Supertagging for Combinatory Categorial Grammar

Stephen Clark
Division of Informatics, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW
Scotland, UK
`Stephen.Clark@ed.ac.uk`

1. Introduction

Supertagging was introduced for Lexicalised Tree Adjoining Grammar (LTAG) to reduce the number of elementary trees assigned to a word, thereby increasing parsing efficiency (Bangalore and Joshi, 1994). Bangalore and Joshi (1999) have shown that techniques used for POS-tagging can be applied successfully to the supertagging problem.

Parsing efficiency is also an issue for Combinatory Categorial Grammar (CCG, Steedman (2000)) since many words can have many possible CCG categories. We have developed a supertagger for CCG, similar to the POS-tagger of Ratnaparkhi (1996). Maximum entropy models are used to estimate the probability that a word is assigned a particular category, given the local context. These probabilities are then used to select a sub-set of the possible categories for a word.

The next section gives a brief introduction to CCG, and Section 3 describes the category set used in the experiments. Section 4 describes a "single-category" supertagger, and gives figures for its accuracy. Section 5 shows how the supertagger can be adapted to output more than one category per word, to produce a "multi-tagger", and we show the effect the multi-tagger has on the speed and coverage of a CCG parser.

2. Combinatory Categorial Grammar

A grammar in CCG consists of a lexicon, which pairs words with lexical categories, and a set of combinatory rules, which specify how categories combine. Categories are either atomic or complex. Examples of atomic categories include S (sentence), N (noun), NP (noun phrase) and PP (prepositional phrase). Features on the S category can be used to indicate types of sentence (or clause); for example: S[dcl] (declarative), S[to] (to-infinitival), S[b] (bare-infinitival), S[adj] (adjectival).

Complex categories are functors which specify the type and directionality of the arguments, and the type of the result. For example, one of the categories for the verb *likes* specifies that one noun phrase (NP) is required to the right of the verb, and one to the left, resulting in a sentence (as in *John likes sweets*):

likes := (S[dcl]\NP)/NP

Another category for *likes* specifies that a to-infinitival clause is required to the right of the verb (as in *John likes to eat sweets*):

likes := (S[dcl]\NP)/(S[to]\NP)

Functor categories can also express modification, as in the following adverbial category for *really*:

really := (S\NP)/(S\NP)

The following derivation shows how categories combine. This derivation uses only two combinatory rules: forward application (>) and backward application (<).

John	really	likes	to	eat	sweets
NP	(S\NP)/(S\NP)	(S[dcl]\NP)/(S[to]\NP)	(S[to]\NP)/(S[b]\NP)	(S[b]\NP)/NP	NP
				S[b]\NP (>)	
			S[to]\NP (>)		
		S[dcl]\NP (>)			
	S[dcl]\NP (>)				
S[dcl] (<)					

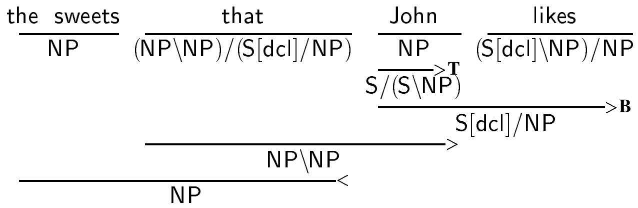
Further combinatory rules are needed to deal with syntactic phenomena such as coordination and extraction.

 Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 101–106. Universitá di Venezia.

frequency cut-off	# cat types in cat-set	# cat tokens in 2-21 not in cat-set	# sentences in 2-21 with missing cat	# cat tokens in 00 not in cat-set	# sentences in 00 with missing cat
1	1,206	0	0	11 (0.02%)	11 (0.6%)
5	512	1,157 (0.1%)	1,032 (2.6%)	49 (0.1%)	44 (2.3%)
10	398	1,898 (0.2%)	1,667 (4.3%)	79 (0.2%)	67 (3.5%)
20	304	3,190 (0.4%)	2,756 (7.0%)	123 (0.3%)	104 (5.5%)

Table 1: Category coverage for seen data (sections 2-21) and unseen data (section 00) under various cut-offs

In the following object-extraction example, *type-raising* (>**T**) turns the atomic NP category for *John* into a functor category looking for a verb-phrase to its right; *forward composition* (>**B**) then combines the type-raised category with the category for *likes*:



Note that, in this paper, we assume type-raising is dealt with by the parser rather than at the level of the lexicon. Thus the supertagger does not deal with type-raised categories.

3. The Lexical Category Set

The category set used here has been derived from a treebank of CCG (normal-form) derivations, in which each word is tagged with a lexical category.[1] The treebank, which we call CCG-bank, has been created by Julia Hockenmaier (Hockenmaier and Steedman, 2002a), and derived semi-automatically from the Penn Treebank (Marcus, Santorini and Marcinkiewicz, 1993). Below is an example sentence from the CCG-bank, together with the lexical categories.[2]

Pierre N/N Vinken N , , 61 N/N years N old (S[adj]\NP)\NP , , will (S[dcl]\NP)/(S[b]\NP)
join ((S[b]\NP)/PP)/NP the NP/N board N as PP/NP a NP/N nonexecutive N/N
director N Nov. ((S\NP)\(S\NP))/N 29 N . .

The category set was obtained from sections 2-21 of the CCG-bank (corresponding to the Penn Treebank sections), which contain 39,161 sentences, 903,661 category tokens, and 1,206 category types. Many of the category types occur only a few times in the data, and some arise through noise in the Penn Treebank, or errors introduced by the CCG extraction program. We therefore investigated using a subset of the 1,206 category types by applying a frequency cut-off.

Table 1 shows how many category tokens in Sections 2-21 do not appear in the category set entailed by the cut-off, and also shows the percentage of sentences that have at least one category token not in the set. The same figures are given for Section 00, which has 1,900 sentences and 44,544 category tokens. The figures show that the size of the category set can be reduced by as much as 3/4, without greatly increasing the percentage of missing category tokens in the data (unseen and seen).

The LTAG supertag set used by Bangalore and Joshi (1999) contained 300 supertags (for their WSJ experiments). To obtain a CCG category set as compact as that requires a cut-off as high as 20. However, Bangalore and Joshi took their supertags from the manually created XTAG grammar, which presumably contains a much cleaner set of supertags than an automatically extracted grammar.

A more useful comparison is with the work of Chen and Vijay-Shanker (2000), who extract an LTAG automatically from the Penn Treebank. A number of strategies are used for extracting the supertags (referred to as *tree*

1. There is a distinction between lexical categories and categories created during a derivation. Since we are only interested in lexical categories in this paper, we may sometimes use *category* to mean *lexical category*.
2. The category for *join* has a PP-complement, which is arguably incorrect. Since the distinction between complements and adjuncts is not always reliably marked in the Penn Treebank, the procedure for identifying complements and adjuncts does lead to some erroneous data in the CCG-bank.

Experiment 1		Experiment 2		Experiment 3		Experiment 4	
Condition	Contextual predicates						
w_i is not rare	$w_i = X$	$\forall w_i$	$t_i = X$	$\forall w_i$	$t_i = X$	$\forall w_i$	$t_i = X$
w_i is rare	X is prefix of w_i, $\lvert X \rvert \leq 4$		$t_{i-1} = X$		$t_{i-1} = X$		$t_{i-1} = X$
	X is suffix of w_i, $\lvert X \rvert \leq 4$		$t_{i-2} = X$		$t_{i-2} = X$		$t_{i-2} = X$
	w_i contains a number		$t_{i+1} = X$		$t_{i+1} = X$		$t_{i+1} = X$
	w_i contains uppercase character		$t_{i+2} = X$		$t_{i+2} = X$		$t_{i+2} = X$
	w_i contains a hyphen				$w_{i-3} = X$		$c_{i-2}c_{i-1} = XY$
$\forall w_i$	$c_{i-1} = X$				$w_{i+3} = X$		$w_{i-2}w_{i-1} = XY$
	$c_{i-2} = X$				$t_{i-3} = X$		$w_{i+1}w_{i+2} = XY$
	$w_{i-1} = X$				$t_{i+3} = X$		$t_{i-2}t_{i-1} = XY$
	$w_{i-2} = X$						$t_{i+1}t_{i+2} = XY$
	$w_{i+1} = X$						
	$w_{i+2} = X$						

Table 2: Contextual predicates used in the experiments. The predicates shown for experiments 2, 3 and 4 are in addition to those used in 1; the rare word predicates are only used in experiment 1.

frames by Chen and Vijay-Shanker), but the size of the resulting sets range between 2,366 and 8,996 supertags. Chen and Vijay-Shanker also experimented with a cut-off, with a value of 3 reducing the size of the supertag sets by at least 1/2, and a cut-off of 9 producing sets ranging between around 800 and 1,800 supertags. These numbers suggest that the CCG category sets extracted from CCG-bank are more compact than Chen and Vijay-Shanker's LTAG supertag sets (for equivalent cut-off values), although we should point out that CCG-bank has received a significant amount of manual clean-up.

4. The supertagger

The supertagger uses conditional maximum entropy models to estimate the probability of words being assigned particular categories. We chose to implement a maximum entropy supertagger, rather than the HMM supertagger used by Bangalore and Joshi (1999), because of the ease with which additional features can be integrated into the model. The use of conditional models, rather than the generative model of the HMM, also makes it easy to define a "multi-tagger", as we show in Section 5.

The probability of a category, c, given a context, h, is defined as follows:

$$p(c|h) = \frac{1}{Z(h)} e^{\sum_i \lambda_i f_i(c,h)} \tag{1}$$

The functions $f_i(c,h)$ define "features" of the category and context, and $Z(h)$ is a normalisation constant. An example feature is as follows:

$$f_j(c,h) = \begin{cases} 1 & \text{if } \texttt{curr\_word\_is\_the}(h) = \mathit{true} \ \& \ c = \mathsf{NP/N} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The feature function takes the value 1 if the current word (i.e. word to be tagged) is *the* and the category is $\mathsf{NP/N}$. The weight (λ_j) corresponding to the feature contributes to the probability $p(c|h)$ when h contains *the* as the current word and $c = \mathsf{NP/N}$. Generalised Iterative Scaling is used to estimate the values of the weights.

The predicate `curr_word_is_the` is an example of what Ratnaparkhi (1998) calls a *contextual predicate*. Contextual predicates identify elements of the context that might be useful in predicting the lexical category. Possible contextual predicates for the supertagger include the current word, certain properties of the current word (such as suffix and prefix information), the POS-tag of the current word, words either side of the current word, and the POS-tags of those words.

We experimented with a number of contextual predicates, and different window sizes for the context. Table 2 gives "templates" for the contextual predicates used in a series of experiments. The first set used in experiment 1 is based on that used by Ratnaparkhi (1998). (The notation in the table is also borrowed from Ratnaparkhi (1998).) The current word, w_i, is *rare* if it appears less than a certain number of times in the training data. A cut-off value

	expt.1	expt.2	expt.3	expt.4
CCG supertagger	88.1	90.4	90.5	90.5
Baseline	71.2			
CCG (Nielsen)	87.7			
CCG (TnT)	87.8			
LTAG (Srinivas)	92.2			
LTAG (Chen)	78.9			

Table 3: Results for the single supertagger

of 5 was used here. If w_i is not rare, the word itself forms a contextual predicate, otherwise contextual predicates based on the suffixes and prefixes (up to 4 characters) are used, plus some others listed in the table. In addition, the two categories before w_i (c_{i-1}, c_{i-2}) form contextual predicates, along with the two words before and after w_i. Thus in experiment 1 a 5-word window is providing the context.

Experiment 2 introduces the POS-tags (t_j) of the words in the 5-word window, but does not use the rare word features, since we found that ignoring the rare word features once the POS-tags had been introduced did not affect the results. The POS-tagger of Ratnaparkhi (1998) was used to provide the POS-tags. Experiment 3 extends the context to a 7-word window. Experiment 4 retains the 5-word window, but forms complex features by considering pairs of categories, words, and POS-tags.

Following Ratnaparkhi (1998), a simple feature selection technique was used for all the experiments: only those features that appear at least 10 times in the training data are considered (except the current word features which only have to appear 5 times).

Initially we developed a supertagger that chooses the most probable sequence of categories, given the sentence. The probability of a category sequence, C, given a sentence S, is defined as follows:

$$p(C|S) = \prod_i p(c_i|h_i) \tag{3}$$

where c_i is the ith category in the sequence and h_i is the context for the ith word. A beam search is used, so that only the top N sequences are retained at each point in the tagging process. A value of $N = 10$ was used here.

The supertagger consults a "tag-dictionary", which contains, for each word, the set of categories the word was seen with in the data. If a word appears at least K times, the supertagger only considers the categories in the word's category set. If a word appears less than K times, all categories are considered.

Table 3 gives the results on Section 23 of the CCG-bank (ignoring punctuation) using the category set with 398 types, and a value of $K = 20$ for the tag dictionary.[3] Sections 2-21 were used for training. The baseline result is obtained by choosing the category the current word appears with most frequently in the data, and assigning category N to unseen words.

Results are also given for two HMM supertaggers: Brants' TnT tagger (Brants, 2000), and a supertagger developed by Nielsen (2001).[4] The Nielsen figure was obtained using a slightly older version of the CCG-bank, and does include punctuation. The Brants' figure was obtained by training the publically available version of TnT on the latest version of the CCG-bank, and does not include punctuation.[5] The HMM results are similar to those for the maximum entropy experiment 1, which uses a similar set of features. The improvement in experiment 2 over the HMM supertaggers is because the HMM supertaggers did not use POS-tag information.

Results are also given for LTAG supertagging (Bangalore and Joshi, 1999; Chen and Vijay-Shanker, 2000) on the Penn Treebank. The Srinivas supertagger uses a manually created set of elementary trees, whereas the Chen supertagger uses an automatically extracted tag set, which explains the difference in the LTAG results.

We have integrated the supertagger with a CCG parser (Hockenmaier and Steedman, 2002b), and attempted to parse the sentences from the Penn Treebank. If the supertagger assigns a single category to each word, the parser is able to provide an analysis for around 91% of the sentences in Section 23. To try and increase the coverage we investigated assigning more than one category to each word.

3. Very similar results were obtained for values of K between 5 and 100 using Section 00 as a development set.
4. The TnT tagger was designed for POS-tagging, but it can be easily adapted to the supertagging problem by simply training it on supertagged data.
5. Including punctuation increases the figure by around 1%, because the categories for punctuation marks such as comma and period are simply the punctuation marks themselves.

CCG multi-tagger		# cats/word	LTAG multi-tagger	# cats/word
$\beta = 0.1$	96.2	1.6	94.6	1.9
$\beta = 0.05$	97.2	2.0	95.8	2.2
$\beta = 0.01$	98.4	3.8	97.0	3.4

Table 4: Results for the multi-tagger

5. Using the supertagger as a multi-tagger

A feature of the conditional model in equation 1 is that it is very easy to define a multi-tagger: simply assign all categories to a word whose probabilities (given by equation 1) are within some factor, β, of the highest probability category. Note that equation 3 is no longer used, since we are no longer estimating the probability of a sequence of categories. The feature set used in experiment 4 for the single tagger was used (but without the previous category features), with the category set of 398 category types, and $K = 20$. The results on Section 23 are given in Table 4, for various values of β, together with the average number of categories per word.

The LTAG results of Chen, Bangalore and Vijay-Shanker (1999), using a supertag set derived from the XTAG grammar, are given for comparison. Table 3 showed the single tagger results to be better for LTAG, at least for this supertag set (Chen et al. report a similar result), but the multi-tagger results are better for CCG, using a much simpler approach. Chen, Bangalore and Vijay-Shanker (1999) used a clustering technique to assign more than one category to each word.

Using a multi-tagger rather than a single-tagger in conjunction with the Hockenmaier and Steedman (2002a) parser greatly increases the coverage. For a β value of 0.01, over 99% of Section 23 could be parsed. To give some indication of how changes in β affect the speed of the parser, we took the first 500 sentences from Section 23 and recorded the parse times for various values of β. The times depend on various parameter settings used by the parser, such as beam width, but the figures reported here are intended only to give a rough indication of how the supertagger can speed up the parser. A β value of 0.0001 (which gives an average of 28 categories per word) was used to provide a baseline figure. Changing β to 0.01 (3.8 categories per word) increased the speed of the parser by a factor of 2.7 over the baseline figure. Changing β to 0.1 (1.6 categories per word) increased the speed of the parser by a factor of 4.7 over the baseline figure (but with a slight loss in coverage).

6. Conclusion

The supertagger described here can be used to significantly speed up a CCG parser. However, assigning a single category to each word in a sentence allows the parser of Hockenmaier and Steedman (2002a) to provide an analysis for only 91% of the sentences from Section 23 of the Penn Treebank. A simple multi-tagging approach can greatly increase the coverage, while keeping the average number of categories to a manageable number. The accuracy of the supertagger compares favourably with existing LTAG supertaggers, particularly those which use an automatically extracted LTAG.

One question which has not been discussed in this paper is whether supertagging is easier for LTAG or CCG. The comparison with the work of Chen and Vijay-Shanker (2000) suggests that it may be easier to do supertagging with an automatically extracted CCG than with an automatically extracted LTAG, although of course this depends crucially on the extracted grammars in each case.

There is also the question of how much work remains to be done by the parser after supertagging has taken place. Bangalore (2000) argues that, for LTAG, simple heuristics can be used to satisfy the dependency requirements encoded in the supertags. Whether such an approach would work for CCG remains an open question.

7. Acknowledgements

This research was funded by EPSRC grant GR/M96889/01. I would like to thank Julia Hockenmaier for providing the CCG-bank and for getting the results for her parser, and Mark Steedman for his guidance and help with this work.

References

Bangalore, Srinivas. 2000. A Lightweight Dependency Analyser for Partial Parsing. *Natural Language Engineering*, 6(2):113–138.

Bangalore, Srinivas and Aravind Joshi. 1994. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the 15th COLING Conference*, pages 154–160, Kyoto, Japan.

Bangalore, Srinivas and Aravind Joshi. 1999. Supertagging: An Approach to Almost Parsing. *Computational Linguistics*, 25(2):237–265.

Brants, Thorsten. 2000. TnT - a statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*.

Chen, John, Srinivas Bangalore and K. Vijay-Shanker. 1999. New Models for Improving Supertag Disambiguation. In *Proceedings of the 9th Meeting of EACL*, Bergen, Norway.

Chen, John and K. Vijay-Shanker. 2000. Automated Extraction of TAGS from the Penn Treebank. In *Proceedings of IWPT 2000*, Trento, Italy.

Hockenmaier, Julia and Mark Steedman. 2002a. Acquiring Compact Lexicalized Grammars from a Cleaner Treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (to appear)*, Las Palmas, Spain.

Hockenmaier, Julia and Mark Steedman. 2002b. Generative Models for Statistical Parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL (to appear)*, Philadelphia, PA.

Marcus, Mitchell, Beatrice Santorini and Mary Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Nielsen, Lief. 2001. Supertagging with Categorial Grammar. Master's thesis, University of Cambridge.

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Part-Of-Speech Tagger. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.

Ratnaparkhi, Adwait. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

Steedman, Mark. 2000. *The Syntactic Process*. Cambridge, MA: The MIT Press.