UWM: Applying an Existing Trainable Semantic Parser to Parse Robotic Spatial Commands

Rohit J. Kate
University of Wisconsin-Milwaukee
Milwaukee, WI
katerj@uwm.edu

Abstract

This paper describes Team UWM's system for the Task 6 of SemEval 2014 for doing supervised semantic parsing of robotic spatial commands. An existing semantic parser, KRISP, was trained using the provided training data of natural language robotic spatial commands paired with their meaning representations in the formal robot command language. The entire process required very little manual effort. Without using the additional annotations of word-aligned semantic trees, the trained parser was able to exactly parse new commands into their meaning representations with 51.18% best F-measure at 72.67% precision and 39.49% recall. Results show that the parser was particularly accurate for short sentences.

1 Introduction

Semantic parsing is the task of converting natural language utterances into their complete formal meaning representations which are executable for some application. Example applications of semantic parsing include giving natural language commands to robots and querying databases in natural language. Some old semantic parsers were developed manually to work for specific applications (Woods, 1977; Warren and Pereira, 1982). However, such semantic parsers were generally brittle and building them required a lot of manual effort. In addition, these parsers could not be ported to any other application without again putting significant manual effort.

More recently, several semantic parsers have been developed using machine learning (Zelle and Mooney, 1996; Ge and Mooney, 2005; Zettlemoyer and Collins, 2005; Wong and Mooney, 2006; Kate and Mooney, 2006; Lu et al., 2008; Kwiatkowski et al., 2011). In this approach, training data is first created for the domain of interest. Then using one of the many machine learning methods and semantic parsing frameworks, a semantic parser is automatically learned from the training data (Mooney, 2007). The trained semantic parser is then capable of parsing new natural language utterances into their meaning representations. Semantic parsers built using machine learning tend to be more robust and can be easily ported to other application domains with appropriate domain-specific training data.

The Task 6 of SemEval 2014 provided a new application domain for semantic parsing along with training and test data. The domain involved giving natural language commands to a robotic arm which would then move blocks on a board (Dukes, 2013). The domain was inspired from the classic AI system SHRDLU (Winograd, 1972). The training data contained 2500 examples of sentences paired with their meaning representations in the Robot Command Language (RCL) which was designed for this domain (Dukes, 2013). The test data contained 909 such example pairs.

We trained an existing and freely available[1] semantic parser KRISP (Kate and Mooney, 2006) using the training data for this domain. Besides changing the format of the data for running KRISP and writing a context-free grammar for the meaning representation language RCL, the entire process required minimal manual effort. The author spent less than a week's time for participating in the Task 6, and most of it was spent in running the experiments. This demonstrates that trainable semantic parsers like KRISP can be rapidly adopted to new domains. In the Results section we show different precisions and recalls it ob-

[1] http://www.cs.utexas.edu/users/ml/krisp/

Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 823–827,
Dublin, Ireland, August 23-24, 2014.

tained at different confidence levels in the form of a precision-recall curve. The results also show that the parser was particularly accurate on shorter sentences. Two major reasons that prevented KRISP from performing better on this domain were - its high computational demand for memory which prevented it from being trained beyond 1500 training examples, and some variability in the meaning representation language RCL that negatively affected training as well as evaluation.

2 Background: KRISP Semantic Parser

KRISP (Kernel-based Robust Interpretation for Semantic Parsing) is a trainable semantic parser (Kate and Mooney, 2006) that uses Support Vector Machines (SVMs) (Cristianini and Shawe-Taylor, 2000) as the machine learning method with string-subsequence kernel (Lodhi et al., 2002). It takes natural language utterances and their corresponding formal meaning representation as the training data along with the context-free grammar of the meaning representation language (MRL). The key idea in KRISP is that every production of the MRL is treated as a semantic concept. For every MRL production, an SVM classifier is trained so that it can give for any input natural language substring of words the probability that it expresses the corresponding semantic concept. Once these classifiers are trained, parsing a sentence reduces to finding the most probable semantic derivation of the sentence in which different productions cover different parts of the sentence and together form a complete meaning representation. Figure 1 shows an example semantic derivation of a robotic spatial command. Productions of RCL grammar (Table 1) are shown at tree nodes depicting different parts of the sentence they cover.

Since the training data is not in the form of such semantic derivations, an EM-like iterative algorithm is used to collect appropriate positive and negative examples in order to train the classifiers (Kate and Mooney, 2006). Positive examples are collected from correct semantic derivations derived by the parser learned in the previous iteration, and negative examples are collected from the incorrect semantic derivations.

KRISP was shown to work well on the US geography database query domain (Tang and Mooney, 2001) as well as on the RoboCup Coach Language (CLang) domain (Kate et al., 2005). It was also shown to be particularly robust to noise in the natural language utterances (Kate and Mooney, 2006). KRISP was later extended to do semi-supervised semantic parsing (Kate and Mooney, 2007b), to learn from ambiguous supervision in which multiple sentences could be paired with a single meaning representation in the training data (Kate and Mooney, 2007a), and to transform the MRL grammar to improve semantic parsing (Kate, 2008).

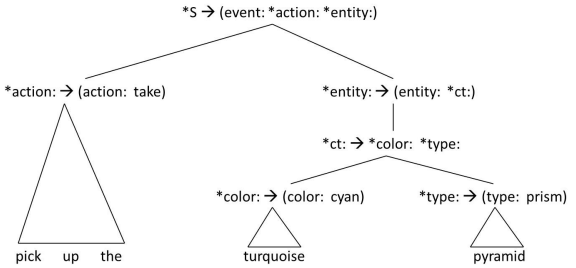

Figure 1: Semantic derivation of the robotic spatial command *"pick up the turquoise pyramid"* obtained by KRISP during testing which gives the correct RCL representation *(event: (action: take) (entity: (color: cyan) (type: prism)))*.

3 Methods

In order to apply KRISP to the Task 6 of SemEval 2014, the format of the provided data was first changed to the XML-type format that KRISP accepts. The data contained several instances of co-references which was also part of RCL, but KRISP was not designed to handle co-references and expects them to be pre-resolved. We observed that almost all co-references in the meaning representations, indicated by "reference-id" token, resolved to the first occurrence of an "entity" element in the meaning representation. This was found to be true for more than 99% of the cases. We used this observation to resolve co-references during semantic parsing in the following way. As a pre-processing step, we first remove from the meaning representations all the "id:" tokens (these resolve the references) but keep the "reference-id:" tokens (these encode presence of co-references). The natural language sentences are not modified in any way and the parser learns from the training data to relate words like "it" and "one" to the RCL token "reference-id". After KRISP generates a meaning representation during testing, as a post-processing step, "id: 1" is added to the first "entity" element in the meaning representation if it contains the "reference-id:" token.

The context-free grammar for RCL was not provided by the Task organizers. There are multi-

ple ways to write a context-free grammar for a meaning representation language and those that conform better to natural language work better for semantic parsing (Kate, 2008). We manually wrote grammar for RCL which mostly followed the structure of the meaning representations as they already conformed highly to natural language commands and hence writing the grammar was straightforward. KRISP runs faster if there are fewer non-terminals on the right-hand-side (RHS) of the grammar because that makes the search for the most probable semantic derivation faster. Hence we kept non-terminals on RHS as few as possible while writing the grammar. Table 1 shows the entire grammar for RCL that we wrote which was given to KRISP. The non-terminals are indicated with a "*" in their front. We point out that KRISP needs grammar only for the meaning representation language (an application will need it anyway if the statements are to be executed) and not for the natural language.

KRISP's training algorithm could be aided by providing it with information about which natural language words are usually used to express the concept of a production. For example, word "red" usually expresses "*color: → (color: red)". The data provided with the Task 6 came with the word-aligned semantic trees which indicated which natural language words corresponded to which meaning representation components. This information could have been used to aid KRISP, however, we found many inconsistencies and errors in the provided word-aligned semantic trees and chose not to use them. In addition, KRISP seemed to learn most of that information on its own anyway.

The Task 6 also included integrating semantic parsing with spatial planning. This meant that if the semantic parser generates an RCL representation that does not make sense for the given block configuration on the board, then it could be dismissed and the next best RCL representation could be considered. Besides generating the best meaning representation for a natural language utterance, KRISP is also capable of generating multiple possible meaning representations sorted by their probabilities. We could have used this capability to output only the best RCL representation that is valid for the given board configuration. Unfortunately, unfamiliarity with the provided Java API for the spatial planner and lack of time prevented us from doing this.

*action: → (action: move)
*action: → (action: drop)
*action: → (action: take)
*cardinal: → (cardinal: 1)
*cardinal: → (cardinal: 2)
*cardinal: → (cardinal: 3)
*cardinal: → (cardinal: 4)
*color: → (color: magenta)
*color: → (color: red)
*color: → (color: white)
*color: → (color: cyan)
*color: → (color: green)
*color: → (color: yellow)
*color: → (color: blue)
*color: → (color: gray)
*indicator: → (indicator: rightmost)
*indicator: → (indicator: back)
*indicator: → (indicator: center)
*indicator: → (indicator: right)
*indicator: → (indicator: leftmost)
*indicator: → (indicator: individual)
*indicator: → (indicator: nearest)
*indicator: → (indicator: front)
*indicator: → (indicator: left)
*reference-id: → (reference-id: 1)
*relation: → (relation: right)
*relation: → (relation: forward)
*relation: → (relation: within)
*relation: → (relation: above)
*relation: → (relation: nearest)
*relation: → (relation: adjacent)
*relation: → (relation: front)
*relation: → (relation: left)
*relation: → (relation: backward)
*type: → (type: type-reference-group)
*type: → (type: board)
*type: → (type: prism)
*type: → (type: cube)
*type: → (type: type-reference)
*type: → (type: cube-group)
*type: → (type: corner)
*type: → (type: robot)
*type: → (type: stack)
*type: → (type: edge)
*type: → (type: region)
*type: → (type: tile)
*type: → (type: reference)
*indicator: → *indicator: *indicator:
*color: → *color: *color:
*ct: → *color: *type:
*ict: → *indicator: *ct:
*ctr: → *ct: *reference-id:
*cct: → *cardinal: *ct:
*ed: → *entity: (destination: *spatial-relation:)
*entity: → (entity: *type:)
*entity: → (entity: *type: *reference-id:)
*entity: → (entity: *type: *spatial-relation:)
*entity: → (entity: *ct:)
*entity: → (entity: *indicator: *type:)
*entity: → (entity: *ict:)
*entity: → (entity: *ict: *spatial-relation:)
*entity: → (entity: *cardinal: *type:)
*entity: → (entity: *cct:)
*entity: → (entity: *cct: *spatial-relation:)
*entity: → (entity: *ctr:)
*entity: → (entity: *ct: *spatial-relation:)
*entity: → (entity: *ctr: *spatial-relation:)
*measure: → (measure: *entity:)
*mr: → *measure: *relation:
*spatial-relation: → (spatial-relation: *relation: *entity:)
*spatial-relation: → (spatial-relation: *mr:)
*spatial-relation: → (spatial-relation: *mr: *entity:)
*S → (sequence: *S *S)
*S → (event: *action: *ed:)
*S → (event: *action: *entity:)

Table 1: Grammar for the Robot Command Language (RCL) given to KRISP for semantic parsing. The non-terminals are indicated with a "*" in their front. The start symbol is *S.

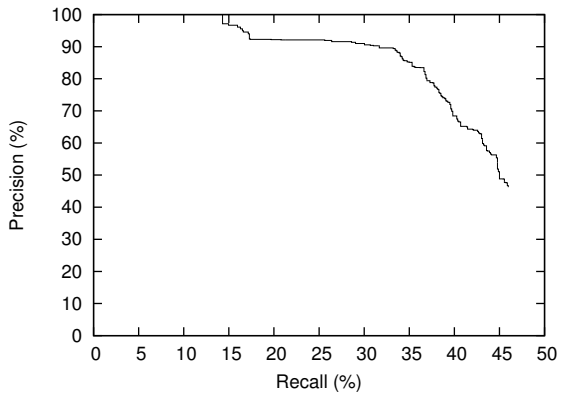

Figure 2: Precision-recall curve for the semantic parsing output on test sentences.

Sentence length	Accuracy (Correct/Total)
1-3	100.00% (15/15)
4-7	71.20% (136/191)
8-11	51.76% (147/284)
12-15	41.80% (79/189)
16-19	22.22% (28/126)
20-23	15.71% (11/70)
24-27	3.23% (1/31)
28-31	33.33% (1/3)
All	45.98% (418/909)

Table 2: Accuracy of semantic parsing across test sentences of varying lengths.

4 Results

We found that KRISP could not be trained beyond 1500 examples in this domain because the number of negative examples that are generated during the training process would become too large for the available memory size. This is something that could be fixed in the future by suitably sampling negative examples. Using the first 1500 training examples, we evaluated KRISP's performance on the provided 909 test examples. A generated RCL representation is considered correct only if it exactly matches the correct answer; no partial credit is given. In order to avoid generating incorrect meaning representations when it is not confident, KRISP uses a threshold and if the confidence (probability) of the best semantic derivation is below this threshold, it does not generate any meaning representation. This threshold was set to 0.05 as was previously done for other domains.

Performance was measured in terms of precision (the percentage of generated meaning representations that were correct) and recall (the percentage of all sentences for which correct meaning representations were obtained). Given that KRISP also gives confidences with its output meaning representations, we can compute precisions and recalls at various confidence levels. Figure 2 shows the entire precision-recall curve thus obtained. The best F-measure (harmonic mean of precision and recall) on this curve is 51.18% pdf at 72.67% precision and 39.49% recall. The precision at the highest recall was 45.98% which we had reported as our official evaluation result for the SemEval Task 6.

We further analyzed the results according to the lengths of the sentences and found that KRISP was very accurate with shorter sentences and became progressively less accurate as the lengths of the sentences increase. Table 2 shows these results. This could be simply because the longer the sentence, the more the likelihood of making an error, and since no partial credit is given, the entire output meaning representation is deemed incorrect.

On further error analysis we observed that there was some variability in the meaning representations. The "move" and "drop" actions seemed to mean the same thing and were used alternatively. For example in the training data, the utterance *"place the red block on single blue block"* had "(action: drop)" in the corresponding meaning representation, while *"place red cube on grey cube"* had "(action: move)", but there is no apparent difference between the two cases. There were many such instances. This was confusing KRISP's training algorithm because it would collect the same phrase sometimes as a positive example and sometimes as a negative example. This also affected the evaluation, because KRISP would generate "move" which won't match "drop", or vice-versa, and the evaluator will call it an error.

5 Conclusions

We participated in the SemEval 2014 Task 6 of supervised semantic parsing of robotic spatial commands. We used an existing semantic parser learner, KRISP, and trained it on this domain which required minimum time and effort from our side. The trained parser was able to map natural language robotic spatial commands into their formal robotic command language representations with good accuracy, particularly for shorter sentences.

References

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Kais Dukes. 2013. Semantic annotation of robotic spatial commands. In *Proceedings of the Language and Technology Conference (LTC-2013)*, Poznan, Poland.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 9–16, Ann Arbor, MI, July.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, pages 913–920, Sydney, Australia, July.

Rohit J. Kate and Raymond J. Mooney. 2007a. Learning language semantics from ambiguous supervision. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, pages 895–900, Vancouver, Canada, July.

Rohit J. Kate and Raymond J. Mooney. 2007b. Semi-supervised learning for semantic parsing using support vector machines. In *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)*, pages 81–84, Rochester, NY, April.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 1062–1068, Pittsburgh, PA, July.

Rohit J. Kate. 2008. Transforming meaning representation grammars to improve semantic parsing. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL-2008)*, pages 33–40. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2011)*, pages 1512–1523. Association for Computational Linguistics.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. 2:419–444.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, Honolulu, HI, October.

Raymond J. Mooney. 2007. Learning for semantic parsing. In A. Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing: Proceedings of the 8th International Conference (CICLing-2007), Mexico City*, pages 311–324. Springer Verlag, Berlin.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning (ECML-2001)*, pages 466–477, Freiburg, Germany.

David H. D. Warren and Fernando C. N. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3-4):110–122.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press, Orlando, FL.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT-NAACL-06)*, pages 439–446, New York City, NY.

William A. Woods. 1977. Lunar rocks in natural English: Explorations in natural language question answering. In Antonio Zampoli, editor, *Linguistic Structures Processing*. Elsevier North-Holland, New York.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1050–1055, Portland, OR, August.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence (UAI-2005)*, Edinburgh, Scotland, July.