Learning Web-based Procedures by Reasoning over Explanations and Demonstrations in Context

Shashank Srivastava[1*] **Oleksandr Polozov**[2] **Nebojsa Jojic**[2] **Christopher Meek**[2]
[1]University of North Carolina, Chapel Hill [2]Microsoft Research, Redmond
ssrivastava@cs.unc.edu {polozov, jojic, meek}@microsoft.com

Abstract

We explore learning web-based tasks from a human teacher through natural language explanations and a single demonstration. Our approach investigates a new direction for semantic parsing that models *explaining a demonstration* in a context, rather than mapping explanations to demonstrations. By leveraging the idea of inverse semantics from program synthesis to reason backwards from observed demonstrations, we ensure that all considered interpretations are consistent with executable actions in any context, thus simplifying the problem of search over logical forms. We present a dataset of explanations paired with demonstrations for web-based tasks. Our methods show better task completion rates than a supervised semantic parsing baseline (40% relative improvement on average), and are competitive with simple exploration-and-demonstration based methods, while requiring no exploration of the environment. In learning to align explanations with demonstrations, basic properties of natural language syntax emerge as learned behavior. This is an interesting example of pragmatic language acquisition without any linguistic annotation.

1 Introduction

People routinely perform repetitive web-based tasks, involving sequences of clicking and typing actions. These include activities such as forwarding emails, booking flight tickets, ordering pizza, etc. These activities largely consist of small sequences of actions in an environment with restricted semantics, and are potentially amenable to automation. In this work, we explore whether an AI agent can be taught such tasks through natural language explanations and a single demonstration by a user (as one might teach such a task to a human assistant).

*Work done while the first author was at Microsoft Research.

Figure 1: AI assistants that can be taught web-based procedures by their users can have diverse practical applications. Here, we explore learning very simple tasks from the Mini World-of-Bits framework using natural language explanations and a single demonstration of the task

From the perspective of language understanding, this involves challenges such as converting instructional language to actions, resolving ambiguities through pragmatics, and learning script-like behavior. The web domain is rich in textual, structural and spatial features, allowing for exploration of multiple types of grounding behavior including spatial and visual language understanding, as well as reasoning over semi-structured data. Also, despite its richness, the tasks involved usually do not require much background knowledge.

From a practical perspective, teachable AI assistants can change the way people interact with computers. Today's conversational assistants such as Alexa or Cortana act on a small number of preprogrammed language commands (e.g., "What is the weather going to be like?"). However, they cannot be taught new functionalities important to a user (as in Figure 1). Enabling users to *teach* computers personalized procedures through explained demonstrations can make conversational AI systems fundamentally more useful.

In Section 2, we situate our work in the broader body of work on grounded semantic parsing and learning from language. Section 3 summarizes our framework and dataset. In Section 4, we describe our approach in detail. Here, we investigate a new paradigm for interpreting language in grounded contexts. Instead of mapping statements to logical forms that then execute in a context as in traditional semantic parsing, the method considers the set of possible typing and clicking actions in a context, identifies features of corresponding web elements and their relationships with other elements on the webpage, and aligns these to natural language explanations through a generative model. Section 5 describes the empirical evaluation. Our contributions are:

- An approach towards learning web-based tasks from a single explained demonstration.
- A dataset of explanations and demonstrations for tasks from the MiniWoB framework.
- Empirical results showing that explained demonstrations can be an effective mode of supervision for learning such tasks. Language can significantly reduce the number of samples needed compared to learning from demonstrations alone.

2 Related Work

Semantic Parsing: Supervised models for converting statements to logical forms have long been studied in a wide range of settings (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Kwiatkowksi et al., 2010; Yin and Neubig, 2017). More recent approaches focused on using weaker forms of supervision such as denotations or observations of world state (Berant et al., 2013; Clarke et al., 2010; Krishnamurthy and Mitchell, 2012) and semi-supervised methods aimed at efficient prototyping (Pasupat and Liang, 2015; Wang et al., 2015). These methods require more readily available supervision, such as question/answer pairs for model training, rather than annotations of logical forms. (Artzi and Zettlemoyer, 2013) learn to follow instructions in the context of robot navigation by conditioning parsing on environmental context. Artzi and Zettlemoyer (2011) use conversational feedback as a signal to induce logical forms for individual utterances from transcripts of conversations in a dialog-based setup. Some other recent approaches (Long et al., 2016; Guu et al., 2017) explore learning language from sequences of utterances and interactions in simple environments, which is conceptually similar to our work. Muhlgay et al. (2019) and Guu et al. (2017) explore better strategies to search the space of logical forms. While all of these methods are related to multiple facets of work, our method diverges from them in that the space of candidate logical forms is driven by the constraints of possible actions in an environment rather than the natural language utterance. This guarantees that all of the considered logical forms during search are consistent with executable actions in any novel context. Finally, some recent methods (Andreas et al., 2016) marginalize over latent interpretations of language in context of downstream tasks. We use a similar Bayesian approach, where actions are chosen by marginalizing logical forms (rather than choosing a single interpretation of an explanation).

Interactive Learning from Language: Several frameworks have leveraged natural language supervision to learn new tasks, starting with early work on the SHRLDU system (Winograd, 1972) and Interactive Task Learning (Laird et al., 2017). In particular, several reinforcement learning approaches have been explored in text-based environments for learning strategies, following instruction manuals, game playing, etc. (Branavan et al., 2009; Goldwasser and Roth, 2014; Misra et al., 2018; Narasimhan et al., 2015). These approaches leverage the ability to explore and interact with the environment to learning policies that lead to favourable outcomes. This is different from our goal here, where the agent needs to learn from a single explained demonstration of a task, and no interactivity with the environment is assumed. Some recent approaches have shown language explanations to be effective for learning realistic tasks including relation extraction, concept learning and question answering (Hancock et al., 2018; Srivastava et al., 2017, 2018; Andreas et al., 2018).

In terms of the goal and problem formulation, our approach extends multiples lines of previous work. Quirk et al. (2015)'s work is similar to ours in motivation in learning user-specified recipes, but has no aspects of grounding or demonstrations.(Wang et al., 2016) explore interactive parser training through language games in context of block-world environments. Pasupat et al. (2018) explore mapping natural language to specific elements on complex and realistic web-pages, although not in context of learning from demonstrations. Our framework directly extends previous work on learning web-based tasks from the Mini

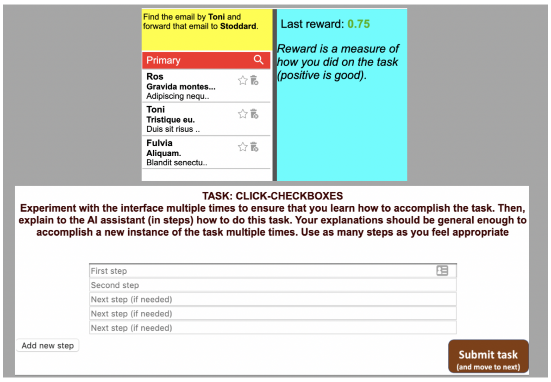

Figure 2: Crowd-worker interface used for collecting natural language explanations and demonstrations

Word-of-Bits framework using multiple demonstrations and exploration of the environment (Shi et al., 2017; Liu et al., 2018). In particular, our DSL extends the constraint language defined in Liu et al. (2018) to explore learning from explained demonstrations instead.

3 Framework and dataset

We build on the Mini World-of-bits (MiniWoB) framework (Shi et al., 2017), a collection of web-based tasks initially proposed as a testbed for reinforcement learning agents. The tasks vary in difficulty in terms of the number of actions required, variability between instances of the task, and types of reasoning involved (including clicking specified buttons, forwarding emails and playing tic-tac-toe). See the top half of Figure 2 for an example of a task. Each task consists of a task description (yellow box), and an interactive web interface.

While previous methods have focused on learning sequential decision making to complete these tasks through a mixture of exploration (the framework provides simulators, where correctly completing a task yields a reward) and behavior cloning (by observing multiple demonstrations from human users); our focus is on learning to complete these tasks in a one-shot sense (without any exploration). This is because the one-shot case is a much more realistic scenario for learning web-based procedures from a teacher. In practical situations (where there are no simulators), it would not be feasible for an AI agent to learn to book flights by booking multiple incorrect tickets, or manage a user's email by sending multiple incorrect emails. On the other hand, a paradigm where the agent attempts to generalize from a single demonstration and explanations can be feasible for many more such scenarios.

Task: Forward an email
Click on the segment that mentions Maureen
Click on the button name "Forward" at the bottom of the page
Type in the word 'Amata' in front of the row tagged 'to'
Click on the arrow button at the top of the page

Task: Select a radio button
Focus on the word sequence after Select
Click on the radio button to the left of the word sequence
Press submit

Figure 3: Examples of collected explanations

3.1 Dataset

We created a dataset of natural language explanations paired with demonstrations by human users for tasks from the MiniWoB framework. For this, crowdsourced workers on Amazon Mechanical Turk were asked to demonstrate how to complete these tasks and provide stepwise explanations to an AI assistant on how to complete the task. Since users would be unfamiliar with most of the tasks, for each task they were allowed to experiment with the interface as many times as they liked, and only the final demonstration was logged.

In all, we collect 520 demonstrations (each consisting of a sequence of click/type actions in the context of a MiniWoB task) paired with stepwise explanation sequences. Figure 3 shows samples of collected explanations. On average, each explanation sequence contains 3.3 explanations. The dataset contains 1719 explanations in total (individual steps), averaging 8.4 words per explanation. The size of the vocabulary of the explanations is 995. In general, workers found the teaching process to be engaging, with an average rating of 8.3 on a 1-10 scale on how they enjoyed the HIT in a post-completion survey. The dataset is available at `https://aka.ms/Web-D-E`.

Data characteristics: From a manual analysis of 100 randomly selected explanation sequences and task demonstrations, we find that in almost all cases (97%), the sequence of actions described in the explanations corresponds to the sequence of actions in the demonstration. More than 85% of explanations mention a clicking or typing action, while around 10% identify an entity/string on the webpage that is used in an action in the next step (e.g., the first explanation for the second task in Figure 3). Around 3% of the explanations correspond to conditionals and hypotheticals, which go beyond the scope of our approach. Roughly 15% of the explanations mention multiple entities on the webpage – usually specifying one element in relation to the other (e.g., *"the radio button to the right of the text-box"*).

Return Type	Operator	Type	Example invocation/description
Action	`Action Click (element)`		*click the icon ...*
	`Action TypeString (element, string)`		*enter the destination ...*
Identify Web Element(s)	`HasTag (tag)`	Semantic	*find the button ...*
	`HasText (string)`		*...that says 'submit'*
	`HasTextIncluding (string)`		*... the email that mentions Jeanette ...*
	`HasPosnHigh()`	Spatial	*the button at the top ...*
	`HasPosnLow()`		
	`RelnNear (element)`		
	`RelnSameRow (element)`		
	`RelnSameCol (element)`		*the icon next to ...*
	`RelnBelow (element)`		
	`RelnAbove (element)`		
	`RelnRightOf (element)`		
	`RelnLeftOf (element)`		*the link below the icon ...*
	`HasNumericIndex (int)`	Count	*the last option in the list ...*
Identify String(s)	`IndexedWord(int)`	From task description	*the last word ...*
	`BeforeWord(string)`		*the city after "from:"*
	`AfterWord(string)`		
	`FindMatchingContext(string, context)`		*enter "Seattle" as the source city ...*

Table 1: Major operators in DSL for learning of web-based procedures

3.2 DSL for semantic parsing

We define a domain specific language (DSL) for describing web-based procedures in terms of DOM elements by expanding on the constraint language in Liu et al. (2018). The DSL operators correspond to actions on DOM elements, element features and relations between them. The DSL defines the vocabulary of logical forms for parsing of user explanations, and grounds sensors and effectors in the web environment. Table 1 summarizes the DSL. There are three types of operations: (1) click and type actions on specified web elements (with a specified string, in case of a type action), (2) operations that filter elements on a page that satisfy a criterion, and (3) operations that filter strings based on a criterion. We include a special operator `FindMatchingContext` to accommodate cases in which the users provide explanations for an instance of a task with specific arguments mentioned in the task description (e.g., see the last row in Table 1). In this case, the operator can pick out the corresponding argument for the new instance by looking at the surrounding context in the new task description. The evaluation of logical forms in the DSL in the context of a webpage consists of set operations over all DOM elements on the webpage (and text-spans of up to two tokens for string operators). For example, the logical form `HasTag(type=button)` will evaluate to the set of elements on a page that have a HTML tag type with value `button`.

4 Learning from explanations and demonstrations

Our approach for learning web-based tasks, which we call *LED* – for *Learning from Explained Demonstrations*, models the process of explaining a demonstration of a task in a grounded context. We assume that the reasoning behind each action in a demonstration can be described by a logical form, l, in the DSL.[1] *LED*'s essential idea is that preferred logical forms are both (1) *consistent* with the user demonstration, d, in the observed context, c, and (2) *relevant* to the user's language explanations, x.

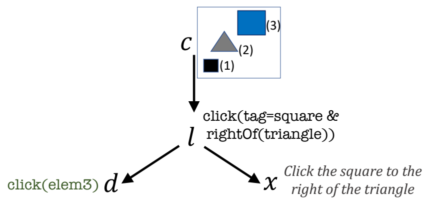

Figure 4: Modeling principle for Learning from Explained Demonstrations (LED). We prefer logical forms (l) that are both consistent with the user demonstration (d) in the context (c), and relevant to the user's explanations (x).

Figure 4 illustrates this for a toy-example, where the context consists of a web-page with three elements, the demonstration consists of a single action, and a corresponding explanation is provided. Based on the observed demonstration (that `elem3` was clicked), it is hard to infer the reason behind clicking it. Multiple logical forms in the DSL can be *consistent* with clicking `elem3` in this context. e.g., it is at the top of the page, its color is blue, etc. However, these interpretations would not justify the provided explanation as those logical forms are not *relevant* to the explanation. Modeling relevance between logical forms and explanations can help identify the reasoning behind user demonstrations.

This framing diverges from traditional semantic parsing, where statements x are mapped to logi-

[1]We do not infer individual logical forms corresponding to an explanation, since we marginalize over all logical forms that resolve to the same action in a context.

cal forms l (e.g., database queries), which are then are executed against a context c (e.g., a knowledge base) to get a denotation (corresponds here with a demonstration) d. i.e., $d = [\![l(x)]\!]_c$. In this model-theoretic view of semantics, parsed logical forms are not informed by the environmental context until execution. In comparison, *LED* roots logical forms in the observed context, and thus pragmatic consistency is ensured by design.[2] We maximize the log-likelihood of observing the explanations given the demonstration in a grounded context:

$$\mathcal{L}(\theta) = \log p(x|d, c) = \log \sum_l \underbrace{p(x|l)}_{\text{relevance}} \underbrace{p(l|d, c)}_{\text{consistency}} \quad (1)$$

Here, the first term corresponds to scoring relevance between logical forms and explanations (modeled using a semantic parsing model). The second term enforces consistency between candidate logical forms and the demonstration in the context, and can be deterministically evaluated. As we see in Section 4.2, consistency is enforced by temperature-based annealing during training.

4.1 Grounded Logical forms as latent variables

Eqn 1 marginalizes over latent logical forms. To make this tractable, we represent a logical form in a grounded context as an assignment of a tuple of discrete variables, $l := (e_0, f_0, r, e_1, f_1, a, t, f_t)$. These variables indicate things such as which DOM element is acted upon (e_0), if its relation (r) with another element on the page (e_1) is relevant, and so on. These are defined below.

- $e_0 \in \text{domElements}(c)$ denotes the DOM-element on which an action is performed. (e.g., $e_0 = elem3$ in Fig 4) This is observed from the demonstration, thus $p(e_0) = \mathbb{I}_{e_0 = e_{observed}}$.[3]
- $f_0 = (f_{01} \dots f_{0n_F})$ is a set of selector variables, where f_{0i} denotes if feature i of element e_0 is relevant for choosing it. Its domain is $\{\phi \cup \mathcal{F}_i\}$, where $\mathcal{F}_i$ is the range of values feature i can take. $f_{0i} = \phi$ denotes that the feature was not relevant for choosing e_0 (e.g., $f_{0\,\text{color}} = \phi$ in Fig 4). If $f_{0i} \neq \phi$, it can only take the observed value of the feature for e_0 in the context (e.g., $f_{0\,\text{tag}} = square$ in Fig 4). In Table 1, these correspond to operators that return web-elements and have names with prefix `Has`.
- r denotes if relation r between e_0 and another element on the webpage is relevant for choosing it. Its domain is $\{\phi \cup \mathcal{R}\}$, where $\mathcal{R}$ is the set of (binary) relations between elements in the DSL. In Table 1, these are operators that have names with prefix `Reln`. $r = \phi$ denotes that the no relation was relevant for choosing e_0. If $r \neq \phi$, it can only take the value of a relation that exists between e_0 and another element. (e.g., in Fig 4, r can't take the value `LeftOf`, since $elem3$ is the rightmost element in the context). Our choice of having a single variable for r disallows logical forms with multiple or nested relations. This was guided by an analysis of our dataset, where none of the collected explanations show such behavior.
- e_1 denotes that relation r between elements e_0 and e_1 is relevant for choosing e_0. Its domain is $\{\phi \cup \text{domElements}(c)\}$. $e_1 = \phi$ if and only if $r = \phi$, i.e. if no relation is relevant for choosing e_0. If $r = reln$, e_1 can only take values of elements such that $reln(e_0, e_1)$ is true in the context.
- $f_1 = (f_{11} \dots f_{1n_F})$ is a set of selector variables, where f_{1i} denotes if feature i of element e_1 is relevant. e.g., for *'click the checkbox next to the button that says submit'*, the `HasText` feature of the button is relevant). $f_{1i} = \phi$ denotes that feature i was not relevant. If $f_{1i} \neq \phi$, it can only take the observed value of the feature for e_1.
- a denotes the action performed on e_0 (click or type). This is observed from the demonstration.
- t denotes the string to type, if $a = type$. This is observed from the demonstration (and is a substring of the task description text).
- $f_t = (f_{t1} \dots f_{tn_T})$ is a set of selector variables, where f_{tj} denotes if the text feature j of t is relevant for choosing it (In Table 1, operators with a string return type correspond to text features).

Inverse Semantics: Assignments of values to these variables represents a search in the DSL space, since given any context, there is a mapping a from logical forms to an assignment of these variables. A key idea here is that, borrowing from program synthesis, we can leverage the *inverse semantics* of operators in the DSL (Polozov and Gulwani, 2015) to guarantee consistency of logical forms with the grounded context. i.e., at any step, the space of candidate logical forms we consider is consistent with the observed demonstration. This is possible because in our case, computing the inverse semantics for all operators in the DSL is feasible.[4]

[2] For example, in Figure 4, `click(tag=triangle & rightOf(square))` won't be considered for the provided utterance, as it is inconsistent with the context.

[3] $\mathbb{I}_{condition}$ denotes an indicator function for $condition$.

[4] Since there is only a relatively small number of candidates

As just described, our approach will use the context of the webpage leverage DSL inverse semantics to maintain an implicit set of candidate logical forms that are consistent with the observed demonstration. We will use variational inference to infer the logical forms that are most relevant to the seen explanations, and choose the action to take based on the inferred distribution over logical forms.

4.2 Model Description

In Eqn 1, the second term corresponds to a prior probability overs logical forms given a demonstration and context (webpage). Our representation of logical forms as latent variable assignments (from Section 4.1) enables us to decompose this probability into local factor distributions. We choose these local priors to correspond to distributions that are uniform over assignments that are consistent, and has zero support otherwise, similar to previous work on pragmatic reasoning (Frank and Goodman, 2012; Monroe et al., 2017). In other words, these distributions are proportional to indicator function over valid assignments of variables in each factor. As seen below, these define a prior over l that is also proportional to a simple indicator function over values of l that are consistent with the observed demonstration and context.

$$\begin{aligned} p(l \mid d, c) &= p(e_0, f_0, r, e_1, f_1, a, t, f_t \mid d, c) \\ &= p(e_0|d)\ p(f_0|e_0, c)\ p(e_1, r|e_0, c) \\ &\quad \times p(f_1|e_1, c)\ p(a, t|d)\ p(f_t|t, c) \\ &\propto \mathbb{I}_{Valid(e_0,d)}\ \mathbb{I}_{Valid(f_0,e_0,c)}\ \mathbb{I}_{Valid(e_1,r,e_0,c)} \\ &\quad \times \mathbb{I}_{Valid(f_1,e_1,c)}\ \mathbb{I}_{Valid(a,t,d)}\ \mathbb{I}_{Valid(f_t,t,c)} \\ &= \mathbb{I}_{Valid(l,d,c)} \end{aligned} \tag{2}$$

Substituting this in Eqn 1 and using Jensen's inequality, any distribution q over logical forms provides a lower-bound on the log-likelihood:

$$\begin{aligned} \mathcal{L}(\theta) &\geq \sum_l q(l) \log \frac{p(x|l)\ \mathbb{I}_{Valid(l)}}{q(l)} \\ &= \sum_l q(l) \big(\log p(x|l) + \log \mathbb{I}_{Valid(l)} \big) + \mathcal{H}_q \end{aligned} \tag{3}$$

where $\mathcal{H}_q$ is the entropy for distribution q. In Sec 4.1, we represent l as a tuple of variables. Next, we make a mean field approximation by assuming the distribution $q(l)$ decomposes as:

$$q(l) = q(e_0, a, t) \prod_i q_{f_{0i}} q_{e_1} q_r \prod_i q_{f_{1i}} \prod_j q_{f_{tj}} \tag{4}$$

Focusing on the unobserved variables (given a demonstration), we have $q(l) = q_{f_0} q_{e_1} q_r q_{f_1} q_{f_t}$.[5]

Parsing model: We assume that the probability of an explanation decomposes into the probability of individual words as $\log p_\theta(x|l) = \sum_{w \in x} \log p(w|f_0, r, f_1, f_t, a)$. Further, we assume that individual words are generated from features, relations and actions in the logical form as:

$$\begin{aligned} \log p(w|f_0, r, f_1, f_t, a) &= \log \frac{1}{C} \sum_k^{\{f_0, r, f_1, f_t, a\}} p(w|k)^{z_{kw}} p(z_{kw}) \\ &\geq \sum_k^{\{f_0, r, f_1, f_t, a\}} b_{kw} \big(\log p(w|k, z_{kw}) + \log p(z_{kw}) \big) + \mathcal{H}_{b_{kw}} \end{aligned} \tag{5}$$

Here, k is an index over values of f_0, r, f_1, f_t and a. z_{kw} denotes an alignment between a particular value of a feature, relation or action (k) and word w in the explanation, in which case the word is generated from the distribution $p(w|k)$. The presence of a summation inside of a logarithm makes maximizing this objective hard. We again use Jensen's inequality to get a bound by introducing variational distributions b_{kw} over alignments z_{kw}. b_{kw} can be thought of as representing the proportions of an explanation word contributed by specific feature values, relations or actions k in the logical form. Each $p(w|k)$ is parameterized as a multinomial distribution, θ_{kw}, over the vocabulary.

Training and Inference: Our model training follows a variational EM approach, where in the E-step, we perform inference for the latent logical form variables and alignment proportions, keeping the model parameters as fixed. In the M-step, we update the parameters, θ_{kw}, taking the variational distributions and alignments as fixed. Combining Eqn 2, Eqn 3 and Eqn 5, we get:

$$\begin{aligned} \mathcal{L}(\theta) \geq \sum_l q_{f_0} q_{e_1} q_r q_{f_1} q_{f_t} \Big(\big(\sum_w \sum_k b_{kw} [\log \theta_{kw}^{z_{kw}} \\ + \log p(z_{kw})] + \mathcal{H}_{b_{kw}} \big) + \log \mathbb{I}_{Valid(l,d,c)} \Big) \\ + \mathcal{H}_{f_0} + \mathcal{H}_{e_1} + \mathcal{H}_r + \mathcal{H}_{f_1} + \mathcal{H}_{f_t} \end{aligned} \tag{6}$$

Maximizing this objective w.r.t. the variational

– DOM elements or strings on the webpage – to search over. Compare this with an operation in arithmetic, e.g., `add(int, int)`, which might require a search over infinite co-domains.

[5] Using q_{f_0} as shorthand notation for the product of variational distributions $\prod_i q_{f_{0i}}$, and so on.

distributions yields the following E-step updates:[6]

$$q_{f_{0i}}(v_{f_{0i}}) \propto \exp\Big(\sum_{w} b_{v_{f_{0i}}w}\log(\theta_{v_{f_{0i}}w}) + \log \mathbb{I}_{Valid(v_{f_{0i}}, e_0, c)}\Big)$$
$$q_{e_1}(v_{e_1}) \propto \exp\Big((\sum_{f_{1i}}\sum_{w} b_{f_{1i}w}\sum_{v_{f_{1i}}} q_{f_{1i}}(v_{f_{1i}})\log\theta_{v_{f1iw}} + \log \mathbb{I}_{Valid\,(v_{f_{1i}}, e_1, c)}) + (\sum_{v_r} q_r(v_r)\sum_{w} b_{v_r w}\log\theta_{v_r w} + \log \mathbb{I}_{Valid(e_1, v_r, e_0, c)})\Big)$$
$$q_r(v_r) \propto \exp\Big(\sum_{v_{e_1}} q_{e_1}(v_{e_1})(\sum_{w} b_{v_r w}\log\theta_{v_r w}) + \log \mathbb{I}_{Valid(e_1, v_r, e_0, c)}\Big)$$
$$q_{f_{1i}}(v_{f_{1i}}) \propto \exp\Big(\sum_{v_{e_1}} q_{e_1}(v_{e_1})(\sum_{w} b_{v_{f_{1i}}w}\log\theta_{v_{f_{1i}}w}) + \log \mathbb{I}_{Valid(v_{f_{1i}}, e_1, c)}\Big)$$
$$q_{f_{tj}}(v_{f_{tj}}) \propto \exp\Big(\sum_{w} b_{v_{f_{tj}}w}\log(\theta_{v_{f_{tj}}w}) + \log \mathbb{I}_{Valid(v_{f_{tj}}, t, c)}\Big) \quad (7)$$

Similarly, the updates for the alignment proportions (taking $p(z_{kw})$ in Eqn 6 to be uniform) are:

$$b_{kw} \propto \exp\big(\sum_{k} q_k(k)\log\theta_{kw}\big) \quad (8)$$

LED(+syntax): The above approach allows for arbitrary alignments between words and features, relations or actions in the grounded logical form (k), essentially representing x as a bag-of-words.[7] We also explore a variant that models x as a sequence of tokens by introducing a prior over joint alignments $\mathbf{z}_{kx} = z_{k_1 w_1} \ldots z_{k_T w_T}$ in a sentence $x := w_1 \ldots w_T$ (in Eqn 5). This is done by simply modeling $p(\mathbf{z}_{kx})$ with pairwise transition probabilities as $p(\mathbf{z}_{kx}) := \prod_n p(z_{k_t}|z_{k_{t-1}}) = \prod_n \mathcal{T}_{k_t, k_{t-1}}$. In this case, updates for alignment proportions (Eqn. 8) correspond with emission probabilities in a HMM (which we omit here for brevity).

Since the updates in Eqn 7 and Eqn 8 are cyclic, in each E-step, we make 20 iterations of updates to the variational distributions and alignment proportions in a round-robin schedule. We note that consistency is enforced during training by the log-of-indicator-variable terms in Eqn 7. This is because any inconsistent assignments get a score of $\log(0)$, which tends to negative infinity. However, to ensure smooth training (and alleviate modeling issues from our mean field approximation), we leverage an annealing based strategy, where we incrementally increase the penalty for $\log(0)$ terms during training as $-N/2$ for the N'th EM iteration (for large N, this also is a prohibitive penalty). In our experiments, this was seen to improve training.

In the M-step, we maximize the objective w.r.t. θ_k:

$$\theta_k(w) \propto \exp\Big(\sum_{n}\sum_{w \in x_n} b^n_{kw} q_k\Big) \quad (9)$$

The one exception is a special copy mechanism for string-valued features. For these, θ_{kw} is not learned, but simply corresponds to an indicator function denoting if w matches the value of the feature. e.g., $\theta_{\text{HasText('submit'),'submit'}} = 1$.

5 Experiments

We next discuss *LED*'s empirical performance.

5.1 Procedure Learning performance

First, we evaluate the method for completion rates on tasks from the MiniWoB framework. Following Liu et al. (2018), we filtered 40 tasks from the MiniWoB framework (Shi et al., 2017) that require only clicking and typing actions. During training of the *LED* model, we sample an explained demonstration for each of the 40 tasks, and models are trained on the aggregate of these (the model sees one explanation-demonstration pair for a task). For testing, models are evaluated on a new instance of a task, where the model greedily computes the demonstration d (specifying a click or typing action on a web element in the current DOM) that would maximize $p(x|d, c)$ (see Eqn 1) and executes the corresponding actions. The method then moves to the next explanations. This requires an enumeration of all possibly clicking and typing actions that can be performed in a context c at every step.[8] Since the number of actions in a demonstration can be different from the number of steps in the explanation, we heuristically align the sequence of actions in demonstrations to the sequence of sentences in the explanations in our dataset based on a small manually defined list of trigger words.

A direct comparison of *LED* with other approaches is not possible, since they differ considerably in the type of supervision and resources used. Nonetheless, here we compare *LED*'s performance with the following two methods to get a coarse sense of its effectiveness:

[6] The optimal value for the concave problem $\sum_j x_j \log \frac{y_j}{x_j}$ s.t. $\sum_j x_j = 1$ is achieved when $x^*_j \propto y_j$.

[7] E.g., this won't differentiate between "*click the URL below the button*" and "*click the button below the URL*".

[8] This is possible since the set of actionable elements on a webpage, and the set of candidate strings that can be typed (up to two length tokens from task description) are not large.

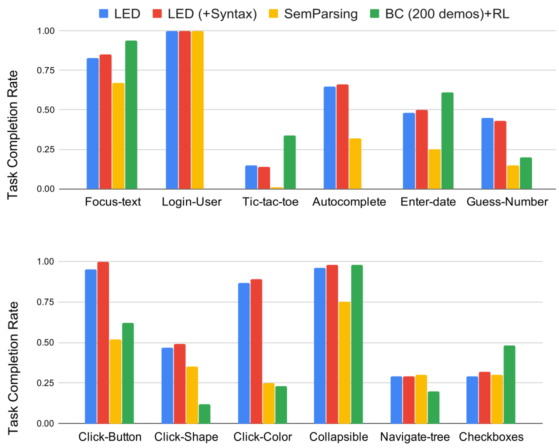

Figure 5: Task-completion rates for MiniWoB tasks with varying difficulty. Rates are calculated over 100 new instances of each task

1. *SemParse*: This is a supervised semantic parsing baseline, trained on a manually annotated dataset of around 300 explanations labeled with their DSL logical forms (covering roughly one annotated explanation sequence for every task). The model is based on a sequence-to-sequence neural semantic parser from Jia and Liang (2016). During testing, the method parses the sequence of explanations to logical forms, and sequentially (attempts to) executes the predicted logical forms. In contrast, *LED* requires no logical form annotations. However, it leverages the inverse semantics of the DSL operators, which may not be feasible for every DSL.
2. *BC+RL*: This is the original approach from Shi et al. (2017), who proposed the MiniWoB framework and consists of behavior cloning and exploration. This learns a task by supervised learning on about 200 demonstrations, followed by exploration via reinforcement learning to fine-tune the learned policies. In comparison, *LED* requires no exploration of the environment but leverages additional supervision in the form of natural language explanations. Multiple methods have since explored other RL-based approaches, resulting in much improved performance (Liu et al., 2018; Luo, 2019; Jia et al., 2019). In particular, Liu et al. (2018) leverage a constraint language similar to our DSL to train a RL policy to get large gains in performance. However, all these methods require multiple demonstrations and exploration of the environment.

Figure 5 shows task completion performance for different methods on a subset of tasks from the MiniWoB framework. We compute task completion rates over 100 randomly selected test instances of each task. The differences between instances involves different arguments for a task and differences in the state of the environment. Firstly, we note that the *LED* approaches consistently outperforms *SemParse* across all tasks. This is a strong result, since *LED* does not have access to logical form annotations for explanations as *SemParse* does. This strongly indicates that knowledge of the pragmatic context is important for language interpretation in this domain, since our approach which roots logical forms in observed demonstrations performs better or as well for all but one task.

We note that there is a large variance among tasks in terms of amenability to learning from explanations or exploration. For tasks like *tic-tac-toe*, explanation-based methods perform poorly as expected, since learning the game involves reasoning that is hard to explain through step-wise explanation of a demonstration, but can be more naturally learned from exploration. On the other hand, explanation-based methods perform well on tasks that are easily expressed through language. On the whole, the *LED* approaches and are roughly competitive with *BC+RL*, while requiring no exploration and only a single demonstration. Note that unlike exploration-based methods, *LED* and *SemParse* can potentially generalize to new tasks during testing (where no demonstration is seen during training) from explanations and context only.

We also note that *LED(+Syntax)* generally outperforms vanilla *LED*, although the effect size is not large. However, this trend is statistically significant (binomial test, $p < 0.1$).

Approach	Action-prediction accuracy
LED (+Syntax)	0.45
LED	0.43
SemParse	0.39
Random	0.28

Table 2: Semantic parsing performance (predicted action match) for interpreting individual explanations in a context

5.2 Language Interpretation performance

Next, we quantitatively evaluate the parsing performance of our method at the level of individual explanations (rather than task completion rate). For this, we evaluate the trained models on explanations from a set of 80 demonstrations from the dataset (unseen during training), where we calculate the match between the predicted action from an explanation in the context, and the actual action in the logged demonstration (accuracy of pre-

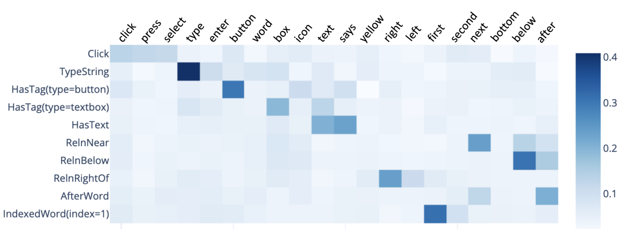

Figure 6: Heatmap showing learned values of θ_{kw} for 20 frequent words w in and representative values of k. Darker shades correspond to higher probability values.

dicted action in a context). Table 2 summarizes this performance, which shows a similar trend as Section 5.1. Both LED methods perform substantially better than *SemParse*, and all three methods perform much better than randomly choosing the next executable action in the context ($Random$). We note again that *LED*'s involves no logical form annotations, and is driven purely by grounding explanations in observed demonstrations.

5.3 Visualizing learned language

Figure 6 depicts the learned lexicon by visualizing a representative subset of learned θ_{kw} values for *LED (+Syntax)* (from Sec 4.2) as a heatmap. We note that the model correctly induces mappings between words and DSL operators. The rows and columns are manually ordered to emphasize the block diagonal structure.

Table 3 shows the learned transition probabilities, $\mathcal{T}_{k_1,k_2}$, for *LED (+Syntax)*. To reduce model size, we share parameters for values of k corresponding to types f_0, r, f_1, f_t and a. A common template about the general structure of user explanations is reflected from the parameter values. Most explanations start with the description of the action a, followed by mentioning features that identify the relevant element f_0. In fact, f_0 distributions generate the majority of words in most explanations. Relation mentions, when present, usually follow this, in turn followed by features corresponding to f_0, reflective of a VSO word order in most explanations. Diagonal values are substantially higher, indicating that words describing specific objects and actions tend to cluster together, as would be expected from the semantics of natural language.

5.4 Common Errors

From a qualitative error analysis, we note that most errors in task learning come from three sources. Firstly, although the method learns reasonable mappings between words and semantic operators, the method often misaligns attributes of different elements, even with the *LED(+Syntax)* model. This is likely because the training data is not adequate to learn these constraints, and methods that enforce these through informed priors maybe more effective. Another common error is due to challenges with anaphora resolution and discourse referents. Finally, a large number of explanations are not explicit in describing the sequence of actions required to perform a task, and some needed actions remain unmentioned. While this would be expected in realistic computer-human interactions, fixing these errors is beyond the scope of the current method.

	a	f_0	r	f_1	f_t
a	0.12	0.70	0.07	0.04	0.07
f_0	0.05	0.82	0.08	0.04	0.01
r	0.01	0.12	0.57	0.26	0.04
f_1	0.03	0.07	0.22	0.63	0.04
f_t	0.08	0.30	0.09	0.06	0.47

Table 3: Learned transition probabilities between latent variable categories for *LED (+Syntax)*. These reflect a prominence of VSO sentence structures in user explanations.

6 Conclusion

Our work here is a step in the direction of teachable AI agents that can learn new behavior from conversational interactions with ordinary users. In terms of technique, our bottom-up approach to generating logical forms ensures consistency between interpretations and the ambient context during search. Conversely, this would be complicated in domains with rich composition and nesting in logical forms, which go beyond simple features and relations. e.g., "*click the third email from Jeanette*", and where modeling inverse semantics is infeasible.

Here, we posed the learning of web-based tasks as similar to instruction-following problem, with no aspect of interactivity or exploration of the environment. In future work, the possibility of learning from a mix of explanations, exploration and a limited budget of interaction with the environment can be explored. Also, language grounding models that incorporate richer alignments between explanations and demonstrations can lead to more effective learning. Since *LED* only requires tokenization as pre-processing, it can possibly extend to low resource scenarios. In terms of problem framing, interactive use-cases that enable the agent to ask questions when it is confused may also be realistic. Future work can also explore curriculum learning in this domain, by first learning simpler tasks, which can be compositionally invoked in explanations for complex tasks.

References

Jacob Andreas, Dan Klein, and Sergey Levine. 2018. Learning with latent language. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2166–2179. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 39–48. IEEE Computer Society.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 421–432, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62.

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*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90, Suntec, Singapore. Association for Computational Linguistics.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27, Uppsala, Sweden. Association for Computational Linguistics.

Michael C. Frank and Noah D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Mach. Learn.*, 94(2):205–232.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1051–1062, Vancouver, Canada. Association for Computational Linguistics.

Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. 2018. Training classifiers with natural language explanations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1884–1895. Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Sheng Jia, Jamie Kiros, and Jimmy Ba. 2019. DOM-Q-NET: grounded RL on structured language. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765, Jeju Island, Korea. Association for Computational Linguistics.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA. Association for Computational Linguistics.

John E. Laird, Kevin A. Gluck, John R. Anderson, Kenneth D. Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario D. Salvucci, Matthias Scheutz, Andrea Thomaz, J. Gregory Trafton, Robert E. Wray, Shiwali Mohan, and James R. Kirk. 2017. Interactive task learning. *IEEE Intell. Syst.*, 32(4):6–21.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. 2018. Reinforcement learning on web interfaces using workflow-guided exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Berlin, Germany. Association for Computational Linguistics.

Katie Luo. 2019. Goal-induced inverse reinforcement learning. Technical Report EECS-2019-81, University of California, Berkeley.

Dipendra Misra, Ming-Wei Chang, Xiaodong He, and Wen-tau Yih. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2442–2452, Brussels, Belgium. Association for Computational Linguistics.

Will Monroe, Robert X.D. Hawkins, Noah D. Goodman, and Christopher Potts. 2017. Colors in context: A pragmatic neural model for grounded language understanding. *Transactions of the Association for Computational Linguistics*, 5:325–338.

Dor Muhlgay, Jonathan Herzig, and Jonathan Berant. 2019. Value-based search in execution space for mapping instructions to programs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1942–1954, Minneapolis, Minnesota. Association for Computational Linguistics.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Lisbon, Portugal. Association for Computational Linguistics.

Panupong Pasupat, Tian-Shun Jiang, Evan Liu, Kelvin Guu, and Percy Liang. 2018. Mapping natural language commands to Web elements. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4970–4976, Brussels, Belgium. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. 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)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Oleksandr Polozov and Sumit Gulwani. 2015. FlashMeta: A framework for inductive program synthesis. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA 2015, pages 107–126, New York, NY, USA. ACM.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for If-This-Then-That recipes. 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)*, pages 878–888, Beijing, China. Association for Computational Linguistics.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. 2017. World of bits: An open-domain platform for Web-based agents. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3135–3144.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2017. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1527–1536, Copenhagen, Denmark. Association for Computational Linguistics.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 306–316. Association for Computational Linguistics.

Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning language games through interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. 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)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.

Terry Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, pages 658–666. AUAI Press.