Learning Language Games through Interaction

Sida I. Wang **Percy Liang** **Christopher D. Manning**
Computer Science Department
Stanford University
`{sidaw,pliang,manning}@cs.stanford.edu`

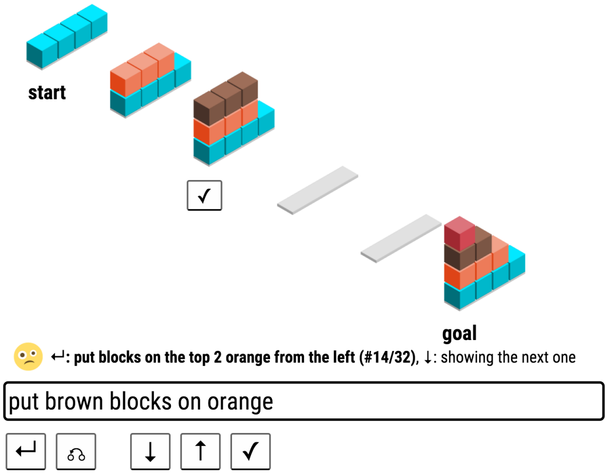
Abstract

We introduce a new language learning setting relevant to building adaptive natural language interfaces. It is inspired by Wittgenstein's language games: a human wishes to accomplish some task (e.g., achieving a certain configuration of blocks), but can only communicate with a computer, who performs the actual actions (e.g., removing all red blocks). The computer initially knows nothing about language and therefore must learn it from scratch through interaction, while the human adapts to the computer's capabilities. We created a game called SHRDLURN in a blocks world and collected interactions from 100 people playing it. First, we analyze the humans' strategies, showing that using compositionality and avoiding synonyms correlates positively with task performance. Second, we compare computer strategies, showing that modeling pragmatics on a semantic parsing model accelerates learning for more strategic players.

1 Introduction

Wittgenstein (1953) famously said that *language derives its meaning from use*, and introduced the concept of *language games* to illustrate the fluidity and purpose-orientedness of language. He described how a builder B and an assistant A can use a primitive language consisting of four words—*'block'*, *'pillar'*, *'slab'*, *'beam'*—to successfully communicate what block to pass from A to B. This is only one such language; many others would also work for accomplishing the cooperative goal.

This paper operationalizes and explores the idea of language games in a learning setting, which we call *interactive learning through language games* (ILLG). In the ILLG setting, the two parties do not initially speak a common language, but nonetheless need to collaboratively accomplish a goal. Specifically, we created a game called SHRDLURN,[1] in homage to the seminal work of Winograd (1972). As shown in Figure 1, the objective is to transform a start state into a goal state, but the only action the human can take is entering an utterance. The computer parses the utterance and produces a ranked list of possible interpretations according to its current model. The human scrolls through the list and chooses the intended one, simultaneously advancing the state of the blocks and providing feedback to the computer. Both the human and the computer wish to reach the goal state



↵: put blocks on the top 2 orange from the left (#14/32), ↓: showing the next one

put brown blocks on orange

Figure 1: The SHRDLURN game: the objective is to transform the start state into the goal state. The human types in an utterance, and the computer (which does not know the goal state) tries to interpret the utterance and perform the corresponding action. The computer initially knows nothing about the language, but through the human's feedback, learns the human's language while making progress towards the game goal.

[1] Demo: `http://shrdlurn.sidaw.xyz`

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2368–2378,
Berlin, Germany, August 7-12, 2016. ©2016 Association for Computational Linguistics

(only known to the human) with as little scrolling as possible. For the computer to be successful, it has to learn the human's language quickly over the course of the game, so that the human can accomplish the goal more efficiently. Conversely, the human must also accommodate the computer, at least partially understanding what it can and cannot do.

We model the computer in the ILLG as a semantic parser (Section 3), which maps natural language utterances (e.g., *'remove red'*) into logical forms (e.g., `remove(with(red))`). The semantic parser has no seed lexicon and no annotated logical forms, so it just generates many candidate logical forms. Based on the human's feedback, it performs online gradient updates on the parameters corresponding to simple lexical features.

During development, it became evident that while the computer was eventually able to learn the language, it was learning less quickly than one might hope. For example, after learning that *'remove red'* maps to `remove(with(red))`, it would think that *'remove cyan'* also mapped to `remove(with(red))`, whereas a human would likely use mutual exclusivity to rule out that hypothesis (Markman and Wachtel, 1988). We therefore introduce a pragmatics model in which the computer explicitly reasons about the human, in the spirit of previous work on pragmatics (Golland et al., 2010; Frank and Goodman, 2012; Smith et al., 2013). To make the model suitable for our ILLG setting, we introduce a new online learning algorithm. Empirically, we show that our pragmatic model improves the online accuracy by 8% compared to our best non-pragmatic model on the 10 most successful players (Section 5.3).

What is special about the ILLG setting is the real-time nature of learning, in which the human also learns and adapts to the computer. While the human can teach the computer any language—English, Arabic, Polish, a custom programming language—a good human player will choose to use utterances that the computer is more likely to learn quickly. In the parlance of communication theory, the human *accommodates* the computer (Giles, 2008; Ireland et al., 2011). Using Amazon Mechanical Turk, we collected and analyzed around 10k utterances from 100 games of SHRDLURN. We show that successful players tend to use compositional utterances with a consistent vocabulary and syntax, which matches the inductive biases of the computer (Section 5.2). In addition, through this interaction, many players adapt to the computer by becoming more consistent, more precise, and more concise.

On the practical side, natural language systems are often trained once and deployed, and users must live with their imperfections. We believe that studying the ILLG setting will be integral for creating adaptive and customizable systems, especially for resource-poor languages and new domains where starting from close to scratch is unavoidable.

2 Setting

We now describe the interactive learning of language games (ILLG) setting formally. There are two players, the human and the computer. The game proceeds through a fixed number of levels. In each level, both players are presented with a starting state $s \in \mathcal{Y}$, but only the human sees the goal state $t \in \mathcal{Y}$. (e.g. in SHRDLURN, $\mathcal{Y}$ is the set of all configurations of blocks). The human transmits an utterance x (e.g., *'remove red'*) to the computer. The computer then constructs a ranked list of candidate actions $Z = [z_1, \dots, z_K] \subseteq \mathcal{Z}$ (e.g., `remove(with(red))`, `add(with(orange))`, etc.), where $\mathcal{Z}$ is all possible actions. For each $z_i \in Z$, it computes $y_i = [\![z_i]\!]_s$, the successor state from executing action z_i on state s. The computer returns to the human the ordered list $Y = [y_1, \dots, y_K]$ of successor states. The human then chooses y_i from the list Y (we say the computer is *correct* if $i = 1$). The state then updates to $s = y_i$. The level ends when $s = t$, and the players advance to the next level.

Since only the human knows the goal state t and only the computer can perform actions, the only way for the two to play the game successfully is for the human to somehow encode the desired action in the utterance x. However, we assume the two players do not have a shared language, so the human needs to pick a language and teach it to the computer. As an additional twist, the human does not know the exact set of actions $\mathcal{Z}$ (although they might have some preconception of the computer's capabilities).[2] Finally, the human only sees the outcomes of the computer's actions, not the actual logical actions themselves.

We expect the game to proceed as follows: In the beginning, the computer does not understand

[2] This is often the case when we try to interact with a new software system or service before reading the manual.

what the human is saying and performs arbitrary actions. As the computer obtains feedback and learns, the two should become more proficient at communicating and thus playing the game. Herein lies our key design principle: *language learning should be necessary for the players to achieve good game performance*.

SHRDLURN. Let us now describe the details of our specific game, SHRDLURN. Each state $s \in \mathcal{Y}$ consists of stacks of colored blocks arranged in a line (Figure 1), where each stack is a vertical column of blocks. The actions $\mathcal{Z}$ are defined compositionally via the grammar in Table 1. Each action either adds to or removes from a set of stacks, and a set of stacks is computed via various set operations and selecting by color. For example, the action `remove(leftmost(with(red)))` removes the top block from the leftmost stack whose topmost block is red. The compositionality of the actions gives the computer non-trivial capabilities. Of course, the human must teach a language to harness those capabilities, while not quite knowing the exact extent of the capabilities. The actual game proceeds according to a curriculum, where the earlier levels only need simpler actions with fewer predicates.

We designed SHRDLURN in this way for several reasons. First, visual block manipulations are intuitive and can be easily crowdsourced, and it can be fun as an actual game that people would play. Second, the action space is designed to be compositional, mirroring the structure of natural language. Third, many actions z lead to the same successor state $y = [\![z]\!]_s$; e.g., the *'leftmost stack'* might coincide with the *'stack with red blocks'* for some state s and therefore an action involving either one would result in the same outcome. Since the human only points out the correct y, the computer must grapple with this indirect supervision, a reflection of real language learning.

3 Semantic parsing model

Following Zettlemoyer and Collins (2005) and most recent work on semantic parsing, we use a log-linear model over logical forms (actions) $z \in \mathcal{Z}$ given an utterance x:

$$p_\theta(z \mid x) \propto \exp(\theta^\top \phi(x, z)), \qquad (1)$$

where $\phi(x, z) \in \mathbb{R}^d$ is a feature vector and $\theta \in \mathbb{R}^d$ is a parameter vector. The denotation y (successor state) is obtained by executing z on a state s; formally, $y = [\![z]\!]_s$.

Features. Our features are n-grams (including skip-grams) conjoined with tree-grams on the logical form side. Specifically, on the utterance side (e.g., *'stack red on orange'*), we use unigrams (*'stack'*, $*$, $*$), bigrams (*'red'*, *'on'*, $*$), trigrams (*'red'*, *'on'*, *'orange'*), and skip-trigrams (*'stack'*, $*$, *'on'*). On the logical form side, features corresponds to the predicates in the logical forms and their arguments. For each predicate h, let $h.i$ be the i-th argument of h. Then, we define *tree-gram* features $\psi(h, d)$ for predicate h and depth $d = 0, 1, 2, 3$ recursively as follows:

$$\psi(h, 0) = \{h\},$$
$$\psi(h, d) = \{(h, i, \psi(h.i, d-1)) \mid i = 1, 2, 3\}.$$

The set of all features is just the cross product of utterance features and logical form features. For example, if x = *'enlever tout'* and z = `remove(all())`, then features include:

(*'enlever'*, `all`) (*'tout'*, `all`)
(*'enlever'*, `remove`) (*'tout'*, `remove`)
(*'enlever'*, (`remove`, 1, `all`))
(*'tout'*, (`remove`, 1, `all`))

Note that we do not model an explicit alignment or derivation compositionally connecting the utterance and the logical form, in contrast to most traditional work in semantic parsing (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Liang et al., 2011; Kwiatkowski et al., 2010; Berant et al., 2013), instead following a looser model of semantics similar to (Pasupat and Liang, 2015). Modeling explicit alignments or derivations is only computationally feasible when we are learning from annotated logical forms or have a seed lexicon, since the number of derivations is much larger than the number of logical forms. In the ILLG setting, neither are available.

Generation/parsing. We generate logical forms from smallest to largest using beam search. Specifically, for each size $n = 1, \ldots, 8$, we construct a set of logical forms of size n (with exactly n predicates) by combining logical forms of smaller sizes according to the grammar rules in Table 1. For each n, we keep the 100 logical forms z with the highest score $\theta^\top \phi(x, z)$ according to the current model θ. Let Z be the set of logical forms on the final beam, which contains logical forms of all sizes n. During training, due to pruning at

Rule	Semantics	Description
Set	$\texttt{all()}$	all stacks
Color	$\texttt{cyan\|brown\|red\|orange}$	primitive color
Color → Set	$\texttt{with}(c)$	stacks whose top block has color c
Set → Set	$\texttt{not}(s)$	all stacks except those in s
Set → Set	$\texttt{leftmost\|rightmost}(s)$	leftmost/rightmost stack in s
Set Color → Act	$\texttt{add}(s, c)$	add block with color c on each stack in s
Set → Act	$\texttt{remove}(s)$	remove the topmost block of each stack in s

Table 1: The formal grammar defining the compositional action space $\mathcal{Z}$ for SHRDLURN. We use c to denote a Color, and s to denote a Set. For example, one action that we have in SHRDLURN is: *'add an orange block to all but the leftmost brown block'* ↦ `add(not(leftmost(with(brown))),orange)`.

intermediate sizes, Z is not guaranteed to contain the logical form that obtains the observed state y. To mitigate this effect, we use a curriculum so that only simple actions are needed in the initial levels, giving the human an opportunity to teach the computer about basic terms such as colors first before moving to larger composite actions.

The system executes all of the logical forms on the final beam Z, and orders the resulting denotations y by the maximum probability of any logical form that produced it.[3]

Learning. When the human provides feedback in the form of a particular y, the system forms the following loss function:

$$\ell(\theta, x, y) = -\log p_\theta(y \mid x, s) + \lambda||\theta||_1, \tag{2}$$

$$p_\theta(y \mid x, s) = \sum_{z:[\![z]\!]_s = y} p_\theta(z \mid x). \tag{3}$$

Then it makes a single gradient update using AdaGrad (Duchi et al., 2010), which maintains a per-feature step size.

4 Modeling pragmatics

In our initial experience with the semantic parsing model described in Section 3, we found that it was able to learn reasonably well, but lacked a reasoning ability that one finds in human learners. To illustrate the point, consider the beginning of a game when $\theta = 0$ in the log-linear model $p_\theta(z \mid x)$. Suppose that human utters *'remove red'* and then identifies $z_{\texttt{rm-red}} = \texttt{remove(with(red))}$ as the correct logical form. The computer then performs a gradient update on the loss function (2), upweighting features such as (*'remove'*, $\texttt{remove}$) and (*'remove'*, $\texttt{red}$).

Next, suppose the human utters *'remove cyan'*. Note that $z_{\texttt{rm-red}}$ will score higher than all other formulas since the (*'remove'*, $\texttt{red}$) feature will fire again. While statistically justified, this behavior fails to meet our intuitive expectations for a smart language learner. Moreover, this behavior is not specific to our model, but applies to any statistical model that simply tries to fit the data without additional prior knowledge about the specific language. While we would not expect the computer to magically guess *'remove cyan'* ↦ $\texttt{remove(with(cyan))}$, it should at least push down the probability of $z_{\texttt{rm-red}}$ because $z_{\texttt{rm-red}}$ intuitively is already well-explained by another utterance *'remove red'*.

This phenomenon, *mutual exclusivity*, was studied by Markman and Wachtel (1988). They found that children, during their language acquisition process, reject a second label for an object and treat it instead as a label for a novel object.

The pragmatic computer. To model mutual exclusivity formally, we turn to probabilistic models of pragmatics (Golland et al., 2010; Frank and Goodman, 2012; Smith et al., 2013; Goodman and Lassiter, 2015), which operationalize the ideas of Grice (1975). The central idea in these models is to treat language as a cooperative game between a speaker (human) and a listener (computer) as we are doing, but where the listener has an explicit model of the speaker's strategy, which in turn models the listener. Formally, let $S(x \mid z)$ be the speaker's strategy and $L(z \mid x)$ be the listener's

[3] We tried ordering based on the sum of the probabilities (which corresponds to marginalizing out the logical form), but this had the degenerate effect of assigning too much probability mass to y being the set of empty stacks, which can result from many actions.

	$z_{\texttt{rm-red}}$	$z_{\texttt{rm-cyan}}$	$z_3, z_4, \ldots$
		$p_\theta(z \mid x)$	
'remove red'	0.8	0.1	0.1
'remove cyan'	**0.6**	0.2	0.2
		$S(x \mid z)$	
'remove red'	0.57	0.33	0.33
'remove cyan'	0.43	0.67	0.67
		$L(z \mid x)$	
'remove red'	0.46	0.27	0.27
'remove cyan'	0.24	**0.38**	**0.38**

Table 2: Suppose the computer saw one example of *'remove red'* $\mapsto z_{\texttt{rm-red}}$, and then the human utters *'remove cyan'*. **top**: the literal listener, $p_\theta(z \mid x)$, mistakingly chooses $z_{\texttt{rm-red}}$ over $z_{\texttt{rm-cyan}}$. **middle**: the pragmatic speaker, $S(x \mid z)$, assigns a higher probability to to *'remove cyan'* given $z_{\texttt{rm-cyan}}$; **bottom**: the pragmatic listener, $L(z \mid x)$ correctly assigns a lower probability to $z_{\texttt{rm-red}}$ where $p(z)$ is uniform.

strategy. The speaker takes into account the literal semantic parsing model $p_\theta(z \mid x)$ as well as a prior over utterances $p(x)$, while the listener considers the speaker $S(x \mid z)$ and a prior $p(z)$:

$$S(x \mid z) \propto \left(p_\theta(z \mid x)p(x)\right)^\beta, \tag{4}$$

$$L(z \mid x) \propto S(x \mid z)p(z), \tag{5}$$

where $\beta \geq 1$ is a hyperparameter that sharpens the distribution (Smith et al., 2013). The computer would then use $L(z \mid x)$ to rank candidates rather than p_θ. Note that our pragmatic model only affects the ranking of actions returned to the human and does not affect the gradient updates of the model p_θ.

Let us walk through a simple example to see the effect of modeling pragmatics. Table 2 shows that the literal listener $p_\theta(z \mid x)$ assigns high probability to $z_{\texttt{rm-red}}$ for both *'remove red'* and *'remove cyan'*. Assuming a uniform $p(x)$ and $\beta = 1$, the pragmatic speaker $S(x \mid z)$ corresponds to normalizing each column of p_θ. Note that if the pragmatic speaker wanted to convey $z_{\texttt{rm-cyan}}$, there is a decent chance that they would favor *'remove cyan'*. Next, assuming a uniform $p(z)$, the pragmatic listener $L(z \mid x)$ corresponds to normalizing each row of $S(x \mid z)$. The result is that conditioned on *'remove cyan'*, $z_{\texttt{rm-cyan}}$ is now more likely than $z_{\texttt{rm-red}}$, which is the desired effect.

The pragmatic listener models the speaker as a cooperative agent who behaves in a way to maximize communicative success. Certain speaker behaviors such as avoiding synonyms (e.g., not *'delete cardinal'*) and using a consistent word ordering (e.g, not *'red remove'*) fall out of the game theory.[4] For speakers that do not follow this strategy, our pragmatic model is incorrect, but as we get more data through game play, the literal listener $p_\theta(z \mid x)$ will sharpen, so that the literal listener and the pragmatic listener will coincide in the limit.

```
$\forall z, C(z) \leftarrow 0$
$\forall z, Q(z) \leftarrow \epsilon$
repeat
    receive utterance $x$ from human
    $L(z \mid x) \propto \frac{P(z)}{Q(z)} p_\theta(z \mid x)^\beta$
    send human a list $Y$ ranked by $L(z \mid x)$
    receive $y \in Y$ from human
    $\theta \leftarrow \theta - \eta \nabla_\theta \ell(\theta, x, y)$
    $Q(z) \leftarrow Q(z) + p_\theta(z \mid x)^\beta$
    $C(z) \leftarrow C(z) + p_\theta(z \mid x, [\![z]\!]_s = y)$
    $P(z) \leftarrow \frac{C(z)+\alpha}{\sum_{z' : C(z') > 0} \left(C(z')+\alpha\right)}$
until game ends
```

Algorithm 1: Online learning algorithm that updates the parameters of the semantic parser θ as well as counts C, Q required to perform pragmatic reasoning.

Online learning with pragmatics. To implement the pragmatic listener as defined in (5), we need to compute the speaker's normalization constant $\sum_x p_\theta(z \mid x)p(x)$ in order to compute $S(x \mid z)$ in (4). This requires parsing all utterances x based on $p_\theta(z \mid x)$. To avoid this heavy computation in an online setting, we propose Algorithm 1, where some approximations are used for the sake of efficiency. First, to approximate the intractable sum over all utterances x, we only use the examples that are seen to compute the normalization constant $\sum_x p_\theta(z \mid x)p(x) \approx \sum_i p_\theta(z \mid x_i)$. Then, in order to avoid parsing all previous examples again using the current parameters for each new example, we store $Q(z) = \sum_i p_{\theta_i}(z \mid x_i)^\beta$, where θ_i is the parameter after the model updates on the i^{th} example x_i. While θ_i is different from the current parameter θ, $p_\theta(z \mid x_i) \approx p_{\theta_i}(z \mid x_i)$ for the relevant example x_i, which is accounted for

[4] Of course, synonyms and variable word order occur in real language. We would need a more complex game compared to SHRDLURN to capture this effect.

by both θ_i and θ.

In Algorithm 1, the pragmatic listener $L(z \mid x)$ can be interpreted as an importance-weighted version of the sharpened literal listener p_θ^β, where it is downweighted by $Q(z)$, which reflects which z's the literal listener prefers, and upweighted by $P(z)$, which is just a smoothed estimate of the actual distribution over logical forms $p(z)$. By construction, Algorithm 1 is the same as (4) except that it uses the normalization constant Q based on stale parameters θ_i after seeing example, and it uses samples to compute the sum over x. Following (5), we also need $p(z)$, which is estimated by $P(z)$ using add-α smoothing on the counts $C(z)$. Note that $Q(z)$ and $C(z)$ are updated *after* the model parameters are updated for the current example.

Lastly, there is a small complication due to only observing the denotation y and not the logical form z. We simply give each consistent logical form $\{z \mid [\![z]\!]_s = y\}$ a pseudocount based on the model: $C(z) \leftarrow C(z) + p_\theta(z \mid x, [\![z]\!]_s = y)$ where $p_\theta(z \mid x, [\![z]\!]_s = y) \propto \exp(\theta^\mathsf{T}\phi(x,z))$ for $[\![z]\!]_s = y$ (0 otherwise).

Compared to prior work where the setting is specifically designed to require pragmatic inference, pragmatics arises naturally in ILLG. We think that this form of pragmatics is the most important during learning, and becomes less important if we had more data. Indeed, if we have a lot of data and a small number of possible zs, then $L(z|x) \approx p_\theta(z|x)$ as $\sum_x p_\theta(z|x)p(x) \rightarrow p(z)$ when $\beta = 1$.[5] However, for semantic parsing, we would not be in this regime even if we have a large amount of training data. In particular, we are nowhere near that regime in SHRDLURN, and most of our utterances / logical forms are seen only once, and the importance of modeling pragmatics remains.

5 Experiments

5.1 Setting

Data. Using Amazon Mechanical Turk (AMT), we paid 100 workers 3 dollars each to play SHRDLURN. In total, we have 10223 utterances along with their starting states s. Of these, 8874 utterances are labeled with their denotations y; the rest are unlabeled, since the player can try any utterance without accepting an action. 100 players completed the entire game under identical settings. We deliberately chose to start from scratch for every worker, so that we can study the diversity of strategies that different people used in a controlled setting.

Each game consists of 50 blocks tasks divided into 5 levels of 10 tasks each, in increasing complexity. Each level aims to reach an end goal given a start state. Each game took on average 89 utterances to complete.[6] It only took 6 hours to complete these 100 games on AMT and each game took around an hour on average according to AMT's *work time* tracker (which does not account for multi-tasking players). The players were provided minimal instructions on the game controls. Importantly, we gave no example utterances in order to avoid biasing their language use. Around 20 players were confused and told us that the instructions were not clear and gave us mostly spam utterances. Fortunately, most players understood the setting and some even enjoyed SHRDLURN as reflected by their optional comments:

- *That was probably the most fun thing I have ever done on mTurk.*
- *Wow this was one mind bending games [sic].*

Metrics. We use the *number of scrolls* as a measure of game performance for each player. For each example, the number of scrolls is the position in the list Y of the action selected by the player. It was possible to complete this version of SHRDLURN by scrolling (all actions can be found in the first 125 of Y)—22 of the 100 players failed to teach an actual language, and instead finished the game mostly by scrolling. Let us call them *spam players*, who usually typed single letters, random words, digits, or random phrases (e.g. *'how are you'*). Overall, spam players had to scroll a lot: 21.6 scrolls per utterance versus only 7.4 for the non-spam players.

5.2 Human strategies

Some example utterances can be found in Table 3. Most of the players used English, but vary in their adherence to conventions such as use of determiners, plurals, and proper word ordering. 5 players invented their own language, which are more precise, more consistent than general English. One player used Polish, and another used Polish notation (bottom of Table 3).

[5] Technically, we also need p_θ to be *well-specified.*

[6] This number is not 50 because some block tasks need multiple steps and players are also allowed to explore without reaching the goal.

Most successful players (1st–20th)		
rem cy pos 1, stack or blk pos 4, rem blk pos 2 thru 5, rem blk pos 2 thru 4, stack bn blk pos 1 thru 2, fill bn blk, stack or blk pos 2 thru 6, rem cy blk pos 2 fill rd blk **(3.01)**	remove the brown block, remove all orange blocks, put brown block on orange blocks, put orange blocks on all blocks, put blue block on leftmost blue block in top row **(2.78)**	Remove the center block, Remove the red block, Remove all red blocks, Remove the first orange block, Put a brown block on the first brown block, Add blue block on first blue block **(2.72)**
Average players (21th–50th)		
reinsert pink, take brown, put in pink, remove two pink from second layer, Add two red to second layer in odd intervals, Add five pink to second layer, Remove one blue and one brown from bottom layer **(9.17)**	remove red, remove 1 red, remove 2 4 orange, add 2 red, add 1 2 3 4 blue, emove 1 3 5 orange, add 2 4 orange, add 2 orange, remove 2 3 brown, add 1 2 3 4 5 red, remove 2 3 4 5 6, remove 2, add 1 2 3 4 6 red **(8.37)**	move second cube, double red with blue, double first red with red, triple second and fourth with orange, add red, remove orange on row two, add blue to column two, add brown on first and third **(7.18)**
Least successful players (51th–)		
holdleftmost, holdbrown, holdleftmost, blueonblue, brownonblue1, blueonorange, holdblue, holdorange2, blueonred2 , holdends1, holdrightend, hold2, orangeonorangerightmost **(14.15)**	'add red cubes on center left, center right, far left and far right', 'remove blue blocks on row two column two, row two column four', remove red blocks in center left and center right on second row **(12.6)**	laugh with me, red blocks with one aqua, aqua red alternate, brown red red orange aqua orange, red brown red brown red brown, space red orange red, second level red space red space red space **(14.32)**
Spam players (∼ 85th–100)		
next, hello happy, how are you, move, gold, build goal blocks, 23,house, gabboli, x, run,,xav, d, j, xcv, dulicate goal (21.7)		
Most interesting		
usuń brązowe klocki, postaw pomarańczowy klocek na pierwszym klocku, postaw czerwone klocki na pomarańczowych, usuń pomarańczowe klocki w górnym rzędzie	rm scat + 1 c, + 1 c, rm sh, + 1 2 4 sh, + 1 c, - 4 o, rm 1 r, + 1 3 o, full fill c, rm o, full fill sh, - 1 3, full fill sh, rm sh, rm r, + 2 3 r, rm o, + 3 sh, + 2 3 sh, rm b, - 1 o, + 2 c,	mBROWN,mBLUE,mORANGE RED+ORANGEˆORANGE, BROWN+BROWNm1+BROWNm3, ORANGE +BROWN +ORANGEˆm1+ ORANGEˆm3 + BROWNˆˆ2 + BROWNˆˆ4

Table 3: Example utterances, along with the average number of scrolls for that player in parentheses. Success is measured by the number of scrolls, where the more successful players need less scrolls. 1) The 20 most successful players tend to use consistent and concise language whose semantics is similar to our logical language. 2) Average players tend to be slightly more verbose and inconsistent (left and right), or significantly different from our logical langauge (middle). 3) Reasons for being unsuccessful vary. Left: no tokenization, middle: used a coordinate system and many conjunctions; right: confused in the beginning, and used a language very different from our logical language.

Overall, we find that many players adapt in ILLG by becoming more consistent, less verbose, and more precise, even if they used standard English at the beginning. For example, some players became more consistent over time (e.g. from using both *'remove'* and *'discard'* to only using *'remove'*). In terms of verbosity, removing function words like determiners as the game progresses is a common adaptation. In each of the following examples from different players, we compare an utterance that appeared early in the game to a similar utterance that appeared later: *'Remove the red ones'* **became** *'Remove red.'*; *'add brown on top of red'* **became** *'add orange on red'*; *'add red blocks to all red blocks'* **became** *'add red to red'*; *'dark red'* **became** *'red'*; one player used *'the'* in all of the first 20 utterances, and then never used *'the'* in the last 75 utterances.

Players also vary in precision, ranging from overspecified (e.g. *'remove the orange cube at the left'*, *'remove red blocks from top row'*) to underspecified or requiring context (e.g. *'change colors'*, *'add one blue'*, *'Build more blocus'*, *'Move the blocks fool'*,*'Add two red cubes'*). We found that some players became more precise over time, as they gain a better understanding of ILLG.

Most players use utterances that actually do not match our logical language in Table 1, even the successful players. In particular, numbers are often used. While some concepts always have the same effect in our blocks world (e.g. *'first block'* means `leftmost`), most are different. More concretely, of the top 10 players, 7 used numbers of some form and only 3 players matched our logical language. Some players who did not match the logical language performed quite well nevertheless. One possible explanation is because the action required is somewhat constrained by the logical language and some tokens can have unintended interpretations. For example, the computer can correctly interpret numerical positional references, as long as the player only refers to the leftmost and rightmost positions. So if the player says *'rem blk pos 4'* and *'rem blk pos 1'*, the computer can interpret *'pos'* as `rightmost` and interpret the bigram (*'pos'*, *'1'*) as `leftmost`. On the other hand, players who deviated significantly by describing the desired state declaratively (e.g. *'red orange red'*, *'246'*) rather than using actions, or a coordinate system (e.g. *'row two column two'*) performed poorly. Although players do not have to match our logical language exactly to perform well, being similar is definitely helpful.

Compositionality. As far as we can tell, all players used a compositional language; no one invented unrelated words for each action. Interestingly, 3 players did not put spaces between words. Since we assume monomorphemic words separated by spaces, they had to do a lot of scrolling as a result (e.g., 14.15 with utterances like *'orangeonorangerightmost'*).

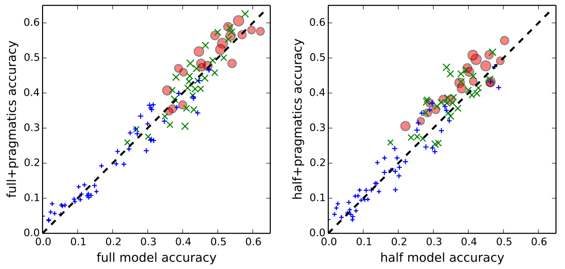
5.3 Computer strategies

We now present quantitative results on how quickly the computer can learn, where our goal is to achieve high accuracy on new utterances as we make just a single pass over the data. The number of scrolls used to evaluate player is sensitive to outliers and not as intuitive as accuracy. Instead, we consider *online accuracy*, described as follows. Formally, if a player produced T utterances $x^{(j)}$ and labeled them $y^{(j)}$, then

$$\text{online accuracy} \overset{\text{def}}{=} \frac{1}{T} \sum_{j=1}^{T} \mathbb{I}\left[y^{(j)} = [\![z^{(j)}]\!]_{s^{(j)}}\right],$$

where $z^{(j)} = \arg\max_z p_{\theta^{(j-1)}}(z|x^{(j)})$ is the model prediction based on the previous parameter $\theta^{(j-1)}$. Note that the online accuracy is defined with respect to the player-reported labels, which only corresponds to the actual accuracy if the player is precise and honest. This is not true for most spam players.

Compositionality. To study the importance of compositionality, we consider two baselines. First, consider a non-compositional model (*memorize*) that just remembers pairs of complete utterance and logical forms. We implement this using indicator features on features (x, z), e.g., (*'remove all the red blocks'*, $z_{\texttt{rm-red}}$), and use a large learning rate. Second, we consider a model (*half*) that treats utterances compositionally with unigrams, bigrams, and skip-trigrams features, but the logical forms are regarded as non-compositional, so we have features such as (*'remove'*, $z_{\texttt{rm-red}}$), (*'red'*, $z_{\texttt{rm-red}}$), etc.



(a) (b)

Figure 2: Pragmatics improve online accuracy. In these plots, each marker is a player. red o: players who ranked 1–20 in terms of minimizing number of scrolls, green x: players 20–50; blue +: lower than 50 (includes spam players). Marker sizes correspond to player rank, where better players are depicted with larger markers. **2a**: online accuracies with and without pragmatics on the full model; **2b**: same for the half model.

	players ranked by # of scrolls			
Method	top 10	top 20	top 50	all 100
memorize	25.4	24.5	22.5	17.6
half model	38.7	38.4	36.0	27.0
half + prag	43.7	42.7	39.7	29.4
full model	48.6	47.8	44.9	33.3
full + prag	52.8	49.8	45.8	33.8

Table 4: Average online accuracy under various settings. memorize: featurize entire utterance and logical form non-compositionally; half model: featurize the utterances with unigrams, bigrams, and skip-grams but conjoin with the entire logical form; full model: the model described in Section 3; +prag: the models above, with our online pragmatics algorithm described in Section 4. Both compositionality and pragmatics improve accuracy.

Table 4 shows that the full model (Section 3) significantly outperforms both the *memorize* and *half* baselines. The learning rate $\eta = 0.1$ is selected via cross validation, and we used $\alpha = 1$ and $\beta = 3$ following Smith et al. (2013).

Pragmatics. Next, we study the effect of pragmatics on online accuracy. Figure 2 shows that modeling pragmatics helps successful players (e.g., top 10 by number of scrolls) who use precise and consistent languages. Interestingly, our pragmatics model did not help and can even hurt the less successful players who are less precise and consistent. This is expected behavior: the pragmatics model assumes that the human is cooperative and behaving rationally. For the bottom half of the players, this assumption is not true, in which case the pragmatics model is not useful.

6 Related Work and Discussion

Our work connects with a broad body of work on grounded language, in which language is used in some environment as a means towards some goal. Examples include playing games (Branavan et al., 2009, 2010; Reckman et al., 2010) interacting with robotics (Tellex et al., 2011, 2014), and following instructions (Vogel and Jurafsky, 2010; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013) Semantic parsing utterances to logical forms, which we leverage, plays an important role in these settings (Kollar et al., 2010; Matuszek et al., 2012; Artzi and Zettlemoyer, 2013).

What makes this work unique is our new interactive learning of language games (ILLG) setting, in which a model has to learn a language from *scratch* through interaction. While online gradient descent is frequently used, for example in semantic parsing (Zettlemoyer and Collins, 2007; Chen, 2012), we using it in a truly online setting, taking one pass over the data and measuring online accuracy (Cesa-Bianchi and Lugosi, 2006).

To speed up learning, we leverage computational models of pragmatics (Jäger, 2008; Golland et al., 2010; Frank and Goodman, 2012; Smith et al., 2013; Vogel et al., 2013). The main difference is these previous works use pragmatics with a trained base model, whereas we learn the model online. Monroe and Potts (2015) uses learning to improve the pragmatics model. In contrast, we use pragmatics to speed up the learning process by capturing phenomena like mutual exclusivity (Markman and Wachtel, 1988). We also differ from prior work in several details. First, we model pragmatics in the online learning setting where we use an online update for the pragmatics model. Second, unlikely the reference games where pragmatic effects plays an important role by design, SHRDLURN is not specifically designed to require pragmatics. The improvement we get is mainly due to players trying to be consistent in their language use. Finaly, we treat both the utterance and the logical forms as featurized compositional objects. Smith et al. (2013) treats utterances (i.e. words) and logical forms (i.e. objects) as categories; Monroe and Potts (2015) used features, but also over flat categories.

Looking forward, we believe that the ILLG setting is worth studying and has important implications for natural language interfaces. Today, these systems are trained once and deployed. If these systems could quickly adapt to user feedback in real-time as in this work, then we might be able to more readily create systems for resource-poor languages and new domains, that are customizable and improve through use.

Acknowledgments

DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF-15-1-0462. The first author is supported by a NSERC PGS-D fellowship. In addition, we thank Will Monroe, and Chris Potts for their insightful comments and discussions on pragmatics.

Reproducibility

All code, data, and experiments for this paper are available on the CodaLab platform:
`https://worksheets.codalab.org/worksheets/0x9fe4d080bac944e9a6bd58478cb05e5e`
The client side code is here:
`https://github.com/sidaw/shrdlurn/tree/acl16-demo`
and a demo: `http://shrdlurn.sidaw.xyz`

References

Y. Artzi and L. Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)* 1:49–62.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 82–90.

S. Branavan, L. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*. pages 1268–1277.

N. Cesa-Bianchi and G. Lugosi. 2006. *Prediction, learning, and games*. Cambridge University Press.

D. L. Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Association for Computational Linguistics (ACL)*.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 859–865.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

M. Frank and N. D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science* 336:998–998.

H. Giles. 2008. *Communication accommodation theory*. Sage Publications, Inc.

D. Golland, P. Liang, and D. Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Empirical Methods in Natural Language Processing (EMNLP)*.

N. Goodman and D. Lassiter. 2015. *Probabilistic Semantics and Pragmatics: Uncertainty in Language and Thought*. The Handbook of Contemporary Semantic Theory, 2nd Edition Wiley-Blackwell.

H. P. Grice. 1975. Logic and conversation. *Syntax and semantics* 3:41–58.

M. E. Ireland, R. B. Slatcher, P. W. Eastwick, L. E. Scissors, E. J. Finkel, and J. W. Pennebaker. 2011. Language style matching predicts relationship initiation and stability. *Psychological Science* 22(1):39–44.

G. Jäger. 2008. Game theory in semantics and pragmatics. Technical report, University of Tübingen.

T. Kollar, S. Tellex, D. Roy, and N. Roy. 2010. Grounding verbs of motion in natural language commands to robots. In *International Symposium on Experimental Robotics (ISER)*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1223–1233.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*. pages 590–599.

E. Markman and G. F. Wachtel. 1988. Children's use of mutual exclusivity to constrain the meanings of words. *Cognitive Psychology* 20:125–157.

C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*. pages 1671–1678.

W. Monroe and C. Potts. 2015. Learning in the Rational Speech Acts model. In *Proceedings of 20th Amsterdam Colloquium*.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

H. Reckman, J. Orkin, and D. Roy. 2010. Learning meanings of words and constructions, grounded in a virtual game. In *Conference on Natural Language Processing (KONVENS)*.

N. J. Smith, N. D. Goodman, and M. C. Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in Neural Information Processing Systems (NIPS)*.

S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. 2014. Asking for help using inverse semantics. In *Robotics: Science and Systems (RSS)*.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

A. Vogel, M. Bodoia, C. Potts, and D. Jurafsky. 2013. Emergence of gricean maxims from multi-agent decision theory. In *North American Association for Computational Linguistics (NAACL)*. pages 1072–1081.

A. Vogel and D. Jurafsky. 2010. Learning to follow navigational directions. In *Association for Computational Linguistics (ACL)*. pages 806–814.

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.

L. Wittgenstein. 1953. *Philosophical Investigations*. Blackwell, Oxford.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*. pages 960–967.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*. pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. pages 678–687.