Unanimous Prediction for 100% Precision with Application to Learning Semantic Mappings

Fereshte Khani
Stanford University
fereshte@cs.stanford.edu

Martin Rinard
MIT
rinard@lcs.mit.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

Can we train a system that, on any new input, either says "don't know" or makes a prediction that is guaranteed to be correct? We answer the question in the affirmative provided our model family is well-specified. Specifically, we introduce the unanimity principle: only predict when all models consistent with the training data predict the same output. We operationalize this principle for semantic parsing, the task of mapping utterances to logical forms. We develop a simple, efficient method that reasons over the infinite set of all consistent models by only checking two of the models. We prove that our method obtains 100% precision even with a modest amount of training data from a possibly adversarial distribution. Empirically, we demonstrate the effectiveness of our approach on the standard GeoQuery dataset.

1 Introduction

If a user asks a system "*How many painkillers should I take?*", it is better for the system to say "don't know" rather than making a costly incorrect prediction. When the system is learned from data, uncertainty pervades, and we must manage this uncertainty properly to achieve our precision requirement. It is particularly challenging since training inputs might not be representative of test inputs due to limited data, covariate shift (Shimodaira, 2000), or adversarial filtering (Nelson et al., 2009; Mei and Zhu, 2015). In this unforgiving setting, can we still train a system that is *guaranteed* to either abstain or to make the correct prediction?

Our present work is motivated by the goal of

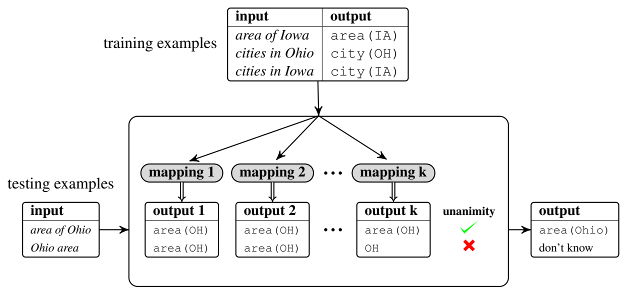


Figure 1: Given a set of training examples, we compute $\mathcal{C}$, the set of all mappings consistent with the training examples. On an input x, if all mappings in $\mathcal{C}$ unanimously predict the same output, we return that output; else we return "don't know".

building reliable question answering systems and natural language interfaces. Our goal is to learn a semantic mapping from examples of utterance-logical form pairs (Figure 1). More generally, we assume the input x is a bag (multiset) of source atoms (e.g., words $\{$*area*, *of*, *Ohio*$\}$), and the output y is a bag of target atoms (e.g., predicates $\{\texttt{area}, \texttt{OH}\}$). We consider learning mappings M that decompose according to the multiset sum: $M(x) = \uplus_{s \in x} M(s)$ (e.g., $M(\{\textit{Ohio}\}) = \{\texttt{OH}\}$, $M(\{\textit{area,of,Ohio}\}) = \{\texttt{area,OH}\}$). The main challenge is that an individual training example (x, y) does not tell us which source atoms map to which target atoms.[1]

How can a system be 100% sure about something if it has seen only a small number of possibly non-representative examples? Our approach is based on what we call the *unanimity* principle (Section 2.1). Let $\mathcal{M}$ be a model family that contains the true mapping from inputs to outputs. Let $\mathcal{C}$ be the subset of mappings that are consistent

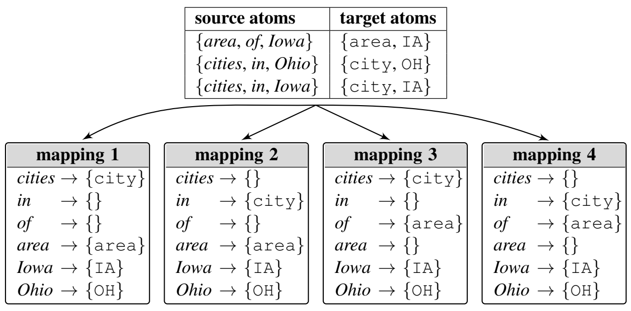
[1] A semantic parser further requires modeling the context dependence of words and the logical form structure joining the predicates. Our framework handles these cases with a different choice of source and target atoms (see Section 4.2).

with the training data. If all mappings $M \in \mathcal{C}$ unanimously predict the same output on a test input, then we return that output; else we return "don't know" (see Figure 1). The unanimity principle provides robustness to the particular input distribution, so that we can tolerate even adversaries (Mei and Zhu, 2015), provided the training outputs are still mostly correct.

To operationalize the unanimity principle, we need to be able to efficiently reason about the predictions of all consistent mappings $\mathcal{C}$. To this end, we represent a mapping as a matrix M, where M_{st} is number of times target atom t (e.g., OH) shows up for each occurrence of the source atom s (e.g., *Ohio*) in the input. We show that unanimous prediction can be performed by solving two integer linear programs. With a linear programming relaxation (Section 3), we further show that checking unanimity over $\mathcal{C}$ can be done very efficiently without any optimization but rather by checking the predictions of just two random mappings, while still guaranteeing 100% precision with probability 1 (Section 3.2).

We further relax the linear program to a linear system, which gives us a geometric view of the unanimity: We predict on a new input if it can be expressed as a "linear combination" of the training inputs. As an example, suppose we are given training data consisting of (CI) *cities in Iowa*, (CO) *cities in Ohio*, and (AI) *area of Iowa* (Figure 1). We can compute (AO) *area of Ohio* by analogy: (AO) = (CO) - (CI) + (AI). Other reasoning patterns fall out from more complex linear combinations.

We can handle noisy data (Section 3.4) by asking for unanimity over additional slack variables. We also show how the linear algebraic formulation enables other extensions such as learning from denotations (Section 5.1), active learning (Section 5.2), and paraphrasing (Section 5.3). We validate our methods in Section 4. On artificial data generated from an adversarial distribution with noise, we show that unanimous prediction obtains 100% precision, whereas point estimates fail. On GeoQuery (Zelle and Mooney, 1996), a standard semantic parsing dataset, where our model assumptions are violated, we still obtain 100% precision. We were able to reach 70% recall on recovering predicates and 59% on full logical forms.



source atoms	target atoms
{*area, of, Iowa*}	{area, IA}
{*cities, in, Ohio*}	{city, OH}
{*cities, in, Iowa*}	{city, IA}

mapping 1	mapping 2	mapping 3	mapping 4
cities → {city}	*cities* → {}	*cities* → {city}	*cities* → {}
in → {}	*in* → {city}	*in* → {}	*in* → {city}
of → {}	*of* → {}	*of* → {area}	*of* → {area}
area → {area}	*area* → {area}	*area* → {}	*area* → {}
Iowa → {IA}	*Iowa* → {IA}	*Iowa* → {IA}	*Iowa* → {IA}
Ohio → {OH}	*Ohio* → {OH}	*Ohio* → {OH}	*Ohio* → {OH}

Figure 2: Given the training examples in the top table, there are exactly four mappings consistent with these training examples.

2 Setup

We represent an input x (e.g., *area of Ohio*) as a bag (multiset) of *source atoms* and an output y (e.g., area(OH)) as a bag of *target atoms*. In the simplest case, *source atoms* are words and *target atoms* are predicates—see Figure 2(top) for an example.[2] We assume there is a true mapping M^* from a source atom s (e.g., *Ohio*) to a bag of target atoms $t = M^*(s)$ (e.g., $\{\texttt{OH}\}$). Note that M^* can also map a source atom s to no target atoms ($M^*(\textit{of}) = \{\}$) or multiple target atoms ($M^*(\textit{grandparent}) = \{\texttt{parent}, \texttt{parent}\}$). We extend M^* to bag of source atoms via multiset sum: $M^*(x) = \uplus_{s \in x} M^*(s)$.

Of course, we do not know M^* and must estimate it from training data. Our training examples are input-output pairs $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$. For now, we assume that there is no noise so that $y_i = M^*(x_i)$; Section 3.4 shows how to deal with noise. Our goal is to output a *mapping* $\hat{M}$ that maps each input x to either a bag of target atoms or "don't know." We say that $\hat{M}$ has 100% precision if $\hat{M}(x) = M^*(x)$ whenever $\hat{M}(x)$ is not "don't know." The chief difficulty is that the source atoms x_i and the target atoms y_i are unaligned. While we could try to infer the alignment, we will show that it is unnecessary for obtaining 100% precision.

2.1 Unanimity principle

Let $\mathcal{M}$ be the set of mappings (which contains the true mapping M^*). Let $\mathcal{C}$ be the subset of map-

[2]Our semantic parsing experiments (Section 4.2) use more complex source and target atoms to capture some context and structure.

$$S = \begin{array}{r|cccccc|}
 & \textit{area} & \textit{of} & \textit{Ohio} & \textit{cities} & \textit{in} & \textit{Iowa} \\
\textit{area of Iowa} & 1 & 1 & 0 & 0 & 0 & 1 \\
\textit{cities in Ohio} & 0 & 0 & 1 & 1 & 1 & 0 \\
\textit{cities in Iowa} & 0 & 0 & 0 & 1 & 1 & 1
\end{array}
\quad
M = \begin{array}{r|cccc|}
 & \texttt{area} & \texttt{city} & \texttt{OH} & \texttt{IA} \\
\textit{area} & 1 & 0 & 0 & 0 \\
\textit{of} & 0 & 0 & 0 & 0 \\
\textit{Ohio} & 0 & 0 & 1 & 0 \\
\textit{cities} & 0 & 1 & 0 & 0 \\
\textit{in} & 0 & 0 & 0 & 0 \\
\textit{Iowa} & 0 & 0 & 0 & 1
\end{array}
\quad
T = \begin{array}{r|cccc|}
 & \texttt{area} & \texttt{city} & \texttt{OH} & \texttt{IA} \\
\texttt{area(IA)} & 1 & 0 & 0 & 1 \\
\texttt{city(OH)} & 0 & 1 & 1 & 0 \\
\texttt{city(IA)} & 0 & 1 & 0 & 1
\end{array}$$

Figure 3: Our training data encodes a system of linear equations $SM = T$, where the rows of S are inputs, the rows of T are the corresponding outputs, and M specifies the mapping between source and target atoms.

pings consistent with the training examples.

$$\mathcal{C} \overset{\text{def}}{=} \{M \in \mathcal{M} \mid M(x_i) = y_i, \forall i = 1, \dots, n\} \tag{1}$$

Figure 2 shows the four mappings consistent with the training set in our running example. Let $\mathcal{F}$ be the set of *safe inputs*, those on which all mappings in $\mathcal{C}$ agree:

$$\mathcal{F} \overset{\text{def}}{=} \{x : |\{M(x) : M \in \mathcal{C}\}| = 1\}. \tag{2}$$

The *unanimity principle* defines a mapping $\hat{M}$ that returns the unanimous output on $\mathcal{F}$ and "don't know" on its complement. This choice obtains the following strong guarantee:

Proposition 1. *For each safe input $x \in \mathcal{F}$, we have $\hat{M}(x) = M^*(x)$. In other words, M obtains 100% precision.*

Furthermore, $\hat{M}$ obtains the best possible recall given this model family subject to 100% precision, since for any $x \notin \mathcal{F}$ there are at least two possible outputs generated by consistent mappings, so we cannot safely guess one of them.

3 Linear algebraic formulation

To solve the learning problem laid out in the previous section, let us recast the problem in linear algebraic terms. Let n_{s} (n_{t}) be the number of source (target) atom types. First, we can represent the bag x (y) as a n_{s}-dimensional (n_{t}-dimensional) row vector of counts; for example, the vector form of "*area of Ohio*" is $\begin{array}{cccccc} \textit{area} & \textit{of} & \textit{Ohio} & \textit{cities} & \textit{in} & \textit{Iowa} \\ [1 & 1 & 1 & 0 & 0 & 0] \end{array}$.

We represent the mapping M as a non-negative integer-valued matrix, where M_{st} is the number of times target atom t appears in the bag that source atom s maps to (Figure 3). We also encode the n training examples as matrices: S is an $n \times n_{\text{s}}$ matrix where the i-th row is x_i; T as an $n \times n_{\text{t}}$ matrix where the i-th row is y_i. Given these matrices, we can rewrite the set of consistent mappings (2) as:

$$\mathcal{C} = \{M \in \mathbb{Z}_{\geq 0}^{n_{\text{s}} \times n_{\text{t}}} : SM = T\}. \tag{3}$$

See Figure 3 for the matrix formulation of S and T, along with one possible consistent mapping M for our running example.

3.1 Integer linear programming

Finding an element of $\mathcal{C}$ as defined in (3) corresponds to solving an integer linear program (ILP), which is NP-hard in the worst case, though there exist relatively effective off-the-shelf solvers such as Gurobi. However, *one* solution is not enough. To check whether an input x is in the safe set $\mathcal{F}$ (2), we need to check whether *all* mappings $M \in \mathcal{C}$ predict the same output on x; that is, xM is the same for all $M \in \mathcal{C}$.

Our insight is that we can check whether $x \in \mathcal{F}$ by solving just two ILPs. Recall that we want to know if the output vector xM can be different for different $M \in \mathcal{C}$. To do this, we pick a random vector $v \in \mathbb{R}^{n_{\text{t}}}$, and consider the scalar projection xMv. The first ILP maximizes this scalar and the second one minimizes it. If both ILPs return the same value, then with probability 1, we can conclude that xM is the same for all mappings $M \in \mathcal{C}$ and thus $x \in \mathcal{F}$. The following proposition formalizes this:

Proposition 2. *Let x be any input. Let $v \sim \mathcal{N}(0, I_{n_t \times n_t})$ be a random vector. Let $a = \min_{M \in \mathcal{C}} xMv$ and $b = \max_{M \in \mathcal{C}} xMv$. With probability 1, $a = b$ iff $x \in \mathcal{F}$.*

Proof. If $x \in \mathcal{F}$, there is only one output xM, so $a = b$. If $x \notin \mathcal{F}$, there exists two $M_1, M_2 \in \mathcal{C}$ for which $xM_1 \neq xM_2$. Then $w \overset{\text{def}}{=} x(M_1 -$

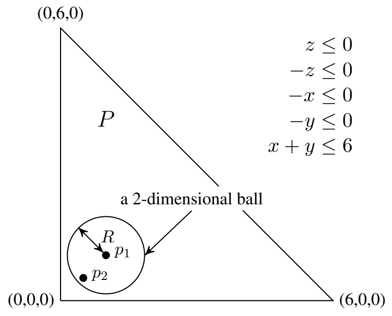


Figure 4: Our goal is to find two points p_1, p_2 in the relative interior of a polytope P defined by inequalities shown on the right. The inequalities $z \leq 0$ and $-z \leq 0$ are always active. Therefore, P is a 2-dimensional polytope. One solution to the LP (6) is $\alpha^* = 1, p^* = (1,1,0), \xi^{*\top} = [0,0,1,1,1]$, which results in $p_1 = (1,1,0)$ with $R = 1/\sqrt{2}$. The other point p_2 is chosen randomly from the ball of radius R.

$M_2) \in \mathbb{R}^{1 \times n_t}$ is nonzero. The probability of $wv = 0$ is zero because the space orthogonal to w is a $(n_t - 1)$-dimensional space while v is drawn from a n_t-dimensional space. Therefore, with probability 1, $xM_1 v \neq xM_2 v$. Without loss of generality, $a \leq xM_1 v < xM_2 v \leq b$, so $a \neq b$. □

3.2 Linear programming

Proposition 2 requires solving two non-trivial ILPs per input *at test time*. A natural step is to relax the integer constraint so that we solve two LPs instead.

$$\mathcal{C}_{\text{LP}} \overset{\text{def}}{=} \{M \in \mathbb{R}_{\geq 0}^{n_s \times n_t} \mid SM = T\} \quad (4)$$

$$\mathcal{F}_{\text{LP}} \overset{\text{def}}{=} \{x : |\{M(x) : M \in \mathcal{C}_{\text{LP}}\}| = 1\}. \quad (5)$$

The set of consistent mappings is larger ($\mathcal{C}_{\text{LP}} \supseteq \mathcal{C}$), so the set of safe inputs is smaller ($\mathcal{F}_{\text{LP}} \subseteq \mathcal{F}$). Therefore, if we predict only on $\mathcal{F}_{\text{LP}}$, we still maintain 100% precision, although the recall could be lower.

Now we will show how to exploit the convexity of $\mathcal{C}_{\text{LP}}$ (unlike $\mathcal{C}$) to avoid solving any LPs at test time at all. The basic idea is that if we choose two mappings $M_1, M_2 \in \mathcal{C}_{\text{LP}}$ "randomly enough", whether $xM_1 = xM_2$ is equivalent to unanimity over $\mathcal{C}_{\text{LP}}$. We could try to sample M_1, M_2 uniformly from $\mathcal{C}_{\text{LP}}$, but this is costly. We instead show that "less random" choice suffices. This is formalized as follows:

Proposition 3. *Let X be a finite set of test inputs. Let d be the dimension of $\mathcal{C}_{\text{LP}}$. Let M_1 be any mapping in $\mathcal{C}_{\text{LP}}$, and let vec(M_2) be sampled from a proper density over a d-dimensional ball lying in $\mathcal{C}_{\text{LP}}$ centered at vec(M_1). Then, with probability 1, for all $x \in X$, $xM_1 = xM_2$ implies $x \in \mathcal{F}_{\text{LP}}$.*

Proof. We will prove the contrapositive. If $x \notin \mathcal{F}_{\text{LP}}$, then xM is not the same for all $M \in \mathcal{C}_{\text{LP}}$. Without loss of generality, assume not all $M \in \mathcal{C}_{\text{LP}}$ agree on the i-th component of xM. Note that $(xM)_i = \text{tr}(Me_i x)$, which is the inner product of vec(M) and vec$(e_i x)$. Since $(xM)_i$ is not the same for all $M \in \mathcal{C}_{\text{LP}}$ and $\mathcal{C}_{\text{LP}}$ is convex, the projection of $\mathcal{C}_{\text{LP}}$ onto vec$(e_i x)$ must be a one-dimensional polytope. For both vec(M_1) and vec(M_2) to have the same projection on vec$(e_i x)$, they would have to both lie in a $(d-1)$-dimensional polytope orthogonal to vec$(e_i x)$. Since vec(M_2) is sampled from a proper density over a d-dimensional ball, this has probability 0. □

Algorithm. We now provide an algorithm to find two points p_1, p_2 inside a general d-dimensional polytope $P = \{p : Ap \leq b\}$ satisfying the conditions of Proposition 3, where for clarity we have simplified the notation from vec(M_i) to p_i and $\mathcal{C}_{\text{LP}}$ to P.

We first find a point p_1 in the relative interior of P, which consists of points for which the fewest number of inequalities j are active (i.e., $a_j p = b_j$). We can achieve this by solving the following LP from Freund et al. (1985):

$$\max \mathbf{1}^\top \xi \text{ s.t. } Ap + \xi \leq \alpha b, 0 \leq \xi \leq \mathbf{1}, \alpha \geq 1. \quad (6)$$

Here, ξ_j is a lower bound on the slack of inequality j, and α scales up the polytope so that all the ξ_j that can be positive are exactly 1 in the optimum solution. Importantly, if $\xi_j = 0$, constraint j is *always active* for all solutions $p \in P$. Let (p^*, ξ^*, α^*) be an optimal solution to the LP. Then define A_1 as the submatrix of A containing rows j for which $\xi_j^* = 1$, and A_0 consist of the remaining rows for which $\xi_j^* = 0$.

The above LP gives us $p_1 = p^*/\alpha^*$, which lies in the relative interior of P (see Figure 4). To obtain p_2, define a radius $R \overset{\text{def}}{=} (\alpha \max_{j:\xi_j^*=1} \|a_j\|_2)^{-1}$. Let the columns of matrix N form an orthonormal basis of the null space of A_0. Sample v from a unit d-dimensional ball centered at 0, and set $p_2 = p_1 + RNv$.

To show that $p_2 \in P$: First, p_2 satisfies the always-active constraints j, $a_j^\top(p_1 + RNv) = b_j$,

Algorithm 1 Our linear programming approach.

```
procedure TRAIN
Input: Training examples
Output: Generic mappings $(M_1, M_2)$
    Define $\mathcal{C}_{\text{LP}}$ as explained in (4).
    Compute $M_1$ and a radius $R$ by solving an LP (6).
    Sample $M_2$ from a ball with radius $R$ around $M_1$.
    return $(M_1, M_2)$
end procedure

procedure TEST
Input: input $x$, mappings $(M_1, M_2)$
Output: A guaranteed correct $y$ or "don't know"
    Compute $y_1 = xM_1$ and $y_2 = xM_2$.
    if $y_1 = y_2$ then return $y_1$
    else return "don't know"
    end if
end procedure
```

by definition of null space. For non-active j, the LP ensures that $a_j^\top p_1 + \alpha^{-1} \le b_j$, which implies $a_j^\top (p_1 + RNv) \le b_j$.

Algorithm 1 summarizes our overall procedure: At training time, we solve a single LP (6) and draw a random vector to obtain M_1, M_2 satisfying Proposition 3. At test time, we simply apply M_1 and M_2, which scales only linearly with the number of source atoms in the input.

3.3 Linear system

To obtain additional intuition about the unanimity principle, let us relax $\mathcal{C}_{\text{LP}}$ (4) further by removing the non-negativity constraint, which results in a linear system. Define the relaxed set of consistent mappings to be all the solutions to the linear system and the relaxed safe set accordingly:

$$\mathcal{C}_{\text{LS}} \overset{\text{def}}{=} \{M \in \mathbb{R}^{n_s \times n_t} \mid SM = T\} \tag{7}$$

$$\mathcal{F}_{\text{LS}} \overset{\text{def}}{=} \{x : |\{M(x) : M \in \mathcal{C}_{\text{LS}}\}| = 1\}. \tag{8}$$

Note that $\mathcal{C}_{\text{LS}}$ is an affine subspace, so each $M \in \mathcal{C}_{\text{LS}}$ can be expressed as $M_0 + BA$, where M_0 is an arbitrary solution, B is a basis for the null space of S and A is an arbitrary matrix. Figure 5 presents the linear system for four training examples. In the rare case that S has full column rank (if we have many training examples), then the left inverse of S exists, and there is exactly one consistent mapping, the true one ($M^* = S^\dagger T$), but we do not require this.

Let's try to explore the linear algebraic structure in the problem. Intuitively, if we know *area of Ohio* maps to `area(OH)` and *Ohio* maps to `OH`, then we should conclude *area of* maps to `area` by subtracting the second example from the first. The following proposition formalizes and generalizes this intuition by characterizing the relaxed safe set:

Proposition 4. *The vector x is in row space of S iff $x \in \mathcal{F}_{\text{LS}}$.*

$$\overbrace{\begin{array}{l|cccccc|l} & \textit{area} & \textit{of} & \textit{Ohio} & \textit{cities} & \textit{in} & \textit{Iowa} & \\ \textit{area of Iowa} & 1 & 1 & 0 & 0 & 0 & 1 & +1 \\ \textit{cities in Ohio} & 0 & 0 & 1 & 1 & 1 & 0 & +1 \\ \textit{cities in Iowa} & 0 & 0 & 0 & 1 & 1 & 1 & -1 \\ \hline \textit{area of Ohio} & 1 & 1 & 1 & 0 & 0 & 0 & \end{array}}^{S} \qquad \overbrace{\begin{array}{l|cccc|l} & \texttt{area} & \texttt{city} & \texttt{OH} & \texttt{IA} & \\ \texttt{area(IA)} & 1 & 0 & 0 & 1 & +1 \\ \texttt{city(OH)} & 0 & 1 & 1 & 0 & +1 \\ \texttt{city(IA)} & 0 & 1 & 0 & 1 & -1 \\ \hline \texttt{area(OH)} & 1 & 0 & 1 & 0 & \end{array}}^{T}$$

Figure 6: Under the linear system relaxation, we can predict the target atoms for the new input *area of Ohio* by adding and subtracting training examples (rows of S and T).

Proof. If x is in the row space of S, we can write x as a linear combination of S for some coefficients $\alpha \in \mathbb{R}^n$: $x = \alpha^\top S$. Then for all $M \in \mathcal{C}_{\text{LS}}$, we have $SM = T$, so $xM = \alpha^\top SM = \alpha^\top T$, which is the unique output[3] (See Figure 6). If $x \in \mathcal{F}_{\text{LS}}$ is safe, then there exists a y such that for all $M \in \mathcal{C}_{\text{LS}}$, $xM = y$. Recall that each element of $\mathcal{C}_{\text{LS}}$ can be decomposed into $M_0 + BA$. For $x(M_0 + BA)$ to be the same for each A, x should be orthogonal to each column of B, a basis for the null space of S. This means that x is in the row space of S. $\square$

Intuitively, this proposition says that stitching new inputs together by adding and subtracting existing training examples (rows of S) gives you exactly the relaxed safe set $\mathcal{F}_{\text{LS}}$.

Note that relaxations increases the set of consistent mappings ($\mathcal{C}_{\text{LS}} \supseteq \mathcal{C}_{\text{LP}} \supseteq \mathcal{C}$), which has the contravariant effect of shrinking the safe set ($\mathcal{F}_{\text{LS}} \subseteq \mathcal{F}_{\text{LP}} \subseteq \mathcal{F}$). Therefore, using the relaxation (predicting when $x \in \mathcal{F}_{\text{LS}}$) still preserves 100% precision.

3.4 Handling noise

So far, we have assumed that our training examples are noiseless, so that we can directly add the

[3]There might be more than one set of coefficients (α_1, α_2) for writing x. However, they result to a same output: $\alpha_1^\top S = \alpha_2^\top S \implies \alpha_1^\top SM = \alpha_2^\top SM \implies \alpha_1^\top T = \alpha_2^\top T$.

$$\overbrace{\begin{array}{cccccc} \textit{area} & \textit{of} & \textit{Ohio} & \textit{cities} & \textit{in} & \textit{Iowa} \\ \hline 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{array}}^{S} \times M = \overbrace{\begin{array}{cccc} \texttt{area} & \texttt{city} & \texttt{OH} & \texttt{IA} \\ \hline 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{array}}^{T} \implies M = \overbrace{\begin{array}{c|cccc} & \texttt{area} & \texttt{city} & \texttt{OH} & \texttt{IA} \\ \hline \textit{area} & 1 & 0 & 0 & 0 \\ \textit{of} & 0 & 0 & 0 & 0 \\ \textit{Ohio} & 0 & 0 & 1 & 0 \\ \textit{cities} & 0 & 1 & 0 & 0 \\ \textit{in} & 0 & 0 & 0 & 0 \\ \textit{Iowa} & 0 & 0 & 0 & 1 \end{array}}^{M_0} + \overbrace{\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & -1 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}^{B} \times \overbrace{\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \end{bmatrix}}^{A}$$

Figure 5: Under the linear system relaxation, all solutions M to $SM = T$ can be expressed as $M = M_0 + BA$, where B is the basis for the null space of S and A is arbitrary. Rows s of B which are zero (*Ohio* and *Iowa*) correspond to the safe source atoms (though not the only safe inputs).

constraint $SM = T$. Now assume that an adversary has made at most n_{mistakes} additions to and deletions of target atoms across the examples in T, but of course we do not know which examples have been tainted. Can we still guarantee 100% precision?

The answer is yes for the ILP formulation: we simply replace the exact match condition ($SM = T$) with a weaker one: $\|SM - T\|_1 \leq n_{\text{mistakes}}$ (*). The result is still an ILP, so the techniques from Section 3.1 readily apply. Note that as n_{mistakes} increases, the set of candidate mappings grows, which means that the safe set shrinks.

Unfortunately, this procedure is degenerate for linear programs. If the constraint (*) is not tight, then $M + E$ also satisfies the constraint for any matrix E of small enough norm. This means that the consistent mappings $\mathcal{C}_{\text{LP}}$ will be full-dimensional and certainly not be unanimous on any input.

Another strategy is to remove examples from the dataset if they could be potentially noisy. For each training example i, we run the ILP (*) on all but the i-th example. If the i-th example is not in the resulting safe set (2), we remove it. This procedure produces a noiseless dataset, on which we can apply the noiseless linear program or linear system from the previous sections.

4 Experiments

4.1 Artificial data

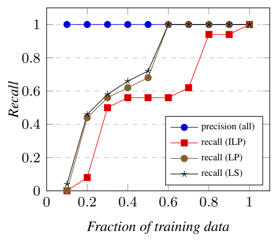
We generated a true mapping M^* from 50 source atoms to 20 target atoms so that each source atom maps to 0–2 target atoms. We then created 120 training examples and 50 test examples, where the length of every input is between 5 and 10. The source atoms are divided into 10 clusters, and each input only contains source atoms from one cluster.

Figure 7a shows the results for $\mathcal{F}$ (integer linear programming), $\mathcal{F}_{\text{LP}}$ (linear programming), and $\mathcal{F}_{\text{LS}}$ (linear system). All methods attain 100% precision, and as expected, relaxations lead to lower recall, though they all can reach 100% recall given enough data.

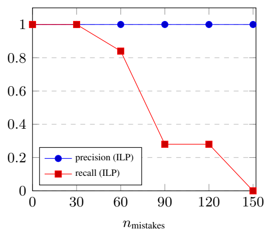
Comparison with point estimation. Recall that the unanimity principle $\hat{M}$ reasons over the entire set of consistent mappings, which allows us to be robust to changes in the input distribution, e.g., from training set attacks (Mei and Zhu, 2015). As an alternative, consider computing the point estimate M_p that minimizes $\|SM - T\|_2^2$ (the solution is given by $M_p = S^{\dagger}T$). The point estimate, by minimizing the average loss, implicitly assumes i.i.d. examples. To generate output for input x we compute $y = xM_p$ and round each coordinate y_t to the closest integer. To obtain a precision-recall tradeoff, we set a threshold ϵ and if for all target atoms t, the interval $[y_t - \epsilon, y_t + \epsilon)$ contains an integer, we set y_t to that integer; otherwise we report "don't know" for input x.

To compare unanimous prediction $\hat{M}$ and point estimation M_p, for each $f \in \{0.2, 0.5, 0.7\}$, we randomly generate 100 subsampled datasets consisting of an f fraction of the training examples. For M_p, we sweep ϵ across $\{0.0, 0.1, \ldots, 0.5\}$ to obtain a ROC curve. In Figure 7c(left/right), we select the distribution that results in the maximum/minimum difference between $F_1(\hat{M})$ and $F_1(M_p)$ respectively. As shown, $\hat{M}$ has always 100% precision, while M_p can obtain less 100% precision over its full ROC curve. An adversary can only hurt the recall of unanimous prediction.

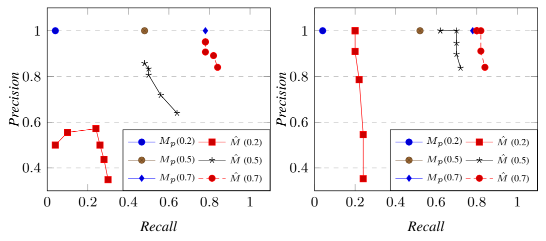
Noise. As stated in Section 3.4, our algorithm has the ability to guarantee 100% precision even when the adversary can modify the outputs. As we increase the number of predicate additions/deletions (n_{mistakes}), Figure 7b shows that precision remains at 100%, while recall naturally decreases in response to being less confident about



(a) All the relaxations reach 100% recall; relaxation results in slightly slower convergence.



(b) Size of the safe set shrinks with increasing number of mistakes in the training data.



(c) Performance of the point estimate (M_p) and unanimous prediction ($\hat{M}$) when the inputs are chosen adversarially for M_p (left) and for $\hat{M}$ (right).

Figure 7: Our algorithm always obtains 100% precision with (a) different amounts of training examples and different relaxations, (b) existence of noise, and (c) adversarial input distributions.

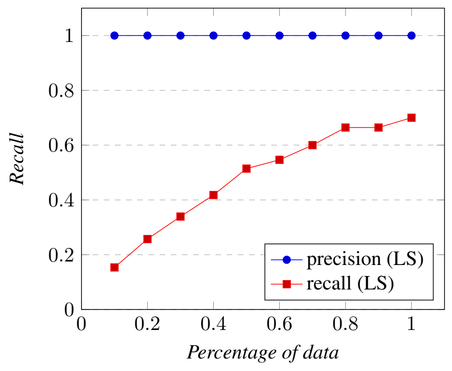


Figure 8: We maintain 100% precision while recall increases with the number of training examples.

the training outputs.

4.2 Semantic parsing on GeoQuery

We now evaluate our approach on the standard GeoQuery dataset (Zelle and Mooney, 1996), which contains 880 utterances and their corresponding logical forms. The utterances are questions related to the US geography, such as: "*what river runs through the most states*". We use the standard 600/280 train/test split (Zettlemoyer and Collins, 2005). After replacing entity names by their types[4] based on the standard entity lexicon, there are 172 different words and 57 different predicates in this dataset.

Handling context. Some words are polysemous in that they map to two predicates: in "*largest river*" and "*largest city*", the word *largest* maps to `longest` and `biggest`, respectively. Therefore, instead of using words as source atoms, we use bigrams, so that each source atom always maps to the same target atoms.

Reconstructing the logical form. We define target atoms to include more information than just the predicates, which enables us to reconstruct logical forms from the predicates. We use the variable-free functional logical forms (Kate et al., 2005), in which each target atom is a predicate conjoined with its argument order (e.g., `loc_1` or `loc_2`). Table 1 shows two different choices of target atoms. At test time, we search over all possible "compatible" ways of combining target atoms into logical forms. If there is exactly one, then we return that logical form and abstain otherwise. We call a predicate combination "compatible" if it appears in the training set.

We put a "null" word at the end of each sentence, and collapsed the `loc` and `traverse` predicates. To deal with noise, we minimized $\|SM - T\|_1$ over real-valued mappings and removed any example (row) with non-zero residual. We perform all experiments using the linear system relaxation. Training takes under 30 seconds.

Figure 8 shows precision and recall as a function of the number of the training examples. We obtain 70% recall over predicates on the test examples. 84% of these have a unique compatible way of combining target atoms into a logical form, which results in a 59% recall on logical forms.

Though our modeling assumptions are incorrect for real data, we were still able to get 100% precision for all training examples. Interestingly, the linear system (which allows negative mappings) helps model GeoQuery dataset better than the linear program (which has a non-negativity constraint). There exists a predicate `all:e` in GeoQuery that is in every sentence unless the ut-

[4]If an entity name has more than one type we replace it by concatenating all of its possible types.

utterances	**logical form (A)**	**target atoms (A)**	**logical form (B)**	**target atoms (B)**
cities traversed by the Columbia	`city(x),loc(x,Columbia)`	`city,loc,Columbia`	`city(loc_1(Columbia))`	`city,loc_1,Columbia`
cities of Texas	`city(x),loc(Texas,x)`	`city,loc,Texas`	`city(loc_2(Texas))`	`city,loc_2,Texas`

Table 1: Two different choices of target atoms: (A) shows predicates and (B) shows predicates conjoined with their argument position. (A) is sufficient for simply recovering the predicates, whereas (B) allows for logical form reconstruction.

terance contains a proper noun. With negative mappings, *null* maps to `all:e`, while each proper noun maps to its proper predicate *minus* `all:e`.

There is a lot of work in semantic parsing that tackles the GeoQuery dataset (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Kwiatkowski et al., 2010; Liang et al., 2011), and the state-of-the-art is 91.1% precision and recall (Liang et al., 2011). However, none of these methods can guarantee 100% precision, and they perform more feature engineering, so these numbers are not quite comparable. In practice, one could use our unanimous prediction approach in conjunction with others: For example, one could run a classic semantic parser and simply certify 59% of the examples to be correct with our approach. In critical applications, one could use our approach as a first-pass filter, and fall back to humans for the abstentions.

5 Extensions

5.1 Learning from denotations

Up until now, we have assumed that we have input-output pairs. For semantic parsing, this means annotating sentences with logical forms (e.g., *area of Ohio* to `area(OH)`) which is very expensive. This has motivated previous work to learn from question-answer pairs (e.g., *area of Ohio* to `44825`) (Liang et al., 2011). This provides weaker supervision: For example, `44825` is the area of Ohio (in squared miles), but it is also the zip code of Chatfield. So, the true output could be either `area(OH)` or `zipcode(Chatfield)`.

In this section, we show how to handle this form of weak supervision by asking for unanimity over additional selection variables. Formally, we have $D = \{(x_1, Y_1), \dots, (x_n, Y_n)\}$ as a set of training examples, here each Y_i consists of k_i candidate outputs for x_i. In this case, the unknowns are the mapping M as before along with a selection vector π_i, which specifies which of the k_i outputs in Y_i is equal to $x_i M$. To implement the unanimity principle, we need to consider the set of all consistent solutions (M, π).

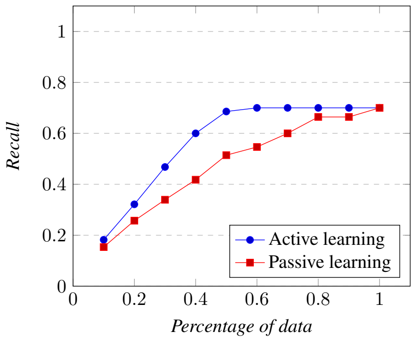


Figure 9: When we choose examples to be linearly independent, we only need half the number of examples to achieve the same performance.

We construct an integer linear program as follows: Each training example adds a constraint that the output of it should be exactly one of its candidate output. For the i-th example, we form a matrix $T_i \in \mathbb{R}^{k_i \times n_t}$ with all the k_i candidate outputs. Formally we want $x_i M = \pi_i T_i$. The entire ILP is:

$$\begin{aligned} &\forall i,\ x_i M = \pi_i T_i \\ &\forall i,\ \textstyle\sum_j \pi_{ij} = 1 \\ &\pi, M \ge 0 \end{aligned}$$

Given a new input x, we return the same output if xM is same for all consistent solutions (M, π). Note that we can effectively "marginalize out" π. We can also relax this ILP into an linear program following Section 3.2.

5.2 Active learning

A side benefit of the linear system relaxation (Section 3.3) is that it suggests an active learning procedure. The setting is that we are given a set of inputs (the matrix S), and we want to (adaptively) choose which inputs (rows of S) to obtain the output (corresponding row of T) for.

Proposition 4 states that under the linear system formulation, the set of safe inputs $\mathcal{F}_{\text{LS}}$ is exactly the same as the row space of S. Therefore, if we ask for an input that is already in the row space of S, this will not affect $\mathcal{F}_{\text{LS}}$ at all. The algorithm is then simple: go through our training inputs $x_1, \dots, x_n$ one by one and ask for the output only if it is not in the row space of the previously-added inputs $x_1, \dots, x_{i-1}$.

Figure 9 shows the recall when we choose examples to be linearly independent in this way in comparison to when we choose examples randomly. The active learning scheme requires half as many labeled examples as the passive scheme to reach the same recall. In general, it takes $\text{rank}(S) \leq n$ examples to obtain the same recall as having labeled all n examples. Of course, the precision of both systems is 100%.

5.3 Paraphrasing

Another side benefit of the linear system relaxation (Section 3.3) is that we can easily partition the safe set $\mathcal{F}_{\text{LS}}$ (8) into subsets of utterances which are paraphrases of each other. Two utterances are paraphrase of each other if both map to the same logical form, e.g., "*Texas's capital*" and "*capital of Texas*". Given a sentence $x \in \mathcal{F}_{\text{LS}}$, our goal is to find all of its paraphrases in $\mathcal{F}_{\text{LS}}$.

As explained in Section 3.3, we can represent each input x as a linear combination of S for some coefficients $\alpha \in \mathbb{R}^n$: $x = \alpha^\top S$. We want to find all $x' \in \mathcal{F}_{\text{LS}}$ such that x' is guaranteed to map to the same output as x. We can represent $x' = \beta^\top S$ for some coefficients $\beta \in \mathbb{R}^n$. The outputs for x and x' are thus $\alpha^\top T$ and $\beta^\top T$, respectively. Thus we are interested in β's such that $\alpha^\top T = \beta^\top T$, or in other words, $\alpha - \beta$ is in the null space of $T^\top$. Let B be a basis for the null space of $T^\top$. We can then write $\alpha - \beta = Bv$ for some v. Therefore, the set of paraphrases of $x \in \mathcal{F}_{\text{LS}}$ are:

$$\text{Paraphrases}(x) \stackrel{\text{def}}{=} \{(\alpha - Bv)^\top S : v \in \mathbb{R}^n\}. \tag{9}$$

6 Discussion and related work

Our work is motivated by the semantic parsing task (though it can be applied to any set-to-set prediction task). Over the last decade, there has been much work on semantic parsing, mostly focusing on learning from weaker supervision (Liang et al., 2011; Goldwasser et al., 2011; Artzi and Zettlemoyer, 2011; Artzi and Zettlemoyer, 2013), scaling up beyond small databases (Cai and Yates, 2013; Berant et al., 2013; Pasupat and Liang, 2015), and applying semantic parsing to other tasks (Matuszek et al., 2012; Kushman and Barzilay, 2013; Artzi and Zettlemoyer, 2013). However, only Popescu et al. (2003) focuses on precision. They also obtain 100% precision, but with a hand-crafted system, whereas we *learn* a semantic mapping.

The idea of computing consistent hypotheses appears in the classic theory of version spaces for binary classification (Mitchell, 1977) and has been extended to more structured settings (Vanlehn and Ball, 1987; Lau et al., 2000). Our version space is used in the context of the unanimity principle, and we explore a novel linear algebraic structure. Our "safe set" of inputs appears in the literature as the complement of the disagreement region (Hanneke, 2007). They use this notion for active learning, whereas we use it to support unanimous prediction.

There is classic work on learning classifiers that can abstain (Chow, 1970; Tortorella, 2000; Balsubramani, 2016). This work, however, focuses on the classification setting, whereas we considered more structured output settings (e.g., for semantic parsing). Another difference is that we operate in a more adversarial setting by leaning on the unanimity principle.

Another avenue for providing user confidence is probabilistic calibration (Platt, 1999), which has been explored more recently for structured prediction (Kuleshov and Liang, 2015). However, these methods do not guarantee precision for *any* training set and test input.

In summary, we have presented the unanimity principle for guaranteeing 100% precision. For the task of learning semantic mappings, we leveraged the linear algebraic structure in our problem to make unanimous prediction efficient. We view our work as a first step in learning reliable semantic parsers. A natural next step is to explore our framework with additional modeling improvements—especially in dealing with context, structure, and noise.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x593676a278fc4e5abe2d8bac1e3df486/`.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments. We are also grateful for a Future Of Life Research Award and NSF grant CCF-1138967, which supported this work.

References

Y. Artzi and L. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 421–432.

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

A. Balsubramani. 2016. Learning to abstain from binary prediction. *arXiv preprint arXiv:1602.08151.*

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

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL).*

C. K. Chow. 1970. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46.

R. M. Freund, R. Roundy, and M. J. Todd. 1985. Identifying the set of always-active constraints in a system of linear inequalities by a single linear program. Technical report, Massachusetts Institute of Technology, Alfred P. Sloan School of Management.

D. Goldwasser, R. Reichart, J. Clarke, and D. Roth. 2011. Confidence driven unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*, pages 1486–1495.

S. Hanneke. 2007. A bound on the label complexity of agnostic active learning. In *International Conference on Machine Learning (ICML)*, pages 353–360.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1062–1068.

V. Kuleshov and P. Liang. 2015. Calibrated structured prediction. In *Advances in Neural Information Processing Systems (NIPS).*

N. Kushman and R. Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 826–836.

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

T. A. Lau, P. Domingos, and D. S. Weld. 2000. Version space algebra and its application to programming by demonstration. In *International Conference on Machine Learning (ICML)*, pages 527–534.

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

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

S. Mei and X. Zhu. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *Association for the Advancement of Artificial Intelligence (AAAI).*

T. M. Mitchell. 1977. Version spaces: A candidate elimination approach to rule learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 305–310.

B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. Tygar, and K. Xia. 2009. Misleading learners: Co-opting your spam filter. In *Machine learning in cyber trust*, pages 17–51.

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

J. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74.

A. Popescu, O. Etzioni, and H. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *International Conference on Intelligent User Interfaces (IUI)*, pages 149–157.

H. Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

F. Tortorella. 2000. An optimal reject rule for binary classifiers. In *Advances in Pattern Recognition*, pages 611–620.

K. Vanlehn and W. Ball. 1987. A version space approach to learning context-free grammars. *Machine learning*, 2(1):39–74.

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

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

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