UW-MRS: Leveraging a Deep Grammar for Robotic Spatial Commands

Woodley Packard
University of Washington
sweaglesw@sweaglesw.org

Abstract

This paper describes a deep-parsing approach to SemEval-2014 Task 6, a novel context-informed supervised parsing and semantic analysis problem in a controlled domain. The system comprises a hand-built rule-based solution based on a pre-existing broad coverage deep grammar of English, backed up by a off-the-shelf data-driven PCFG parser, and achieves the best score reported among the task participants.

1 Introduction

SemEval-2014 Task 6 involves automatic translation of natural language commands for a robotic arm into structured "robot control language" (**RCL**) instructions (Dukes, 2013a). Statements of RCL are trees, with a fixed vocabulary of content words like `prism` at the leaves, and markup like `action:` or `destination:` at the nonterminals. The yield of the tree largely aligns with the words in the command, but there are frequently substitutions, insertions, and deletions.

A unique and interesting property of this task is the availability of highly relevant machine-readable descriptions of the spatial context of each command. Given a candidate RCL fragment describing an object to be manipulated, a spatial planner provided by the task organizers can automatically enumerate the set of task-world objects that match the description. This information can be used to resolve some of the ambiguity inherent in natural language.

The commands come from the Robot Commands Treebank (Dukes, 2013a), a crowdsourced corpus built using a *game with a purpose* (von Ahn, 2006). Style varies considerably, with missing determiners, missing or unexpected punctuation, and missing capitalization all common (Dukes, 2013b). Examples (1) and (2) show typical commands from the dataset.

(1) drop the blue cube

(2) Pick yellow cube and drop it on top of blue cube

Although the natural language commands vary in their degree of conformance to what might be called standard English, the hand-built gold standard RCL annotations provided with them (e.g. Figure 1) are commendable in their uniformity and accuracy, in part because they have been automatically verified against the formal *before* and *after* scene descriptions using the spatial planner.

```
(event:  (action:  drop)
         (entity:  (color:  blue)
                   (type:  cube))
```

Figure 1: RCL corresponding to Example (1).

2 Related Work

Automatic interpretation of natural language is a difficult and long-standing research problem. Some approaches have taken a relatively shallow view; for instance, ELIZA (Weizenbaum, 1966) used pattern matching to somewhat convincingly participate in an English conversation. Approaches taking a deeper view tend to parse utterances into structured representations. These are usually abstract and general-purpose in nature, e.g. the syntax trees produced by mainstream PCFG parsers and the DRS produced by the Boxer system (Bos, 2008). As a notable exception, Dukes (2014) presents a novel method to produce RCL output directly.

The English Resource Grammar (**ERG**; Flickinger, 2000) employed as a component in the present work is a broad-coverage precision hand-written unification grammar of English, following the Head-driven Phrase Structure Grammar theory of syntax (Pollard & Sag, 1994). The ERG produces Minimal Recursion Semantics

Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 812–816,
Dublin, Ireland, August 23-24, 2014.

(**MRS**; Copestake et al., 2005) analyses, which are flat structures that explicitly encode predicate argument relations (and other data). A simplified MRS structure is shown in Figure 2. With minor modifications to allow determinerless NPs and some unexpected measure noun lexemes (as in "two *squares* to the left", etc), the ERG yields analyses for 99% of the commands in the training portion of the Robot Command Treebank.

$$\big(\text{INDEX} = e, \big\{\texttt{pron}(x), \texttt{cube\_n}(y),$$
$$\texttt{drop\_v\_cause}(e, x, y), \texttt{blue\_a}(\_, y)\big\}\big)$$

Figure 2: Highly simplified view of the MRS produced by the ERG for Example (1).

3 ERG-based RCL Synthesis

This section outlines the method my system employs to synthesize RCL outputs from the MRS analyses produced by the ERG. The ERG provides a ranked list of candidate MRS analyses for each input. As a first step, grossly inappropriate analyses are ruled out, e.g. those proposing non-imperative main verbs or domain-inappropriate parts of speech ("block" as a verb). An attempt is made to convert each remaining analysis into a candidate RCL statement. If conversion is successful, the result is tested for coherence with respect to the known world state, using the supplied spatial planner. An RCL statement is incoherent if it involves picking up or moving an entity which does not exist, or if its command type (`take, move, drop`) is incompatible with the current state of the robot arm, e.g. `drop` is incoherent when the robot arm is not holding anything. Processing stops as soon as a coherent result is found.[1]

3.1 From MRS to RCL

Given an individual (imperative) MRS structure, the first step in conversion to RCL is to identify the sequence of top-level verbal predications. The `INDEX` property of the MRS provides an entry point. In a simple command like Example (1), the `INDEX` will point to a single verbal predication, whereas in a compound command such as Example (2), the `INDEX` will point to a coordination predication, which itself will have left and right arguments which must be visited recursively. Each verbal predication visited in this manner generates an `event:` RCL statement whose `action:` property is determined by a looking up the verbal predicate in a short hand-written table (e.g. `drop_v_cause` maps to `action: drop`). If the predicate is not found in the table, the most common action `move` is guessed.

Every RCL `event:` element must have an `entity:` subelement, representing the object to be moved by the action. Although in principle MRS makes no guarantees about the generalizability of the semantic interpretation of argument roles across different predicates, in practice the third argument of every verbal predicate relevant to this domain represents the object to be moved; hence, synthesis of an `event:` proceeds by inspecting the third argument of the MRS predicate which gave rise to it. Some types of `event:` also involve a `destination:` subelement, which encodes the location where the entity should come to rest. When present, a verbal predicate's *fourth* argument almost always identifies a prepositional predication holding this information, although there are exceptions (e.g. for `_move_v_from-to_rel` it is the fifth). When no such resultative role is present, the first prepositional modifier (if any) of the verbal event variable is used for the `destination:` subelement.

Synthesis of an `entity:` element from a referential index like y in Figure 2 or a `spatial-relation:` element from a prepositional predication proceeds in much the same way: the RCL `type:` or `relation:` is determined by a simple table lookup, and subelements are built based on connections indicated in the MRS. One salient difference is the treatment of predicates that are not found in their respective lookup tables. Whereas unknown command predicates default to the most common action `move`, unknown modifying spatial relations are simply dropped,[2] and unknown entity types cause conversion to fail, on the theory that an incorrect parse is likely. Prudent rejection of suspect parses only rarely eliminates all available analyses, and generally helps to find the most appropriate one. On development data, the first analysis produced by the ERG was

[1] Practically speaking, conversion from MRS to RCL is accomplished by a relatively short C program embodying these rules and steps (about 1500 lines in the final version): http://sweaglesw.org/svn/semeval-2014-task6/tags/dublin

[2] If the spatial relation is part of a mandatory `destination:` element, this can then cause conversion to fail.

convertible for 87% of commands, and the first RCL hypothesis was spatially coherent for 96% of commands. These numbers indicate that the parse ranking component of the ERG works quite well.

3.2 Polishing the Rules

I split the 2500 task-supplied annotated commands into a randomly-divided training set (2000 commands) and development set (500 commands). Throughout this work, the development set was only used for estimating performance on unseen data and tuning system combination settings; the contents of the development set were never inspected for rule writing or error analysis purposes. Although the conversion architecture outlined above constitutes an effective framework, there were quite a few details to be worked through, such as the construction of the lookup tables, identification of cases requiring special handling, elimination of undesirable parses, modest extension of the ERG, etc. An error-analysis tool which performed a fine-grained comparison of the synthesized RCL statements with the gold-standard ones and agglomerated common error types proved invaluable when writing rules. [3] Polishing the system in this manner took about two weeks of part-time effort; I maintained a log giving a short summary of each tweak (e.g. "map `_center_n_of_rel` to `type: region`"). These tweaks required varying amounts of time to implement, from a few seconds up to perhaps an hour; system accuracy as a function of the number of such tweaks is shown in Figure 3.

3.3 Anaphora and Ellipsis

Some commands use anaphora to evoke the identity or type of previously mentioned entities. Typically, the pronoun "it" refers to a specific entity while the pronoun "one" refers to the type of an entity (e.g. "Put the red cube on the blue one."). Empirically, the antecedent is nearly always the first `entity:` element in the RCL statement, and this heuristic works well in the system. A small fraction of commands ($< 0.5\%$ of the training data) elide the pronoun, in commands like "Take the blue tetrahedron and place in front left corner." In principle these could be detected and accommodated through the addition of a simple mal-rule to the ERG (Bender et al., 2004), but for simplicity my system ignores this problem, leading to errors.

[3] The error-analysis tool walks the system and gold RCL trees in tandem, recording differences and printing the most common mismatches. It consists of about 100 lines of Python and shell script, and took perhaps an hour to build.

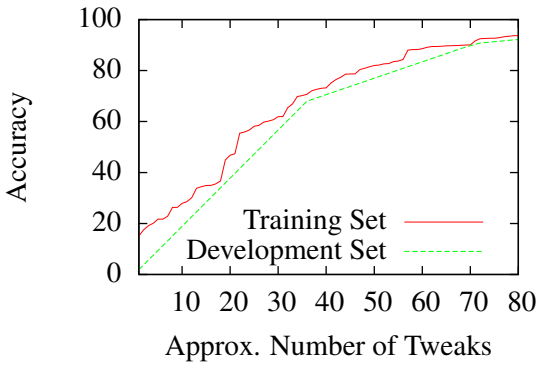

Figure 3: Tuning the MRS-to-RCL conversion system by tweaking/adding rules. Development-set accuracy was only checked occasionally during rule-writing to avoid over-fitting.

4 Robustness Strategies

If none of the analyses produced by the ERG result in coherent RCL statements, the system produces no output. On the one hand this results in quite a high precision: on the training data, 96.75% of the RCL statements produced are exactly correct. On the other hand, in some scenarios a lower precision result may be preferable to no result. The ERG-based system fails to produce any output for 3.1% of the training data inputs, a number that should be expected to increase for unseen data (since conversion can sometimes fail when the MRS contains unrecognized predicates).

In order to produce a best-guess answer for these remaining items, I employed the Berkeley parser (Petrov et al., 2006), a state-of-the-art data-driven system that induces a PCFG from a user-supplied corpus of strings annotated with parse trees. The RCL treebank is not directly suitable as training material for the Berkeley parser, since the yield of an RCL tree is not identical to (or even in 1-to-1 correspondence with) the words of the input utterance. In the interest of keeping things simple, I produced a phrase structure translation of the RCL treebank by simply discarding the elements of the RCL trees that did not correspond to any input, and inserting `(X word)` nodes for input words that were not aligned to any RCL fragment. The question of where in the tree to insert these `X` nodes is presumably of considerable importance, but again in the interest of simplicity I simply clustered them together with the first RCL-

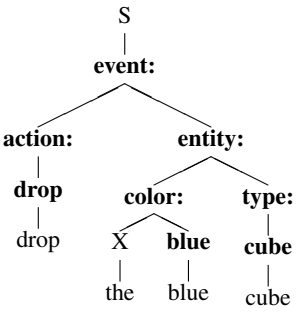

Figure 4: Automatic phrase structure tree translation of the RCL statement shown in Figure 1.

aligned word appearing after them. Unaligned input tokens at the end of the sentence were added as siblings of the root node. Figure 4 shows the phrase structure tree resulting from the translation of the RCL statement shown in Figure 1.

Using this phrase structure treebank, the Berkeley parser tools make it possible to automatically derive a similar phrase structure tree for any input string, and indeed when the input string is a command such as the ones of interest in this work, the resulting tree is quite close to an RCL statement. Deletion of the `X` nodes yields a robust system that frequently produces the exact correct RCL, at least for those items where only input-aligned RCL leaves are required. The most common type of non-input-aligned RCL fragment is the `id:` element, identifying the antecedent of an anaphor. As with the ERG-based system, a heuristic selecting the first `entity` as the antecedent whenever an anaphor is present works quite well.

Improving the output of the statistical system via tweaks of the type used in the ERG-based system was much more challenging, due to the relative impoverishedness of the information made available by the parser. Accurately detecting situations to improve without causing collateral damage proved difficult. However, the base accuracy of the statistical system was quite good, and when used as a back-off it improved overall system scores considerably, as shown in Table 5.

5 Results and Discussion

The combined system performs best on both portions of the data. Over the development data, the MRS-based system performs considerably better than the statistical system, in part due to the use of spatial planning in the MRS-based system (time did not permit adding spatial planning to the statistical system). The statistical system has a slightly higher recall than the MRS-only system without spatial planning, but the MRS-only system has a higher precision — markedly so on the evaluation data. This is consistent with previous findings combining precision grammars with statistical systems (Packard et al., 2014).

System	Dev P	Dev R	Eval P	Eval R
MRS-only ($-$**SP**)	90.7	88.0	92.1	80.3
MRS-only ($+$**SP**)	95.4	92.2	96.1	82.4
Robust-only ($-$**SP**)	88.2	88.2	81.5	81.5
Combined ($-$**SP**)	90.8	90.8	90.5	90.5
Combined ($+$**SP**)	95.0	95.0	92.5	92.5
ERG coverage		98.6		91.0

Figure 5: Evaluation results. $\pm$**SP** indicates whether or not spatial planning was used. The robust and combined systems always returned a result, so P = R.

ERG coverage dropped precipitously from roughly 99% on the development data to 91% on the evaluation data. This is likely the major cause of the 10% absolute drop in the recall of the MRS-only system. The fact that the robust statistical system encounters a comparable drop on the evaluation data suggests that the text is qualitatively different from the (also held-out) development data. One possible explanation is that whereas the development data was randomly selected from the 2500 task-provided training commands, the evaluation data was taken as the sequentially following segment of the treebank, resulting in the same distribution of game-with-a-purpose participants (and hence writing styles) between the training and development sets but a different distribution for the evaluation data. [4]

Dukes (2014) reports an accuracy of 96.53%, which appears to be superior to the present system; however, that system appears to have used more training data than was available for the shared task, and averaged scores over the entire treebank, making direct comparison difficult.

Acknowledgements

I am grateful to Dan Flickinger, Emily Bender and Stephan Oepen for their many helpful suggestions.

[4] Reviewers suggested dropping sentence initial punctuation and reading "cell" as "tile." This trick boosts the MRS-only recall to 91.1% and the combined system to 94.5%, demonstrating both the frailty of NLP systems to unexpected inputs and the presence of surprises in the evaluation data. ERG coverage rose from 91.0% to 98.6%.

References

Bender, E. M., Flickinger, D., Oepen, S., Walsh, A., & Baldwin, T. (2004). Arboretum: Using a precision grammar for grammar checking in CALL. In *Instil/icall symposium 2004.*

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In J. Bos & R. Delmonte (Eds.), *Semantics in text processing. step 2008 conference proceedings* (pp. 277–286). College Publications.

Copestake, A., Flickinger, D., Pollard, C., & Sag, I. (2005). Minimal recursion semantics: An introduction. *Research on Language & Computation*, *3*(2), 281–332.

Dukes, K. (2013a). Semantic annotation of robotic spatial commands. In *Language and Technology Conference (LTC).* Poznan, Poland.

Dukes, K. (2013b). Train robots: A dataset for natural language human-robot spatial interaction through verbal commands. In *International Conference on Social Robotics (ICSR). Embodied Communication of Goals and Intentions Workshop.*

Dukes, K. (2014). Contextual Semantic Parsing using Crowdsourced Spatial Descriptions. *Computation and Language. arXiv:1405.0145 [cs.CL].*

Flickinger, D. (2000). On building a more efficient grammar by exploiting types. *Natural Language Engineering*, *6*(01), 15-28.

Packard, W., Bender, E. M., Read, J., Oepen, S., & Dridan, R. (2014). Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *Proceedings of the 52nd annual meeting of the Association for Computational Linguistics.* Baltimore, USA.

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics* (pp. 433–440).

Pollard, C., & Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. Chicago, USA: The University of Chicago Press.

von Ahn, L. (2006). Games with a purpose. *Computer*, *39*(6), 92–94.

Weizenbaum, J. (1966). ELIZA — a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, *9*(1), 36–45.