Improving Dependency Parsers using Combinatory Categorial Grammar

Bharat Ram Ambati **Tejaswini Deoskar** **Mark Steedman**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
bharat.ambati@ed.ac.uk, {tdeoskar,steedman}@inf.ed.ac.uk

Abstract

Subcategorization information is a useful feature in dependency parsing. In this paper, we explore a method of incorporating this information via Combinatory Categorial Grammar (CCG) categories from a supertagger. We experiment with two popular dependency parsers (Malt and MST) for two languages: English and Hindi. For both languages, CCG categories improve the overall accuracy of both parsers by around 0.3-0.5% in all experiments. For both parsers, we see larger improvements specifically on dependencies at which they are known to be weak: long distance dependencies for Malt, and verbal arguments for MST. The result is particularly interesting in the case of the fast greedy parser (Malt), since improving its accuracy without significantly compromising speed is relevant for large scale applications such as parsing the web.

1 Introduction

Dependency parsers can recover much of the predicate-argument structure of a sentence, while being relatively efficient to train and extremely fast at parsing. Dependency parsers have been gaining in popularity in recent times due to the availability of large dependency treebanks for several languages and parsing shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007a; Bharati et al., 2012).

Ambati et al. (2013) showed that the performance of Malt (Nivre et al., 2007b) on the free word order language, Hindi, is improved by using lexical categories from Combinatory Categorial Grammar (CCG) (Steedman, 2000). In this paper, we extend this work and show that CCG categories are useful even in the case of English, a typologically different language, where parsing accuracy of dependency parsers is already extremely high. In addition, we also demonstrate the utility of CCG categories to MST (McDonald et al., 2005) for both languages. CCG lexical categories contain subcategorization information regarding the dependencies of predicates, including long-distance dependencies. We show that providing this subcategorization information in the form of CCG categories can help both Malt and MST on precisely those dependencies for which they are known to have weak rates of recovery. The result is particularly interesting for Malt, the fast greedy parser, as the improvement in Malt comes without significantly compromising its speed, so that it can be practically applied in web scale parsing. Our results apply both to English, a fixed word order and morphologically simple language, and to Hindi, a free word order and morphologically rich language, indicating that CCG categories from a supertagger are an easy and robust way of introducing lexicalized subcategorization information into dependency parsers.

2 Related Work

Parsers using different grammar formalisms have different strengths and weaknesses, and prior work has shown that information from one formalism can improve the performance of a parser in another formalism. Sagae et al. (2007) achieved a 1.4% improvement in accuracy over a state-of-the-art HPSG parser by using dependencies from a dependency parser for constraining wide-coverage rules in the HPSG parser. Coppola and Steedman (2013) incorporated higher-order dependency features into a cube decoding phrase-structure parser and obtained significant gains on dependency recovery for both in-domain and out-of-domain test sets.

Kim et al. (2012) improved a CCG parser using dependency features. They extracted n-best parses from a CCG parser and provided dependency

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 159–163,
Gothenburg, Sweden, April 26-30 2014.

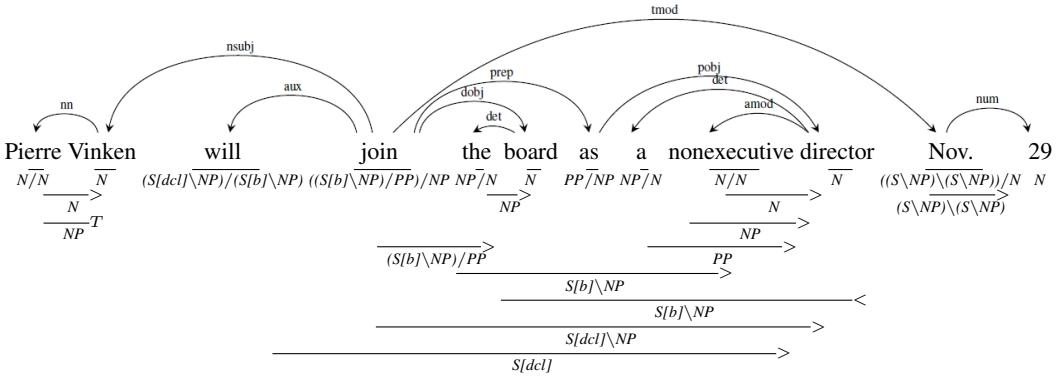


Figure 1: A CCG derivation and the Stanford scheme dependencies for an example sentence.

features from a dependency parser to a re-ranker with an improvement of 0.35% in labelled F-score of the CCGbank test set. Conversely, Ambati et al. (2013) showed that a Hindi dependency parser (Malt) could be improved by using CCG categories. Using an algorithm similar to Cakici (2005) and Uematsu et al. (2013), they first created a Hindi CCGbank from a Hindi dependency treebank and built a supertagger. They provided CCG categories from a supertagger as features to Malt and obtained overall improvements of 0.3% and 0.4% in unlabelled and labelled attachment scores respectively.

3 Data and Tools

Figure 1 shows a CCG derivation with CCG lexical categories for each word and Stanford scheme dependencies (De Marneffe et al., 2006) for an example English sentence. (Details of CCG and dependency parsing are given by Steedman (2000) and Kübler et al. (2009).)

3.1 Treebanks

In English dependency parsing literature, Stanford and CoNLL dependency schemes are widely popular. We used the Stanford parser's built-in converter (with the basic projective option) to generate Stanford dependencies and Penn2Malt[1] to generate CoNLL dependencies from Penn Treebank (Marcus et al., 1993). We used standard splits, training (sections 02-21), development (section 22) and testing (section 23) for our experiments. For Hindi, we worked with the Hindi Dependency Treebank (HDT) released as part of Coling 2012 Shared Task (Bharati et al., 2012). HDT contains 12,041 training, 1,233 development and 1,828 testing sentences.

We used the English (Hockenmaier and Steedman, 2007) and Hindi CCGbanks (Ambati et al., 2013) for our experiments. For Hindi we used two lexicons: a fine-grained one (with morphological information) and a coarse-grained one (without morphological information).

3.2 Supertaggers

We used Clark and Curran (2004)'s supertagger for English, and Ambati et al. (2013)'s supertagger for Hindi. Both are Maximum Entropy based CCG supertaggers. The Clark and Curran (2004) supertagger uses different features like word, part-of-speech, and contextual and complex bi-gram features to obtain a 1-best accuracy of 91.5% on the development set. In addition to the above mentioned features, Ambati et al. (2013) employed morphological features useful for Hindi. The 1-best accuracy of Hindi supertagger for fine-grained and coarse-grained lexicon is 82.92% and 84.40% respectively.

3.3 Dependency Parsers

There has been a significant amount of work on parsing English and Hindi using the Malt and MST parsers in the recent past (Nivre et al., 2007a; Bharati et al., 2012). We first run these parsers with previous best settings (McDonald et al., 2005; Zhang and Nivre, 2012; Bharati et al., 2012) and treat them as our baseline. In the case of English, Malt uses arc-standard and stack-projective parsing algorithms for CoNLL and Stanford schemes respectively and LIBLINEAR learner (Fan et al., 2008) for both the schemes. MST uses 1st-order features, and a projective parsing algorithm with 5-best MIRA training for both the schemes. For Hindi, Malt uses the arc-standard parsing algorithm with a LIBLINEAR learner. MST uses 2nd-order features, non-projective algorithm with 5-best MIRA training.

For English, we assigned POS-tags using a perceptron tagger (Collins, 2002). For Hindi, we also did all our experiments using automatic features

[1] http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

Language	*Experiment*	*Malt*		*MST*	
		UAS	*LAS*	*UAS*	*LAS*
English	Stanford Baseline	90.32	87.87	90.36	87.18
	Stanford + CCG	**90.56** (2.5)**	**88.16** (2.5)**	**90.93** (5.9)**	**87.73** (4.3)**
	CoNLL Baseline	89.99	88.73	90.94	89.69
	CoNLL + CCG	**90.38** (4.0)**	**89.19** (4.1)**	**91.48** (5.9)**	**90.23** (5.3)**
Hindi	Baseline	88.67	83.04	90.52	80.67
	Fine CCG	88.93** (2.2)	83.23* (1.1)	**90.97** (4.8)**	**80.94* (1.4)**
	Coarse CCG	**89.04** (3.3)**	**83.35* (1.9)**	90.88** (3.8)	80.73* (0.4)

Table 1: Impact of CCG categories from a supertagger on dependency parsing. Numbers in brackets are percentage of errors reduced. McNemar's test compared to baseline, * = $p < 0.05$; ** = $p < 0.01$ (Hindi Malt results (grey background) are from Ambati et al. (2013)).

(POS, chunk and morphological information) extracted using a Hindi shallow parser[2].

4 CCG Categories as Features

Following Ambati et al. (2013), we used supertags which occurred at least K times in the training data, and backed off to coarse POS-tags otherwise. For English K=1, i.e., when we use CCG categories for all words, gave the best results. K=15 gave the best results for Hindi due to sparsity issues, as the data for Hindi is small. We provided a supertag as an atomic symbol similar to a POS tag and didn't split it into a list of argument and result categories. We explored both Stanford and CoNLL schemes for English and fine and coarse-grained CCG categories for Hindi. All feature and parser tuning was done on the development data. We assigned automatic POS-tags and supertags to the training data.

4.1 Experiments with Supertagger output

We first used gold CCG categories extracted from each CCGbank as features to the Malt and MST, to get an upper bound on the utility of CCG categories. As expected, gold CCG categories boosted the Unlabelled Attachment Score (UAS) and Labelled Attachment Score (LAS) by a large amount (4-7% in all the cases).

We then experimented with using automatic CCG categories from the English and Hindi supertaggers as a feature to Malt and MST. With automatic categories from a supertagger, we got statistically significant improvements (McNemar's test, $p < 0.05$ for Hindi LAS and $p < 0.01$ for the rest) over the baseline parsers, for all cases (Table 1). Since the CCGbanks used to train the supertaggers are automatically generated from the constituency or dependency treebanks used to train the dependency parsers, the improvements are indeed due to reparameterization of the model to include CCG categories and not due to additional hand annotations in the CCGbanks. This shows that the rich subcategorization information provided by automatically assigned CCG categories can help Malt and MST in realistic applications.

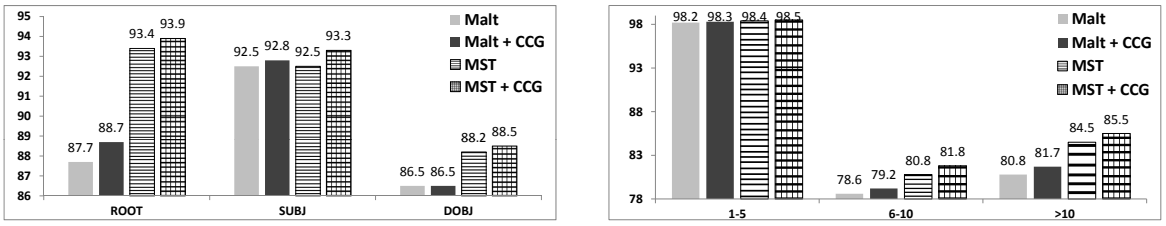
For English, in case of Malt, we achieved 0.3% improvement in both UAS and LAS for Stanford scheme. For CoNLL scheme, these improvements were 0.4% and 0.5% in UAS and LAS respectively. For MST, we got around 0.5% improvements in all cases.

In case of Hindi, fine-grained supertags gave larger improvements for MST. We got final improvements of 0.5% and 0.3% in UAS and LAS respectively. In contrast, for Malt, Ambati et al. (2013) had shown that coarse-grained supertags gave larger improvements of 0.3% and 0.4% in UAS and LAS respectively. Due to better handling of error propagation in MST, the richer information in fine-grained categories may have surpassed the slightly lower supertagger performance, compared to coarse-grained categories.

4.2 Analysis: English

We analyze the impact of CCG categories on different labels (label-wise) and distance ranges (distance-wise) for CoNLL scheme dependencies (We observed a similar impact for the Stanford scheme dependencies as well). Figure 2a shows the F-score for three major dependency labels, namely, `ROOT` (sentence root), `SUBJ` (subject), `OBJ` (object). For Malt, providing CCG categories gave an increment of 1.0%, 0.3% for `ROOT` and `SUBJ` labels respectively. For MST, the improvements for `ROOT` and `SUBJ` were 0.5% and 0.8% respectively. There was no significant improvement for `OBJ` label, especially in the case of Malt.

[2]http://ltrc.iiit.ac.in/analyzer/hindi/



(a) Label-wise impact

(b) Distance-wise impact

Figure 2: Label-wise and Distance-wise impact of supertag features on Malt and MST for English

Figure 2b shows the F-score of dependencies based on the distance ranges between words. The percentage of dependencies in the $1-5$, $6-10$ and >10 distance ranges are 88.5%, 6.6% and 4.9% respectively out of the total of around 50,000 dependencies. For both Malt and MST, there was very slight improvement for short distance dependencies ($1-5$) but significant improvements for longer distances ($6-10$ and >10). For Malt, there was an improvement of 0.6% and 0.9% for distances $6-10$, and >10 respectively. For MST, these improvements were 1.0% and 1.0% respectively.

4.3 Analysis: Hindi

In the case of Hindi, for MST, providing CCG categories gave an increment of 0.5%, 0.4% and 0.3% for `ROOT`, `SUBJ` and `OBJ` labels respectively in F-score over the baseline. Ambati et al. (2013) showed that for Hindi, providing CCG categories as features improved Malt in better handling of long distance dependencies.

The percentage of dependencies in the $1-5$, $6-10$ and >10 distance ranges are 82.2%, 8.6% and 9.2% respectively out of the total of around 40,000 dependencies. Similar to English, there was very slight improvement for short distance dependencies ($1-5$). But for longer distances, $6-10$, and >10, there was significant improvement of 1.3% and 1.3% respectively for MST. Ambati et al. (2013) reported similar improvements for Malt as well.

4.4 Discussion

Though valency is a useful feature in dependency parsing (Zhang and Nivre, 2011), Zhang and Nivre (2012) showed that providing valency information dynamically, in the form of the number of dependencies established in a particular state during parsing, did not help Malt. However, as we have shown above, providing this information as a static lexical feature in the form of CCG categories does help Malt. In addition to specifying the number of arguments, CCG categories also contain syntactic type and direction of those arguments. However, providing CCG categories as features to zpar (Zhang and Nivre, 2011) didn't have significant impact as it is already using similar information.

4.5 Impact on Web Scale Parsing

Greedy parsers such as Malt are very fast and are practically useful in large-scale applications such as parsing the web. Table 2, shows the speed of Malt, MST and zpar on parsing English test data in CoNLL scheme (including POS-tagging and supertagging time). Malt parses 310 sentences per second, compared to 35 and 11 of zpar and MST respectively. Clearly, Malt is orders of magnitude faster than MST and zpar. After using CCG categories from the supertagger, Malt parses 245 sentences per second, still much higher than other parsers. Thus we have shown a way to improve Malt without significantly compromising speed, potentially enhancing its usefulness for web scale parsing.

Parser	*Ave. Sents / Sec*	*Total Time*
MST	11	3m 36s
zpar	35	1m 11s
Malt	310	0m 7.7s
Malt + CCG	245	0m 10.2s

Table 2: Time taken to parse English test data.

5 Conclusion

We have shown that informative CCG categories, which contain both local subcategorization information and capture long distance dependencies elegantly, improve the performance of two dependency parsers, Malt and MST, by helping in recovering long distance relations for Malt and local verbal arguments for MST. This is true both in the case of English (a fixed word order language) and Hindi (free word order and morphologically richer language), extending the result of Ambati et al. (2013). The result is particularly interesting in the case of Malt which cannot directly use valency information, which CCG categories provide indirectly. It leads to an improvement in performance without significantly compromising speed and hence promises to be applicable to web scale processing.

References

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2013. Using CCG categories to improve Hindi dependency parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 604–609, Sofia, Bulgaria.

Akshar Bharati, Prashanth Mannem, and Dipti Misra Sharma. 2012. Hindi Parsing Shared Task. In *Proceedings of Coling Workshop on Machine Translation and Parsing in Indian Languages*, Kharagpur, India.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164, New York City, New York.

Ruken Cakici. 2005. Automatic induction of a CCG grammar for Turkish. In *Proceedings of the ACL Student Research Workshop*, pages 73–78, Ann Arbor, Michigan.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of COLING-04*, pages 282–288.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the conference on Empirical methods in natural language processing*, EMNLP '02, pages 1–8.

Greg Coppola and Mark Steedman. 2013. The effect of higher-order dependency features in discriminative phrase-structure parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 610–616, Sofia, Bulgaria.

Marie Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *In LREC 2006*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June.

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.

Sunghwan Mac Kim, Dominick Ng, Mark Johnson, and James Curran. 2012. Improving combinatory categorial grammar parse reranking with dependency grammar features. In *Proceedings of COLING 2012*, pages 1441–1458, Mumbai, India.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

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.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 91–98, Ann Arbor, Michigan.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Kenji Sagae, Yusuke Miyao, and Jun'ichi Tsujii. 2007. HPSG parsing with shallow dependency constraints. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 624–631, Prague, Czech Republic.

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

Sumire Uematsu, Takuya Matsuzaki, Hiroki Hanaoka, Yusuke Miyao, and Hideki Mima. 2013. Integrating multiple dependency corpora for inducing wide-coverage Japanese CCG resources. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1042–1051, Sofia, Bulgaria.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA.

Yue Zhang and Joakim Nivre. 2012. Analyzing the effect of global learning and beam-search on transition-based dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 1391–1400, Mumbai, India, December.