Learning from Natural Instructions

Dan Goldwasser Dan Roth
Department of Computer Science
University of Illinois at Urbana-Champaign
$\{goldwas1, danr\}$@illinois.edu

Abstract

Machine learning is traditionally formalized and researched as the study of learning concepts and decision functions from labeled examples, requiring a representation that encodes information about the domain of the decision function to be learned. We are interested in providing a way for a human teacher to interact with an automated learner using *natural instructions*, thus allowing the teacher to communicate the relevant domain expertise to the learner without necessarily knowing anything about the internal representations used in the learning process.

In this paper we suggest to view the process of learning a decision function as a natural language *lesson interpretation problem* instead of learning from labeled examples. This interpretation of machine learning is motivated by human learning processes, in which the learner is given a lesson describing the target concept directly, and a few instances exemplifying it. We introduce a learning algorithm for the *lesson interpretation problem* that gets feedback from its performance on the final task, while learning jointly (1) how to interpret the lesson and (2) how to use this interpretation to do well on the final task. This approach alleviates the supervision burden of traditional machine learning by focusing on supplying the learner with only human-level task expertise for learning.

We evaluate our approach by applying it to the rules of the Freecell solitaire card game. We show that our learning approach can eventually use natural language instructions to learn the target concept and play the game legally. Furthermore, we show that the learned semantic interpreter also generalizes to previously unseen instructions.

1 Introduction

Machine learning has traditionally focused on learning concepts from labeled examples: given a set of labeled examples the learner constructs a decision function generalizing over the observed training data. While this approach has been tremendously successful for many learning domains, it carries an inherent drawback - the learner can only be as good as the data it is given. Learning therefore depends on annotating considerable amounts of training data, an expensive and time consuming process. Furthermore, successful learning often depends on machine learning expertise required for designing and calibrating the learning environment.

In this work we aim to alleviate some of this difficulty by suggesting a different kind of learning protocol - a *lesson based* learning protocol, instead of example based learning. This protocol draws motivation from human learning processes, in which the learner is given a "knowledge injection", in the form of a lesson describing a high level concept and a few specific examples to validate the correct understanding of the lesson. Example-based learning, an inductive process by nature, generalizes over the labeled examples. This contrasts directly with our approach, which attempts to learn the correct hypothesis directly. The proposed approach carries with it an immediate advantage, as it allows the system designer to focus on task-related expertise. To ensure this advantage we focus on natural instructions, describing the target concept in natural language (NL).

While the promise of this approach is clear, successful learning in these settings depends on correctly communicating relevant knowledge to the learning system. Unfortunately this proves to be a non trivial task: allowing human users to communicate effectively with computer systems in a natural manner is one of the longest standing goals of artificial intelligence. This problem is typically framed as a NL interpretation task, mapping between NL input and a formal meaning interpretation, expressed in a logical language understandable by the target computer system. Current works approach this task using supervised machine learning methods, which are difficult and expensive as the original learning problem we hoped to avoid.

Our learning framework avoids this difficulty by making the connection between the learning tasks explicit. In our settings the learner has the ability to test its understanding, while this is clearly insufficient for constructing the target hypothesis it allows the learner to reject incorrect lesson interpretations. We exploit this property to provide feedback to the semantic interpretation learning process, and base the learning algorithm on this feedback.

We test our approach in an actionable settings, in which the

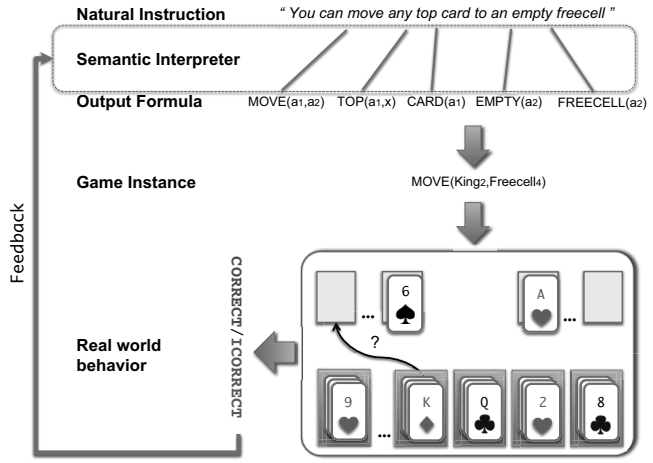


Figure 1: Language interpretation setup - from textual input to real world behavior.

learner can evaluate its hypothesis by taking actions in a (simulated) world environment. Our experiments were conducted over the Freecell solitaire card game, in which the lessons describe preconditions on actions. We show that using our protocol an automated system can be taught to play games using NL instructions.

2 Learning From Natural Instructions

In this section we give a bird's eye view of our framework and its components. The purpose of our framework is to learn a classification function capturing an observable desired behavior of the learning system. However unlike traditional machine learning algorithms, our framework does not learn from annotated examples, but rather from natural instructions describing the target concept.

The learning process in which the system improves pertains only to its ability to better understand instructions. The only supervision signal available to the learner is the system's behavior given the instruction interpretation, following the intuition that correct behavior corresponds to a correct interpretation of the input instructions. Sec. 3 describes the learning process in detail.

In our experiments we used the Freecell solitaire card game, taking as input instructions describing the legality of game actions and used the instructions' interpretation to predict which moves are legal given specific game states. Fig. 1 describes an example of this process.

2.1 Semantic Interpretation

We formulate semantic interpretation as a structured prediction problem, mapping a NL input (denoted $\mathbf{x}$), to its highest ranking logical interpretation (denoted $\mathbf{z}$). In order to correctly parametrize and weight the possible outputs, the decision relies on an intermediate representation: an alignment between textual fragments and their meaning representation (denoted $\mathbf{y}$). In our experiments the input sentences $\mathbf{x}$ are natural language Freecell Solitaire game instructions, describing actions legality. The output formula representation $\mathbf{z}$ is described using a formal language, we provide further details about it in Sec. 2.2.

The prediction function, mapping a sentence to its corresponding interpretation, is formalized as follows:

$$\hat{\mathbf{z}} = F_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y}\in\mathcal{Y},\mathbf{z}\in\mathcal{Z}} \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y},\mathbf{z}) \quad (1)$$

Where Φ is a feature function defined over an input sentence $\mathbf{x}$, alignment $\mathbf{y}$ and output $\mathbf{z}$. The weight vector $\mathbf{w}$ contains the model's parameters, whose values are determined by the learning process.

We refer to the $\arg\max$ above as the inference problem. Given an input sentence, solving this inference problem based on Φ and $\mathbf{w}$ is what compromises our interpretation process. Sec. 4 provides more details about the feature representation and inference procedure used.

2.2 Target Representation

The output of semantic interpretation is a logical formula, grounding the semantics of the input sentence in the Freecell domain. We use a subset of first order logic consisting of typed constants (corresponding to specific cards, values, etc.) and functions, which capture relations between domains entities, and properties of entities (e.g., $\texttt{value} : E \rightarrow N$). Our formulation of the Freecell domain is a slightly modified version of the Freecell domain defined in the Planning Domain Definition Language (PDDL), which is used for evaluating automated planning systems. Throughout the paper we denote the set of logical symbols in the freecell domain as $\mathbf{D}$.

The domain consists of 87 constants and 11 predicates. A **game state** contains specific instantiations of the domain symbols, describing the relations between the entities in the state. Given a game state, a logical formula defined over the domain symbols can be evaluated. For example, with regard to the game state described in Fig. 1, the formula $\texttt{freecell}(x_1)\ \texttt{top}(x_2,x_1)\ \texttt{value}(x_2,6)$, stating that a card with a value of 6 is on top of at least one freecell, will be evaluated to a `true`.

As can be observed in this example, dependency between the values of logical predicates is expressed via argument sharing. We use this mechanism to construct meaningful logical formulas from text, that can be evaluated given game states. Since our goal is to predict the legality of game actions (i.e., horn rules), we refer to the predicate corresponding to an action, `move`, as the **head** predicate and as a convention we denote its arguments as a_1, a_2. We define $\texttt{args}(\texttt{p})$, to be a function mapping a predicate p to its list of argument variables, and denote by p^i the i-th argument in this list.

2.3 Feedback from Real World Behavior

Observing the behavior resulting from instruction interpretation comprises the only feedback mechanism available to our learner. We envision a human learning process, in which in addition to high level instructions the learner receives a small number of examples used for clarification. For each concept taught, the learner had access to 10 positive and 10 negative

labeled examples chosen randomly[1].

Throughout the paper we abstract over the implementation details, and refer to the feedback mechanism as a binary function $Feedback : \mathcal{Z} \rightarrow \{+1, -1\}$, informing the learner whether a predicted logical form $\mathbf{z}$ when executed on the actual game states domain produces the desired outcome.

3 Updating the Semantic Interpreter

In this paper we advocate the idea of instructable computing, in which an automated learning system is taught a concept via direct instructions rather than by examples. The role of traditional machine learning is therefore shifted from learning the target concept to learning to interpret natural instructions, explaining the target concept.

The learning problem is defined as finding a good set of parameters for the inference function described in Eq. 1, such that when applied to NL instructions the corresponding output formula will result in a correct behavior. Typically, such prediction functions are trained in a supervised settings in which the learner has access to training examples, consisting of the input sentences and their corresponding logical forms $\{(\mathbf{x}^l, \mathbf{z}^l)\}_{l=1}^N$ (e.g., [Zettlemoyer and Collins, 2005; Wong and Mooney, 2007]). However our learning framework does not have access to this type of data, and relies only on feedback obtained from world interaction - by trying out the interpretation. This setup gives rise to our algorithmic learning approach, iteratively performing the following steps- generating rules from instructions, receiving feedback by acting in the world and updating the interpretation function parameters accordingly. We explain the technical aspects of this protocol in the following subsection.

3.1 Learning Structures from Binary Feedback

From a machine learning perspective the algorithm learns a structured predictor, $F_{\mathbf{w}}(\mathbf{x})$, using binary feedback. In general, the goal of structured learning is to learn, given "gold" structures, the parameters to a ranking function, such that the correct structure will be ranked first. Since the only indication of correctness our framework has is game data prediction accuracy, we use this feedback to approximate the structured learning goal, by promoting output structures that result in the desired world behavior so that their score will increase. Our algorithm samples the space of possible structures (alignments and logical forms ($\mathcal{Y} \times \mathcal{Z}$)) for a given input $\mathbf{x}$, modifying the model's parameter vector ($\mathbf{w}$) to encourage correct structures.

Using this intuition we can cast the problem of learning a weight vector for Equation (1) as a binary classification problem where we directly consider structures the feedback assigns $+1$ as positive examples and those assigned -1 as negative. We represent the input to the binary classifier as a feature vector $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ normalized by the size of the input sentence. [2]

[1] Since our goal is to learn the target concept from instructions rather than from examples, we designed a weak feedback mechanism which cannot be used to learn the target concept directly

[2] Normalization is required to ensure that each sentence contributes equally to the binary learning problem regardless of the sentence's length.

Algorithm 1 Learning from Natural Instructions

Input: Sentences $\{\mathbf{x}^l\}_{l=1}^N$,
$Feedback : \mathcal{Z} \rightarrow \{+1, 1\}$,
initial weight vector $\mathbf{w}$

1: $S_l \leftarrow \{\}$ for all $l = 1, \ldots, N$
2: **repeat**
3: **for** $l = 1, \ldots, N$ **do**
4: $\hat{\mathbf{y}}, \hat{\mathbf{z}} = \arg\max_{\mathbf{y},\mathbf{z}} \mathbf{w}^T \Phi(\mathbf{x}^l, \mathbf{y}, \mathbf{z})$
5: $f = Feedback(\hat{\mathbf{z}})$
6: add $(\Phi(\mathbf{x}^l, \hat{\mathbf{y}}, \hat{\mathbf{z}})/|\mathbf{x}^l|, f)$ to S_l
7: **end for**
8: $\mathbf{w} \leftarrow Learn(S)$ where $S = \cup_l S_l$
9: **until** no S_l has new unique examples
10: **return** $\mathbf{w}$

Algorithm 1 outlines the approach in detail. The first stage of the algorithm iterates over all the training input sentences and computes the best logical form $\hat{\mathbf{z}}$ and alignment $\hat{\mathbf{y}}$ by solving the inference problem (line 4). The output structures are evaluated using the feedback function (line 5), and a new training example is created by extracting features from the triple containing the sentence, alignment and logical form and the feedback is used as a label. This training example is added to the working set of training examples for this input sentence (line 6). All the feedback training examples are used to train a binary classifier whose weight vector is used in the next iteration (line 8). The algorithm repeats until no new unique training examples are added to any of the working sets for any input sentence.

In our experiments we used the SVM algorithm with linear kernel and squared hinge loss to train the model.

4 Inference

Semantic interpretation, as formulated in Eq. 1, is an inference procedure selecting the top ranking output logical formula. In practice this decision is decomposed into smaller decisions, capturing local mappings of input tokens to logical fragments and their composition into larger fragments. These decisions are converted into a feature representation by a feature function Φ, and parameterized by a weight vector.

We formulate the inference process over these decision as an Integer Linear Program (ILP), maximizing the overall score of active decisions, subject to constraints ensuring the validity of the output formula. The flexibility of ILP has previously been advantageous in natural language processing tasks [Roth and Yih, 2007; Martins *et al.*, 2009] as it allows us to easily incorporate constraints declaratively. These constraints help facilitate learning as they shape the space of possible output structures, thus requiring the learned model's parameters to discriminate between a smaller set of candidates.

4.1 Decision Variables and Objective Function

The inference decision is defined over two types of decision variables. The first type, referred to as a **first order decision**, encodes a lexical mapping decision as a binary variable α_{cs},

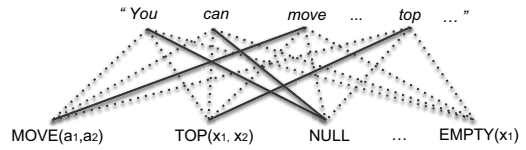
indicating that a constituent c is aligned with a logical symbol `s`. The pairs connected by the alignment ($\mathbf{y}$) in Fig. 2(a) are examples of such decisions.

The final output structure $\mathbf{z}$ is constructed by composing individual predicates into a complete formula, this is formulated as an *argument sharing decision* indicating if two functions take the same variable as input. We refer to this type of decisions as a **second order decision**, encoded as a binary variable, β_{cs^i,dt^j} indicating if the j-th argument of `t` (associated with constituent d) and the i-th argument of `s` (associated with constituent c) refer to the same variable or constant symbol. For example, the decision variables representation of the formula presented in Fig. 2(b): `move`(a_1, a_2) `top`(a_1, x_2) includes an active second order variable indicating that the corresponding predicates share an argument.

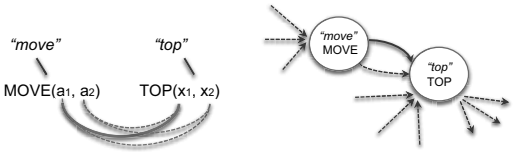
Objective Function given an input sentence, we consider the space of possible semantic interpretations as the space of possible assignments to the decision variables. The semantic interpretation decision is done by selecting a subset of variables maximizing a linear objective function, defined as follows -

$$F_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\alpha,\beta} \sum_{c\in\mathbf{x}} \sum_{s\in D} \alpha_{cs} \cdot {\mathbf{w_1}}^T \Phi_1(\mathbf{x}, c, s)$$
$$+ \sum_{c,d\in\mathbf{x}} \sum_{s,t\in D} \sum_{i,j} \beta_{cs^i,dt^j} \cdot {\mathbf{w_2}}^T \Phi_2(\mathbf{x}, c, s^i, d, t^j) \qquad (2)$$

where i, j iterate over `args(s)` and `args(t)` respectively.



(a) 1^{st}-order decisions



(b) 2^{nd}-order decisions (c) Flow variables

Figure 2: An example of inference variables space for a given input. The dashed edges correspond to non-active decision variables and the bold lines to active variables, corresponding to the output structure `move`(a_1, a_2) `top`(a_1, x_2). Active variables include - 1^{st}-order: $\alpha_{(\text{“}move''\text{,move}^1)}, \alpha_{(\text{“}top''\text{,top}^1)}$, 2^{nd}-order: $\beta_{(\text{“}move''\text{,move}^1),(\text{“}top''\text{,top}^1)}$, and positive flow: $f_{(\text{“}move''\text{,move}^1),(\text{“}top''\text{,top}^1)}$

4.2 Constraints

Given an input sentence, the space of possible interpretations is subsumed by the space of possible assignments to the decisions variables. For example, there is a clear dependency between α-variables and β-variables assignments, as functions can only share a variable (β decision) if they appear in the output formula (α decisions). In order to prevent spurious assignments, we restrict the decision space. We take advantage of the flexible ILP framework, and encode these restrictions as global constraints over Eq. 2.

Lexical Mapping Decisions

- An input constituent can only be associated with at most one logical symbol.
$$\forall c \in \mathbf{x},\ \textstyle\sum_{s\in\mathbf{D}} \alpha_{cs} \leq 1$$
- The head predicate (e.g., `move`) must be active.
$$\textstyle\sum_{c\in\mathbf{x}} \alpha_{c,move} = 1$$

Argument Sharing Decisions

- Variable sharing is only possible when variable types match.
- If two predicates share a variable, then these predicates must be active.
$$\forall c, d \in \mathbf{x}, \forall s, t \in \mathbf{D},\ \beta_{cs^i,dt^j} \implies \alpha_{cs} \wedge \alpha_{dt}$$
where i, j range over `args(s)` and `args(t)` respectively.

Global Structure: Connectivity Constraints In addition to constraints over local decision we are also interested in ensuring a correct global structure of the output formula. We impose constraints forcing an overall *fully-connected* output structure, in which each logical symbol appearing in the output formula is connected to the head predicate via argument sharing. This property ensures that the value of each logical construct in the output is dependent on the head predicate's arguments. In order to clarify this idea, consider the output logical formula described in example 1. The value of that formula, when given a game state, is evaluated to TRUE if the game state contains at least one vacant freecell, *not necessarily the target freecell* specified by the head predicate arguments.

Example 1 (Disconnected Output Structure)
`move`(a_1, a_2)
`top`(a_1, x_1) `card`(a_1) `freecell`(x_2) `empty`(x_2)

We encode the connectivity property by representing the decision space as a graph, and forcing the graph corresponding to the output prediction to have a connected tree structure using flow constraints. Fig. 2(c) provides an example of this formulation.

Let $G = (V, E)$ be a directed graph, where V contains vertices corresponding to α variables and E contains edges corresponding to β variables, each adding two directional edges. We refer to vertices corresponding to $\alpha_{c,\text{move}}$ variables as head vertices. Clearly, the output formula will be fully-connected if and only if the graph corresponding to the output structure is connected. We associate a flow variable f_{cs^i,dt^j} with every edge in the graph, and encode the following constraints over the flow variables to ensure that the resulting graph is connected.

- Only active edges can have a positive flow.

$$\beta_{cs^i,dt^j} = 0 \implies f_{cs^i,dt^j} = 0 \wedge f_{dt^j,cs^i} = 0$$

- The total outgoing flow from all head vertices must be equal to the number of logical symbols appearing in the formula.

$$\sum f_{*,\texttt{move}^i,*,*} = \sum_{c\in\mathbf{x}} \sum_{s\in\mathbf{D}\setminus\{\texttt{move}\}} \alpha_{cs}$$

  For readability reasons, we use * to indicate all possible values for constituents in $\mathbf{x}$ and logical symbols in $\mathbf{D}$.

- Each non-head vertex consumes one unit of flow.

$$\forall d \in \mathbf{x}, \forall t \in \mathbf{D},\ \sum f_{*,*,dt^j} - \sum f_{dt^j,*,*} = 1$$

4.3 Features

The inference problem defined in Eq. (2) uses two feature functions: Φ_1 for first order decision and Φ_2 for second order decisions. In general, Φ_1 represents lexical information while Φ_2 represents syntactic and semantic dependencies between sub-structures.

First-order decision features Φ_1 Determining if a logical symbol is aligned with a specific constituent depends mostly on lexical information. Following previous work (e.g., [Zettlemoyer and Collins, 2005]) we create a small lexicon, mapping logical symbols to surface forms.

We rely on external knowledge [Miller *et al.*, 1990] to extend the initial lexicon, and add features which measure the lexical similarity between a constituent and a logical symbol's surface forms (as defined by the lexicon). In order to disambiguate preposition constituents, an additional feature is added. This feature considers the current lexical context (one word to the left and right) in addition to word similarity.

Second-order decision features Φ_2 Second order decisions rely on syntactic information. We use the dependency tree [Klein and Manning, 2003] of the input sentence. Given a second-order decision $\beta_{cs,dt}$, the dependency feature takes the normalized distance between the head words in the constituents c and d. In addition, a set of features indicate which logical symbols are usually composed together, without considering their alignment to text.

5 Experiments

We applied our learning framework to instructional text, and evaluated the output formulas on Freecell game data. In this section we describe our experimental evaluation, we begin by describing the experimental setup and report the results in the subsequent section.

Experimental Setup The decision function described by the text classifies the legality of Freecell game moves given a game state. Since this rule is dependent on the move action target location we break this decision into three target concepts, each capturing the legality of moving a card to a different location. We denote the three target concepts as FREECELL, HOMECELL and TABLEAU. The following examples describe the FREECELL and TABLEAU target concepts and their textual descriptions.

Example 2 (FREECELL concept and its description)
$\texttt{move}(a_1, a_2)$
$\texttt{top}(a_1, x_1)\ \texttt{card}(a_1)\ \texttt{freecell}(a_2)\ \texttt{empty}(a_2)$

- "You can move any of the top cards to an empty freecell"
- "Any playable card can be moved to a freecell if it is empty"

Example 3 (TABLEAU concept and its description)
$\texttt{move}(a_1, a_2)$
$\texttt{top}(a_1, x_1)\ \texttt{card}(a_1)\ \texttt{tableau}(a_2)\ \texttt{top}(x_2, a_2)$
$\texttt{color}(a_1, x_3)\ \texttt{color}(x_2, x_4)\ \texttt{not-equal}(x_3, x_4)$
$\texttt{value}(a_1, x_5)\ \texttt{value}(x_2, x_6)\ \texttt{successor}(x_5, x_6)$

- "A top card can be moved to a tableau if it has a different color than the color of the top tableau card, and the cards have successive values"

In order to evaluate our framework we associate with each target concept instructional text describing the target rule, and game data over which the predicted structures are evaluated. Each target concept is associated with 25 different instructions describing the target rule, and 900 relevant game moves sampled randomly. To avoid bias the game moves contain an equal number of positive and negative examples. Note that these examples are used for evaluation only.

We evaluated the performance of our learning system by measuring the proportion of correct predictions for each of the target concepts on the game data. The accuracy for each target concept is measured by averaging the accuracy score of each of the individual instruction interpretations.

The semantic interpreter was initialized using a simple rule based procedure, assigning uniform scores to input constituents appearing in the lexicon (first-order decisions) and penalizing second order decisions corresponding to input constituents which are far apart on the dependency tree of the input sentence.

Experimental Approach Our experiments were designed to evaluate the learner's ability to generalize beyond the limited supervision offered by the feedback function. Generalization is evaluated along two lines: (1) Evaluating the quality of the learned target concept: the ability of the system to perform well on unseen solitaire game data (2) Evaluating the quality of the learned semantic interpretation model. In this case after the learning process terminates the system is given a set of new textual instructions, and its performance is evaluated based on the quality of the newly generated rules. To accommodate this set up we performed a 5-fold cross validation over the textual data, and report the averaged results.

Results Our results are summarized in Table 1 and 2, the first describing the ability of our learning algorithm to generalize to new game data, and the second - to new instructions.

A natural baseline for the prediction problem is to simply return FALSE regardless of the input - this ensures a baseline performance of 0.5 . The performance achieved without

Target Concept	Initial Model	Learned Model
FREECELL	0.78	0.956
HOMECELL	0.532	0.672
TABLEAU	0.536	0.628

Table 1: Results for the Freecell game rules. Accuracy was evaluated over previously unseen game moves using the classification rules learned from the instructions used in training. The Initial Model column describes the performance of the rules generated by the initial interpretation model (i.e., before learning).

Target Concept	Initial Model	Learned Model
FREECELL	0.78	0.967
HOMECELL	0.532	0.668
TABLEAU	0.536	0.608

Table 2: Results for the Freecell game rules. Accuracy was evaluated over previously unseen game moves using classification rules generated from *previously unseen instructions*. Semantic interpretation was done using the learned semantic interpreter.

learning (using the initialized model) barely improves on that baseline, after learning this figure consistently improves.

As can be noticed in the examples above, target concepts have different levels of difficulty - the FREECELL concept is the easiest one, both in terms of the output structure and the text used to describe it. The results indeed support this observation, and performance for this task is excellent. The other tasks are more difficult, resulting in a more modest improvement, however the improvement due to learning is still clear.

Finally, we can see that our learning algorithm is also able to learn a good semantic interpreter which generalizes well to previously unseen instructions.

6 Related Work

Human-centric learning protocols In this work we study a novel learning protocol based on learning from instructions given by a human teacher. Instructable computing approaches leveraging human expertise, have typically been studied in a reinforcement learning setting, in which a human teacher provides additional feedback to the learning process (a few recent examples include [Isbell *et al.*, 2006; Knox and Stone, 2009; Thomaz and Breazeal, 2006]). The role of human intervention in our learning framework is different, as we simulate a NL lesson scenario. The approach closest to ours is described in [Kuhlmann *et al.*, 2004], integrating NL advice into a reinforcement learner. However in their setting the language interpretation model is trained independently from the learner in a fully supervised process.

Semantic Interpretation of Natural Language Converting NL into a formal meaning representation is referred to as *semantic parsing*. This task has been studied extensively in the natural language processing community, typically by employing supervised machine learning approaches. Early works [Zelle and Mooney, 1996; Tang and Mooney, 2000] employed inductive logic programming approaches to learn a semantic parser. More recent works apply statistical learning methods to the problem [Kate and Mooney, 2006; Wong and Mooney, 2007; Zettlemoyer and Collins, 2005; 2009]. These works rely on annotated training data, consisting of sentences and their corresponding logical forms.

We learn to interpret NL instructions from game interaction feedback, instead of supervised learning. Learning in similar settings for semantic interpretation has been studied recently by several works: [Chen and Mooney, 2008; Liang *et al.*, 2009; Branavan *et al.*, 2009; Clarke *et al.*, 2010] use an external world context as a supervision signal for semantic interpretation. However the semantic interpretation task is different than ours - the NL input is completely situated in an external world state, while in our case the NL input describes a high level rule abstracting over specific states.

Leveraging textual instructions to improve game rules learning has been studied previously in [Eisenstein *et al.*, 2009] for Freecell solitaire. In that work textual interpretation was limited to mining repeating patterns and using them as features for learning the game rules over considerable amounts of game training data. Incorporating natural language advice in a game playing framework was also studied [Branavan *et al.*, 2011]. In their settings text interpretation is used to augment the state space representation in a reinforcement learning framework.

7 Conclusions

In this paper we investigate the feasibility of a new type of machine learning, based on language interpretation rather than labeled examples. This process, motivated by human learning processes, takes as input a natural language lesson describing the target concept and outputs a logical formula capturing the learning system understanding of the lesson. This approach has both theoretical and practical advantages, as it reduces the annotation cost and focuses the learning process on human-level task expertise rather than on machine learning and technical expertise.

To fulfill its promise this type of learning requires communicating effectively with the learning system in a natural, human-level manner. This introduces the major challenge in lesson based learning - interpreting natural language instructions. To avoid the difficulty of training a semantic interpreter independently, we introduce a novel learning algorithm that learns both tasks jointly by exploiting the dependency between the *target concept* learning task and the *language interpretation* learning task.

Acknowledgments We thank the anonymous reviewers for their helpful feedback. This research is supported by the Defense Advanced Research Projects Agency (DARPA) Bootstrapped Learning Program and Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

[Branavan *et al.*, 2009] S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. Reinforcement learning for mapping instructions to actions. In *ACL*, 2009.

[Branavan *et al.*, 2011] S.R.K. Branavan, D. Silver, and R. Barzilay. Playing games with language in a monte-carlo framework. In *ACL*, 2011.

[Chen and Mooney, 2008] D. Chen and R. Mooney. Learning to sportscast: a test of grounded language acquisition. In *ICML*, 2008.

[Clarke *et al.*, 2010] J. Clarke, D. Goldwasser, M. Chang, and D. Roth. Driving semantic parsing from the world's response. In *CoNLL*, 7 2010.

[Eisenstein *et al.*, 2009] J. Eisenstein, J. Clarke, D. Goldwasser, and D. Roth. Reading to learn: Constructing features from semantic abstracts. In *EMNLP*, 2009.

[Isbell *et al.*, 2006] C.L Isbell, M. Kearns, S. Singh, C. Shelton, P. Stone, and D. Kormann. Cobot in lambdamoo: An adaptive social statistics agent. In *AAMAS*, 2006.

[Kate and Mooney, 2006] R. Kate and R. Mooney. Using string-kernels for learning semantic parsers. In *ACL*, 2006.

[Klein and Manning, 2003] D. Klein and C. D. Manning. Fast exact inference with a factored model for natural language parsing. In *NIPS*, 2003.

[Knox and Stone, 2009] B. Knox and P. Stone. Interactively shaping agents via human reinforcement. In *KCAP*, 2009.

[Kuhlmann *et al.*, 2004] G. Kuhlmann, P. Stone, R. J. Mooney, and J. W. Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *AAAI workshops*, 2004.

[Liang *et al.*, 2009] P. Liang, M. I. Jordan, and D. Klein. Learning semantic correspondences with less supervision. In *ACL*, 2009.

[Martins *et al.*, 2009] A. Martins, N. A. Smith, and E. Xing. Concise integer linear programming formulations for dependency parsing. In *ACL*, 2009.

[Miller *et al.*, 1990] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 1990.

[Roth and Yih, 2007] D. Roth and W. Yih. Global inference for entity and relation identification via a linear programming formulation. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, 2007.

[Tang and Mooney, 2000] L. Tang and R. Mooney. Automated construction of database interfaces: integrating statistical and relational learning for semantic parsing. In *EMNLP*, 2000.

[Thomaz and Breazeal, 2006] A. L. Thomaz and C. Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*, 2006.

[Wong and Mooney, 2007] Y.W. Wong and R. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*, 2007.

[Zelle and Mooney, 1996] J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic proramming. In *AAAI*, 1996.

[Zettlemoyer and Collins, 2005] L. Zettlemoyer and M. Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005.

[Zettlemoyer and Collins, 2009] L. Zettlemoyer and M. Collins. Learning context-dependent mappings from sentences to logical form. In *ACL*, 2009.