TLDR: Simple task and parameters extractor model

Neural Semantic Parsing with Anonymization for Command Understanding in General-Purpose Service Robots

Nick Walker[0000−0001−7711−0003], Yu-Tang Peng[0000−0002−4916−5557], and Maya Cakmak[0000−0001−8457−6610]

Paul G. Allen School of Computer Science & Engineering
University of Washington, Seattle WA 98195, USA
{nswalker,pengy25,mcakmak}@cs.washington.edu

Abstract. Service robots are envisioned to undertake a wide range of tasks at the request of users. Semantic parsing is one way to convert natural language commands given to these robots into executable representations. Methods for creating semantic parsers, however, rely either on large amounts of data or on engineered lexical features and parsing rules, which has limited their application in robotics. To address this challenge, we propose an approach that leverages neural semantic parsing methods in combination with contextual word embeddings to enable the training of a semantic parser with little data and without domain specific parser engineering. Key to our approach is the use of an anonymized target representation which is more easily learned by the parser. In most cases, this simplified representation can trivially be transformed into an executable format, and in others the parse can be completed through further interaction with the user. We evaluate this approach in the context of the RoboCup@Home *General Purpose Service Robot* task, where we have collected a corpus of paraphrased versions of commands from the standardized command generator. Our results show that neural semantic parsers can predict the logical form of unseen commands with 89% accuracy. We release our data and the details of our models to encourage further development from the RoboCup and service robotics communities.

Keywords: Natural Language Understanding · General-Purpose Service Robot · Semantic Parsing.

1 Introduction

General-purpose service robots (GPSRs) are envisioned as capable helpers that will assist with everything from chores around the home to finding open conference rooms in the office. These robots are distinguished by their ability to accomplish a wide variety of possible goals by recomposing basic capabilities spanning navigation, manipulation, perception and interaction. One especially desirable interface for these robots is natural language, because users are already familiar with using language to ask for assistance.

Command understanding is commonly framed as executable semantic parsing [3,19], where the objective is to convert the user command into a logical representation that unambiguously captures the meaning of the command, decoupled from the surface characteristics of the language. For instance, both "Bring me a red apple from the kitchen" and "Could you get me a red apple from kitchen please" might be transformed into a λ-calculus representation like bring(λ\$1.(apple(\$1) $\wedge$ red(\$1) $\wedge$ at(\$1, "kitchen"))). This representation can then be grounded immediately by finding satisfactory entities from the robot's ontology or later as a part of executing the resulting plan.

Traditionally, creating usable semantic parsers in a low-data domain has required an expert to craft lexical features or parsing rules. Recent neural semantic parsing approaches have lessened the need for domain specific engineering, but are not easily applied to robotics settings because of the dearth of available annotated data. This data-deficit is likely to persist because the contextual nature of commands makes it challenging to collect data without expensive extended interactions, and there is no deployed base of service robots with which interactions can be gathered en masse.

In this paper, we propose an approach to command understanding in a general-purpose service robot that can make use of recent advances in semantic parsing. Key to our approach is the use of a simplified target representation which trades immediate executablitity under certain circumstances for ease of learning. Further, we leverage pretrained contextual word embeddings to improve the model's generalizability. We evaluate this approach on a new corpus of command-semantics pairs created by crowd-sourcing paraphrases of the generated language used in the *General Purpose Service Robot* task from RoboCup@Home, demonstrating that learned parsers can predict the correct logical form of an unseen command with 89% accuracy.

2 Related Work

Several works have investigated neural methods for semantic parsing [20,14,16]. Our work is based on recent advances in translating language to logical forms using sequence-to-sequence encoder-decoder neural models [6]. Architectures that enforce the constraints of the target representation have been proposed, including recurrent neural network grammars [8], sequence-to-tree methods [6], course-to-fine methods [7], as well as other applications of constrained decoding tailored for semantic parsing [15]. Our work does not enforce decoding constraints, sacrificing potential performance gains for a higher degree of portability across target representations.

Anonymization, also referred to as delexicalization, is frequently used to overcome data sparsity in natural and spoken language systems. Our work is similar in spirit to the argument identification method used by Dong and Lapata [6], but instead of anonymizing entities into unique tokens, we abstract them by type into tokens representing their class. Perhaps closest to our approach is work on weakly-supervised parser learning aided by abstracted representations [12],

which similarly proposes leveraging a small lexicon to simplify a visual reasoning domain. Copy mechanisms [13], which enable sequence-to-sequence models to copy portions of their input into their output, are also frequently used to combat data sparsity. Recent work has suggested that copying may dominate delexicalization for several semantic parsing tasks [4]. We do not consider a copy mechanism in this work, as avenues for integrating the robot's knowledge are less available when relying on a learned copying behavior.

The application of contextualized word embeddings to various language tasks is an active area and has seen several early applications to semantic parsing [9]. To our knowledge, this work is the first to evaluate whether contextual embeddings improve performance for semantic parsing in low-data domains.

Many researchers have investigated methods for giving commands to robots [24,19]. Our work is most closely aligned with research from Thomason et al. [25] on continually learning a command parsing and grounding system via dialogue. While we do not consider learning from dialogue interactions, we make the assumption that our understanding system will be used as part of a dialogue agent that can ask for confirmation or corrections. Further, where Thomason et al. initialized their parser by engineering a CCG lexicon against a set of user commands, we use neural methods that directly learn the correspondence between commands and logical forms.

While many teams have built systems motivated specifically by the RoboCup@Home GPSR task, they have largely adopted techniques that depend on knowledge of the command generation grammar [17]. Perhaps the most sophisticated system is that of Bastianelli et al., a frame-semantics based spoken language understanding system for service robots which can integrate and utilize visual information [1,2]. In contrast, our work considers solely the language aspect of command understanding, and takes advantage of the flexible nature of neural translation models to avoid prescribing a particular representation.

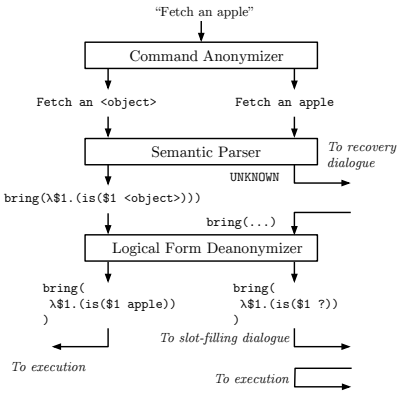
3 Approach

The thrust of our approach is to leverage the robot's knowledge base to simplify input utterances where possible, then parse them into an abstracted λ-calculus representation where argument slots are left as tokens representing the class of the argument. This simplifies the output space of the parser and enables us to use neural semantic parsing, which would otherwise be infeasible due to a lack of in-domain data. When important information is lost in this simplification, it can be retrieved easily by asking the user for clarification. An overview of this framing is shown in Fig. 1. We augment a standard encoder-decoder model with contextual embeddings to further ameliorate the challenge of data-scarcity.

3.1 Command Anonymizer

Though service robots operate in an open world and thus cannot assume complete knowledge of the environment, they are usually equipped with ontologies specifying basic knowledge about objects and locations. We leverage this

(a)



(b)

Fig. 1: (a) The Toyota Human Support Robot is used as a standard platform in the RoboCup@Home competition. (b) A high level overview of how the approach converts a natural language command to a logical form. The left path traces successful execution of each step. The right path shows how failures can be addressed by gathering additional information via dialogue interaction.

to anonymize commands where possible: Any token in the input command for which we have an name entry in the knowledgebase is replaced with a special token denoting its class. For instance, the command "Fetch an apple from the kitchen" would be anonymized to "Fetch an <object> from the <location>."

When anonymization is successful, it reduces the complexity of the semantic parsing task. For commands with arguments that fall within the robot's ontology, the model is no longer required to generalize parses across instantiations with slightly different entities. Further, anonymization guarantees that newly added entities can be used in the same commands that worked for previous entities. Because users will frequently refer to previously unknown objects or use unfamiliar language to refer to known objects however, the semantic parser must still be robust to partially- or even completely non-anonymized commands.

3.2 Semantic Parser

Following Dong and Lapata [6], our parser is a sequence-to-sequence (seq2seq) bidirectional LSTM encoder-decoder model with a bilinear attention mechanism which takes language input and translates it to a logical form. This architecture is shown in Fig. 2. Input tokens are represented with a concatenation of a contextualized word embedding and their 100D GloVe embedding [21]. For anonymized commands, the embedding provides a signal for which class of command is being asked for, as the interchangeability of verbs is captured. For partially anonymized commands, which the model receives when anonymization is unsuccessful, the

embeddings may help if the unknown referent phrase embeds near arguments seen during training.

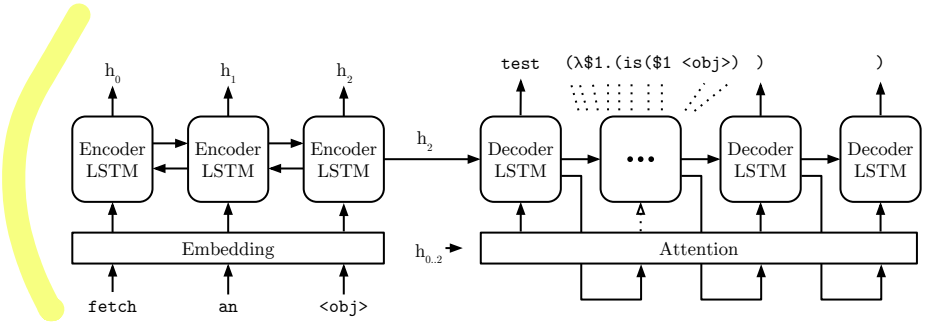


Fig. 2: The outline of the seq2seq architecture, shown encoding a command and decoding a logical form. Special start and end tokens are omitted for clarity. Arrows denote representations passing between modules.

We adopt λ-calculus for our logical representation, but counter to typical practice where output forms are fully specified, we use an intermediate, anonymized form. Similar to the anonymized form of a command, arguments to predicates are represented as abstract class tokens that encode the type of the argument but not its identity. For instance, the command "Navigate to the kitchen, look for the apple, and give it to Bill" has corresponding anonymized form

$$\text{bring}(\lambda\$1.(\text{is\_a}(\$1, \text{<object>}) \wedge \text{at}(\$1, \text{<room>})), \lambda\$1.(\text{person}(\$1) \wedge \text{name}(\$1, \text{<name>}))).$$

This representation retains the expressive power of λ-calculus, which can capture compositional aspects of common commands (e.g. "Bring me the [apple [from the counter [by the refrigerator]]]"), but discards the separate challenge of properly assigning arguments to leaf predicates. Our observation is that, by applying this abstraction, the space of output logical forms is significantly contracted for a set of representative robot commands, enabling us to train useful models even with relatively little data.

3.3 Logical Form Deanonymizer

The output of the semantic parser is an anonymized logical form. In order for the robot to execute the command however, this form must be deanonymized into a fully specified logical expression.

In cases where the anonymizer had replaced spans in the original input utterance, the deanonymizer has access to the correspondence between the class

tokens and the original text that they replaced. In many cases, this allows the deanonymizer to automatically reconstruct the full logical form. In some cases, however, there may be multiple possible matchings between class tokens in the anonymized command and the anonymized logical form. For instance, for the command "Move the apple from the kitchen counter to the dining table", there will be two <location> tokens. Additionally, in cases where anonymization failed, the deanonymizer does not have knowledge of what input spans correspond to the anonymous tokens in the semantic parse. Our observation is that these scenarios are easily addressed via dialogue. Similar to the approach used by Thomason et al. [25], we propose that the deanonymizer disambiguate argument assignments via a slot filling dialogue policy. The same dialogue can be used to add entities to the ontology so they can be properly anonymized in the future.

4 Data

To our knowledge, there is no GPSR-style corpus of sufficient scale to evaluate our approach, so we constructed a set of anonymized command-semantics pairs based on the command generator[1] used in the *General Purpose Service Robot* task from the 2018 international RoboCup@Home domestic service robotics competition [18]. In the GPSR task, robots are read a generated command which they must carry out in a mock-apartment arena. These commands are intended to encompass all of the capabilities that are assessed in other tasks in the competition, including things like finding people and fetching objects. We supplemented the synthetic data from the generator with paraphrases gathered via crowdsourcing. In this section, we describe the details of the construction of these datasets.[2]

4.1 Generated

The generator is specified as a probabilistic context-free grammar with equal weighting on all production rules, split into three categories. Each category introduces both more complicated commands as well as more complicated desired goals. Example commands are provided for each category in Table 2 and summary information about the grammar is shown in Table 1.

The generator is distributed with an ontology describing objects, object categories, person names, gestures, question-answer pairs, common sayings, and household locations. Because commands have multiple highly-branching non-terminals representing entities from the ontology, the number of full expansions grows exponentially in the size of the ontology. However, we observe that the number of distinct anonymized commands is actually quite low, and the number of distinct logical forms is lower still.

[1] The generator is available at https://github.com/kyordhel/GPSRCmdGen. At the time of this work, the 2019 generator had not been finalized.

[2] The data and splits used for our experiments are available at https://doi.org/10.5281/zenodo.3244800.

Table 1: Summary of our annotations to the 2018 *General Purpose Service Robot* task command generator, broken down by category. "Annotations" refers to the number of logical templates that were annotated atop the grammar rules. "Logical forms" is the number of unique anonymized logical forms produced by expanding the annotations. Because categories overlap slightly, we count commands and logical forms as belonging to the first category that they appear in. Length is measured in tokens and includes all tokens that would be given or expected from a model. On average, 58% of logical form tokens are parentheses, quotation marks, or λ-calculus variable type markers.

Category	**1**	**2**	**3**	**All**
Anonymized commands	192	352	667	1211
Logical forms	17	39	45	101
Annotations	28	53	44	125
Complexity Measures				
Average command length	11.9	12.2	10.4	11.3
Average logical form length	28.4	27.0	22.6	25.1
Commands to forms ratio	11.3	9.0	14.8	12.0

We defined a logical domain consisting of 27 predicates (7 actions, 20 descriptive) and used them to create 125 annotations that provide logical forms for all 1211 distinct anonymized commands. These predicates cover concepts such as names, basic prepositions (e.g. `at`, `left_of`), and entity types (e.g. `person`, `object`).

There is no general correspondence between the structure of the generation grammar and the structure of the resulting logical representation; however expansions near the leaves frequently take forms that can be neatly tagged with a semantic template. Thus we annotate partial expansions from the generator with an accompanying semantic template which contains some of the same non-terminals, creating a shallow synchronous context free grammar that produces both commands and their logical representations when expanded.

For instance, a partial expansion that produces "bring" commands contains a nonterminal which produces synonymous verbs and another non-terminal which produces different objects. Its annotation incorporates the object non-terminal but discards the verb non-terminal as its expansions do not affect the semantics of the command:

$$\$\text{vbbring me the } \$\text{object} = \text{bring}(\lambda\$1.(\text{is\_a}(\$1, \$\mathit{object})))$$

Continuing to expand both the command and this semantic template will result in a fully specified command-semantics pair. To produce pairs of anonymized

commands and anonymized logical forms, we simply modify productions associated with ontology entities to produce our class tokens.

4.2 Paraphrases

Although the GPSR command generator is designed with naturalism in mind, it is nonetheless artificial. In order to understand how our approach will fair given more realistic input, we applied a similar methodology as that of Wang, Berant and Liang to crowdsource paraphrases of our generated dataset [26]. Crowd workers were provided fully-expanded generated commands and prompted to provide a paraphrased version—new text that captures all of the same information but uses different words or phrasing. We nudged workers to provide substantial paraphrases by presenting a warning UI if their input was below a threshold of Levenshtein (character) and Jaccard (word) distances from the original command. A similar warning was presented if the same metrics indicated that the paraphrase contained almost no overlap, as this almost always indicated that important information was discarded. To help filter out low-quality responses, we required crowd-workers to write their own commands based on their impression of what the robot could do. Spam responses to these questions consistently indicated spam responses to the paraphrasing task.

We used Amazon Mechanical Turk to collect 1836 paraphrases from 95 workers, ensuring there were at least 10 paraphrases per logical form. The mean Levenshtein and Jaccard distances between a paraphrase and its original command are 28.0 and 0.59 respectively. Sample paraphrases are shown in Table 2.

Table 2: Examples of fully-expanded commands, their anonymized forms, crowdsourced paraphrases, and corresponding logical forms from each category of the GPSR task.

Category	**Example**
1	tell me how many coke there are on the freezer tell me how many <object> there are on the <location> “how many cokes are left in the freezer” say(count(λ\$1.(is_a(\$1, <object>)) $\wedge$ at(\$1, <location>)))
2	tell me what’s the largest object from the bar tell me what’s the largest object from the <location> “Which is the largest object on top of the bar” say(λ\$1.(largest(\$1) $\wedge$ at(\$1, <location>)))
3	Could you give me the object on top of the glass from the coffee table Could you give me the object on top of the <object> from the <location> “can you bring be the thing from the coffee table thats on top of the glass” bring(λ\$1.(λ\$2.(is_a(\$2, <object>) $\wedge$ on_top_of(\$1, \$2))), <location>)

5 Experiments

We evaluate whether it is feasible to learn usable semantic parsers under our approach given the relatively small amount of data available. Our metric is exact-match accuracy; the percentage of logical forms that a model predicts exactly correctly on held-out test data.

Though we are primarily interested in how well models perform on the paraphrased data—as this best represents real language that a robot may encounter—we take advantage of the generated-paraphrased corpus' parallel nature to investigate a range of interesting configurations:

1. *Train generated, test generated* lets us see whether a learned model can approximate the performance of a grammar-based chart-parser, even without having access to the full grammar.
2. *Train generated, test paraphrased* exposes the extent to which the generator captures aspects of natural commands.
3. *Train paraphrased, test paraphrased* shows a model's capacity to generalize across real commands.
4. *Train generated and paraphrased, test paraphrased* can reveal whether there are any benefits to augmenting the paraphrased data with synthetic language.

To better approximate the conditions of a real service robot deployment, we do not use prior knowledge of entities in our experiments. Generated data consists purely of pairs of anonymized commands and anonymized logical forms. Paraphrased commands are not anonymized before being processed by the model.

We split both the generated data and the paraphrased data 70%/10%/20% into training, validation and test sets. Splits are such that no command appears in more than one set. As noted by Finegan et al., testing only generalization to unseen commands can reward models that learn to simply classify commands and produce memorized logical forms [10]. To evaluate whether models are able to produce correct, unseen logical forms, we use an additional logical split. This split forces the pools of logical forms in each part of the dataset to be disjoint. We ensure that the logical split roughly matches the proportions of the command split, and that the split is synchronized across generated and paraphrased data so that they can be combined without causing data leakage.

5.1 Training Regime

We use the Adam optimizer to minimize the cross entropy of the predicted output with the ground truth labels. Training is performed for 150 epochs with early stopping (patience=10) based on accuracy evaluation on the validation set. We use an encoder dropout probability of .1. Test- and validation-time decoding is performed using beam search with a width of 5. Our experiments are built on top of AllenNLP [11] which uses the PyTorch deep learning framework.[3]

[3] The details of our implementation, including the full parameterization of our experiments, is available at https://doi.org/10.5281/zenodo.3246755.

We provide results using each of ELMo [22], OpenAI GPT1 [23], BERT$_{base}$, BERT$_{large}$ [5], with a comparison against a model that forgoes a contextual word embedding and simply uses GloVe. We leave the contextual word embedding frozen during training and instead allow tuning of the GloVe weights to avoid catastrophic forgetting.

5.2 Baseline Models

We compare our models against two simple baselines. GRAMMAR-ORACLE chart parses test samples using the generation grammar and looks up the corresponding annotation to return a logical form. Because it always has full access to the grammar, its predictions depend only on whether the test data are within the grammar. The K-nearest neighbors (KNN) model predicts the label of a test command by searching for its nearest neighbor amongst the training and validation data. We found empirically that using Jaccard distance and $K = 1$ worked well. This model is naturally incapable of predicting labels it has never seen before, so it always scores 0% when evaluated on a logical split.

6 Results

The results of our experiments are shown in Table 3. The columns of the table and our discussion are ordered to match the sequence of the descriptions given in Section 5.

1. When trained and evaluated on synthetic data, neural semantic parsing models easily fit to the surface characteristics of the data, achieving accuracy levels of 98.8% on unseen commands, despite not having access to the underlying grammar.
2. Models trained with the generated data achieve at most 27.6% accuracy on unseen paraphrased commands. All models are bested by the KNN baseline. As the generated training data doesn't contain any entities, it is unsurprising that test performance on the completely non-anonymized paraphrasing data is poor. The results indicate that around a quarter of the paraphrased commands can be resolved purely by looking for structural patterns learned from the generated language.
3. The best model trained on paraphrased data alone is able to achieve 78.5% accuracy on unseen commands, indicating that neural semantic parsers are reasonably capable of handling a realistic GPSR task. Comparing the best BERT-based model against the model with no contextual word embedding indicates that these large pretrained models can provide around an 8% performance improvement for this task.
4. Training with both the generated and paraphrased data leads to a consistent and large performance boost across all models, yielding the best performing model as evaluated on the paraphrasing test set. This result is possibly a reflection of headroom for improvement if more real data were available to the model.

Mirroring previously reported results on text-to-SQL tasks [10], the generalization achieved across unseen commands does not extend to unseen logical forms.

Table 3: Accuracy of models and ablations on different datasets. "C" indicates results from data split on commands while "λ" indicates results from data split on logical forms.

Train	Gen.				Para.		G. + P.	
Test	Gen.		Para.		Para.		Para.	
Split	C	λ	C	λ	C	λ	C	λ
GRAMMAR-ORACLE	100.0	100.0	1.1	0.9	1.1	0.9	1.1	0.9
KNN	63.0	0.0	**42.0**	0.0	42.8	0.0	49.8	0.0
SEQ2SEQ	95.9	0.0	13.0	0.0	64.4	0.0	79.6	0.0
+ GloVe	**98.8**	26.3	12.4	0.0	70.2	0.0	85.3	6.3
+ GloVe;ELMo	**98.8**	36.2	21.3	0.0	77.3	22.1	85.4	34.4
+ GloVe;OpenAI	97.9	24.6	27.6	1.8	78.2	26.3	89.0	**37.9**
+ GloVe;BERT$_{base}$	96.3	**56.9**	12.2	3.3	75.4	**31.3**	87.6	37.6
+ GloVe;BERT$_{large}$	97.9	54.7	27.1	**9.6**	**78.5**	30.4	**89.4**	37.6

6.1 Error Analysis

We manually inspected the predictions of the best paraphrase parsing model, seq2seq with GloVe and BERT$_{large}$, to better understand the model's generalization and common errors.

Table 4: Multiple inputs mapping to the same, incorrect logical form

x_1	"Bring an umbrella"
x_2	"Navigate to the hallway"
x_3	"Do this then that"
$y_1 = y_2 = y_3$ (go " <room> ")	

Unexpected defaults As shown in Table 4, we observed that disparate commands can produce the same prediction. One explanation is that no probable

solutions are found during decoding, so output tends to a default that is helpful for fitting the training data. Sampling additional commands that yield `UNKNOWN` might address this. Alternatively, a confidence threshold could be used to discard low scoring decodings.

Table 5: Predictions can change based on a single word

x_4 "Bring an umbrella to me"
y_4 (bring (λ $1 e (is_a $1 "<object>")))
x_5 "Bring an umbrella to Bob"
y_5 (go "<room>")
x_6 "Bring an umbrella to him"
y_6 (guide (λ $1 e (person $1) (name $1 "<name>")) "<location>")

Overly sensitive As illustrated in Table 5, inputs that differ by a single word can be parsed to drastically different logical forms. Thus we suspect that the model put undue emphasis on the different arguments when predicting the first predicate.

7 Discussion

We have shown that the task of understanding commands in a general-purpose service robot domain can be successfully addressed using neural semantic parsing methods. Key to this success is the use of contextual word embeddings and a abstracted target representation which simplifies the learning task. Though anonymization may require a clarification dialogue for complex commands, such a dialogue would, in practice, occur regardless simply to confirm the command.

The dataset produced for this work is a strong common basis for command-taking in general-purpose service robots. Though the task and the generator are set in the home, the types of goals that arise from the commands apply to a wide variety domains. Thus, we expect that this data can be used to accelerate future efforts to bootstrap command understanding systems for service robots. We hope that the ease with which it is possible to train capable systems under this framework will motivate members of the RoboCup@Home and service robotics communities to expand their expectations of robot language understanding systems.

Acknowledgements We thank Yuqian Jiang, Jesse Thomason, and the anonymous reviewers for their helpful feedback. This work was supported by HONDA award "Curious Minded Machines" and the National Science Foundation award IIS-1552427 "CAREER: End-User Programming of General-Purpose Robots."

References

1. Bastianelli, E., Castellucci, G., Croce, D., Iocchi, L., Basili, R., Nardi, D.: Huric: a human robot interaction corpus. In: Chair), N.C.C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., Piperidis, S. (eds.) Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). European Language Resources Association (ELRA), Reykjavik, Iceland (may 2014)
2. Bastianelli, E., Croce, D., Vanzo, A., Basili, R., Nardi, D.: A discriminative approach to grounded spoken language understanding in interactive robotics. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016. pp. 2747–2753 (2016)
3. Chen, D.L., Mooney, R.J.: Learning to interpret natural language navigation instructions from observations pp. 859–865 (August 2011)
4. Damonte, M., Goel, R., Chung, T.: Practical semantic parsing for spoken language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers). pp. 16–23. Association for Computational Linguistics, Minneapolis - Minnesota (Jun 2019)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Dong, L., Lapata, M.: Language to logical form with neural attention. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 33–43. Association for Computational Linguistics (2016). https://doi.org/10.18653/v1/P16-1004
7. Dong, L., Lapata, M.: Coarse-to-fine decoding for neural semantic parsing. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 731–742. Association for Computational Linguistics, Melbourne, Australia (Jul 2018)
8. Dyer, C., Kuncoro, A., Ballesteros, M., Smith, N.A.: Recurrent neural network grammars. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 199–209. Association for Computational Linguistics, San Diego, California (Jun 2016). https://doi.org/10.18653/v1/N16-1024
9. Einolghozati, A., Pasupat, P., Gupta, S., Shah, R., Mohit, M., Lewis, M., Zettlemoyer, L.: Improving semantic parsing for task oriented dialog. arXiv preprint arXiv:1902.06000 (2019)
10. Finegan-Dollak, C., Kummerfeld, J.K., Zhang, L., Ramanathan, K., Sadasivam, S., Zhang, R., Radev, D.: Improving text-to-sql evaluation methodology. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 351–360. Association for Computational Linguistics (2018)
11. Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.F., Peters, M., Schmitz, M., Zettlemoyer, L.S.: Allennlp: A deep semantic natural language processing platform (2017)
12. Goldman, O., Latcinnik, V., Nave, E., Globerson, A., Berant, J.: Weakly supervised semantic parsing with abstract examples. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1809–1819. Association for Computational Linguistics, Melbourne, Australia (Jul 2018)

13. Gu, J., Lu, Z., Li, H., Li, V.O.: Incorporating copying mechanism in sequence-to-sequence learning. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1631–1640. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/P16-1154
14. Kočiskỳ, T., Melis, G., Grefenstette, E., Dyer, C., Ling, W., Blunsom, P., Hermann, K.M.: Semantic parsing with semi-supervised sequential autoencoders. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Austin, Texas (November 2016)
15. Krishnamurthy, J., Dasigi, P., Gardner, M.: Neural semantic parsing with type constraints for semi-structured tables. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 1516–1526. Association for Computational Linguistics, Copenhagen, Denmark (Sep 2017). https://doi.org/10.18653/v1/D17-1160
16. Lewis, M., Lee, K., Zettlemoyer, L.: Lstm ccg parsing. In: Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (2016)
17. Matamoros, M., Harbusch, K., Paulus, D.: From commands to goal-based dialogs: A roadmap to achieve natural language interaction in robocup@home. CoRR **abs/1902.00754** (2019)
18. Matamoros, M., Rascon, C., Hart, J., Holz, D., van Beek, L.: Robocup@home 2018: Rules and regulations. http://www.robocupathome.org/rules/2018_rulebook.pdf (2018)
19. Matuszek, C., Herbst, E., Zettlemoyer, L., Fox, D.: Learning to Parse Natural Language Commands to a Robot Control System, pp. 403–415. Springer International Publishing, Heidelberg (2013). https://doi.org/10.1007/978-3-319-00065-7_28
20. Misra, D., Artzi, Y.: Neural shift-reduce ccg semantic parsing. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 1775–1786. Association for Computational Linguistics (2016)
21. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
22. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proc. of NAACL (2018)
23. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
24. Tellex, S., Kollar, T., Dickerson, S., Walter, M.R., Banerjee, A.G., Teller, S., Roy, N.: Understanding natural language commands for robotic navigation and mobile manipulation. In: Proceedings of the National Conference on Artificial Intelligence (AAAI) (2011)
25. Thomason, J., Padmakumar, A., Sinapov, J., Walker, N., Jiang, Y., Yedidsion, H., Hart, J., Stone, P., Mooney, R.J.: Improving grounded natural language understanding through human-robot dialog. In: International Conference on Robotics and Automation (ICRA) (2019)
26. Wang, Y., Berant, J., Liang, P.: Building a semantic parser overnight. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 1332–1342. Association for Computational Linguistics, Beijing, China (Jul 2015). https://doi.org/10.3115/v1/P15-1129