parser's handling of backward composition, parallelised the supertagger training process, and implemented the MIRA and AP algorithms for feature weight estimation.

The change in backward composition handling provided a 50% speed boost and a further 30% was gained for parsing Wikipedia by using parsed Wikipedia as extra training data. As more Wikipedia data was used speed on the WSJ fell below the baseline, indicating that domain adaptation was occurring.

Models trained using perceptron based algorithms performed just as well, but were trained two orders of magnitude faster. Extending the feature set led to small but statistically significant improvements, including two models that achieve an F-score of 84.17% for labelled dependencies on Section 00 of the WSJ.

Initially the system produced an F-score of 83.22% on Section 00 of the WSJ, could parse the WSJ and Wikipedia at 31.7 and 30.8 sentences per second respectively, and took two hours to train the supertagging model, using only forty thousand sentences for training. Our changes enabled the construction of a model in under four minutes that achieves an F-score of 83.79 on WSJ, and speeds of 62.8 and 69.7 sentences per second for WSJ and Wikipedia respectively, ie. 2.0 times faster for WSJ, and 2.3 times faster for Wikipedia.

8 Acknowledgements

We thank the reviewers for helpful feedback. This work was supported by the Capital Markets Cooperative Research Centre Limited and a University of Sydney Merit Scholarship and aspects of the work were completed at the Summer Research Workshop on Machine Learning for Language Engineering at the Center for Language and Speech Processing, Johns Hopkins University.

References

S. Bird, E. Loper, and E. Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

C. G. Broyden. 1970. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90.

R. Chandrasekar and S. Bangalore. 1997a. Gleaning information from the web: Using syntax to filter out irrelevant information. In *World Wide Web, Stanford University*.

R. Chandrasekar and S. Bangalore. 1997b. Using supertags in document filtering: The effect of increased context on information retrieval effectiveness. In *Proceedings of RANLP '97*.

J. Chen, S. Bangalore, and V. K. Shanker. 1999. New models for improving supertag disambiguation. In *Proceedings of the 9th Meeting of EACL*, pages 188–195, Bergen, Norway, June.

J. Chen, S. Bangalore, M. Collins, and O. Rambow. 2002. Reranking an n-gram supertagger. In *Proceedings of the TAG+ Workshop*, pages 259–268, Venice, Italy, May.

N. Chinchor. 1992. Statistical significance of muc-6 results.

S. Clark and J. R. Curran. 2003. Log-linear models for wide-coverage ccg parsing. In *Proceedings of the Conf. on EMNLP*, pages 97–104. ACL.

S. Clark and J. R. Curran. 2007a. Perceptron training for a wide-coverage lexicalized-grammar parser. In *ACL 2007 Workshop on Deep Linguistic Processing*, pages 9–16, Prague, Czech Republic, June. ACL.

S. Clark and J. R. Curran. 2007b. Wide-coverage eficient statistical parsing with ccg and log-linear models. *Comp. Ling.*, 33(4):493–552.

S. Clark, J. Curran, and M. Osborne. 2003. Boostrapping pos-taggers using unlabelled data. In *Proceedings of CoNLL*.

S. Clark, A. Copestake, J. R. Curran, Y. Zhang, A. Herbelot, J. Haggerty, B. Ahn, C. Van Wyk, J. Roesner, J. K. Kummerfeld, and T. Dawborn. 2009. Large-scale syntactic processing: Parsing the web. Technical report, JHU CLSP Workshop.

S. Clark. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of TAG+6*, pages 19–24, Venice, Italy, May.

M. Collins and B. Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the ACL*, pages 111–118, Barcelona, Spain.

M. Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conf. on EMNLP*, pages 1–8. ACL, July.

N. Cooper. 2007. Improved statistical models for supertagging.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480.

J. Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th Meeting of the ACL*, pages 79–86, Santa Cruz, CA.

R. Fletcher. 1970. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322,.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

D. Goldfarb. 1970. A family of variable metric updates derived by variational means. *Mathematics of Computation*, 24:23–26.

J. Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

A. K. Joshi and S. Bangalore. 1994. Disambiguation of super parts of speech (or supertags): almost parsing. In *Proceedings of the 15th Conf. on Computational Linguistics*, pages 154–160, Kyoto, Japan, August.

T. Kasami. 1967. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA.

T. Kiss and J. Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Comp. Ling.*, 32(4):485–525.

D. McClosky, E. Charniak, and M. Johnson. 2006. Effective self-training for parsing. In *Proceedings of the NAACL Conf.*

F. Rosenblatt. 1958. The perceptron - a probabilistic model for information - storage and organization in the brain. *Psychological Review*, 65(6):386–408.

A. Sarkar, F. Xia, and A. K. Joshi. 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *Proceedings of COLING*, pages 37–42.

A. Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the NAACL conference*, pages 1–8, Pittsburgh, Pennsylvania. ACL.

A. Sarkar, 2007. *Combining Supertagging and Lexicalized Tree-Adjoining Grammar Parsing*. MIT Press.

D. F. Shanno. 1970. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24:647–656.

M. Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Meeting of the ACL*, pages 189–196. ACL.

D. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Y. Zhang, T. Matsuzaki, and J. Tsujii. 2009. Hpsg supertagging: A sequence labeling view. In *Proceedings of the 11th IWPT*, pages 210–213, Paris, France, October. ACL.