Integrated Learning of Dialog Strategies and Semantic Parsing

Aishwarya Padmakumar **Jesse Thomason** **Raymond J. Mooney**
Department of Computer Science, University of Texas at Austin
{aish, jesse, mooney}@cs.utexas.edu

Abstract

Natural language understanding and dialog management are two integral components of interactive dialog systems. Previous research has used machine learning techniques to individually optimize these components, with different forms of direct and indirect supervision. We present an approach to integrate the learning of *both* a dialog strategy using reinforcement learning, and a semantic parser for robust natural language understanding, using only natural dialog interaction for supervision. Experimental results on a simulated task of robot instruction demonstrate that joint learning of both components improves dialog performance over learning either of these components alone.

1 Introduction

Natural language understanding and dialog management are two integral components of a dialog system. Current research typically deals with optimizing only one of these components. We present an approach to integrate the learning of *both* a dialog strategy using reinforcement learning, and a semantic parser for robust natural language understanding, using only natural dialog interaction for supervision.

Research in dialog systems has primarily been focused on the problems of accurate dialog state tracking and learning a policy for the dialog system to respond appropriately in various scenarios. Dialogs are typically modeled using Partially Observable Markov Decision Processes (POMDPs), and various reinforcement learning algorithms have been proposed and evaluated for the task of learning optimal policies over these representations to accomplish user goals using as short and natural a dialog as possible (Gašić and Young, 2014; Pietquin et al., 2011; Young et al., 2013). However, such systems typically assume a fixed language understanding component that is available a priori.

Semantic parsing is the task of mapping natural language to a formal meaning representation. It has the potential to allow for more robust mapping of free-form natural language to a representation that can be used to interpret user intentions and track dialog state. This is done by leveraging the compositionality of meaning inherent in language. Prior work has shown that a semantic parser, incrementally updated from conversations, is helpful in dialogs for communicating commands to a mobile robot (Thomason et al., 2015). We show that incremental learning of a POMDP-based dialog policy allows for further improvement in dialog success.

A major challenge with combining the above parser and dialog policy learning techniques is that reinforcement learning (RL) algorithms assume that the dialog agent is operating in a *stationary* environment. This assumption is violated when the parser is updated between conversations. For example, the improved semantic parser may be able to extract more information from a response to a question, which the old parser could not parse. So the RL algorithm may have earlier assumed that asking such a question is not useful, but this is not the case with the updated parser. Our results show that this effect can be mitigated if we break the allowed budget of training dialogs into batches, updating both parser and policy after each batch. As the next training batch gets collected using the updated parser, the policy can be updated using this experience to adapt better to it. We demonstrate, using crowd-sourced results with a simulated robot, that by integrating learning of *both* a dialog manager *and* a semantic parser in

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 547–557,
Valencia, Spain, April 3-7, 2017. ©2017 Association for Computational Linguistics

this manner, task success is improved over cases where the components are trained individually.

2 Related Work

Prior work has used dialog to facilitate robot task learning, e.g. She et al. (2014), but does not account for uncertainty or dynamic changes to the language understanding module when developing a system policy. Some works use a POMDP model and common-sense knowledge (Zhang and Stone, 2015) or generate clarification questions in a probabilistic manner (Tellex et al., 2014), but these too assume that a fixed and well-trained natural language understanding component is available a-priori. Kollar et al. (2013) use a probabilistic parsing and grounding model to understand natural language instructions and extend their knowledge base by asking questions. However, unlike this work, they do not use semantic parsing to leverage the compositionality of language, and also use a fixed hand-coded policy for dialog.

There has been considerable work in semantic parsing using both direct supervision in the form of annotated meaning representations (Wong and Mooney, 2007; Kwiatkowski et al., 2013; Berant et al., 2013) and indirect signals from downstream tasks (Artzi and Zettlemoyer, 2011; Artzi and Zettlemoyer, 2013; Thomason et al., 2015). Artzi and Zettlemoyer (2011) use clarification dialogs to train semantic parsers for an airline reservation system without explicit annotation of meaning representations. More related to our work is that of Thomason et al. (2015), who incorporated this general approach into a system for instructing a mobile robot; however, they use a simple model of dialog state and a fixed, hand-coded dialog policy. We show that learning a dialog policy in addition to this, is more beneficial than only parser learning. We also use a richer state representation that incorporates multiple hypotheses from the semantic parser.

There has also been considerable work in goal-directed dialog systems in domains such as information provision (Young et al., 2013). These systems model dialog as a POMDP and focus on either the problem of tracking belief state accurately over large state spaces (Young et al., 2010; Thomson and Young, 2010; Mrkšić et al., 2015; El Asri et al., 2016) or efficiently learning a dialog policy over this state space (Gašić and Young, 2014; Pietquin et al., 2011; Png et al., 2012). However, these systems typically assume a fixed natural language understanding component. In this work, we combine language learning with principled dialog strategy learning.

More recently, there has been work on modeling various components of a dialog system using neural networks (Mrkšić et al., 2015; Wen et al., 2015). There have also been some end-to-end neural network systems that simultaneously learn dialog policy and language comprehension for goal directed dialog (Wen et al., 2016; Williams and Zweig, 2016; Bordes and Weston, 2016), but they do not use a fully compositional semantic parser. Williams and Zweig (2016) use a very simple keyword-spotting based technique for processing input user utterances, which is unlikely to be able to handle out-of-vocabulary expressions for entities. Bordes and Weston (2016) explicitly attempt to handle out-of-vocabulary utterances in a neural dialog system but do not demonstrate much success. We expect that in a domain such as ours where out-of-vocabulary utterances are fairly likely, for example, in different forms of address for a person, a semantic parser that can be incrementally updated from a small number of interactions is likely to perform better. However, an empirical comparison of the two in domains where compositional language understanding is expected to be beneficial, is an interesting direction of future work.

3 Background - Partially Observable Markov Decision Process (POMDP)

A Partially Observable Markov Decision Process (POMDP) is a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$, where $\mathbb{S}$ is a set of states, $\mathbb{A}$ is a set of actions, $\mathbb{T}$ is a transition function, $\mathbb{R}$ is a reward function, $\mathbb{O}$ is a set of observations, $\mathbb{Z}$ is an observation function, γ is a discount factor and b_0 is an initial belief state (Kaelbling et al., 1998). These are defined as follows.

At any instant of time t, the agent is in a state $s_t \in \mathbb{S}$. This state is hidden from the agent and only a noisy observation $o_t \in \mathbb{O}$ of s_t is provided to it. The agent maintains a belief state b_t which is a distribution over all possible states it could be in at time t, where $b_t(s_i)$ gives the probability of being in state s_i at time t. Based on b_t, the agent chooses to take an action $a_t \in \mathbb{A}$ according to a policy π, commonly represented as a probability distribution over actions where $\pi(a_t|b_t)$ is the probability of taking action a_t when the agent is in belief state b_t. On taking action a_t, the agent is given a real-valued reward r_t, transitions to a state s_{t+1}, and receives a noisy observation o_{t+1} of s_{t+1}.

State transitions occur according to the probability distribution $P(s_{t+1}|s_t, a_t) = \mathbb{T}(s_t, a_t, s_{t+1})$, observations are related to the states by the probability distribution $P(o_t|s_t, a_{t-1}) = \mathbb{Z}(o_t, s_t, a_{t-1})$ and rewards obtained follow the distribution $P(r_t|s_t, a_t) = \mathbb{R}(s_t, a_t, s_{t+1})$.

The objective is to identify a policy π that is optimal in the sense that it maximizes the expected long term discounted reward, called return, given by

$$g = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r_t \right]$$

While there exist both exact and approximate methods for solving POMDPs, these do not usually scale well to the state spaces commonly used in dialog domains. This has led to the development of approximate representations that exploit domain-specific properties of dialog tasks to allow tractable estimation of the belief state and policy optimization (Young et al., 2013).

4 Background - Q-Learning using Kalman Temporal Differences

The quality of a policy π can be estimated using the action value function

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$$

The optimal policy satisfies the Bellman equation,

$$Q^*(s, a) = \mathbb{E}_{s'} \left[\mathbb{R}(s, a, s') + \gamma max_{a' \in \mathbb{A}} Q^*(s', a') \right]$$

When the state space is very large or continuous, Q^π cannot be computed for each state (or belief state) individually and is hence assumed to be a function with parameters θ over some features that represent the state. When the transition or reward dynamics are not constant (*non-stationary* problem), a suitable approximation is the Kalman Temporal Differences framework (Geist and Pietquin, 2010). This casts the function approximation as a filtering problem and solves it using Kalman filtering. The specialization for learning the optimal action value function is called the KTD-Q algorithm.

Filtering problems estimate hidden quantities X from related observations Y, modeling X and Y as random variables. When estimating action values, X corresponds to the function parameters, θ and the observations, Y, are the estimated returns, $r_t + \gamma \max_a \hat{Q}_{\theta_t}(s_{t+1}, a)$. Random noise is added to both of these to allow for parameters to change over time. The update rules are derived from Kalman Filtering Theory and not included here for the sake of brevity.

5 Dialog System

Our system initiates the dialog by requesting the user for a command. The user can command the system to perform two actions: navigation and delivery. Navigation has a single parameter for the destination. For example *"go to Alice's office"* would be a possible way to command the robot to perform a navigation command, whose location is a room that is the office of a person `alice`. Delivery has two parameters: the item to be delivered and the person to receive it. For example, *"bring Alice a hamburger"* would be a possible way to specify a delivery command whose patient is an item `hamburger` and recipient is a person `alice`.

The robot makes an initial guess of the desired action from the user's response, and then may ask clarification questions in case of insufficient understanding. At each step, it can respond with one of four dialog acts: asking the user to repeat their command, confirming a command or an argument value, requesting a specific argument of a command, and executing an action (thereby ending the dialog). A sample dialog is shown in Table 1.

Table 1: An sample clarification dialog

ROBOT	How can I help?
USER	bring the coffee to dr morgan
ROBOT	Sorry I couldn't understand that. Could you reword your original request?
USER	take the coffee to dr morgan
ROBOT	Sorry I couldn't understand that. Could you reword your original request?
USER	give the coffee cup to dr morgan
ROBOT	You want me to bring coffee for George?
USER	no mallory morgan
ROBOT	Whom would you like me to bring something to?
USER	mallory morgan
ROBOT	What would you like me to bring?
USER	coffee
ROBOT	I brought coffee to Mallory. Was this the correct action?
USER	yes

Given a user response, a semantic parser is used to convert utterances to logical forms which are then grounded using world knowledge (e.g. resolving the representation of *"Alice's office"* to a particular room). These grounded logical forms are then used to update the belief state maintained by the system. The next step to be taken by the system, given the belief state, is then chosen based on the current dialog policy. Once the conversation is complete, the parser and policy can be updated appropriately. These steps are outlined in greater detail in sections 5.1 and 5.2.

The dialog is considered a success if the final action taken is correct and a failure otherwise. The user also has the ability to prematurely end the dialog, and any conversation terminated in this manner is also considered a failure.

5.1 Semantic Parser Learning

Semantic parsing maps a natural language sentence such as *"Go to Alice's office"* to a logical form expressed in λ-calculus such as:

$$\begin{aligned}\texttt{walk}(\texttt{the}(\lambda x.\texttt{office}(x) \wedge \\ \texttt{possess}(\texttt{alice}, x) \wedge \\ \texttt{person}(\texttt{alice}))) \end{aligned} \quad (1)$$

Grounding against real-world knowledge, this will identify a room, say room 3512, which is an office that is owned by `alice`.

This formalism reduces the number of lexical entries the system needs to learn by exploiting compositional reasoning over language. For example, if the system learns that *"Alice Ashcraft"* and *"Alice"* both refer to the entity `alice`, no further lexical entries are required to resolve *"Go to Alice Ashcraft's office"* to the same semantic form (1).

In our system, semantic parsing is performed using probabilistic CKY-parsing with a Combinatory Categorial Grammar (CCG) and meanings associated with lexical entries. Perceptron-style updates to parameter values, that minimize the log-likelihood of the training data, are used during training to weight parses to speed search and give confidence scores in parse hypotheses (Zettlemoyer and Collins, 2005).

The parser is trained using paired sentences and logical forms. A small supervised training set is used to initialize the parser. Training continues using pairs obtained through weak supervision collected from user dialogs (Thomason et al., 2015). We use two such types of training pairs. The first consist of responses that are likely to correspond to the complete action, and the logical form induced by the action executed by the robot at the end of the dialog. Such responses are expected from the initial prompt to the user and questions that ask the user to repeat the command. We obtain multiple semantic parses for these responses, and parses that correspond to a complete command, and ground to the action finally taken by the robot, are paired with the response to form one set of training pairs. For example, from the conversation in Table 1, such training examples would be generated by pairing the responses *"bring the coffee to dr morgan"*, *"take the coffee to dr morgan"* and *"give the coffee cup to dr morgan"* with the semantic form `bring(mallory,coffee)`.

The second set of training pairs is obtained from the arguments of the action, such as the patient or location involved. This consists of responses to requests for specific arguments. Again, we consider multiple semantic parses for these responses, and select those that are of the correct syntactic form for a single argument value, and which ground to the corresponding argument value in the final action, to be paired with the response. For example, from the conversation in Table 1, such training examples would be generated by pairing the response *"mallory morgan"* with the semantic form `mallory`, and the response *"coffee"* with the semantic form `coffee`. These paired responses and semantic forms can then be used to retrain the parser between conversations.

This weak supervision may be somewhat noisy because it assumes that the form of the user's response matches the expected response type for the question. However, this is unlikely to generate spurious training examples, because we additionally place constraints on the syntax of the response. For example, if we receive *"Go to Bob's office"* as a response when we expect an argument value, since the response is an imperative sentence, not a noun phrase such as *"Bob's office"*, no training example would be generated from it. Prior experimental results (Artzi and Zettlemoyer, 2011; Thomason et al., 2015) suggest that learning using such weak (potentially noisy) supervision from clarification dialogs is effective at improving semantic parsers.

5.2 Dialog Strategy Learning

We use a POMDP to model dialog and learn a policy (Young et al., 2013), adapting the Hidden Information State model (HIS) (Young et al., 2010) to track the belief state as the dialog progresses. The key idea behind this approach is to group states into equivalence classes called partitions, and maintain a probability for each partition instead of each state. States within a partition are those that are indistinguishable to the system given the current dialog.

More concretely, our belief state can be factored into two main components. The first is the action (such as navigation and delivery) and argument values of the goal (such as the patient or location) which the user is trying to convey, $\mathbf{g} = \{g_a, g_{PAT}, g_{RCP}, g_{LOC}\}$. Goal parameters are represented in terms of semantic roles - *patient* (g_{PAT}), *recipient* (g_{RCP}) and *location* (g_{LOC}), to allow them to generalize across different actions. The second component contains information from the most recent user utterance, $\mathbf{u} = \{u_t, u_a, u_{PAT}, u_{RCP}, u_{LOC}\}$. Here, u_t is the type of the utterance – affirmation, denial, providing information about a complete action, or providing information about a specific argument. The components u_a, u_{PAT}, u_{RCP} and u_{LOC} respectively refer to the action, *patient*, *recipient* and *location* mentioned in the most recent user utterance, any of which can be `null`. This representation allows the method to be applicable to any action that can be expressed using up to 3 arguments.

After every user response, a beam of possible choices for $\mathbf{u}$ can be obtained by grounding the beam of top-ranked parses from the semantic parser. Semantic type-checking is used to disallow violations such as `alice` serving as the location argument of a navigation. However, there are a large number of possible values for $\mathbf{g}$ and we use the idea of partitions (Young et al., 2010) to track their probabilities in a tractable manner. A partition is a set of possible goals $\mathbf{g}^{(i)}$ which are equally probable given the conversation so far. The probability of a partition is the sum of probabilities of all goals in the partition. Initially, all goals are in a single partition of probability 1.

When an utterance hypothesis $\mathbf{u}$ is obtained, every partition currently maintained is split if needed into partitions that are either completely consistent or inconsistent with $\mathbf{u}$. For example, if a partition p has goals containing both navigation and delivery actions, and $\mathbf{u}$ specifies a delivery action, p will have to be split into one partition p_1 with all the navigation goals and another partition p_2 with all the delivery goals. The probability mass of p is divided between p_1 and p_2 in proportion to their sizes, to maintain the invariant that the probability of a partition is the sum of the probabilities of the goals contained in it. Then, given the previous system action $\mathbf{m}$, The belief $b(p, \mathbf{u})$ is calculated as in the HIS model as follows

$$b(p, \mathbf{u}) = k * P(\mathbf{u}) * T(\mathbf{m}, \mathbf{u}) * M(\mathbf{u}, \mathbf{m}, p) * b(p)$$

Here, $P(\mathbf{u})$ is the probability of the utterance hypothesis $\mathbf{u}$ given the user response, which is obtained from the semantic parser. $T(\mathbf{m}, \mathbf{u})$ is the probability that the type of the utterance hypothesis $\mathbf{u_t}$ is compatible with the previous system action $\mathbf{m}$, for example, if the system asks for the confirmation of a goal, the expected type of response is either affirmation or denial. This is determined by system parameters. $M(\mathbf{u}, \mathbf{m}, p)$ is a 0-1 value indicating whether the action and argument values mentioned in the utterance, system action, and partition agree with each other (an example of where they do not is an utterance mentioning an action not present in any goal in the partition) and $b(p)$ is the belief of partition p before the update, obtained by marginalizing out $\mathbf{u}$ from $b(p, \mathbf{u})$. k is a normalization constant that allows the expression to become a valid probability distribution. We also track the number of dialog turns so far.

The belief state is a distribution over all possible hypotheses given the conversation so far. The HIS model allows tracking probabilities of the potentially large number of hypotheses. However, it is difficult to learn a policy over this large a state space in a reasonable number of dialogs. Thus, we learn a dialog policy over a summary state as in previous work (Young et al., 2010; Gašić and Young, 2014). Table 2 contains the features used to learn the policy. Also, the policy is learned over abstract dialog acts (ask user to rephrase the entire goal, ask for a specific parameter, confirm a full/partial goal, execute a goal), which are converted to a system response by using parameters from the most likely hypothesis.

It is important to note that while only the top two hypotheses are used by the policy to choose the next action, it is useful to maintain the belief of all hypotheses because a hypothesis that is initially of low probability may become the most probable after additional turns of dialog.

Probability of top hypothesis
Probability of second hypothesis
Number of goals allowed by the partition in the top hypothesis
Number of parameters of the partition in the top hypothesis, required by its action, that are uncertain (set to the maximum value if there is more than one possible action)
Number of dialog turns used so far
Do the top and second hypothesis use the same partition (0-1)
Type of last user utterance
Action of the partition in the top hypothesis, or *null* if this is not unique

Table 2: Features used in summary space

The choice of policy learning algorithm is important because learning POMDP policies is challenging and dialog applications exhibit properties not often encountered in other reinforcement learning applications (Daubigney et al., 2012). We use KTD-Q (Kalman Temporal Difference Q-learning (Geist and Pietquin, 2010)) to learn the dialog policy as it was designed to satisfy some of these properties and tested in a dialog system with simulated users (Pietquin et al., 2011). The properties we wished to be satisfied by the algorithm were the following:

- Low sample complexity in order to learn from limited user interaction.
- An off-policy algorithm to enable the use of existing dialog corpora to bootstrap the system, and crowdsourcing platforms such as Amazon Mechanical Turk during training and evaluation.
- A model-free rather than a model-based algorithm because it is difficult to design a good transition and observation model for this problem (Daubigney et al., 2012).
- Robustness to non-stationarity because the underlying language understanding component changes with time (Section 5.1), which is likely to change state transitions.

To learn the policy, we provided a high positive reward for correct completion of the task and a high negative reward when the robot chose to execute an incorrect action, or if the user terminated the dialog before the robot was confident about taking an action. The system was also given a per-turn reward of -1 to encourage shorter dialogs.

6 Experimental Evaluation

The learning methods described above were applied to improve an initial dialog system using weak supervision from dialog interaction with real users. The dialog system was initialized using data from the conversation logs of Thomason et al. (2015), which also consist of interactions between a human user and a robot to which a high-level command must be communicated, and which asks clarifying questions when attempting to understand the dialog.

6.1 Initialization

The semantic parser was initialized using a small seed lexicon and trained on a small set of supervised examples constructed using templates for commands gathered from the conversation logs. While the parser can be used even if initialized using only a handful of hand-coded training examples, the increased robustness obtained by training on templated sentences results in less frustrating interaction during initial dialogs.

The RL component was first initialized with a Q-function approximation of the hand-coded policy of Thomason et al. (2015). The hand-coded policy was encoded in the form of if-then rules and had to be mapped to a Q-function appropriate for the KTD-Q algorithm, which assumes the Q-function is a probability distribution with a mean that is a linear function of the feature space. We obtain a set of “training points” for these linear weights by densely sampling the feature space. The hand coded policy is then used to identify the correct action for each of these feature vectors. The target for a training point is a high positive Q value when combined with the correct action and a 0 value when combined with any incorrect action. The weights were then initialized using linear regression over these examples. Finally, we trained the system on the above mentioned conversation logs, improving both the initial POMDP dialog policy and the semantic parser.

The simplest alternative to such an initialization would be to initialize the policy at random, but this would lead to a large number of frustrating dialogs before the system learns a reasonable policy. This can be avoided by training with a simulated user agent. However, such agents are not always realistic and their design requires parameters to be set ideally from existing conversation logs. However, since we use an off-policy algorithm, it is easier to train it directly from conversation logs, rather than develop a sufficiently realistic simulated agent.

Since the KTD-Q algorithm is off-policy, it can be trained using tuples containing the belief state, action taken, next belief state, and reward obtained from these logs. We update the policy using such tuples both in the initial training phase from existing conversation logs, and when updating the policy after collecting batches of conversations in our experiments.

6.2 Platform and setup

Our experiments were done through Mechanical Turk as in previous work (Thomason et al., 2015; Wen et al., 2016). During the training phase, each user interacted with one of four dialog agents (described in section 6.3), selected uniformly at random. Users were not told of the presence of multiple agents and were not aware of which agent they were interacting with. They were given a prompt for either a navigation or delivery task and were asked to have a conversation with the agent to accomplish the given task. No restrictions were placed on the language they could employ. We use visual prompts for the tasks to avoid linguistic priming (e.g. a picture of a hamburger instead of the word "hamburger"). Before users could begin the task, we used a validation step to ensure they were sufficiently fluent in English and understood the objectives of the task. Training dialogs were acquired in 4 batches of 50 dialogs each across all agents. After each batch, agents were updated as described in section 6.3.

A final set of 100 test conversations were then conducted between Mechanical Turk users and the trained agents. These test tasks were novel in comparison to the training data in that although they used the same set of possible actions and argument values, the same combination of action and argument values had not been seen at training time. For example, if one of the test tasks involved delivery of a `hamburger` to `alice`, then there may have been tasks in the training set to deliver a `hamburger` to other people and there may have been tasks to deliver other items to `alice`, but there was no task that involved delivery of a `hamburger` to `alice` specifically.

6.3 Dialog agents

We compared four dialog agents. The first agent performed only parser learning (described in Section 5.1). Its dialog policy was always kept to be a hand coded dialog policy similar to that of Thomason et al. (2015). This was the same hand-coded policy used to initialize the weights of the KTD-Q algorithm. Its parser was incrementally updated after each training batch. This agent is similar to the system used by Thomason et al. (2015) except that it uses the same state space as our other agents, to ensure that any differences in performance are not due to access to less information. Further, while Thomason et al. (2015) use only the top hypothesis from the parser to update the belief state, our agent uses a beam of parses, again to be more comparable to our other agents. In supplementary material, we also include an experiment which demonstrates that using multiple hypotheses from the semantic parser is more beneficial than using only a single one.

The second agent performed only dialog strategy learning. Its parser was always kept to be the initial parser that all agents started out with. Its policy was incrementally updated after each training batch using the KTD-Q algorithm. The third agent performed both parser and dialog learning; but instead of incrementally updating the parser and policy after each batch, they were trained at the end of the training phase using dialogs across all batches. This would not allow the dialog manager to see updated versions of the parser in batches after the first and adapt the policy towards the improving parser. We refer to this as *full* learning of parser and dialog policy. The fourth agent also performed both parser and dialog learning. Its parser and policy were updated incrementally after each training batch. Thus for the next training batch, the changes due to the improvement in the parser from the previous batch could, in theory, be demonstrated in the dialogs and hence contribute towards updating the policy in a manner consistent with it. We refer to this as *batchwise* learning of parser and dialog policy.

We did not include a system that performs no learning on either the parser or policy because it was shown by Thomason et al. (2015) that parser learning combined with a simple hand-coded policy outperforms this. We also did not attempt to update both parser and policy after each dialog because this forces all dialogs to be conducted in sequence, which does not allow us to fully leverage crowdsourcing platforms such as Mechanical Turk.

6.4 Experiment hypothesis

We hypothesized that the agent performing *batchwise* parser and policy learning would outperform the agents performing only parser or only dialog learning as we expect that improving both components is more beneficial. However, we did not necessarily expect the same result from *full* parser and dialog learning because it did not provide any chance to allow updates to propagate even indirectly from one component to another, exposing the RL algorithm to a more *non-stationary* environment. Hence, we also expected *batchwise* learning to outperform *full* learning.

6.5 Results and Discussion

The agents were evaluated on the test set using the following objective performance metrics: the fraction of successful dialogs (see 5) and the length of successful dialogs. We also included a survey at the end of the task asking users to rate on a 1–5 scale whether the robot understood them, and whether they felt the robot asked sensible questions.

Learning involved	% successful dialogs	Avg dialog length	Robot understood	Sensible questions
Parser	75	12.43	2.93	2.79
Dialog	59	11.73	2.55	2.91
Parser & Dialog - *full*	72	12.76	2.79	**3.28**
Parser & Dialog - *batchwise*	**78**	**10.61**	**3.30**	3.17

Table 3: Performance metrics for dialog agents tested. Differences in dialog success and subjective metrics are statistically significant according to an unpaired t-test with $p < 0.05$.

Table 3 gives the agents' performance on these metrics. All differences in dialog success and the subjective metrics are statistically significant according to an unpaired t-test with $p < 0.05$. In dialog length, the improvement of the *batchwise* learning agent over the agents performing only parser or only dialog learning are statistically significant.

As expected, the agent performing *batchwise* parser and dialog learning outperforms the agents performing only parser or only dialog learning, in the latter case by a large margin. We believe the agent performing only parser learning performs much better than the agent performing only dialog learning due to the relatively high sample complexity of reinforcement learning algorithms in general, especially in the partially observable setting. In contrast, the parser changes considerably even from a small number of examples. Also, we observe that *full* learning of both components does not in fact outperform only parser learning. We believe this is because the distribution of hypotheses obtained using the initial parser at training time is substantially different from that obtained using the updated parser at test time. We believe that *batchwise* training mitigates this problem because the distribution of hypotheses changes after each batch of training and the policy when updated at these points can adapt to some of these changes. The optimal size of the batch is a question for further experimentation. Using a larger batch is less likely to overfit updates to a single example but breaking the total budget of training dialogs into more batches allows the RL algorithm to see less drastic changes in the distribution of hypotheses from the parser.

We include an experiment in the supplementary material that quantifies the accuracy improvement of the parsers after training from dialogs. It is more difficult to quantitatively compare the policies before and after learning. Qualitatively, one of the noticeable differences is that the system tends to confirm or act upon lower probability hypotheses than is recommended by the initial hand-coded policy. This is possibly because as the parser improves, its top hypotheses are more likely to be correct, even if they are associated with a lower confidence score from the parser. A demonstration of this can be seen in tables 4 and 5. The learned policy results in a shorter dialog in the same situation because it allows the agent to act upon a hypothesis of lower probability. Also, the learned policy is stochastic, which is very helpful when the agent is not able to understand the user at all. For example, if the agent is unable to parse any of the initial instructions from the user, under a hand-coded policy, as its state has not changed, it would continue to repeat the question it had asked earlier, which prevents it from making any progress. However, in a stochastic policy, other more specific questions are likely to be substituted in between, and responses to these may allow the agent to make progress, which increases dialog success (table 6).

7 Conclusions

In this work, we have demonstrated that continuous dialog strategy learning and semantic parser learning can be successfully combined in a dialog system to enable an agent to better understand commands provided in natural language. Both the semantic parser and the dialog strategy can be automatically improved simultaneously using weak feedback provided during interaction with users rather than manually-labeled or artificially constructed training data. Ongoing parser learning could have confused the RL dialog learner by altering the underlying language understanding system while it was searching for an effective dialog policy. However, our results show that by using an appropriate RL algorithm and batchwise training regimen, this potential difficulty can be avoided, and both language understanding and dialog management can be improved simultaneously.

Acknowledgements

We would like to thank the members of the UT Austin BWI group for several insightful discussions, and Subhashini Venugopalan for her help with revising the paper draft. This research was supported by NSF EAGER (IIS-1548567) and NRI (IIS-1637736) grants.

Table 4: The robot following the learned policy takes the action despite the lower confidence, leading to a shorter dialog.

ROBOT	How can I help?
USER	bring the hamburger to peggy
ROBOT	What would you like me to bring?
USER	the hamburger
	The robot brought a hamburger to Peggy.
ROBOT	Was this the correct action?
USER	yes

Table 5: The robot following the hand-coded policy confirms the action before taking it, leading to a longer dialog.

ROBOT	How can I help?
USER	bring the hamburger to peggy
ROBOT	What would you like me to bring?
USER	the hamburger
ROBOT	You want me to bring hamburger for Peggy?
USER	yes
	The robot brought a hamburger to Peggy.
ROBOT	Was this the correct action?
USER	yes

Table 6: An example where a stochastic policy helps when the robot is unable to understand a full command.

ROBOT	How can I help?
USER	bring evelyn a coffee
ROBOT	Sorry I couldn't understand that. Could you reword your original request?
USER	bring evelyn a cup of coffee
ROBOT	You want me to bring coffee for someone?
USER	yes
ROBOT	Whom would you like me to bring coffee to?
USER	evelyn
ROBOT	Whom would you like me to bring coffee to?
USER	eve
ROBOT	I brought coffee to Eve. Was this the correct action?
USER	yes

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping Semantic Parsers from Conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 421–432.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683.*

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimization. *Journal of Selected Topics in Signal Processing*, 6(8):891–902.

Layla El Asri, Romain Laroche, and Olivier Pietquin. 2016. Compact and Interpretable Dialogue State Representation with Genetic Sparse Distributed Memory. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS).*

Milica Gašić and Steve Young. 2014. Gaussian Processes for POMDP-Based Dialogue Manager Optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

Matthieu Geist and Olivier Pietquin. 2010. Kalman Temporal Differences. *Journal of Artificial Intelligence Research*, 39(1):483–532.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and Acting in

Partially Observable Stochastic Domains. *Artificial intelligence*.

Thomas Kollar, Vittorio Perera, Daniele Nardi, and Manuela Veloso. 2013. Learning Environmental Knowledge from Task-Based Human-Robot Dialog. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4309.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-Domain Dialog State Tracking Using Recurrent Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient Batch Reinforcement Learning for Dialogue Management Optimization. *ACM Transactions on Audio, Speech, and Language Processing*, 7(3):7:1–7:21.

Shaowei Png, Joelle Pineau, and Brahim Chaib-Draa. 2012. Building Adaptive Dialogue Systems Via Bayes-Adaptive POMDPs. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):917–927.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. 2014. Back to the Blocks World: Learning New Actions through Situated Human-Robot Dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 89–97.

Stefanie Tellex, Ross A. Knepper, Adrian Li, Nicholas Roy, and Daniela Rus. 2014. Asking for Help Using Inverse Semantics. In *Proceedings of the 2016 Robotics: Science and Systems Conference (RSS)*.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to Interpret Natural Language Commands through Human-Robot Dialog. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1923–1929.

Blaise Thomson and Steve Young. 2010. Bayesian Update of Dialogue State: A POMDP framework for Spoken Dialogue Systems. *Computer Speech and Language*, 24(4):562–588.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. A Network-based End-to-End Trainable Task-oriented Dialogue System . In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 960–967.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The Hidden Information State Model: A Practical Framework for POMDP-based Spoken Dialogue Management. *Computer Speech and Language*, 24(2):150–174.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, 101(5):1160–1179.

Luke Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*.

Shiqi Zhang and Peter Stone. 2015. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*.

Supplementary Material

A Improvement in parser accuracy

The following experiment is an attempt to quantify the accuracy of the parsers after training from dialog. This was done by hand-annotating the semantic forms for commands from the test set used for the first experiment. The results can be seen in table 7. The parsers are evaluated in terms of $Recall@1$, which is the fraction of times the correct parse is the top parse predicted by the parser, and $Recall@10$, which is the fraction of times the correct parse occurs in the top 10 parses predicted by the parser.

Learning involved	$Recall@1$	$Recall@10$
None	0.564	0.611
Only parser	0.588	**0.671***
Only dialog	0.564	0.623
Parser & dialog - *full*	0.588	**0.647** ˆ
Parser & dialog - *batchwise*	0.576	**0.670***

Table 7: Comparison of performance of initial parser and parsers after updating various components, on paired commands and semantic forms. $*$ indicates that the difference in performance between this and the *Initial* parser on the same metric is statistically significant according to a paired t-test with $p < 0.05$ and ˆ indicates that the difference is trending significance ($p < 0.1$)

.

As expected, we observe that the initial parser (no learning) and the parser from the system performing only dialog learning, perform worse than the others, as the other systems update the parser used by these. The parser of the system performing only dialog learning is in fact a copy of the initial parser and was included only for completeness. Any difference in their performance is due to randomness. The parsers updated from dialogs improve in accuracy but the differences are found to be statistically significant only on $Recall@10$. The modest improvement is unsurprising given that the supervision provided is both noisy and weak. However, as seen in the main paper, even this modest improvement is sufficient to improve overall dialog success.

B Importance of multiple parse hypotheses

Many NLP systems typically return a list of top-n hypotheses, including semantic parsers. We use the entire beam of top-n parses when updating the state. This is expected to be beneficial in cases where that the correct hypothesis is not the top ranked but present in this beam. The following experiment demonstrates that using multiple parses when updating the state improves overall dialog success. We compared an agent that used the same parser and policy as in the batchwise training but only the top ranked parse from the parser to update its state, as opposed to a beam of parses when updating its state. These two systems differed in no other components.

Number of parses considered	% successful dialogs	Dialog length
1	0.59	9.17
10	**0.64**	12.18

Table 8: Comparison of an agent using only the top hypothesis from the semantic parser and another using the top 10 parses. All differences are statistically significant according to an unpaired t-test with $p < 0.05$.

Table 8 shows the usefulness of considering multiple hypotheses from the semantic parser. As expected, the agent using multiple parses performs the correct action a significantly higher fraction of times. The system using a single hypothesis has a shorter average length among its successful dialogs because it rarely succeeds in more complicated dialogs where the system needs repeated clarification or answers to multiple specific questions.