

The OSU System for Surface Realization at Generation Challenges 2011

Rajakrishnan Rajkumar and **Dominic Espinosa** and **Michael White**

Department of Linguistics, The Ohio State University

{*raja, espinosa, mwhite*}@ling.osu.edu

Abstract

This report documents our efforts to develop a Generation Challenges 2011 surface realization system by converting the shared task deep inputs to ones compatible with OpenCCG. Although difficulties in conversion led us to employ machine learning for relation mapping and to introduce several robustness measures into OpenCCG's grammar-based chart realizer, the percentage of grammatically complete realizations still remained well below results using native OpenCCG inputs on the development set, with a corresponding drop in output quality. We discuss known conversion issues and possible ways to improve performance on shared task inputs.

1 Introduction

Our Generation Challenges 2011 shared task system represents an initial attempt to develop a surface realizer for shared task inputs that takes advantage of prior work on broad coverage realization with OpenCCG (White, 2006; Espinosa et al., 2008; Rajkumar et al., 2009; White and Rajkumar, 2009; Rajkumar and White, 2010). OpenCCG is a parsing/generation library for Combinatory Categorical Grammar (Steedman, 2000). CCG is a unification-based categorial grammar formalism defined almost entirely in terms of lexical entries that encode sub-categorization as well as syntactic features. OpenCCG implements a grammar-based chart realization algorithm in the tradition of Kay's (1996) approach to bidirectional processing with unification grammars. The chart realizer takes

as input logical forms represented internally using Hybrid Logic Dependency Semantics (HLDS), a dependency-based approach to representing linguistic meaning (Baldrige and Kruijff, 2002). To illustrate the input to OpenCCG, consider the semantic dependency graph in Figure 1. In the graph, each node has a lexical predication (e.g. **make.03**) and a set of semantic features (e.g. $\langle \text{NUM} \rangle_{\text{sg}}$); nodes are connected via dependency relations (e.g. $\langle \text{ARG0} \rangle$). Such graphs are broadly similar to the “deep” shared task inputs. Note, however, that they are quite different from the shallow input trees, where many of the expected dependencies from coordination, control and relativization are missing. For example, in the figure, both dependents of **make.03** would be missing in the shallow tree, which involve control and relativization (with a null relativizer). As it would be difficult to hallucinate such dependencies, we have only attempted the deep task.

Grammar-based chart realization in the tradition of Kay is capable of attaining high precision, but achieving broad coverage is a challenge, as is robustness to any deviations in the expected input. Previous work on chart realization has primarily used inputs derived from gold standard parses, and indeed, native OpenCCG inputs have been obtained from gold standard derivations in the CCGbank (Hockenmaier and Steedman, 2007). Given the available time, our strategy was to make minor adjustments to OpenCCG's extracted grammars while devoting the bulk of our effort to converting the shared task inputs to be as similar as possible to the native inputs. Difficulties in conversion led us to employ machine learning for relation mapping and to introduce

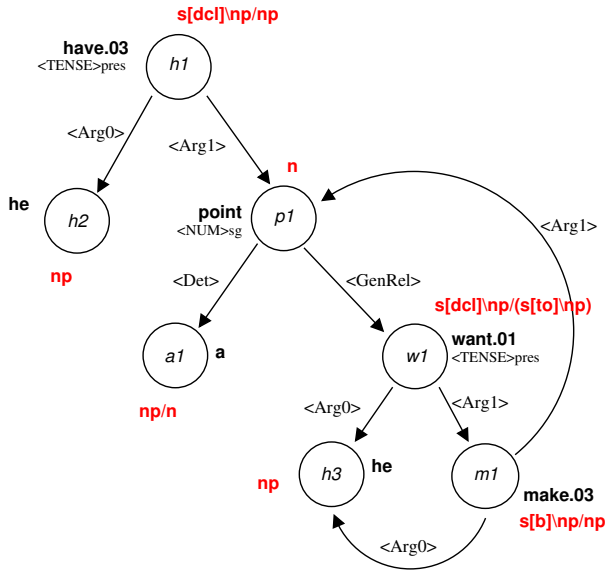


Figure 1: Semantic dependency graph from the CCGbank for *He has a point he wants to make [...]*, along with gold-standard supertags (category labels)

several robustness measures into OpenCCG’s realization algorithm. Nevertheless, the percentage of grammatically complete realizations still remained well below results using native OpenCCG inputs on the development set, with a corresponding drop in output quality.

2 Conversion

In previous work, when extracting HLDS quasi-logical form graphs from the CCGbank, we removed semantically empty function words such as complementizers, infinitival-*to*, expletive subjects, and case-marking prepositions. For improved consistency with shared task inputs, we have instead left expletive subjects and all prepositions (but not complementizers and relativizers) in the native dependency graphs. Even so, the logical forms our system expects differed from the shared task inputs in many ways, the most notable being the structure of conjunctions, possessives and relative clauses, so manual conversion rules were written to handle these cases. In addition, named entities and hyphenated words were collapsed to form atomic logical form predicates, and for simplicity quotes were ignored. The conversion was effected by a Java converter augmented by XSL transforms. Table 1 provides frequencies of converted elements. Finally, to derive

Construction	Frequency
Collapsed NEs	703
Collapsed hyphenations	303
Conjunctions	691
Possessives	214
Relative clauses	90
Punct nodes excised	1672

Table 1: Conversion statistics for 1034 development section shared task graphs

possible word forms for unseen lemmas, morphg (Minnen et al., 2001) was used with heuristically derived POS tags.

3 Relation Tagger

Since the shared task graphs used relations between nodes which were often not easily mappable to native OpenCCG relations, we trained a maxent classifier to tag the most likely relation, as well as an auxiliary maxent classifier to POS tag the graph nodes, much like hypertagging (Espinosa et al., 2008). Training data for the classifier was extracted by comparing each relation between two nodes in the input shared task graph with the corresponding relation in the HLDS logical form. In case a labeled relation did not exist in the HLDS graph, a *NoRel* relation label was assigned. On the development data, we obtained accuracies of 90% for the POS tagger and 90.5% for the relation classifier. A substantial portion of the errors were related to the *NoRel* outcome. Of the 5154 *NoRel* cases in the dev sect, 444 were miscategorized as *Mod*, 344 as *Arg1*, 212 as *Arg0*, and 107 as *Det*. The other major error was that the *Mod* relation was often erroneously misclassified as *NoRel*.

4 Realization Results and Discussion

In spite of the graph structure and relation label changes described above, it still proved necessary to make several adjustments to both OpenCCG as well as the converted graphs. OpenCCG’s strict relation checking had to be relaxed to permit divergences between the relations supplied by a lexical item and the ones in the input graph. In cases where no complete realization could be found, we also employed a novel approach to assembling fragments using MT-inspired glue rules (White, 2011), which enable a more exhaustive search of possible fragment com-

System	Shared Task			Native		
	BLEU	5-best	Coverage	BLEU	5-best	Coverage
OSU.1 (all)	0.4346	0.2483	95%	0.7838	0.5177	95%
OSU.2 (complete)	0.6564	0.3874	19%	0.8341	0.5413	76%

Table 2: Development set scores for all realizations (OSU.1) and grammatically complete realizations only (OSU.2) for the shared task inputs and using native inputs

binations and allow for n -best outputs. Additionally, we added optionality operators into the converted shared task graphs, in order to allow certain features or relations to be used as required by the grammar’s constraints. The most notable cases were an optional $\langle \text{DET} \rangle \text{nil}$ feature for nodes that could be expressed by bare nouns, and making certain relations optional, especially those derived from Nombank that yielded multiple parents for the child node.

For the experiments reported below, as in previous work, we used a lexico-grammar extracted from Sections 02–21 of our enhanced CCGbank with a similar model training procedure. Development set results appear in Table 2. Single-best and weighted 5-best BLEU scores, along with coverage percentages, are given for both the converted shared task inputs as well as native OpenCCG inputs, for comparison. The OSU.1 system includes outputs for all sentences, assembling fragments if no grammatically complete realizations are found; the OSU.2 system only includes outputs for complete realizations.¹ As the table shows, the percentage of grammatically complete realizations for the converted shared task inputs is well below the percentage using native inputs, with a corresponding drop in BLEU scores. Debugging efforts suggest that the remaining relation mismatches and other structural divergences are preventing complete realizations from being derived most of the time. The relative absence of punctuation-related features may also be an issue.

In future work, we plan to explore using machine learning more comprehensively to convert the inputs, beyond just relation tagging. We also plan to explore whether grammars can be induced that are more directly compatible with shared task inputs.

¹Native coverage is less than 100% because of failures to derive a complete LF from the CCGbank; shared task coverage could have been 100% but the system was only run on the same inputs as in the native case.

Acknowledgements

This work was supported in part by NSF grant IIS-0812297 and by an allocation of computing time from the Ohio Supercomputer Center.

References

- Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. ACL-02*.
- Dominic Espinosa, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proc. ACL-08: HLT*.
- 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.
- Martin Kay. 1996. Chart generation. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 200–204, Morristown, NJ, USA. Association for Computational Linguistics.
- G. Minnen, J. Carroll, and D. Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.
- Rajakrishnan Rajkumar and Michael White. 2010. Designing agreement features for realization ranking. In *Proc. Coling 2010: Posters*.
- Rajakrishnan Rajkumar, Michael White, and Dominic Espinosa. 2009. Exploiting named entity classes in CCG surface realization. In *Proc. NAACL HLT 2009 Short Papers*.
- Mark Steedman. 2000. *The syntactic process*. MIT Press, Cambridge, MA, USA.
- Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proc. of EMNLP-09*.
- Michael White. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorical Grammar. *Research on Language and Computation*, 4(1):39–75.
- Michael White. 2011. Glue rules for robust chart realization. In *Proc. of ENLG-11*. To appear.