Faster Parsing by Supertagger Adaptation

Jonathan K. Kummerfeld[a] **Jessika Roesner**[b] **Tim Dawborn**[a] **James Haggerty**[a]
James R. Curran[a*] **Stephen Clark**[c*]

School of Information Technologies[a]
University of Sydney
NSW 2006, Australia

Department of Computer Science[b]
University of Texas at Austin
Austin, TX, USA

Computer Laboratory[c]
University of Cambridge
Cambridge CB3 0FD, UK

james@it.usyd.edu.au[a*] stephen.clark@cl.cam.ac.uk[c*]

Abstract

We propose a novel self-training method for a parser which uses a lexicalised grammar and supertagger, focusing on increasing the speed of the parser rather than its accuracy. The idea is to train the supertagger on large amounts of parser output, so that the supertagger can learn to supply the supertags that the parser will eventually choose as part of the highest-scoring derivation. Since the supertagger supplies fewer supertags overall, the parsing speed is increased. We demonstrate the effectiveness of the method using a CCG supertagger and parser, obtaining significant speed increases on newspaper text with no loss in accuracy. We also show that the method can be used to adapt the CCG parser to new domains, obtaining accuracy and speed improvements for Wikipedia and biomedical text.

1 Introduction

In many NLP tasks and applications, e.g. distributional similarity (Curran, 2004) and question answering (Dumais et al., 2002), large volumes of text and detailed syntactic information are both critical for high performance. To avoid a trade-off between these two, we need to increase parsing speed, but without losing accuracy.

Parsing with lexicalised grammar formalisms, such as Lexicalised Tree Adjoining Grammar and Combinatory Categorial Grammar (CCG; Steedman, 2000), can be made more efficient using a *supertagger*. Bangalore and Joshi (1999) call supertagging *almost parsing* because of the significant reduction in ambiguity which occurs once the supertags have been assigned.

In this paper, we focus on the CCG parser and supertagger described in Clark and Curran (2007). Since the CCG lexical category set used by the supertagger is much larger than the Penn Treebank POS tag set, the accuracy of supertagging is much lower than POS tagging; hence the CCG supertagger assigns multiple supertags[1] to a word, when the local context does not provide enough information to decide on the correct supertag.

The supertagger feeds lexical categories to the parser, and the two interact, sometimes using multiple passes over a sentence. If a spanning analysis cannot be found by the parser, the number of lexical categories supplied by the supertagger is increased. The supertagger-parser interaction influences speed in two ways: first, the larger the lexical ambiguity, the more derivations the parser must consider; second, each further pass is as costly as parsing a whole extra sentence.

Our goal is to increase parsing speed without loss of accuracy. The technique we use is a form of *self-training*, in which the output of the parser is used to train the supertagger component. The existing literature on self-training reports mixed results. Clark et al. (2003) were unable to improve the accuracy of POS tagging using self-training. In contrast, McClosky et al. (2006a) report improved accuracy through self-training for a two-stage parser and re-ranker.

Here our goal is not to improve accuracy, only to maintain it, which we achieve through an *adaptive* supertagger. The adaptive supertagger produces lexical categories that the parser would have used in the final derivation when using the baseline model. However, it does so with much lower ambiguity levels, and potentially during an earlier pass, which means sentences are parsed faster. By increasing the ambiguity level of the adaptive models to match the baseline system, we can also slightly increase supertagging accuracy, which can lead to higher parsing accuracy.

[1] We use *supertag* and *lexical category* interchangeably.

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 345–355,
Uppsala, Sweden, 11-16 July 2010.

Using the parser to generate training data also has the advantage that it is not a domain specific process. Previous work has shown that parsers typically perform poorly outside of their training domain (Gildea, 2001). Using a newspaper-trained parser, we constructed new training sets for Wikipedia and biomedical text. These were used to create new supertagging models adapted to the different domains.

The self-training method of adapting the supertagger to suit the parser increased parsing speed by more than 50% across all three domains, without loss of accuracy. Using an adapted supertagger with ambiguity levels tuned to match the baseline system, we were also able to increase F-score on labelled grammatical relations by 0.75%.

2 Background

Many statistical parsers use two stages: a tagging stage that labels each word with its grammatical role, and a parsing stage that uses the tags to form a parse tree. Lexicalised grammars typically contain a much smaller set of rules than phrase-structure grammars, relying on tags (supertags) that contain a more detailed description of each word's role in the sentence. This leads to much larger tag sets, and shifts a large proportion of the search for an optimal derivation to the tagging component of the parser.

Figure 1 gives two sentences and their CCG derivations, showing how some of the syntactic ambiguity is transferred to the supertagging component in a lexicalised grammar. Note that the lexical category assigned to with is different in each case, reflecting the fact that the prepositional phrase attaches differently. Either we need a tagging model that can resolve this ambiguity, or both lexical categories must be supplied to the parser which can then attempt to resolve the ambiguity by eventually selecting between them.

2.1 Supertagging

Supertaggers typically use standard linear-time tagging algorithms, and only consider words in the local context when assigning a supertag. The C&C supertagger is similar to the Ratnaparkhi (1996) tagger, using features based on words and POS tags in a five-word window surrounding the target word, and defining a local probability distribution over supertags for each word in the sentence, given the previous two supertags. The Viterbi algorithm can be used to find the most probable supertag sequence. Alternatively the Forward-Backward algorithm can be used to efficiently sum over all sequences, giving a probability distribution over supertags for each word which is conditional only on the input sentence.

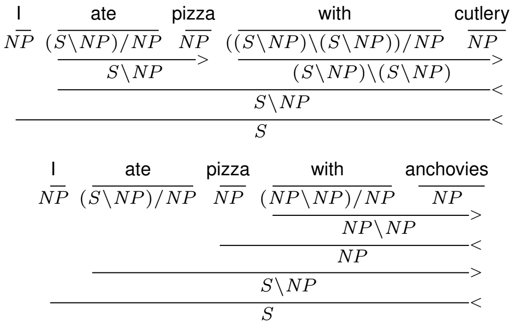
I ate pizza with cutlery
NP $(S\backslash NP)/NP$ NP $((S\backslash NP)\backslash(S\backslash NP))/NP$ NP
$S\backslash NP$ >
$(S\backslash NP)\backslash(S\backslash NP)$ >
$S\backslash NP$ <
S <
I ate pizza with anchovies
NP $(S\backslash NP)/NP$ NP $(NP\backslash NP)/NP$ NP
$NP\backslash NP$ >
NP <
$S\backslash NP$ >
S <

Figure 1: Two CCG derivations with PP ambiguity.

Supertaggers can be made accurate enough for wide coverage parsing using multi-tagging (Chen et al., 1999), in which more than one supertag can be assigned to a word; however, as more supertags are supplied by the supertagger, parsing efficiency decreases (Chen et al., 2002), demonstrating the influence of lexical ambiguity on parsing complexity (Sarkar et al., 2000).

Clark and Curran (2004) applied supertagging to CCG, using a flexible multi-tagging approach. The supertagger assigns to a word all lexical categories whose probabilities are within some factor, β, of the most probable category for that word. When the supertagger is integrated with the C&C parser, several progressively lower β values are considered. If a sentence is not parsed on one pass then the parser attempts to parse the sentence again with a lower β value, using a larger set of categories from the supertagger. Since most sentences are parsed at the first level (in which the average number of supertags assigned to each word is only slightly greater than one), this provides some of the speed benefit of single tagging, but without loss of coverage (Clark and Curran, 2004).

Supertagging has since been effectively applied to other formalisms, such as HPSG (Blunsom and Baldwin, 2006; Zhang et al., 2009), and as an information source for tasks such as Statistical Machine Translation (Hassan et al., 2007). The use of parser output for supertagger training has been explored for LTAG by Sarkar (2007). However, the focus of that work was on improving parser and supertagger accuracy rather than speed.

Previously	,	watch	imports	were	denied	such	duty-free	treatment
S/S	,	N/N	**N**	$(S[dcl]\backslash NP)/(S[pss]\backslash NP)$	$(S[pss]\backslash NP)/NP$	**NP/NP**	**N/N**	**N**
N		**N**		**(S[dcl]\NP)/NP**	$S[pss]\backslash NP$	$(N/N)/(N/N)$		
$S[adj]\backslash NP$				$(S[dcl]\backslash NP)/(S[adj]\backslash NP)$	$(S[pss]\backslash NP)/NP$	N/N		
					$(S[pt]\backslash NP)/NP$			
					(S[dcl]\NP)/NP			

Figure 2: An example sentence and the sets of categories assigned by the supertagger. The first category in each column is correct and the categories used by the parser are marked in bold. The correct category for watch is included here, for expository purposes, but in fact was not provided by the supertagger.

2.2 Semi-supervised training

Previous exploration of semi-supervised training in NLP has focused on improving accuracy, often for the case where only small amounts of manually labelled training data are available. One approach is *co-training*, in which two models with independent views of the data iteratively inform each other by labelling extra training data. Sarkar (2001) applied co-training to LTAG parsing, in which the supertagger and parser provide the two views. Steedman et al. (2003) extended the method to a variety of parser pairs.

Another method is to use a re-ranker (Collins and Koo, 2002) on the output of a system to generate new training data. Like co-training, this takes advantage of a different view of the data, but the two views are not independent as the re-ranker is limited to the set of options produced by the system. This method has been used effectively to improve parsing performance on newspaper text (McClosky et al., 2006a), as well as adapting a Penn Treebank parser to a new domain (McClosky et al., 2006b).

As well as using independent views of data to generate extra training data, multiple views can be used to provide constraints at test time. Hollingshead and Roark (2007) improved the accuracy of a parsing pipeline by using the output of later stages to constrain earlier stages.

The only work we are aware of that uses self-training to improve the *efficiency* of parsers is van Noord (2009), who adopts a similar idea to the one in this paper for improving the efficiency of a Dutch parser based on a manually constructed HPSG grammar.

3 Adaptive Supertagging

The purpose of the supertagger is to cut down the search space for the parser by reducing the set of categories that must be considered for each word. A perfect supertagger would assign the correct category to every word. CCG supertaggers are about 92% accurate when assigning a single lexical category to each word (Clark and Curran, 2004). This is not accurate enough for wide coverage parsing and so a multi-tagging approach is used instead. In the final derivation, the parser uses one category from each set, and it is important to note that having the correct category in the set does not guarantee that the parser will use it.

Figure 2 gives an example sentence and the sets of lexical categories supplied by the supertagger, for a particular value of β.[2] The usual target of the supertagging task is to produce the top row of categories in Figure 2, the correct categories. We propose a new task that instead aims for the categories the parser will use, which are marked in bold for this case. The purpose of this new task is to improve speed.

The reason speed will be improved is that we can construct models that will constrain the set of possible derivations more than the baseline model. We can construct these models because we can obtain much more of our target output, parser-annotated sentences, than we could for the gold-standard supertagging task.

The new target data will contain tagging errors, and so supertagging accuracy measured against the correct categories may decrease. If we obtained perfect accuracy on our new task then we would be removing all of the categories not chosen by the parser. However, parsing accuracy will not decrease since the parser will still receive the categories it would have used, and will therefore be able to form the same highest-scoring derivation (and hence will choose it).

To test this idea we parsed millions of sentences

[2] Two of the categories for such have been left out for reasons of space, and the correct category for watch has been included for expository reasons. The fact that the supertagger does not supply this category is the reason that the parser does not analyse the sentence correctly.

in three domains, producing new data annotated with the categories that the parser used with the baseline model. We constructed new supertagging models that are adapted to suit the parser by training on the combination of these sets and the standard training corpora. We applied standard evaluation metrics for speed and accuracy, and explored the source of the changes in parsing performance.

4 Data

In this work, we consider three domains: newswire, Wikipedia text and biomedical text.

4.1 Training and accuracy evaluation

We have used Sections 02-21 of CCGbank (Hockenmaier and Steedman, 2007), the CCG version of the Penn Treebank (Marcus et al., 1993), as training data for the newspaper domain. Sections 00 and 23 were used for development and test evaluation. A further 113,346,430 tokens (4,566,241 sentences) of raw data from the Wall Street Journal section of the North American News Corpus (Graff, 1995) were parsed to produce the training data for adaptation. This text was tokenised using the C&C tools tokeniser and parsed using our baseline models. For the smaller training sets, sentences from 1988 were used as they would be most similar in style to the evaluation corpus. In all experiments the sentences from 1989 were excluded to ensure no overlap occurred with CCGbank.

As Wikipedia text we have used 794,024,397 tokens (51,673,069 sentences) from Wikipedia articles. This text was processed in the same way as the NANC data to produce parser-annotated training data. For supertagger evaluation, one thousand sentences were manually annotated with CCG lexical categories and POS tags. For parser evaluation, three hundred of these sentences were manually annotated with DepBank grammatical relations (King et al., 2003) in the style of Briscoe and Carroll (2006). Both sets of annotations were produced by manually correcting the output of the baseline system. The annotation was performed by Stephen Clark and Laura Rimell.

For the biomedical domain we have used several different resources. As gold standard data for supertagger evaluation we have used supertagged GENIA data (Kim et al., 2003), annotated by Rimell and Clark (2008). For parsing evaluation, grammatical relations from the BioInfer corpus were used (Pyysalo et al., 2007), with the same post-processing process as Rimell and Clark (2009) to convert the C&C parser output to Stanford format grammatical relations (de Marneffe et al., 2006). For adaptive training we have used 1,900,618,859 tokens (76,739,723 sentences) from the MEDLINE abstracts tokenised by McIntosh and Curran (2008). These sentences were POS-tagged and parsed twice, once as for the newswire and Wikipedia data, and then again, using the bio-specific models developed by Rimell and Clark (2009). Statistics for the sentences in the training sets are given in Table 1.

Source	Sentence Length			Corpus %
	Range	Average	Variance	
News	0-4	3.26	0.64	1.2
	5-20	14.04	17.41	39.2
	21-40	28.76	29.27	49.4
	41-250	49.73	86.73	10.2
	All	24.83	152.15	100.0
Wiki	0-4	2.81	0.60	22.4
	5-20	11.64	21.56	48.9
	21-40	28.02	28.48	24.3
	41-250	49.69	77.70	4.5
	All	15.33	154.57	100.0
Bio	0-4	2.98	0.75	0.9
	5-20	14.54	15.14	41.3
	21-40	28.49	29.34	48.0
	41-250	49.17	68.34	9.8
	All	24.53	139.35	100.0

Table 1: Statistics for sentences in the supertagger training data. Sentences containing more than 250 tokens were not included in our data sets.

4.2 Speed evaluation data

For speed evaluation we held out three sets of sentences from each domain-specific corpus. Specifically, we used 30,000, 4,000 and 2,000 unique sentences of length 5-20, 21-40 and 41-250 tokens respectively. Speeds on these length controlled sets were combined to calculate an overall parsing speed for the text in each domain. Note that more than 20% of the Wikipedia sentences were less than five words in length and the overall distribution is skewed towards shorter sentences compared to the other corpora.

5 Evaluation

We used the hybrid parsing model described in Clark and Curran (2007), and the Viterbi decoder to find the highest-scoring derivation. The multipass supertagger-parser interaction was also used.

The test data was excluded from training data for the supertagger for all of the newswire and Wikipedia models. For the biomedical models tenfold cross validation was used. The accuracy of supertagging is measured by multi-tagging at the first β level and considering a word correct if the correct tag is amongst any of the assigned tags.

For the biomedical parser evaluation we have used the parsing model and grammatical relation conversion script from Rimell and Clark (2009).

Our timing measurements are calculated in two ways. Overall times were measured using the C&C parser's timers. Individual sentence measurements were made using the Intel timing registers, since standard methods are not accurate enough for the short time it takes to parse a single sentence.

To check whether changes were statistically significant we applied the test described by Chinchor (1995). This measures the probability that two sets of responses are drawn from the same distribution, where a score below 0.05 is considered significant.

Models were trained on an Intel Core2Duo 3GHz with 4GB of RAM. The evaluation was performed on a dual quad-core Intel Xeon 2.27GHz with 16GB of RAM.

5.1 Tagging ambiguity optimisation

The number of lexical categories assigned to a word by the CCG supertagger depends on the probabilities calculated for each category and the β level being used. Each lexical category with a probability within a factor of β of the most probable category is included. This means that the choice of β level determines the tagging ambiguity, and so has great influence on parsing speed, accuracy and coverage. Also, the tagging ambiguity produced by a β level will vary between models. A more confident model will have a more peaked distribution of category probabilities for a word, and therefore need a smaller β value to assign the same number of categories.

Additionally, the C&C parser uses multiple β levels. The first pass over a sentence is at a high β level, resulting in a low tagging ambiguity. If the categories assigned are too restrictive to enable a spanning analysis, the system makes another pass with a lower β level, resulting in a higher tagging ambiguity. A maximum of five passes are made, with the β levels varying from 0.075 to 0.001.

We have taken two approaches to choosing β levels. When the aim of an experiment is to improve speed, we use the system's default β levels. While this choice means a more confident model will assign fewer tags, this simply reflects the fact that the model is more confident. It should produce similar accuracy results, but with lower ambiguity, which will lead to higher speed.

For accuracy optimisation experiments we tune the β levels to produce the same average tagging ambiguity as the baseline model on Section 00 of CCGbank. Accuracy depends heavily on the number of categories supplied, so the new models are at an accuracy disadvantage if they propose fewer categories. By matching the ambiguity of the default model, we can increase accuracy at the cost of some of the speed improvements the new models obtain.

6 Results

We have performed four primary sets of experiments to explore the ability of an adaptive supertagger to improve parsing speed or accuracy. In the first two experiments, we explore performance on the newswire domain, which is the source of training data for the parsing model and the baseline supertagging model. In the second set of experiments, we train on a mixture of gold standard newswire data and parser-annotated data from the target domain.

In both cases we perform two experiments. The first aimed to improve speed, keeping the β levels the same. This should lead to an increase in speed as the extra training data means the models are more confident and so have lower ambiguity than the baseline model for a given β value. The second experiment aimed to improve accuracy, tuning the β levels as described in the previous section.

6.1 Newswire speed improvement

In our first experiment, we trained supertagger models using Generalised Iterative Scaling (GIS) (Darroch and Ratcliff, 1972), the limited memory BFGS method (BFGS) (Nocedal and Wright, 1999), the averaged perceptron (Collins, 2002), and the margin infused relaxed algorithm (MIRA) (Crammer and Singer, 2003). Note that these are all alternative methods for estimating the *local* log-linear probability distributions used by the Ratnaparkhi-style tagger. We do not use global tagging models as in Lafferty et al. (2001) or Collins (2002). The training data consisted of Sections 02–21 of CCGbank and progressively larger quantities of parser-annotated NANC data – from zero to four million extra sentences. The results of these tests are presented in Table 2.

	Ambiguity (%)				Tagging Accuracy (%)				F-score				Speed (sents / sec)			
Data	0k	40k	400k	4m	0k	40k	400k	4m	0k	40k	400k	4m	0k	40k	400k	4m
Baseline	1.27				96.34				85.46				39.6			
BFGS	1.27	1.23	1.19	1.18	96.33	96.18	95.95	95.93	85.45	85.51	85.57	85.68	39.8	49.6	71.8	60.0
GIS	1.28	1.24	1.21	1.20	96.44	96.27	96.09	96.11	85.44	85.46	85.58	85.62	37.4	44.1	51.3	54.1
MIRA	1.30	1.24	1.17	1.13	96.44	96.14	95.56	95.18	85.44	85.40	85.38	85.42	34.1	44.8	60.2	**73.3**

Table 2: Speed improvements on newswire, using various amounts of parser-annotated NANC data.

		Sentences			Av. Time Change (ms)			Total Time Change (s)		
	Sentence length	5-20	21-40	41-250	5-20	21-40	41-250	5-20	21-40	41-250
Earlier pass	Lower tag amb.	1166	333	281	-7.54	-71.42	-183.23	-1.1	-29	-26
	Same tag amb.	248	38	8	-2.94	-27.08	-108.28	-0.095	-1.3	-0.44
	Higher tag amb.	530	33	14	-5.84	-32.25	-44.10	-0.40	-1.3	-0.31
Same pass	Lower tag amb.	19288	3120	1533	-1.13	-5.18	-38.05	-2.8	-20	-30
	Same tag amb.	7285	259	35	-0.29	0.94	24.57	-0.28	0.30	0.44
	Higher tag amb.	1133	101	24	-0.25	2.70	8.09	-0.037	0.34	0.099
Later pass	Lower tag amb.	334	114	104	0.90	7.60	-46.34	0.039	1.1	-2.5
	Same tag amb.	14	1	0	1.06	4.26	n/a	0.0019	0.0053	0.0
	Higher tag amb.	2	1	1	-0.13	26.43	308.03	-3.4e-05	0.033	0.16

Table 3: Breakdown of the source of changes in speed. The test sentences are divided into nine sets based on the change in parsing behaviour between the baseline model and a model trained using MIRA, Sections 02-21 of CCGbank and 4,000,000 NANC sentences.

Using the default β levels we found that the perceptron-trained models lost accuracy, disqualifying them from this test. The BFGS, GIS and MIRA models produced mixed results, but no statistically significant decrease in accuracy, and as the amount of parser-annotated data was increased, parsing speed increased by up to 85%.

To determine the source of the speed improvement we considered the times recorded by the timing registers. In Table 3, we have aggregated these measurements based on the change in the pass at which the sentence is parsed, and how the tagging ambiguity changes on that pass. For sentences parsed on two different passes the ambiguity comparison is at the earlier pass. The "Total Time Change" section of the table is the change in parsing time for sentences of that type when parsing ten thousand sentences from the corpus. This takes into consideration the actual distribution of sentence lengths in the corpus.

Several effects can be observed in these results. 72% of sentences are parsed on the same pass, but with lower tag ambiguity (5th row in Table 3). This provides 44% of the speed improvement. Three to six times as many sentences are parsed on an earlier pass than are parsed on a later pass. This means the sentences parsed later have very little effect on the overall speed. At the same time, the average gain for sentences parsed earlier is almost always larger than the average cost for sentences parsed later. These effects combine to produce a particularly large improvement for the sentences parsed at an earlier pass. In fact, despite making up only 7% of sentences in the set, those parsed earlier with lower ambiguity provide 50% of the speed improvement.

It is also interesting to note the changes for sentences parsed on the same pass, with the same ambiguity. We may expect these sentences to be parsed in approximately the same amount of time, and this is the case for the short set, but not for the two larger sets, where we see an increase in parsing time. This suggests that the categories being supplied are more productive, leading to a larger set of possible derivations.

6.2 Newswire accuracy optimised

Any decrease in tagging ambiguity will generally lead to a decrease in accuracy. The parser uses a more sophisticated algorithm with global knowledge of the sentence and so we would expect it to be better at choosing categories than the supertagger. Unlike the supertagger it will exclude categories that cannot be used in a derivation. In the previous section, we saw that training the supertagger on parser output allowed us to develop models that produced the same categories, despite lower tagging ambiguity. Since they were trained on the categories the parser was able to use in derivations, these models should also now be providing categories that are more likely to be useful.

This leads us to our second experiment, opti-

	Tagging Accuracy (%)				F-score				Speed (sents / sec)			
NANC sents	0k	40k	400k	4m	0k	40k	400k	4m	0k	40k	400k	4m
Baseline	96.34				85.46				39.6			
BFGS	96.33	96.42	96.42	**96.66**	85.45	85.55	85.64	**85.98**	39.5	43.7	43.9	42.7
GIS	96.34	96.43	96.53	96.62	85.36	85.47	85.84	85.87	39.1	41.4	41.7	42.6
Perceptron	95.82	95.99	96.30	-	85.28	85.39	85.64	-	45.9	48.0	45.2	-
MIRA	96.23	96.29	96.46	96.63	85.47	85.45	85.55	85.84	37.7	41.4	41.4	42.9

Table 4: Accuracy optimisation on newswire, using various amounts of parser-annotated NANC data.

Train Corpus	Ambiguity			Tag. Acc.			F-score			Speed (sents / sec)		
	News	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio
Baseline	1.267	1.317	1.281	96.34	94.52	90.70	85.46	80.8	75.0	39.6	50.9	35.1
News	1.126	1.151	1.130	95.18	93.56	90.07	85.42	81.2	75.2	**73.3**	**83.9**	**60.3**
Wiki	1.147	1.154	1.129	95.06	93.52	90.03	84.70	81.4	75.5	62.4	**73.9**	58.7
Bio	1.134	1.146	1.114	94.66	93.15	89.88	84.23	80.7	75.9	66.2	**90.4**	**59.3**

Table 5: Cross-corpus speed improvement, models trained with MIRA and 4,000,000 sentences. The highlighted values are the top speed for each evaluation set and results that are statistically indistinguishable from it.

mising accuracy on newswire. We used the same models as in the previous experiment, but tuned the β levels as described in Section 5.1.

Comparing Tables 2 and 4 we can see the influence of β level choice, and therefore tagging ambiguity. When the default β values were used ambiguity dropped consistently as more parser-annotated data was used, and category accuracy dropped in the same way. Tuning the β levels to match ambiguity produces the opposite trend.

Interestingly, while the decrease in supertag accuracy in the previous experiment did not translate into a decrease in F-score, the increase in tag accuracy here does translate into an increase in F-score. This indicates that the supertagger is adapting to suit the parser. In the previous experiment, the supertagger was still providing the categories the parser would have used with the baseline supertagging model, but it provided fewer other categories. Since the parser is not a perfect supertagger these other categories may in fact have been incorrect, and so supertagger accuracy goes down, without changing parsing results. Here we have allowed the supertagger to assign extra categories, which will only increase its accuracy.

The increase in F-score has two sources. First, our supertagger is more accurate, and so the parser is more likely to receive category sets that can be combined into the correct derivation. Also, the supertagger has been trained on categories that the parser is able to use in derivations, which means they are more productive.

As Table 6 shows, this change translates into an improvement of up to 0.75% in F-score on Section 23 of CCGbank. All of the new models in the table make a statistically significant improvement over the baseline.

Model	Tag. Acc. (%)	F-score (%)	Speed (sents/sec)
Baseline	96.51	85.20	39.6
GIS, 4,000k NANC	96.83	**85.95**	42.6
BFGS, 4,000k NANC	96.91	85.90	42.7
MIRA, 4,000k NANC	96.84	85.79	42.9

Table 6: Evaluation of top models on Section 23 of CCGbank. All changes in F-score are statistically significant.

It is also interesting to note that the results in Tables 2, 4 and 6, are similar for all of the training algorithms. However, the training times differ considerably. For all four algorithms the training time is proportional to the amount of data, but the GIS and BFGS models trained on only CCGbank took 4,500 and 4,200 seconds to train, while the equivalent perceptron and MIRA models took 90 and 95 seconds to train.

6.3 Annotation method comparison

To determine whether these improvements were dependent on the annotations being produced by the parser we performed a set of tests with supertagger, rather than parser, annotated data. Three extra training sets were created by annotating newswire sentences with supertags using the baseline supertagging model. One set used the one-best tagger, and two were produced using the most probable tag for each word out of the set supplied by the multi-tagger, with variations in the β value and dictionary cutoff for the two sets.

Train Corpus	Ambiguity		Tag. Acc.			F-score			Speed (sents / sec)		
	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio	News	Wiki	Bio
Baseline	1.317	1.281	96.34	94.52	90.70	85.46	80.8	75.0	39.6	50.9	35.1
News	1.331	1.322	**96.53**	**94.86**	**91.32**	**85.84**	80.1	75.2	41.8	32.6	31.4
Wiki	1.293	1.251	96.28	94.79	91.08	85.02	**81.7**	75.8	40.4	37.2	37.2
Bio	1.287	1.195	96.15	94.28	91.03	84.95	80.6	**76.1**	39.2	52.9	26.2

Table 7: Cross-corpus accuracy optimisation, models trained with GIS and 400,000 sentences.

Annotation method	Tag. Acc.	F-score
Baseline	96.34	85.46
Parser	96.46	85.55
One-best super	95.94	85.24
Multi-tagger a	95.91	84.98
Multi-tagger b	96.00	84.99

Table 8: Comparison of annotation methods for extra data. The multi-taggers used β values 0.075 and 0.001, and dictionary cutoffs 20 and 150, for taggers a and b respectively.

Corpus	Speed (sents / sec)		
Sent length	5-20	21-40	41-250
News	242	44.8	8.24
Wiki	224	42.0	6.10
Bio	268	41.5	6.48

Table 9: Cross-corpus speed for the baseline model on data sets balanced on sentence length.

As Table 8 shows, in all cases the use of supertagger-annotated data led to poorer performance than the baseline system, while the use of parser-annotated data led to an improvement in F-score. The parser has access to a range of information that the supertagger does not, producing a different view of the data that the supertagger can productively learn from.

6.4 Cross-domain speed improvement

When applying parsers out of domain they are typically slower and less accurate (Gildea, 2001). In this experiment, we attempt to increase speed on out-of-domain data. Note that for some of the results presented here it may appear that the C&C parser does not lose speed when out of domain, since the Wikipedia and biomedical corpora contain shorter sentences on average than the news corpus. However, by testing on balanced sets it is clear that speed does decrease, particularly for longer sentences, as shown in Table 9.

For our domain adaptation development experiments, we considered a collection of different models; here we only present results for the best set of models. For speed improvement these were MIRA models trained on 4,000,000 parser-annotated sentences from the target domain.

As Table 5 shows, this training method produces models adapted to the new domain. In particular, note that models trained on Wikipedia or the biomedical data produce lower F-scores[3] than the baseline on newswire. Meanwhile, on the target domain they are adapted to, these models achieve a higher F-score and parse sentences at least 45% faster than the baseline.

The changes in tagging ambiguity and accuracy also show that adaptation has occurred. In all cases, the new models have lower tagging ambiguity, and lower supertag accuracy. However, on the corpus of the extra data, the performance of the adapted models is comparable to the baseline model, which means the parser is probably still be receiving the same categories that it used from the sets provided by the baseline system.

6.5 Cross-domain accuracy optimised

The ambiguity tuning method used to improve accuracy on the newspaper domain can also be applied to the models trained on other domains. In Table 7, we have tested models trained using GIS and 400,000 sentences of parsed target-domain text, with β levels tuned to match ambiguity with the baseline.

As for the newspaper domain, we observe increased supertag accuracy and F-score. Also, in almost every case the new models perform worse than the baseline on domains other than the one they were trained on.

In some cases the models in Table 7 are less accurate than those in Table 5. This is because as well as optimising the β levels we have changed training methods. All of the training methods were tried, but only the method with the best results in newswire is included here, which for F-score when trained on 400,000 sentences was GIS.

The accuracy presented so far for the biomedi-

[3] Note that the F-scores for Wikipedia and biomedical text are reported to only three significant figures as only 300 and 500 sentences respectively were available for parser evaluation.

Train Corpus	F-score
Rimell and Clark (2009)	81.5
Baseline	80.7
CCGbank + Genia	81.5
+ Newswire	81.9
+ Wikipedia	82.2
+ Biomedical	81.7
+ R&C annotated Bio	82.3

Table 10: Performance comparison for models using extra gold standard biomedical data. Models were trained with GIS and 4,000,000 extra sentences, and are tested using a POS-tagger trained on biomedical data.

cal model is considerably lower than that reported by Rimell and Clark (2009). This is because no gold standard biomedical training data was used in our experiments. Table 10 shows the results of adding Rimell and Clark's gold standard biomedical supertag data and using their biomedical POS-tagger. The table also shows how accuracy can be further improved by adding our parser-annotated data from the biomedical domain as well as the additional gold standard data.

7 Conclusion

This work has demonstrated that an adapted supertagger can improve parsing speed and accuracy. The purpose of the supertagger is to reduce the search space for the parser. By training the supertagger on parser output, we allow the parser to reach the derivation it would have found, sooner. This approach also enables domain adaptation, improving speed and accuracy outside the original domain of the parser.

The perceptron-based algorithms used in this work are also able to function online, modifying the model weights after each sentence is parsed. This could be used to construct a system that continuously adapts to the domain it is parsing.

By training on parser-annotated NANC data we constructed models that were adapted to the newspaper-trained parser. The fastest model parsed sentences 1.85 times as fast and was as accurate as the baseline system. Adaptive training is also an effective method of improving performance on other domains. Models trained on parser-annotated Wikipedia text and MEDLINE text had improved performance on these target domains, in terms of both speed and accuracy. Optimising for speed or accuracy can be achieved by modifying the β levels used by the supertagger, which controls the lexical category ambiguity at each level used by the parser.

The result is an accurate and efficient wide-coverage CCG parser that can be easily adapted for NLP applications in new domains without manually annotating data.

Acknowledgements

We thank the reviewers for their helpful feedback. This work was supported by Australian Research Council Discovery grants DP0665973 and DP1097291, the Capital Markets Cooperative Research Centre, and a University of Sydney Merit Scholarship. Part of the work was completed at the Johns Hopkins University Summer Workshop and (partially) supported by National Science Foundation Grant Number IIS-0833652.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Phil Blunsom and Timothy Baldwin. 2006. Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Sydney, Australia.

Ted Briscoe and John Carroll. 2006. Evaluating the accuracy of an unlexicalized statistical parser on the PARC DepBank. In *Proceedings of the Poster Session of the 21st International Conference on Computational Linguistics*, Sydney, Australia.

John Chen, Srinivas Bangalore, and Vijay K. Shanker. 1999. New models for improving supertag disambiguation. In *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 188–195, Bergen, Norway.

John Chen, Srinivas Bangalore, Michael Collins, and Owen Rambow. 2002. Reranking an n-gram supertagger. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 259–268, Venice, Italy.

Nancy Chinchor. 1995. Statistical significance of MUC-6 results. In *Proceedings of the Sixth Message Understanding Conference*, pages 39–43, Columbia, MD, USA.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 282–288, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Stephen Clark, James R. Curran, and Miles Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. In *Proceedings of the seventh Conference on Natural Language Learning*, pages 49–55, Edmonton, Canada.

Michael Collins and Terry Koo. 2002. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Philadelphia, PA, USA.

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

James R. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

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

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 449–54, Genoa, Italy.

Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th International ACMSIGIR Conference on Research and Development*, Tampere, Finland.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, Pittsburgh, PA, USA.

David Graff. 1995. North American News Text Corpus. LDC95T21. Linguistic Data Consortium. Philadelphia, PA, USA.

Hany Hassan, Khalil Sima'an, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 288–295, Prague, Czech Republic.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Kristy Hollingshead and Brian Roark. 2007. Pipeline iteration. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics*, pages 952–959, Prague, Czech Republic.

Jin-Dong Kim, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(1):180–182.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 Dependency Bank. In *Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006a. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, Brooklyn, NY, USA.

David McClosky, Eugene Charniak, and Mark Johnson. 2006b. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 337–344, Sydney, Australia.

Tara McIntosh and James R. Curran. 2008. Weighted mutual exclusion bootstrapping for domain independent lexicon and template acquisition. In *Proceedings of the Australasian Language Technology Workshop*, Hobart, Australia.

Jorge Nocedal and Stephen J. Wright. 1999. *Numerical Optimization*. Springer.

Sampo Pyysalo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. 2007. On the unification of syntactic annotations under the Stanford dependency scheme: a case study on bioinfer and GENIA. In *Proceedings of the ACL workshop on biological, translational, and clinical language processing*, pages 25–32, Prague, Czech Republic.

Adwait Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing*, pages 133–142, Philadelphia, PA, USA.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 475–484, Honolulu, HI, USA.

Laura Rimell and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865.

Anoop Sarkar, Fel Xia, and Aravind K. Joshi. 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *Proceedings of the COLING Workshop on Efficiency in Large-scale Parsing Systems*, pages 37–42, Luxembourg.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 1–8, Pittsburgh, PA, USA.

Anoop Sarkar. 2007. Combining supertagging and lexicalized tree-adjoining grammar parsing. In Srinivas Bangalore and Aravind Joshi, editors, *Complexity of Lexical Descriptions and its Relevance to Natural Language Processing: A Supertagging Approach*. MIT Press, Boston, MA, USA.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Stephen Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 331–338, Budapest, Hungary.

Geertjan van Noord. 2009. Learning efficient parsing. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 817–825. Association for Computational Linguistics.

Yao-zhong Zhang, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. HPSG supertagging: A sequence labeling view. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 210–213, Paris, France.