# Driving Semantic Parsing from the World's Response

**James Clarke** **Dan Goldwasser** **Ming-Wei Chang** **Dan Roth**
Department of Computer Science
University of Illinois
Urbana, IL 61820
{clarkeje,goldwas1,mchang21,danr}@illinois.edu

## Abstract

Current approaches to semantic parsing, the task of converting text to a formal meaning representation, rely on annotated training data mapping sentences to logical forms. Providing this supervision is a major bottleneck in scaling semantic parsers. This paper presents a new learning paradigm aimed at alleviating the supervision burden. We develop two novel learning algorithms capable of predicting complex structures which only rely on a binary feedback signal based on the context of an external world. In addition we reformulate the semantic parsing problem to reduce the dependency of the model on syntactic patterns, thus allowing our parser to scale better using less supervision. Our results surprisingly show that without using any annotated meaning representations learning with a weak feedback signal is capable of producing a parser that is competitive with fully supervised parsers.

## 1 Introduction

Semantic Parsing, the process of converting text into a formal meaning representation (MR), is one of the key challenges in natural language processing. Unlike shallow approaches for semantic interpretation (e.g., semantic role labeling and information extraction) which often result in an incomplete or ambiguous interpretation of the natural language (NL) input, the output of a semantic parser is a complete meaning representation that can be executed directly by a computer program.

Semantic parsing has mainly been studied in the context of providing natural language interfaces to computer systems. In these settings the target meaning representation is defined by the semantics of the underlying task. For example, providing access to databases: a question posed in natural language is converted into a formal database query that can be executed to retrieve information. Example 1 shows a NL input query and its corresponding meaning representation.

**Example 1** *Geoquery input text and output MR*
*"What is the largest state that borders Texas?"*
`largest(state(next_to(const(texas))))`

Previous works (Zelle and Mooney, 1996; Tang and Mooney, 2001; Zettlemoyer and Collins, 2005; Ge and Mooney, 2005; Zettlemoyer and Collins, 2007; Wong and Mooney, 2007) employ machine learning techniques to construct a semantic parser. The learning algorithm is given a set of input sentences and their corresponding meaning representations, and learns a statistical semantic parser — a set of rules mapping lexical items and syntactic patterns to their meaning representation and a score associated with each rule. Given a sentence, these rules are applied recursively to derive the most probable meaning representation. Since semantic interpretation is limited to syntactic patterns identified in the training data, the learning algorithm requires considerable amounts of annotated data to account for the syntactic variations associated with the meaning representation. Annotating sentences with their MR is a difficult, time consuming task; minimizing the supervision effort required for learning is a major challenge in scaling semantic parsers.

This paper proposes a new model and learning paradigm for semantic parsing aimed to alleviate the supervision bottleneck. Following the observation that the target meaning representation is to be executed by a computer program which in turn provides a response or outcome; we propose a *response driven learning framework* capable of exploiting feedback based on the response. The feedback can be viewed as a teacher judging whether the execution of the meaning representation produced the desired response for the input sentence.

This type of supervision is very natural in many situations and requires no expertise, thus can be supplied by any user.

Continuing with Example 1, the response generated by executing a database query would be used to provide feedback. The feedback would be whether the generated response is the correct answer for the input question or not, in this case *New Mexico* is the desired response.

In response driven semantic parsing, the learner is provided with a set of natural language sentences and a feedback function that encapsulates the teacher. The feedback function informs the learner whether its interpretation of the input sentence produces the desired response. We consider scenarios where the feedback is provided as a binary signal, correct $+1$ or incorrect $-1$.

This weaker form of supervision poses a challenge to conventional learning methods: semantic parsing is in essence a structured prediction problem requiring supervision for a set of interdependent decisions, while the provided supervision is binary, indicating the correctness of a generated meaning representation. To bridge this difference we propose two novel learning algorithms suited to the response driven setting.

Furthermore, to account for the many syntactic variations associated with the MR, we propose a new model for semantic parsing that allows us to learn effectively and generalize better. Current semantic parsing approaches extract parsing rules mapping NL to their MR, restricting possible interpretations to previously seen syntactic patterns. We replace the rigid inference process induced by the learned parsing rules with a flexible framework. We model semantic interpretation as a sequence of interdependent decisions, mapping text spans to predicates and use syntactic information to determine how the meaning of these logical fragments should be composed. We frame this process as an Integer Linear Programming (ILP) problem, a powerful and flexible inference framework that allows us to inject relevant domain knowledge into the inference process, such as specific domain semantics that restrict the space of possible interpretations.

We evaluate our learning approach and model on the well studied Geoquery domain (Zelle and Mooney, 1996; Tang and Mooney, 2001), a database consisting of U.S. geographical information, and natural language questions. Our experimental results show that our model with response driven learning can outperform existing models trained with annotated logical forms.

The key contributions of this paper are:

**Response driven learning for semantic parsing** We propose a new learning paradigm for learning semantic parsers without any annotated meaning representations. The supervision for learning comes from a binary feedback signal based a response generated by executing a meaning representation. This type of supervision signal is natural to produce and can be acquired from non-expert users.

**Novel training algorithms** Two novel training algorithms are developed within the response driven learning paradigm. The training algorithms are applicable beyond semantic parsing and can be used in situations where it is possible to obtain binary feedback for a structured learning problem.

**Flexible semantic interpretation process** We propose a novel flexible semantic parsing model that can handle previously unseen syntactic variations of the meaning representation.

## 2 Semantic Parsing

The goal of semantic parsing is to produce a function $F : \mathcal{X} \to \mathcal{Z}$ that maps from the space natural language input sentences, $\mathcal{X}$, to the space of meaning representations, $\mathcal{Z}$. This type of task is usually cast as a structured output prediction problem, where the goal is to obtain a model that assigns the highest score to the correct meaning representation given an input sentence. However, in the task of semantic parsing, this decision relies on identifying a hidden intermediate representation (or an *alignment*) that captures the way in which fragments of the text correspond to the meaning representation. Therefore, we formulate the prediction function as follows:

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

Where $\Phi$ is a feature function that describes the relationships between an input sentence $\mathbf{x}$, alignment $\mathbf{y}$ and meaning representation $\mathbf{z}$. $\mathbf{w}$ is the weight vector which contains the parameters of the model. We refer to the $\arg\max$ above as the inference problem. The feature function combined with the nature of the inference problem defines the semantic parsing model. The key to producing

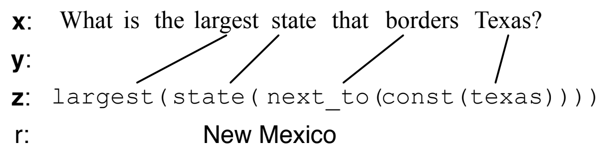
x: What is the largest state that borders Texas?
y:
z: largest(state( next_to(const(texas))))
r: New Mexico

Figure 1: Example input sentence, meaning representation, alignment and answer for the Geoquery domain

a semantic parser involves defining a model and a learning algorithm to obtain $\mathbf{w}$.

In order to exemplify these concepts we consider the Geoquery domain. Geoquery contains a query language for a database of U.S. geographical facts. Figure 1 illustrates concrete examples of the terminology introduce. The input sentences $\mathbf{x}$ are natural language queries about U.S. geography. The meaning representations $\mathbf{z}$ are logical forms which can be executed on the database to obtain a response which we denote with $r$. The alignment $\mathbf{y}$ captures the associations between $\mathbf{x}$ and $\mathbf{z}$.

Building a semantic parser involves defining the model (feature function $\Phi$ and inference problem) and a learning strategy to obtain weights ($\mathbf{w}$) associated with the model. We defer discussion of our model until Section 4 and first focus on our learning strategy.

## 3 Structured Learning with Binary Feedback

Previous approaches to semantic parsing have assumed a fully supervised setting where a training set is available consisting of either: input sentences and logical forms $\{(\mathbf{x}^l, \mathbf{z}^l)\}_{l=1}^N$ (e.g., (Zettlemoyer and Collins, 2005)) or input sentences, logical forms and a mapping between their constituents $\{(\mathbf{x}^l, \mathbf{y}^l, \mathbf{z}^l)\}_{l=1}^N$ (e.g., (Ge and Mooney, 2005)). Given such training examples a weight vector $\mathbf{w}$ can be learned using structured learning methods. Obtaining, through annotation or other means, this form of training data is an expensive and difficult process which presents a major bottleneck for semantic parsing.

To reduce the burden of annotation we focus on a new learning paradigm which uses feedback from a teacher. The feedback signal is binary $(+1, -1)$ and informs the learner whether a predicted logical form $\hat{\mathbf{z}}$ when executed on the target

**Algorithm 1** Direct Approach (Binary Learning)

```
Input: Sentences $\{\mathbf{x}^l\}_{l=1}^N$,
       $Feedback : \mathcal{X} \times \mathcal{Z} \rightarrow \{+1, 1\}$,
       initial weight vector $\mathbf{w}$
 1: $B_l \leftarrow \{\}$ 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:     $f = Feedback(\mathbf{x}^l, \hat{\mathbf{z}})$
 6:     add $(\Phi(\mathbf{x}^l, \hat{\mathbf{y}}, \hat{\mathbf{z}})/|\mathbf{x}^l|, f)$ to $B_l$
 7:   end for
 8:   $\mathbf{w} \leftarrow BinaryLearn(B)$ where $B = \cup_l B_l$
 9: until no $B_l$ has new unique examples
10: return $\mathbf{w}$
```

domain produces the desired response or outcome. This is a very natural method for providing supervision in many situations and requires no expertise. For example, a user can observe the response and provide a judgement. The general form of the teacher's feedback is provided by a function $Feedback : \mathcal{X} \times \mathcal{Z} \rightarrow \{+1, -1\}$.

For the Geoquery domain this amounts to whether the logical form produces the correct response $r$ for the input sentence. Geoquery has the added benefit that the teacher can be automated if we have a dataset consisting of input sentences and response pairs $\{(\mathbf{x}^l, r^l)\}_{l=1}^N$. *Feedback* evaluates whether a logical form produces a response matching $r$:

$$Feedback(\mathbf{x}^l, \mathbf{z}) = \begin{cases} +1 & \text{if } execute(\mathbf{z}) = r^l \\ -1 & \text{otherwise} \end{cases}$$

We are now ready to present our learning with feedback algorithms that operate in situations where input sentences, $\{\mathbf{x}^l\}_{l=1}^N$, and a teacher feedback mechanism, *Feedback*, are available. We do not assume the availability of logical forms.

### 3.1 Direct Approach (Binary Learning)

In general, a weight vector can be considered good if when used in the inference problem (Equation (1)) it scores the correct logical form and alignment (which may be hidden) higher than all other logical forms and alignments for a given input sentence. The intuition behind the *direct approach* is that the feedback function can be used to subsample the space of possible structures (alignments and logical forms ($\mathcal{Y} \times \mathcal{Z}$)) for a given input $\mathbf{x}$. The feedback mechanism indicates whether the structure is good (+1) or bad (−1). 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 the feature vector $\Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ normalized by the size of the input sentence[1] $|\mathbf{x}|$, and the label as the result from $\mathit{Feedback}(\mathbf{x}, \mathbf{z})$.

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 feedback function is queried (line 5) and a training example for the binary predictor created using the normalized feature vector from the triple containing the sentence, alignment and logical form as input and the feedback as the 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. Although the number of possible training examples is very large, in practice the algorithm is efficient and converges quickly. Note that this approach is capable of using a wide variety of linear classifiers as the base learner (line 8).

A policy is required to specify the nature of the working set of training examples ($B_l$) used for training the base classifier. This is pertinent in line 6 of the algorithm. Possible policies include: allowing duplicates in the working set (i.e., $B_l$ is a multiset), disallowing duplicates ($B_l$ is a set), or only allowing one example per input sentence ($\|B_l\| = 1$). We adopt the first approach in this paper.[2]

### 3.2 Aggressive Approach (Structured Learning)

There is important implicit information which the direct approach ignores. It is implicit that when the teacher indicates an input paired with an alignment and logical form is good ($+1$ feedback) that in order to repeat this behavior all other competing structures should be made suboptimal (or bad). To leverage this implicit information we adopt a structured learning strategy in which we consider the prediction as the optimal structure and all others as suboptimal. This is in contrast to the direct approach where only structures that have explicitly received negative feedback are considered subopitmal.

**Algorithm 2** Aggressive Approach (Structured Learning)

```
Input: Sentences $\{\mathbf{x}^l\}_{l=1}^N$,
       $\mathit{Feedback} : \mathcal{X} \times \mathcal{Z} \rightarrow \{+1, 1\}$,
       initial weight vector $\mathbf{w}$
 1: $S_l \leftarrow \emptyset$ 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:     $f = \mathit{Feedback}(\mathbf{x}^l, \hat{\mathbf{z}})$
 6:     if $f$ is $+1$ then
 7:       $S_l \leftarrow \{(\mathbf{x}^l, \hat{\mathbf{y}}, \hat{\mathbf{z}})\}$
 8:     end if
 9:   end for
10:   $\mathbf{w} \leftarrow \mathit{StructLearn}(S, \Phi)$ where $S = \cup_l S_l$
11: until no $S_l$ has changed
12: return $\mathbf{w}$
```

When a structure is found with positive feedback it is added to the training pool for a structured learner. We consider this approach *aggressive* as the structured learner implicitly considers all other structures as being suboptimal. Negative feedback indicates that the structure should not be added to the training pool as it will introduce noise into the learning process.

Algorithm 2 outlines the learning in more detail. As before, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ are predicted using the current weight vector and feedback received (lines 4 and 5). When positive feedback is received a new training instance for a structured learner is created from the input sentence and prediction (line 7) this training instance replaces any previous instance for the input sentence. When negative feedback is received the training pool $S_l$ is not updated. A weight vector is learned using a structured learner where the training data $S$ contains at most one example per input sentence. In the first iteration of the outer loop the training data $S$ will contain very few examples. In each subsequent iteration the newly learned weight vector allows the algorithm to acquire new examples. This is repeated until no

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

[2]The working set $B_l$ for each sentence may contain multiple positive examples with the same and differing alignments.

new examples are added or changed in $S$.

Like the direct approach, this learning framework is makes very few assumptions about the type of structured learner used as a base learner (line 10).[3]

## 4 Model

Semantic parsing is the process of converting a natural language input into a formal logic representation. This process is performed by associating lexical items and syntactic patterns with logical fragments and composing them into a complete formula. Existing approaches rely on extracting a set of *parsing rules*, mapping text constituents to a logical representation, from annotated training data and applying them recursively to obtain the meaning representation. Adapting to new data is a major limitation of these approaches as they cannot handle inputs containing syntactic patterns which were not observed in the training data. For example, assume the training data produced the following set of parsing rules:

**Example 2** *Typical parsing rules*
(1) *NP* [$\lambda x$.`capital(x)`] $\rightarrow$ *capital*
(2) *PP* [ `const(texas)`] $\rightarrow$ *of Texas*
(3) *NNP* [ `const(texas)`] $\rightarrow$ *Texas*
(4) *NP* [`capital(const(texas))`] $\rightarrow$
*NP*[$\lambda x$.`capital(x)`] *PP* [ `const(texas)`]

At test time the parser is given the sentences in Example 3. Despite the lexical similarity in these examples, the semantic parser will correctly parse the first sentence but fail to parse the second because the lexical items belong to different a syntactic category (i.e., the word *Texas* is not part of a preposition phrase in the second sentence).

**Example 3** *Syntactic variations of the same MR*
Target logical form: `capital(const(texas))`
Sentence 1: *"What is the capital of Texas?"*
Sentence 2: *"What is Texas' capital?"*

The ability to adapt to unseen inputs is one of the key challenges in semantic parsing. Several works (Zettlemoyer and Collins, 2007; Kate, 2008) have addressed this issue explicitly by manually defining syntactic transformation rules that can help the learned parser generalize better. Unfortunately these are only partial solutions as a manually constructed rule set cannot cover the many syntactic variations.

Given the previous example, we observe that it is enough to identify that the function `capital(·)` and the constant `const(texas)` appear in the target MR, since there is only a single way to compose these entities into a single formula — `capital(const(texas))`.

Motivated by this observation we define our meaning derivation process over the rules of the MR language and use syntactic information as a way to bias the MR construction process. That is, our inference process considers the *entire* space of meaning representations irrespective of the patterns observed in the training data. This is possible as the MRs are defined by a formal language and formal grammar.[4] The syntactic information present in the natural language is used as soft evidence (features) which guides the inference process to good meaning representations.

This formulation is a major shift from existing approaches that rely on extracting parsing rules from the training data. In existing approaches the space of possible meaning representations is constrained by the patterns in the training data and syntactic structure of the natural language input. Our formulation considers the entire space of meaning representations and allows the model to adapt to previously unseen data and *always* produce a semantic interpretation by using the patterns observed in the input.

We frame our semantic interpretation process as a constrained optimization process, maximizing the objective function defined by Equation 1 which relies on extracting lexical and syntactic features instead of parsing rules. In the remainder of this section we explain the components of our inference model.

### 4.1 Target Meaning Representation

Following previous work, we capture the semantics of the Geoquery domain using a subset of first-order logic consisting of typed constants and functions. There are two types: entities $E$ in the domain and numeric values $N$. Functions describe a functional relationship over types (e.g., `population` $: E \rightarrow N$). A complete logical form is constructed through functional composition; in our formalism this is per-

[3]Mistake driven algorithms that do not enforce margin constraints may not be able to generalize using this protocol since they will repeat the same prediction at training time and therefore will not update the model.

[4]This is true for all meaning representations designed to be executed by a computer system.

formed by the substitution operator. For example, given the function `next_to(x)` and the expression `const(texas)`, substitution replaces the occurrence of the free variable `x`, with the expression, resulting in a new logical form: `next_to(const(texas))`. Due to space limitations we refer the reader to (Zelle and Mooney, 1996) for a detailed description of the Geoquery domain.

### 4.2 Semantic Parsing as Constrained Optimization

Recall that the goal of semantic parsing is to produce the following function (Equation (1)):

$$F_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y},\mathbf{z}} \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

However, given that $\mathbf{y}$ and $\mathbf{z}$ are complex structures it is necessary to decompose the structure into a set of smaller decisions to facilitate efficient inference.

In order to define our decomposition we introduce additional notation: $c$ is a constituent (or word span) in the input sentence $\mathbf{x}$ and $\mathcal{D}$ is the set of all function and constant symbols in the domain. The alignment $\mathbf{y}$ is defined as a set of mappings between constituents and symbols in the domain $\mathbf{y} = \{(c, s)\}$ where $s \in \mathcal{D}$.

We decompose the construction of an alignment and logical form into two types of decisions:
**First-order decisions**. A mapping between constituents and logical symbols (functions and constants).
**Second-order decisions**. Expressing how logical symbols are composed into a complete logical interpretation. For example, whether `next_to` and `state` forms `next_to(state(·))` or `state(next_to(·))`.

Note that for all possible logical forms and alignments there exists a one-to-one mapping to these decisions.

We frame the inference problem as an Integer Linear Programming (ILP) problem (Equation (2)) in which the first-order decisions are governed by $\alpha_{cs}$, a binary decision variable indicating that constituent $c$ is aligned with logical symbol `s`. And $\beta_{cs,dt}$ capture the second-order decisions indicating the symbol `t` (associated with constituent $d$) is an argument to function `s` (associated with constituent $c$).

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

It is clear that there are dependencies between the $\alpha$-variables and $\beta$-variables. For example, given that $\beta_{cs,dt}$ is active, the corresponding $\alpha$-variables $\alpha_{cs}$ and $\alpha_{dt}$ must also be active. In order to ensure a consistent solution we introduce a set of constraints on Equation (2). In addition we add constraints which leverage the typing information inherent in the domain to eliminate logical forms that are invalid in the Geoquery domain. For example, the function `length` only accepts `river` types as input. The set of constraints are:

- A given constituent can be associated with exactly one logical symbol.
- $\beta_{cs,dt}$ is active if and only if $\alpha_{cs}$ and $\alpha_{dt}$ are active.
- If $\beta_{cs,dt}$ is active, `s` must be a function and the types of `s` and `t` should be consistent.
- Functional composition is directional and acyclic.

The flexibility of ILP has previously been advantageous in natural language processing tasks (Roth and Yih, 2007) as it allows us to easily incorporate such constraints.

### 4.3 Features

The inference problem defined in Equation (2) uses two feature functions: $\Phi_1$ and $\Phi_2$.

**First-order decision features** $\Phi_1$ 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.[5] This lexicon is small and only used as a starting point. Existing approaches rely on annotated logical forms to extend the lexicon. However, in our setting we do not have access to annotated logical forms, instead we rely on external knowledge to supply further

[5]The lexicon contains on average 1.42 words per function and 1.07 words per constant. For example the function `next_to` has the lexical entries: *borders*, *next*, *adjacent* and the constant `illinois` the lexical item *illinois*.

information. We add features which measure the lexical similarity between a constituent and a logical symbol's surface forms (as defined by the lexicon). Two metrics are used: stemmed word match and a similarity metric based on WordNet (Miller et al., 1990) which allows our model to account for words not in the lexicon. The WordNet metric measures similarity based on synonymy, hyponymy and meronymy (Do et al., 2010). In the case where the constituent is a preposition, which are notorious for being ambiguous, we add a feature that considers the current lexical context (one word to the left and right) in addition to word similarity.

**Second-order decision features** $\Phi_2$ Determining how to compose two logical symbols relies on syntactic information, in our model 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 $d$. A set of features also indicate which logical symbols are usually composed together, without considering their alignment to text.

## 5 Experiments

In this section we describe our experimental setup, which includes the details of the domain, resources and parameters.

### 5.1 Domain and Corpus

We evaluate our system on the Geoquery domain as described previously. The domain consists of a database and Prolog query language for U.S. geographical facts. The corpus contains of 880 natural language queries paired with Prolog logical form queries ($(\mathbf{x}, \mathbf{z})$ pairs). We follow previous approaches and transform these queries into a functional representation. We randomly select 250 sentences for training and 250 sentences for testing.[6] We refer to the training set as *Response 250* (R250) indicating that each example $\mathbf{x}$ in this data set has a corresponding desired database response $r$. We refer the testing set as *Query 250* (Q250) where the examples only contain the natural language queries.

[6]Our inference problem is less constrained than previous approaches thus we limit the training data to 250 examples due to scalability issues. We also prune the search space by limiting the number of logical symbol candidates per word (on average 13 logical symbols per word).

Precision and recall are typically used as evaluation metrics in semantic parsing. However, as our model inherently has the ability to map any input sentence into the space of meaning representations the trade off between precision and recall does not exist. Thus, we report accuracy: the percentage of meaning representations which produce the correct response. This is equivalent to recall in previous work (Wong and Mooney, 2007; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007).

### 5.2 Resources and Parameters

**Feedback** Recall that our learning framework does not require meaning representation annotations. However, we do require a $Feedback$ function that informs the learner whether a predicted meaning representation when executed produces the desired response for a given input sentence. We automatically generate a set of natural language queries and response pairs $\{(\mathbf{x}, r)\}$ by executing the annotated logical forms on the database. Using this data we construct an automatic feedback function as described in Section 3.

**Domain knowledge** Our learning approaches require an initial weight vector as input. In order to provide an initial starting point, we initialize the weight vector using a similar procedure to the one used in (Zettlemoyer and Collins, 2007) to set weights for three features and a bias term. The weights were developed on the training set using the feedback function to guide our choices.

**Underlying Learning Algorithms** In the direct approach the base linear classifier we use is a linear kernel Support Vector Machine with squared-hinge loss. In the aggressive approach we define our base structured learner to be a structural Support Vector Machine with squared-hinge loss and use hamming distance as the distance function. We use a custom implementation to optimize the objective function using the Cutting-Plane method, this allows us to parrallelize the learning process by solving the inference problem for multiple training examples simultaneously.

## 6 Results

Our experiments are designed to answer three questions:

1. Is it possible to learn a semantic parser *without* annotated logical forms?

| Algorithm | R250 | Q250 |
|---|---|---|
| NOLEARN | 22.2 | — |
| DIRECT | 75.2 | 69.2 |
| AGGRESSIVE | 82.4 | 73.2 |
| SUPERVISED | 87.6 | 80.4 |

Table 1: Accuracy of learned models on R250 data and Q250 (testing) data. NOLEARN: using initialized weight vector, DIRECT: using feedback with the direct approach, AGGRESSIVE: using feedback with the aggressive approach, SUPERVISED: using gold 250 logical forms for training. Note that none of the approaches use any annotated logical forms besides the SUPERVISED approach.

| Algorithm | # LF | Accuracy |
|---|---|---|
| AGGRESSIVE | — | 73.2 |
| SUPERVISED | 250 | 80.4 |
| W&M 2006 | ∼ 310 | ∼ 60.0 |
| W&M 2007 | ∼ 310 | ∼ 75.0 |
| Z&C 2005 | 600 | 79.29 |
| Z&C 2007 | 600 | 86.07 |
| W&M 2007 | 800 | 86.59 |

Table 2: Comparison against previously published results. Results show that with a similar number of logical forms (# LF) for training our SUPERVISED approach outperforms existing systems, while the AGGRESSIVE approach remains competitive without using any logical forms.

2. How much performance do we sacrifice by not restricting our model to parsing rules?

3. What, if any, are the differences in behaviour between the two learning with feedback approaches?

We first compare how well our model performs under four different learning regimes. NOLEARN uses a manually initialized weight vector. DIRECT and AGGRESSIVE use the two response driven learning approaches, where a feedback function but no logical forms are provided. As an upper bound we train the model using a fully SUPERVISED approach where the input sentences are paired with hand annotated logical forms.

Table 1 shows the accuracy of each setup. The model without learning (NOLEARN) gives a starting point with an accuracy of 22.2%. The response driven learning methods perform substantially better than the starting point. The DIRECT approach which uses a binary learner reaches an accuracy of 75.2% on the R250 data and 69.2% on the Q250 (testing) data. While the AGGRESSIVE approach which uses a structured learner sees a bigger improvement, reaching 82.4% and 73.2% respectively. This is only 7% below the fully SUPERVISED upper bound of the model.

To answer the second question, we compare a supervised version of our model to existing semantic parsers. The results are in Table 2. Although the numbers are not directly comparable due to different splits in the data[7], we can see that with a similar number of logical forms for training our SUPERVISED approach outperforms existing systems (Wong and Mooney, 2006; Wong and Mooney, 2007), while the AGGRESSIVE approach remains competitive without using any logical forms. Our SUPERVISED model is still very competitive with other approaches (Zettlemoyer and Collins, 2007; Wong and Mooney, 2007), which used considerably more annotated logical forms in the training phase.

In order to answer the third question, we turn our attention to the differences between the two response driven learning approaches. The DIRECT and AGGRESSIVE approaches use binary feedback to learn, however they utilize the signal differently. DIRECT uses the signal directly to learn a binary classifier capable of replicating the feedback, whereas AGGRESSIVE learns a structured predictor that can repeatedly obtain the logical forms for which positive feedback was received. Thus, although the AGGRESSIVE outperforms the DIRECT approach the concepts each approach learns may be different. Analysis over the training data shows that in 66.8% examples both approaches predict a logical form that gives the correct answer. While AGGRESSIVE correctly answers an additional 16% which DIRECT gets incorrect. In the opposite direction, DIRECT correctly answers 8.8% that AGGRESSIVE does not. Leaving only 8.4% of the examples that both approaches predict incorrect logical forms. This suggests that an approach which combines DIRECT and AGGRESSIVE may be able to improve even further.

Figure 2 shows the accuracy on the entire training data (R250) at each iteration of learning. We see that the AGGRESSIVE approach learns to cover more of the training data and at a faster rate than DIRECT. Note that the performance of the DIRECT approach drops at the first iteration. We hypothesize this is due to imbalances in the binary feedback dataset (too many negative examples) in the first iteration.

[7]It is relatively difficult to compare different approaches in the Geoquery domain given that many existing papers do not use the same data split.

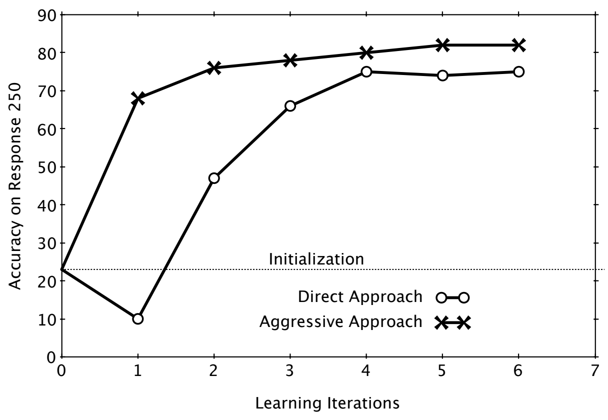

Figure 2: Accuracy on training set as number of learning iterations increases.

## 7 Related Work

Learning to map sentences to a meaning representation has been studied extensively in the NLP community. 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. In (Ge and Mooney, 2005; Nguyen et al., 2006), the input to the learner consists of complete syntactic derivations for the input sentences annotated with logical expressions. Other works (Wong and Mooney, 2006; Kate and Mooney, 2006; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Zettlemoyer and Collins, 2009) try to alleviate the annotation effort by only taking sentence and logical form pairs to train the models. Learning is then defined over hidden patterns in the training data that associate logical symbols with lexical and syntactic elements.

In this work we take an additional step towards alleviating the difficulty of training semantic parsers and present a world response based training protocol. Several recent works (Chen and Mooney, 2008; Liang et al., 2009; Branavan et al., 2009) explore using an external world context as a supervision signal for semantic interpretation. These works operate in settings different to ours as they rely on an external world state that is directly referenced by the input text. Although our framework can also be applied in these settings we do not assume that the text can be grounded in a world state. In our experiments the input text consists of generalized statements which describe some information need that does not correspond directly to a grounded world state.

Our learning framework closely follows recent work on learning from indirect supervision. The direct approach resembles learning a binary classifier over a latent structure (Chang et al., 2010a); while the aggressive approach has similarities with work that uses labeled structures and a binary signal indicating the existence of good structures to improve structured prediction (Chang et al., 2010b).

## 8 Conclusions

In this paper we tackle one of the key bottlenecks in semantic parsing — providing sufficient supervision to train a semantic parser. Our solution is two fold, first we present a new training paradigm for semantic parsing that relies on *natural, human level supervision*. Second, we suggest a new model for semantic interpretation that does not rely on NL syntactic parsing rules, but rather uses the syntactic information to bias the interpretation process. This approach allows the model to generalize better and reduce the required amount of supervision. We demonstrate the effectiveness of our training paradigm and interpretation model over the Geoquery domain, and show that our model can outperform fully supervised systems.

**Acknowledgements** We are grateful to Rohit Kate and Raymond Mooney for their help with the Geoquery dataset. Thanks to Yee Seng Chan, Nick Rizzolo, Shankar Vembu and the three anonymous reviewers for their insightful comments. This material is based upon work supported by the Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181 and by DARPA under the Bootstrap Learning Program. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the AFRL or DARPA.

## References

S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010a. Discriminative learning over constrained latent representations. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

M. Chang, D. Goldwasser, D. Roth, and V. Srikumar. 2010b. Structured output learning with indirect supervision. In *Proc. of the International Conference on Machine Learning (ICML)*.

D. Chen and R. Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *Proc. of the International Conference on Machine Learning (ICML)*.

Q. Do, D. Roth, M. Sammons, Y. Tu, and V.G. Vydiswaran. 2010. Robust, Light-weight Approaches to compute Lexical Similarity. Computer Science Research and Technical Reports, University of Illinois. http://hdl.handle.net/2142/15462.

R. Ge and R. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

R. Kate and R. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

R. Kate. 2008. Transforming meaning representation grammars to improve semantic parsing. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

D. Klein and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Proc. of the Conference on Advances in Neural Information Processing Systems (NIPS)*.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

L. Nguyen, A. Shimazu, and X. Phan. 2006. Semantic parsing with structured svm ensemble classification models. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

L. Tang and R. Mooney. 2000. Automated construction of database interfaces: integrating statistical and relational learning for semantic parsing. In *Proc. of the Conference on Empirical Methods for Natural Language Processing (EMNLP)*.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proc. of the European Conference on Machine Learning (ECML)*.

Y.-W. Wong and R. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*.

Y.-W. Wong and R. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

J. M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proc. of the Annual Conference in Uncertainty in Artificial Intelligence (UAI)*.

L. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning (EMNLP-CoNLL)*.

L. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*.