Learning to Parse Natural Language to Grounded Reward Functions with Weak Supervision

Edward C. Williams[*] edward_c_williams@brown.edu
Mina Rhee[*] mina_rhee@brown.edu
Nakul Gopalan[*] nakul_gopalan@brown.edu
Stefanie Tellex stefie10@cs.brown.edu

Brown University, Providence, Rhode Island
[*] The first three authors contributed equally.

Abstract

In order to intuitively and efficiently collaborate with humans, robots must learn to complete tasks specified using natural language. We represent natural language instructions as goal-state reward functions specified using lambda calculus. To map sentences to such reward function, we learn a weighted linear Combinatory Categorial Grammar (CCG) semantic parser. The parser, including both parameters and the CCG lexicon, is learned from a validation procedure that does not require execution of a planner, annotating reward functions, or labeling parse trees. To learn a CCG lexicon and parse weights, we use coarse lexical generation and validation-driven perceptron weight updates using the approach of Artzi and Zettlemoyer (2013). Initial results on the Cleanup World domain (MacGlashan et al. 2015) demonstrate the potential of our approach. We report an F1 score of 0.79 with a corpus of 5 tasks and 500 corresponding sentences. We are currently growing the set of evaluation tasks and corresponding sentences, and expanding the expressiveness of our semantic representation.

Introduction

Natural language provides an expressive and accessible method to specify goals for robotic agents. For example, a user of a household robot may easily specify navigation (e.g., "go to the TV room") or manipulation (e.g., "place the dishes in the sink") goals without needing to learn complex APIs or user interfaces. In this paper, we propose to learn to map natural language instructions to reward functions that can be executed by conventional planners.

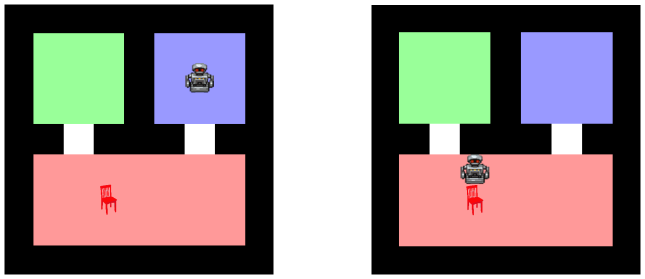
To interface effectively with conventional planning procedures, we formally specify goals using Object-Oriented Markov Decision Process (Diuk, Cohen, and Littman 2008, OO-MDP) reward functions. These function are represented as goal-state proposition functions, which give the agent positive reward for reaching the goal state. Planning then aims to maximize the sum of rewards the agent receives. The use of reward functions as an intermediate representation allows a separation between language interpretation and planning systems.

Existing approaches for learning to map sentences to such reward functions require fully supervised data, including reward function specifications paired with each natural language command (MacGlashan et al. 2015; Arumugam et al. 2017). In this work, we propose to learn a mapping from language to reward functions using demonstrations as annotations. We adopt a grounded lambda-calculus logical form representation for reward functions and induce a Combinatory Categorial Grammar (Steedman 2000, CCG) semantic parser. While existing approaches rely on executing logical forms by planning during training(Artzi and Zettlemoyer 2013; Artzi, Das, and Petrov 2014), our method allows for efficient learning without performing any planning. Our approach does not require annotation of parse trees or reward functions, while avoiding the overhead of planning each considered hypothesis during learning.

We perform tasks in Cleanup Domain (MacGlashan et al. 2015). Cleanup Domain is a simulated mobile manipulator domain, where the set of tasks involve moving to a specific room, or moving next to a particular object. We collected data on Amazon Mechanical Turk and evaluate our system on a held out test dataset. Our initial results demonstrate effective learning of a weakly supervised parser with 0.79 F1 score. We are currently adding more tasks to our dataset, while also expanding the set of predicates in our semantic representation.

Related Work

Zettlemoyer and Collins (2005) presented a supervised method to learn CCG parsers given natural language annotated with lambda-calculus logical forms. Krishnamurthy and Mitchell (2012) demonstrated a weakly supervised method to learn CCG semantic parsers given a syntactically parsed sentence and a knowledge base. Krishnamurthy and Kollar (2013) jointly learned perceptual functions and a CCG parser for identifying objects in an image described



(a) State at task initiation (b) State at task completion

Figure 1: The figure shows an example pair used to collect data. Here we ask the users to give a command to the robot that will result in the pre- and post condition behavior shown in the pair of images.

by users, as well as demonstrating that weakly supervised models achieved similar performance to strongly supervised models. Artzi and Zettlemoyer (2013) used navigation trajectories as weak supervision for CCG parser learning. During learning, proposed lambda-calculus semantic representations were executed to produce a trajectory, which was compared to a ground-truth trajectory to provide a validation signal. While this lambda-calculus lexicon allows the representation of commands describing entire trajectories, our assumption that commands are primarily goal-based allows us to avoid planning during learning.

Tellex et al. (2011) described a method using syntactic parse trees of language to create probabilistic graphical models (PGM) to factorize the distribution over possible groundings for natural language commands. The process of generating trajectories directly from language is expensive, hence Howard, Tellex, and Roy (2014) used a similar PGM, derived from a pre-trained syntactic parser, to generate constraints that be used to plan. Our work also relies on a separation between language processing and planning components of an instruction-following system, although we produce a reward function in an MDP rather than constraints for a general planner. A related line of work looks to map language to reward functions (MacGlashan et al. 2015; Arumugam et al. 2017), either via sequence-to-sequence translation or classification into a fixed set of reward functions. However, these approaches lack the compositionality of a lambda-calculus reward representation.

Next we discuss our methodology for learning grounded reward functions with weak supervision.

Method

We model our tasks in Cleanup Domain (MacGlashan et al. 2015) shown in Figure 1 using Markov Decision Processes (MDPs) (Bellman 1957). An MDP is formally represented as a five-tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{T}$, $\mathcal{R}$, $\mathcal{E}$). Here $\mathcal{S}$ represents the environment's state space; $\mathcal{A}$ represents the agent's action space; $\mathcal{T}(s, a, s')$ is a probability distribution defining the transition dynamics (i.e., the probability that a transition to state s' occurs when action a is taken in state s); $\mathcal{R}(s, a, s')$ represents the reward function returning the numerical reward that the agent receives for transitioning to state s' with action a from state s; and $\mathcal{E} \subset \mathcal{S}$ is a set of terminal states that, once reached, prevent any future action. The goal of planning in an MDP is to find a *policy*—a mapping from states to actions—that maximizes the expected future discounted reward.

In our Cleanup Domain-based data set, there are 5 types of tasks which involve moving into a room (containment) and moving next to an object (adjacency). We specify this our domain as an Object Oriented MDP (OO-MDP) (Diuk, Cohen, and Littman 2008), which allows factored representation of an MDP problem. The state space encodes the position of the objects and the agent, and information about configurations of rooms. The factorization of the environment and actions is done using objects present in the environment, which enables a convenient linkage between language describing objects and the objects themselves. We learn a weighted linear CCG parser (Clark and Curran 2007) that maps natural language commands $x \in X$ to a logical form $y \in Y$ in our semantic representation. For this, we first define a semantic representation that can be provided as a grounded reward function to an MDP planner. Secondly, we collect a data set where each element is an ordered pair of the form (x_i, S_i), where $x_i \in X$ is a natural language command, and S_i is a set of pairs of MDP states. Each pair consists of initial and termination states for the task described by the command. Next we then define a validation function $\mathcal{V}(y, S) \in \{0, 1\}$ that determines if a given semantic parse will produce the correct behavior as described by our training data. Finally, to learn parser weights, we applied the validation-driven perceptron learning algorithm of Artzi and Zettlemoyer (2013). We describe these steps in more detail below.

Semantic Representation and Execution We define a lambda-calculus semantic representation that is tailored to representing a goal-state reward function in an MDP domain. In this work, we make the assumption that natural language commands to our agent define a configuration of the world that the user would like the agent to produce, by re-arranging objects in the world or moving to a different location. We adopt much of the notation from (Artzi and Zettlemoyer 2013; 2011; Zettlemoyer and Collins 2005; Steedman 2000) for our lambda-calculus functions. However, we eschew the neo-Davidsonian event semantics used by Artzi and Zettlemoyer (2013), as our tasks are purely represented by state configurations.

We model nouns as single-argument lambda-calculus functions that map OO-MDP objects in a given state to Boolean values. For example, the phrase "block" would be represented as a function $\lambda x.block(x)$ that, when evaluated in an OO-MDP state, returns true if the given object satisfies the proposition function. Adjectival language, such as "green block," are modeled as conjunctions of these single-argument proposition functions. The function $\lambda x.green(x) \wedge block(x)$ checks two attributes of the object provided as its argument. As an OO-MDP object is parameterized using object classes and attributes, these single-argument proposition functions can be implemented simply as lookup operations on object instances.

We model the definite determiner "the" as a function that maps a proposition function to an object that satisfies the given proposition function, following Artzi and Zettlemoyer (2013). This can be represented as a search over a set of objects in an OO-MDP state. We note that definite determiners are evaluated with respect to the initial state of the task, as we assume object references should be resolved in the state in which the natural language command was issued.

Spatial relationships between two objects are modeled as lambda-calculus functions of arity two. The function $\lambda x.\lambda y.in(x, y)$, for example, uses the spatial dimensions of the object provided as the second argument to determine if it contains the first argument. All relationship proposition functions produce Boolean outputs when evaluated in a given OO-MDP state. Currently, we model two spatial relationships, containment and adjacency.

The above functions can be composed to produce predicates that describe a particular configuration of the world state. For example, the task in Figure 1 can be represented as the lambda-calculus function $near(the(\lambda x.agent(x)), the(\lambda y.red(y) \wedge chair(y)))$.

All lambda-calculus functions were implemented as JScheme (Anderson, Hickey, and Norvig 2013) predicates that operate on states and objects in the BURLAP (MacGlashan 2014) reinforcement learning library.

Parsing Our learning objective is to learn a set of parser weights $\theta \in R^d$ for a weighted linear CCG parser (Clark and Curran 2007) with a d-dimensional feature representation $\Phi(x, y)$. This parser uses a variant of the dynamic-programming CKY algorithm to produce the highest scoring parse $\hat{y}$ from a natural language command x. To perform training and inference, we use the linear CCG parser implemented in the Cornell Semantic Parsing Framework (SPF) (Artzi 2016).

Example Sentences Collected from AMT
"Move to the green room"
"Go by the red chair"
"Move to stand next to the chair"
"Move close to the chair"
"Stand in the blue room"

Table 1: Example sentences collected from the AMT HIT where the users were shown a set of pre-and post condition states and asked to give a command that would instruct the robot to complete the task.

Data We gathered training data using the Amazon Mechanical Turk (AMT) platform. Users were shown three pairs of pre- and post- condition states, all representing the same task in different domain configurations. They were then asked to provide a single command that would instruct the robot to complete the task in every domain configuration. This provided multiple pre- and post- condition pairs for each training example, to incentivize both the learning algorithm and AMT users to produce outputs that are task- rather than configuration-specific. We generated five pre- and post-condition state pairs for twelve different tasks, and sampled three from a set of five for each AMT Human Intelligence Task (HIT). Each possible permutation of three from a set of five needed for a single AMT HIT was used ten times to gather a total of 500 natural language commands. Data was then split the data into 400 commands used for training, and 100 used for quantitative evaluation. Some example commands are shown in Table 1.

Parse Validation To facilitate our validation-driving perceptron learning, we define a parse validation function similar to those of Artzi and Zettlemoyer (2013). We define the function $\mathcal{V}(y, S)$, which takes as input a semantic parse y and set of pre- and post- condition state pairs S and returns 1 if the parse is valid for all pairs of states in S. A parse validates correctly with respect to a state pair if the proposition function is satisfied in the post-condition state, after grounding is performed with respect to the pre- condition state, and not satisfied in the pre- condition state. As our proposition functions define sets of goal states, this is sufficient to check if a parse is valid without invoking a planner.

Coarse Lexical Generation To generate new lexical entries for words and phrases not present in the seed lexicon, we adapt the coarse lexical generation algorithm of (Artzi and Zettlemoyer 2013) to our validation procedure. The algorithm generates new proposed lexicon entries from all possible combinations of factored lexical entries (see (Kwiatkowski et al. 2011) for a detailed description of the lexicon) and words in a given training examples, then discards entries leading to parses that fail to validate.

Learning With our validation function, modified lexical generation procedure, we use the validation-driven perceptron learning algorithm of Artzi and Zettlemoyer (2013) to learn parsing weights. We provided the learning algorithm with a seed lexicon Λ_0, as in Artzi and Zettlemoyer (2013) and Zettlemoyer and Collins (2005), used to initialize the coarse lexical generation procedure. We used a beam width of 75 for lexicon generation during training, which was performed over 10 epochs. We used the implementation of this algorithm in the Cornell SPF (Artzi 2016).

Results

We trained our parser on the 400 training examples described above, and evaluated on 100 pairs of unseen natural language commands annotated with state pair sets. As the commands in test data also did not have logical form annotations, we determined test accuracy using our validation function. We defined a test parse y as correct if it validates with respect to its corresponding state pair set. Our current model has an F1 score of 0.79 on our full data set with a precision of 82.7% and a recall of 77%. We observe that a baseline predicting random reward functions would have an accuracy below 20%, as we learn reward functions for 5 domain-independent tasks as specified above. Note that our method has never seen a complete example of a reward function at training time, and induces every reward function from its seed lexicon and the provided validation function.

A closer look at the data has shown that some AMT users gave given incomplete, or incorrect language specifications, which lead to incorrect parses. Further, many parse errors arise from confusion between the $near$ (adjacency) predicate and in (containment) predicate specified in our ontology, as users often used similar words to describe tasks that involve approaching an object and entering a room. We also hypothesize that part of this confusion is due to the relationship between containment and adjacency predicates. At learning time, adjacency-based tasks are often satisfied by reward functions using containment predicates, leading to suboptimal lexical entries and parser parameters.

The two spatial relationship predicates in our ontology often produce goal-state reward functions whose sets of satisfaction states intersect. Consider Figure 1; the relationship that best satisfies the task is adjacency, that is, asking the agent to go next to the red chair. This adjacency predicate is of the form $near(the(\lambda x.agent(x)), the(\lambda y.red(y) \wedge chair(y)))$. However, another more general predicate of containment can also satisfy this task, that asks the agent to go to the red room. This containment predicate will have the form, $in(the(\lambda x.agent(x)), the(\lambda y.red(y) \wedge room(y)))$. During learning both of these functions would be valid if misleading, possibly leading to the wrong parse being learned and incorrect behavior at testing time. We plan to collect an additional corpus, with more discriminatory examples, to learn subset–superset relationships within our validation procedure.

Conclusion

We presented a method for learning a parser that maps natural language commands to reward functions using a CCG parser via weak supervision. We showed that this parser can be learned using modifications of existing semantic parser learning algorithms, and its outputs are executable as goal-state reward functions with off the shelf planners. Our model produces valid reward functions with an F1 score of 0.79, which is on par with previous weakly supervised semantic parsing baselines (Artzi and Zettlemoyer 2013). However, our method presents a new way to learn goal-based reward function from natural language, without using a planner in the parsing loop. In future we want to learn a larger set of tasks with more complex predicates. We also plan to improve the performance of our model with improved features and further testing. Further, we plan to extend our work to accurately learn subset relationships between spatial proposition functions. For example, the commands ”go to the right of the red block” and ”go near the block” describe state termination sets where one set is contained within the other set.

Acknowledgement

We would like to thank Yoav Artzi for his insightful comments and help with the Cornell SPF library.

References

Anderson, K. R.; Hickey, T. J.; and Norvig, P. 2013. The jscheme language and implementation.

Artzi, Y., and Zettlemoyer, L. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y., and Zettlemoyer, L. 2013. Weakly supervized learning of semantic parsers for mapping instructions to actions. In *Annual Meeting of the Association for Computational Linguistics*.

Artzi, Y.; Das, D.; and Petrov, S. 2014. Learning compact lexicons for ccg semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1273–1283. Doha, Qatar: Association for Computational Linguistics.

Artzi, Y. 2016. Cornell SPF: Cornell Semantic Parsing Framework.

Arumugam, D.; Karamcheti, S.; Gopalan, N.; Wong, L. L.; and Tellex, S. 2017. Accurately and efficiently interpreting human-robot instructions of varying granularities. *CoRR* abs/1704.06616.

Bellman, R. 1957. A Markovian decision process. *Indiana University Mathematics Journal* 6:679–684.

Clark, S., and Curran, J. R. 2007. Wide-coverage efficient statistical parsing with ccg and log-linear models. *Computational Linguistics* 33(4):493–552.

Diuk, C.; Cohen, A.; and Littman, M. L. 2008. An object-oriented representation for efficient reinforcement learning. In *International Conference on Machine Learning*.

Howard, T. M.; Tellex, S.; and Roy, N. 2014. A natural language planner interface for mobile manipulators. In *IEEE International Conference on Robotics and Automation*.

Krishnamurthy, J., and Kollar, T. 2013. Jointly learning to parse and perceive. In *Transactions of the Association for Computational Linguistics*.

Krishnamurthy, J., and Mitchell, T. M. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Kwiatkowski, T.; Zettlemoyer, L.; Goldwater, S.; and Steedman, M. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1512–1523. Association for Computational Linguistics.

MacGlashan, J.; Babeş-Vroman, M.; desJardins, M.; Littman, M. L.; Muresan, S.; Squire, S.; Tellex, S.; Arumugam, D.; and Yang, L. 2015. Grounding english commands to reward functions. In *Robotics: Science and Systems*.

MacGlashan, J. 2014. Brown–UMBC Reinforcement Learning and Planning (BURLAP)–Project Page. http://burlap.cs.brown.edu/.

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

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M. R.; Banerjee, A. G.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI Conference on Artificial Intelligence*.

Zettlemoyer, L., and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*.