Adapting Discriminative Reranking to Grounded Language Learning

Joohyun Kim
Department of Computer Science
The University of Texas at Austin
Austin, TX 78701, USA
scimitar@cs.utexas.edu

Raymond J. Mooney
Department of Computer Science
The University of Texas at Austin
Austin, TX 78701, USA
mooney@cs.utexas.edu

Abstract

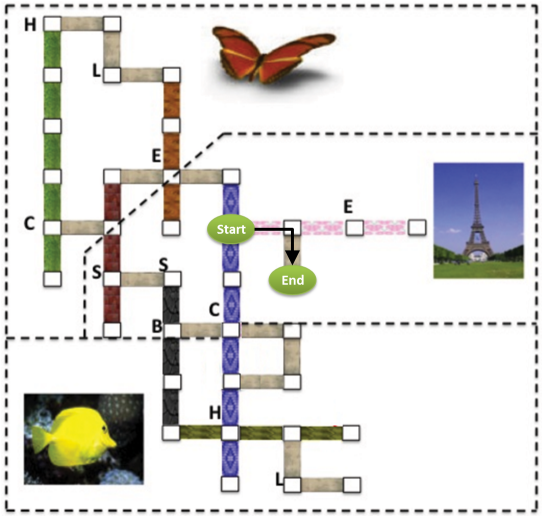
We adapt discriminative reranking to improve the performance of grounded language acquisition, specifically the task of learning to follow navigation instructions from observation. Unlike conventional reranking used in syntactic and semantic parsing, gold-standard reference trees are not naturally available in a grounded setting. Instead, we show how the weak supervision of response feedback (e.g. successful task completion) can be used as an alternative, experimentally demonstrating that its performance is comparable to training on gold-standard parse trees.

1 Introduction

Grounded language acquisition involves learning to comprehend and/or generate language by simply observing its use in a naturally occurring context in which the meaning of a sentence is grounded in perception and/or action (Roy, 2002; Yu and Ballard, 2004; Gold and Scassellati, 2007; Chen et al., 2010). Börschinger et al. (2011) introduced an approach that reduces grounded language learning to unsupervised *probabilistic context-free grammar* (PCFG) induction and demonstrated its effectiveness on the task of sportscasting simulated robot soccer games. Subsequently, Kim and Mooney (2012) extended their approach to make it tractable for the more complex problem of learning to follow natural-language navigation instructions from observations of humans following such instructions in a virtual environment (Chen and Mooney, 2011). The observed sequence of actions provides very weak, ambiguous supervision for learning instructional language since there are many possible ways to describe the same execution path. Although their approach improved accuracy on the navigation task compared to the original work of Chen and Mooney (2011), it was still far from human performance.

Since their system employs a generative model, *discriminative reranking* (Collins, 2000) could potentially improve its performance. By training a discriminative classifier that uses global features of complete parses to identify correct interpretations, a reranker can significantly improve the accuracy of a generative model. Reranking has been successfully employed to improve syntactic parsing (Collins, 2002b), semantic parsing (Lu et al., 2008; Ge and Mooney, 2006), semantic role labeling (Toutanova et al., 2005), and named entity recognition (Collins, 2002c). Standard reranking requires gold-standard interpretations (e.g. parse trees) to train the discriminative classifier. However, grounded language learning does not provide gold-standard interpretations for the training examples. Only the ambiguous perceptual context of the utterance is provided as supervision. For the navigation task, this supervision consists of the observed sequence of actions taken by a human when following an instruction. Therefore, it is impossible to directly apply conventional discriminative reranking to such problems. We show how to adapt reranking to work with such weak supervision. Instead of using gold-standard annotations to determine the correct interpretations, we simply prefer interpretations of navigation instructions that, when executed in the world, actually reach the intended destination. Additionally, we extensively revise the features typically used in parse reranking to work with the PCFG approach to grounded language learning.

The rest of the paper is organized as follows: Section 2 reviews the navigation task and the PCFG approach to grounded language learning. Section 3 presents our modified approach to reranking and Section 4 describes the novel features used to evaluate parses. Section 5 experimentally evaluates the approach comparing to sev-



(a) Sample virtual world of hallways with varying tiles, wallpapers, and landmark objects indicated by letters (e.g. "H" for hat-rack) and illustrating a sample path taken by a human follower.

Instruction:	*"at the very next intersection take a right onto the plain path and follow it to the end of the hall"*
Landmarks plan:	**Travel (steps: 1) ,** Verify (side: CONCRETE HALLWAY) , **Turn (RIGHT) ,** **Verify (** back: WALL , **front: CONCRETE HALLWAY ,** left: EASEL **) ,** **Travel (** steps: 1 **) ,** **Verify (front: WALL)**

(b) A sample natural language instruction and its formal landmarks plan for the path illustrated above. The subset corresponding to the correct formal plan is shown in bold.

Figure 1: Sample virtual world and instruction.

eral baselines. Finally, Section 6 describes related work, Section 7 discusses future work, and Section 8 concludes.

2 Background

2.1 Navigation Task

We address the navigation learning task introduced by Chen and Mooney (2011). The goal is to interpret natural-language (NL) instructions in a virtual environment, thereby allowing a simulated robot to navigate to a specified location. Figure 1a shows a sample path executed by a human following the instruction in Figure 1b. Given *no* prior linguistic knowledge, the task is to learn to interpret such instructions by simply observing humans follow sample directions. Formally speaking, given training examples of the form (e_i, a_i, w_i), where e_i is an NL instruction, a_i is an executed action sequence for the instruction, and w_i is the initial world state, we want to learn to produce an appropriate action sequence a_j given a novel (e_j, w_j).

More specifically, one must learn a semantic parser that produces a plan p_j using a formal *meaning representation* (MR) language introduced by Chen and Mooney (2011). This plan is then executed by a simulated robot in a virtual environment. The MARCO system, introduced by MacMahon et al. (2006), executes the formal plan, flexibly adapting to situations encountered during execution and producing the action sequence a_j. During learning, Chen and Mooney construct a *landmarks* plan c_i for each training example, which includes the *complete* context observed in the world-state resulting from each observed action. The correct plan, p_i, (which is latent and must be inferred) is assumed to be composed from a subset of the components in the corresponding landmarks plan. The landmarks and correct plans for a sample instruction are shown in Figure 1b.

2.2 PCFG Induction for Grounded Language Learning

The baseline generative model we use for reranking employs the unsupervised PCFG induction approach introduced by Kim and Mooney (2012). This model is, in turn, based on the earlier model of Börschinger et al. (2011), which transforms the grounded language learning into unsupervised PCFG induction. The general approach uses grammar-formulation rules which construct CFG productions that form a grammar that effectively maps NL sentences to formal meaning representations (MRs) encoded in its nonterminals. After using Expectation-Maximization (EM) to estimate the parameters for these productions using the ambiguous supervision provided by the grounded-learning setting, it produces a PCFG whose most probable parse for a sentence encodes its correct semantic interpretation. Unfortunately, the initial approach of Börschinger et al. (2011) produces explosively large grammars when applied to more complex problems, such as our navigation task. Therefore, Kim and Mooney enhanced their approach to use a previously learned semantic lexicon to reduce the induced grammar to a tractable size. They also altered the processes for constructing productions and mapping parse trees to MRs in order to make the construction of semantic interpretations more compositional and allow the efficient construction of more complex representa-

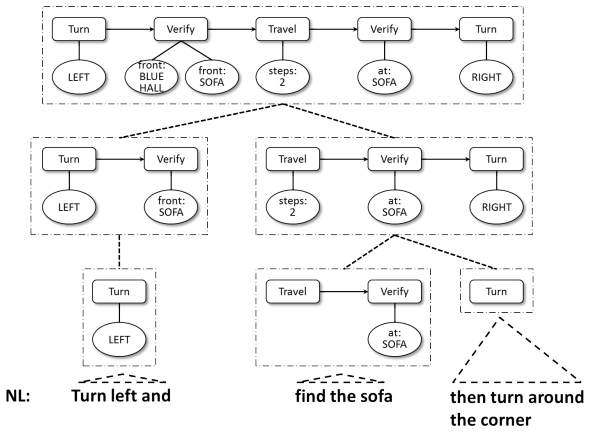


Figure 2: Simplified parse for the sentence *"Turn left and find the sofa then turn around the corner"* for Kim and Mooney's model. Nonterminals show the MR graph, where additional nonterminals for generating NL words are omitted.

tions.

The resulting PCFG can be used to produce a set of most-probable interpretations of instructional sentences for the navigation task. Our proposed reranking model is used to discriminatively reorder the top parses produced by this generative model. A simplified version of a sample parse tree for Kim and Mooney's model is shown in Figure 2.

3 Modified Reranking Algorithm

In reranking, a baseline generative model is first trained and generates a set of candidate outputs for each training example. Next, a second conditional model is trained which uses global features to rescore the candidates. Reranking using an averaged perceptron (Collins, 2002a) has been successfully applied to a variety of NLP tasks. Therefore, we modify it to rerank the parse trees generated by Kim and Mooney (2012)'s model. The approach requires three subcomponents: `1)` a GEN function that returns the list of top n candidate parse trees for each NL sentence produced by the generative model, `2)` a feature function Φ that maps a NL sentence, e, and a parse tree, y, into a real-valued feature vector $\Phi(e, y) \in R^d$, and `3)` a reference parse tree that is compared to the highest-scoring parse tree during training.

However, grounded language learning tasks, such as our navigation task, do not provide reference parse trees for training examples. Instead, our modified model replaces the gold-standard reference parse with the "pseudo-gold" parse tree

Algorithm 1 AVERAGED PERCEPTRON TRAINING WITH RESPONSE-BASED UPDATE

Input: A set of training examples (e_i, y_i^*), where e_i is a NL sentence and $y_i^* = \arg\max_{y \in \text{GEN}(e_i)} \text{EXEC}(y)$
Output: The parameter vector $\bar{W}$, averaged over all iterations $1...T$

procedure PERCEPTRON
  Initialize $\bar{W} = 0$
  for $t = 1...T, i = 1...n$ **do**
    $y_i = \arg\max_{y \in \text{GEN}(e_i)} \Phi(e_i, y) \cdot \bar{W}$
    if $y_i \neq y_i^*$ **then**
      $\bar{W} = \bar{W} + \Phi(e_i, y_i^*) - \Phi(e_i, y_i)$
    end if
  end for
end procedure

whose derived MR plan is most successful at getting to the desired goal location. Thus, the third component in our reranking model becomes an evaluation function EXEC that maps a parse tree y into a real number representing the success rate (w.r.t. successfully reaching the intended destination) of the derived MR plan m composed from y.

Additionally, we improve the perceptron training algorithm by using multiple reference parses to update the weight vector $\bar{W}$. Although we determine the pseudo-gold reference tree to be the candidate parse y^* such that $y^* = \arg\max_{y \in \text{GEN}(e)} \text{EXEC}(y)$, it may not actually be the correct parse for the sentence. Other parses may contain useful information for learning, and therefore we devise a way to update weights using *all* candidate parses whose successful execution rate is greater than the parse preferred by the currently learned model.

3.1 Response-Based Weight Updates

To circumvent the need for gold-standard reference parses, we select a pseudo-gold parse from the candidates produced by the GEN function. In a similar vein, when reranking semantic parses, Ge and Mooney (2006) chose as a reference parse the one which was most similar to the gold-standard semantic annotation. However, in the navigation task, the ultimate goal is to generate a plan that, when actually executed in the virtual environment, leads to the desired destination. Therefore, the pseudo-gold reference is chosen as the candidate parse that produces the MR plan with the great-

est execution success. This requires an external module that evaluates the execution accuracy of the candidate parses. For the navigation task, we use the MARCO (MacMahon et al., 2006) execution module, which is also used to evaluate how well the overall system learns to follow directions (Chen and Mooney, 2011). Since MARCO is nondeterministic when executing underspecified plans, we execute each candidate plan 10 times, and its execution rate is the percentage of trials in which it reaches the correct destination. When there are multiple candidate parses tied for the highest execution rate, the one assigned the largest probability by the baseline model is selected. Our modified averaged perceptron procedure with such a response-based update is shown in Algorithm 1.

One additional issue must be addressed when computing the output of the GEN function. The final plan MRs are produced from parse trees using compositional semantics (see Kim and Mooney (2012) for details). Consequently, the n-best parse trees for the baseline model do not necessarily produce the n-best *distinct* plans, since many parses can produce the same plan. Therefore, we adapt the GEN function to produce the n best distinct plans rather than the n best parses. This may require examining many more than the n best parses, because many parses have insignificant differences that do not affect the final plan. The score assigned to a plan is the probability of the most probable parse that generates that plan. In order to efficiently compute the n best plans, we modify the exact n-best parsing algorithm developed by Huang and Chiang (2005). The modified algorithm ensures that each plan in the computed n best list produces a new distinct plan.

3.2 Weight Updates Using Multiple Parses

Typically, when used for reranking, the averaged perceptron updates its weights using the feature-vector difference between the current best predicted candidate and the gold-standard reference (line 6 in Algorithm 1). In our initial modified version, we replaced the gold-standard reference parse with the pseudo-gold reference, which has the highest execution rate amongst all candidate parses. However, this ignores all other candidate parses during perceptron training. However, it is not ideal to regard other candidate parses as "useless." There may be multiple candidate parses with the same maximum execution rate, and even candidates with lower execution rates could represent the correct plan for the instruction given the weak, indirect supervision provided by the observed sequence of human actions.

Therefore, we also consider a further modification of the averaged perceptron algorithm which updates its weights using multiple candidate parses. Instead of only updating the weights with the single difference between the predicted and pseudo-gold parses, the weight vector $\bar{W}$ is updated with the sum of feature-vector differences between the current predicted candidate and *all* other candidates that have a higher execution rate. Formally, in this version, we replace lines 5–6 of Algorithm 1 with:

1: **for all** $y \in \text{GEN}(e_i)$ where $y \neq y_i$ and $\text{EXEC}(y) > \text{EXEC}(y_i)$ **do**
2: $\quad \bar{W} = \bar{W} + (\text{EXEC}(y) - \text{EXEC}(y_i)) \times (\Phi(e_i, y) - \Phi(e_i, y_i))$
3: **end for**

where $\text{EXEC}(y)$ is the execution rate of the MR plan m derived from parse tree y.

In the experiments below, we demonstrate that, by exploiting multiple reference parses, this new update rule increases the execution accuracy of the final system. Intuitively, this approach gathers additional information from all candidate parses with higher execution accuracy when learning the discriminative reranker. In addition, as shown in line 2 of the algorithm above, it uses the difference in execution rates between a candidate and the currently preferred parse to weight the update to the parameters for that candidate. This allows more effective plans to have a larger impact on the learned model in each iteration.

4 Reranking Features

This section describes the features Φ extracted from parses produced by the generative model and used to rerank the candidates.

4.1 Base Features

The base features adapt those used in previous reranking methods, specifically those of Collins (2002a), Lu et al. (2008), and Ge and Mooney (2006), which are directly extracted from parse trees. In addition, we also include the log probability of the parse tree as an additional feature. Figure 3 shows a sample full parse tree from our baseline model, which is used when explaining the

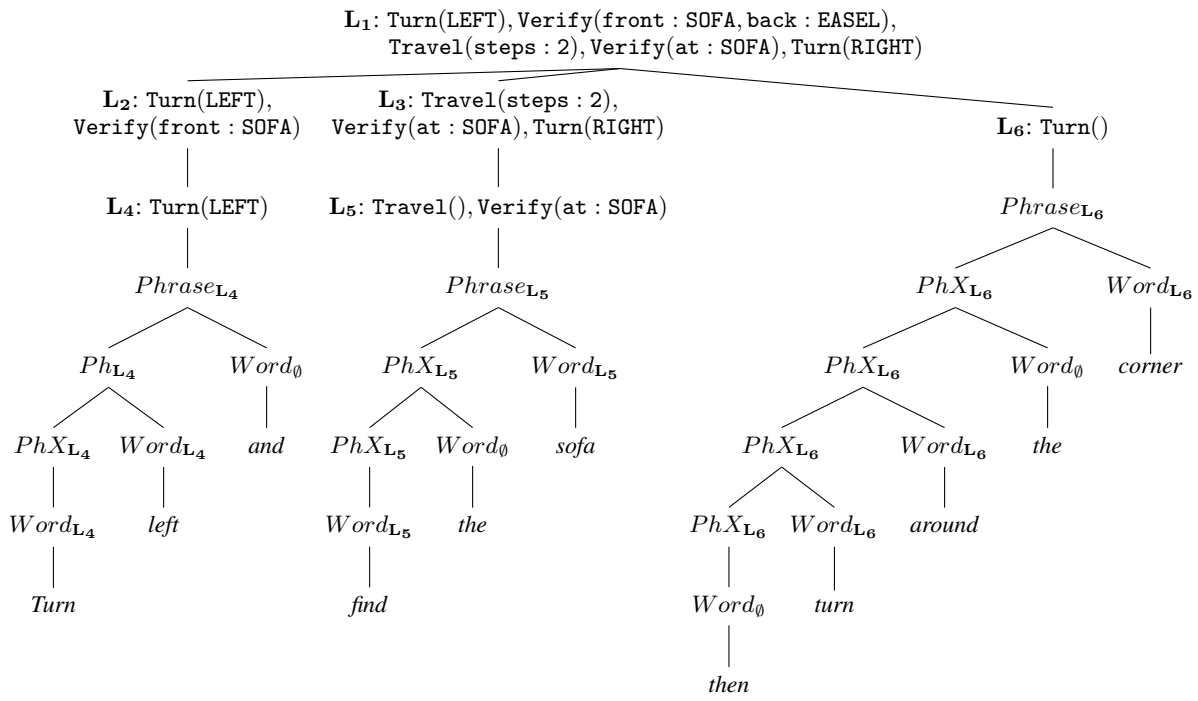


Figure 3: Sample full parse tree for the sentence *"Turn left and find the soft then turn around the corner"* used to explain reranking features. Nonterminals representing MR plan components are shown, which are labeled L_1 to L_6 for ease of reference. Additional nonterminals such as $Phrase$, Ph, PhX, and $Word$ are subsidiary ones for generating NL words from MR nonterminals. They are also shown in order to represent the entire process of how parse trees are constructed (for details, refer to Kim and Mooney (2012)).

reranking features below, each illustrated by an example.

a) PCFG Rule. Indicates whether a particular PCFG rule is used in the parse tree: $f(\mathtt{L_1} \Rightarrow \mathtt{L_2L_3}) = 1$.

b) Grandparent PCFG Rule. Indicates whether a particular PCFG rule *as well as* the nonterminal above it is used in the parse tree: $f(\mathtt{L_3} \Rightarrow \mathtt{L_5L_6}|\mathtt{L1}) = 1$.

c) Long-range Unigram. Indicates whether a nonterminal has a given NL word below it in the parse tree: $f(\mathtt{L_2} \rightsquigarrow \mathbf{left}) = 1$ and $f(\mathtt{L_4} \rightsquigarrow \mathbf{turn}) = 1$.

d) Two-level Long-range Unigram. Indicates whether a nonterminal has a child nonterminal which eventually generates a NL word in the parse tree: $f(\mathtt{L_4} \rightsquigarrow \mathbf{left}|\mathtt{L_2}) = 1$

e) Unigram. Indicates whether a nonterminal produces a given child nonterminal or terminal NL word in the parse tree: $f(\mathtt{L_1} \rightarrow \mathtt{L_2}) = 1$ and $f(\mathtt{L_1} \rightarrow \mathtt{L_3}) = 1$.

f) Grandparent Unigram. Indicates whether a nonterminal has a given child nonterminal/terminal below it, as well as a given parent nonterminal: $f(\mathtt{L_2} \rightarrow \mathtt{L_4}|\mathtt{L_1}) = 1$

g) Bigram. Indicates whether a given bigram of nonterminal/terminals occurs for given a parent nonterminal: $f(\mathtt{L_1} \rightarrow \mathtt{L_2} : \mathtt{L_3}) = 1$.

h) Grandparent Bigram. Same as Bigram, but also includes the nonterminal above the parent nonterminal: $f(\mathtt{L_3} \rightarrow \mathtt{L_5} : \mathtt{L_6}|\mathtt{L_1}) = 1$.

i) Log-probability of Parse Tree. Certainty assigned by the base generative model.

4.2 Predicate-Only Features

The base features above generally include nonterminal symbols used in the parse tree. In the grounded PCFG model, nonterminals are named after components of the semantic representations (MRs), which are complex and numerous. There are $\simeq$ 2,500 nonterminals in the grammar constructed for the navigation data, most of which are very specific and rare. This results in a very large, sparse feature space which can easily lead

the reranking model to over-fit the training data and prevent it from generalizing properly.

Therefore, we also tried constructing more general features that are less sparse. First, we construct generalized versions of the base features in which nonterminal symbols use only predicate names and omit their arguments. In the navigation task, action arguments frequently contain redundant, rarely used information. In particular, the interleaving verification steps frequently include many details that are never actually mentioned in the NL instructions. For instance, a nonterminal for the MR

```
Turn(LEFT),
Verify(at:SOFA,front:EASEL),
Travel(steps:3)
```

is transformed into the predicate-only form

```
Turn(), Verify(), Travel()
```

, and then used to construct more general versions of the base features described in the previous section. Second, another version of the base features are constructed in which nonterminal symbols include action arguments but omit all interleaving verification steps. This is a somewhat more conservative simplification of the nonterminal symbols. Although verification steps sometimes help interpret the actions and their surrounding context, they frequently cause the nonterminal symbols to become unnecessarily complex and specific.

4.3 Descended Action Features

Finally, another feature group which we utilize captures whether a particular atomic action in a nonterminal "descends" into one of its child nonterminals or not. An atomic action consists of a predicate and its arguments, e.g. `Turn(LEFT)`, `Travel(steps:2)`, or `Verify(at:SOFA)`. When an atomic action descends into lower nonterminals in a parse tree, it indicates that it is mentioned in the NL instruction and is therefore important. Below are several feature types related to descended actions that are used in our reranking model:

a) Descended Action. Indicates whether a given atomic action in a nonterminal descends to the next level. In Figure 3, $f(\texttt{Turn}(\texttt{LEFT})) = 1$ since it descends into L_2 and L_4.

b) Descended Action Unigram. Same as Descended Action, but also includes the current nonterminal: $f(\texttt{Turn}(\texttt{LEFT})|\texttt{L}_1) = 1$.

c) Grandparent Descended Action Unigram. Same as Descended Action Unigram, but additionally includes the parent nonterminal as well as the current one: $f(\texttt{Turn}(\texttt{LEFT})|\texttt{L}_2, \texttt{L}_1) = 1$.

d) Long-range Descended Action Unigram. Indicates whether a given atomic action in a nonterminal descends to a child nonterminal and this child generates a given NL word below it: $f(\texttt{Turn}(\texttt{LEFT}) \rightsquigarrow \textbf{left}) = 1$

5 Experimental Evaluation

5.1 Data and Methodology

The navigation data was collected by MacMahon et al. (2006), and includes English instructions and human follower data.[1] The data contains 706 route instructions for three virtual worlds. The instructions were produced by six instructors for 126 unique starting and ending location pairs over the three maps. Each instruction is annotated with 1 to 15 human follower traces with an average of 10.4 actions per instruction. Each instruction contains an average of 5.0 sentences each with an average of 7.8 words. Chen and Mooney (2011) constructed a version of the data in which each sentence is annotated with the actions taken by the majority of followers when responding to this sentence. This single-sentence version is used for training. Manually annotated "gold standard" formal plans for each sentence are used for evaluation purposes only.

We followed the same experimental methodology as Kim and Mooney (2012) and Chen and Mooney (2011). We performed "leave one environment out" cross-validation, i.e. 3 trials of training on two environments and testing on the third. The baseline model is first trained on data for two environments and then used to generate the $n = 50$ best plans for both training and testing instructions. As mentioned in Section 3.1, we need to generate many more top parse trees to get 50 distinct formal MR plans. We limit the number of best parse trees to 1,000,000, and even with this high limit, some training examples were left with less than 50 distinct plans.[2] Each candidate

[1] Data is available at `http://www.cs.utexas.edu/users/ml/clamp/navigation/`

[2] 9.6% of the examples (310 out of total 3237) produced less than 50 distinct MR plans in the evaluation. This was mostly due to exceeding the parse-tree limit and partly because the baseline model failed to parse some NL sentences.

n		1	2	5	10	25	50
Parse Accuracy	F1	74.81	79.08	82.78	85.32	87.52	88.62
Plan Execution	Single-sentence	57.22	63.86	70.93	76.41	83.59	87.02
	Paragraph	20.17	28.08	35.34	40.64	48.69	53.66

Table 1: Oracle parse and execution accuracy for single sentence and complete paragraph instructions for the n best parses.

plan is then executed using MARCO and its rate of successfully reaching the goal is recorded. Our reranking model is then trained on the training data using the n-best candidate parses. We only retain reranking features that appear (i.e. have a value of 1) at least twice in the training data.

Finally, we measure both parse and execution accuracy on the test data. Parse accuracy evaluates how well a system maps novel NL sentences for new environments into correct MR plans (Chen and Mooney, 2011). It is calculated by comparing the system's MR output to the gold-standard MR. Accuracy is measured using F1, the harmonic mean of precision and recall for individual MR constituents, thereby giving partial credit to approximately correct MRs. We then execute the resulting MR plans in the test environment to see whether they successfully reach the desired destinations. Execution is evaluated both for single sentence and complete paragraph instructions. Successful execution rates are calculated by averaging 10 nondeterministic MARCO executions.

5.2 Reranking Results

Oracle results

As typical in reranking experiments, we first present results for an "oracle" that always returns the best result amongst the top-n candidates produced by the baseline system, thereby providing an upper bound on the improvements possible with reranking. Table 1 shows oracle accuracy for both semantic parsing and plan execution for single sentence and complete paragraph instructions for various values of n. For oracle parse accuracy, for each sentence, we pick the parse that gives the highest F1 score. For oracle single-sentence execution accuracy, we pick the parse that gives the highest execution success rate. These single-sentence plans are then concatenated to produce a complete plan for each paragraph instruction in order to measure overall execution accuracy. Since making an error in *any* of the sentences in an instruction can easily lead to the wrong final destination, paragraph-level accuracies are always much lower than sentence-level ones. In order to balance oracle accuracy and the computational effort required to produce n distinct plans, we chose $n = 50$ for the final experiments since oracle performance begins to asymptote at this point.

Response-based vs. gold-standard reference weight updates

Table 2 presents reranking results for our proposed response-based weight update (`Single`) for the averaged perceptron (cf. Section 3.1) compared to the typical weight update method using gold-standard parses (`Gold`). Since the gold-standard annotation gives the correct MR rather than a parse tree for each sentence, `Gold` selects as a single reference parse the candidate in the top 50 whose resulting MR is most similar to the gold-standard MR as determined by its parse accuracy. Ge and Mooney (2006) employ a similar approach when reranking semantic parses.

The results show that our response-based approach (`Single`) has better execution accuracy than both the baseline *and* the standard approach using gold-standard parses (`Gold`). However, `Gold` does perform best on parse accuracy since it explicitly focuses on maximizing the accuracy of the resulting MR. In contrast, by focusing discriminative training on optimizing performance of the ultimate end task, our response-based approach actually outperforms the traditional approach on the final task. In addition, it only utilizes feedback that is naturally available for the task, rather than requiring an expert to laboriously annotate each sentence with a gold-standard MR. Even though `Gold` captures more elements of the gold-standard MRs, it may miss some critical MR components that are crucial to the final navigation task. The overall result is very promising because it demonstrates how reranking can be applied to grounded language learning tasks where gold-standard parses are not readily available.

	Parse Acc	Plan Execution	
	F1	Single	Paragraph
Baseline	74.81	57.22	20.17
Gold	**78.26**	52.57	19.33
Single	73.32	59.65	22.62
Multi	73.43	**62.81**	**26.57**

Table 2: Reranking results comparing our response-based methods using single (Single) or multiple (Multi) pseudo-gold parses to the standard approach using a single gold-standard parse (Gold). Baseline refers to Kim and Mooney (2012)'s system. Reranking results use all features described in Section 4.

Weight update with single vs. multiple reference parses

Table 2 also shows performance when using multiple reference parse trees to update weights (cf. Section 3.2). Using multiple parses (Multi) clearly performs better for all evaluation metrics, particularly execution. As explained in Section 3.2, the single-best pseudo-gold parse provides weak, ambiguous feedback since it only provides a rough estimate of the response feedback from the execution module. Using a variety of preferable parses to update weights provides a greater amount and variety of weak feedback and therefore leads to a more accurate model.[3]

Comparison of different feature groups

Table 3 compares reranking results using the different feature groups described in Section 4. Compared to the baseline model (Kim and Mooney, 2012), each of the feature groups Base (base features), Pred (predicate-only and verification-removed features), and Desc (descended action features) helps improve the performance of plan execution for both single sentence and complete paragraph navigation instructions. Among them, Desc is the most effective group of features. Combinations of the feature groups helps further improve the plan execution performance, and reranking using all of the feature groups (All) performs the best, as expected. However, since our model is optimizing plan execution during training, the results for parse accuracy are always worse than the baseline model.

[3]We also tried extending Gold to use multiple reference parses in the same manner, but this actually degraded its performance for all metrics. This indicates that, unlike Multi, parses other than the best one do not have useful information in terms of optimizing normal parse accuracy. Instead, additional parses seem to add noise to the training process in this case. Therefore, updating with multiple parses does not appear to be useful in standard reranking.

Features	Parse Acc	Plan Execution	
	F1	Single	Paragraph
Baseline	**74.81**	57.22	20.17
Base	71.50	60.09	23.20
Pred	71.61	60.87	24.13
Desc	73.90	61.33	25.00
Base+Pred	69.52	61.49	26.24
Base+Desc	73.66	61.72	25.58
Pred+Desc	72.56	62.36	26.04
All	73.43	**62.81**	**26.57**

Table 3: Reranking results comparing different sets of features. Base refers to base features (cf. Section 4.1), Pred refers to predicate-only features and also includes features based on removing interleaving verification steps (cf. Section 4.2), Desc refers to descended action features (cf. Section 4.3). All refers to all the features including Base, Pred, and Desc. All results use weight update with multiple reference parses (cf. Section 3.2).

6 Related Work

Discriminative reranking is a common machine learning technique to improve the output of generative models. It has been shown to be effective for various natural language processing tasks including syntactic parsing (Collins, 2000; Collins, 2002b; Collins and Koo, 2005; Charniak and Johnson, 2005; Huang, 2008), semantic parsing (Lu et al., 2008; Ge and Mooney, 2006), part-of-speech tagging (Collins, 2002a), semantic role labeling (Toutanova et al., 2005), named entity recognition (Collins, 2002c). machine translation (Shen et al., 2004; Fraser and Marcu, 2006) and surface realization in generation (White and Rajkumar, 2009; Konstas and Lapata, 2012). However, to our knowledge, there has been no previous attempt to apply discriminative reranking to grounded language acquisition, where gold-standard reference parses are not typically available for training reranking models.

Our use of response-based training is similar to work on learning semantic parsers from execution output such as the answers to database queries

(Clarke et al., 2010; Liang et al., 2011). Although the demands of grounded language tasks, such as following navigation instructions, are different, it would be interesting to try adapting these alternative approaches to such problems.

7 Future Work

In the future, we would like to explore the construction of better, more-general reranking features that are less prone to over-fitting. Since typical reranking features rely on the combination and/or modification of nonterminals appearing in parse trees, for the large PCFG's produced for grounded language learning, such features are very sparse and rare. Although the current features provide a significant increase in performance, oracle results imply that an even larger benefit may be achievable.

In addition, employing other reranking methodologies, such as kernel methods (Collins, 2002b), and forest reranking exploiting a packed forest of exponentially many parse trees (Huang, 2008), is another area of future work. We also would like to apply our approach to other reranking algorithms such as SVMs (Joachims, 2002) and MaxEnt methods (Charniak and Johnson, 2005).

8 Conclusions

In this paper, we have shown how to adapt discriminative reranking to grounded language learning. Since typical grounded language learning problems, such as navigation instruction following, do not provide the gold-standard reference parses required by standard reranking models, we have devised a novel method for using the weaker supervision provided by response feedback (e.g. the execution of inferred navigation plans) when training a perceptron-based reranker. This approach was shown to be very effective compared to the traditional method of using gold-standard parses. In addition, since this response-based supervision is weak and ambiguous, we have also presented a method for using multiple reference parses to perform perceptron weight updates and shown a clear further improvement in end-task performance with this approach.

Acknowledgments

We thank anonymous reviewers for their helpful comments to improve this paper. This work was funded by the NSF grant IIS-0712907 and IIS-1016312. Experiments were performed on the Mastodon Cluster, provided by NSF Grant EIA-0303609.

References

Benjamin Börschinger, Bevan K. Jones, and Mark Johnson. 2011. Reducing grounded learning tasks to grammatical inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1416–1425, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 173–180, Ann Arbor, MI, June.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, San Francisco, CA, USA, August.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37:397–435.

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 (CoNLL-2010)*, pages 18–27, Uppsala, Sweden, July. Association for Computational Linguistics.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–69.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 175–182, Stanford, CA, June.

Michael Collins. 2002a. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, Philadelphia, PA, July.

Michael Collins. 2002b. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 263–270, Philadelphia, PA, July.

Michael Collins. 2002c. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 489–496, Philadelphia, PA.

Alexander Fraser and Daniel Marcu. 2006. Semi-supervised training for statistical word alignment. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL-06)*, pages 769–776, Stroudsburg, PA, USA. Association for Computational Linguistics.

R. Ge and R. J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, Sydney, Australia, July.

Kevin Gold and Brian Scassellati. 2007. A robot that uses existing vocabulary to infer non-visual word meanings from observation. In *Proceedings of the 22nd national conference on Artificial intelligence - Volume 1*, AAAI'07, pages 883–888. AAAI Press.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594, Columbus, Ohio, June. Association for Computational Linguistics.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Canada.

Joohyun Kim and Raymond J. Mooney. 2012. Unsupervised PCFG induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning*, EMNLP-CoNLL '12.

Ioannis Konstas and Mirella Lapata. 2012. Concept-to-text generation via discriminative reranking. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 369–378, Stroudsburg, PA, USA. Association for Computational Linguistics.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL*, Portland, Oregon, June. Association for Computational Linguistics.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, Honolulu, HI, October.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: connecting language, knowledge, and action in route instructions. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1475–1482. AAAI Press.

Deb Roy. 2002. Learning visually grounded words and syntax for a scene description task. *Computer Speech and Language*, 16(3):353–385.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 177–184, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 589–596, Ann Arbor, MI, June.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 410–419, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chen Yu and Dana H. Ballard. 2004. On the integration of grounding language and learning objects. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, pages 488–493.