Simpler Context-Dependent Logical Forms via Model Projections

Reginald Long
Stanford University
reggylong@cs.stanford.edu

Panupong Pasupat
Stanford University
ppasupat@cs.stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

We consider the task of learning a context-dependent mapping from utterances to denotations. With only denotations at training time, we must search over a combinatorially large space of logical forms, which is even larger with context-dependent utterances. To cope with this challenge, we perform successive projections of the full model onto simpler models that operate over equivalence classes of logical forms. Though less expressive, we find that these simpler models are much faster and can be surprisingly effective. Moreover, they can be used to bootstrap the full model. Finally, we collected three new context-dependent semantic parsing datasets, and develop a new left-to-right parser.

1 Introduction

Suppose we are only told that a piece of text (a command) in some context (state of the world) has some denotation (the effect of the command)—see Figure 1 for an example. How can we build a system to learn from examples like these with no initial knowledge about what any of the words mean?

We start with the classic paradigm of training semantic parsers that map utterances to logical forms, which are executed to produce the denotation (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Zettlemoyer and Collins, 2009; Kwiatkowski et al., 2010). More recent work learns directly from denotations (Clarke et al., 2010; Liang, 2013; Berant et al., 2013; Artzi and Zettlemoyer, 2013), but in this setting, a constant struggle is to contain the exponential explosion of possible logical forms. With no initial lexicon and longer context-dependent texts, our situation is exacerbated.

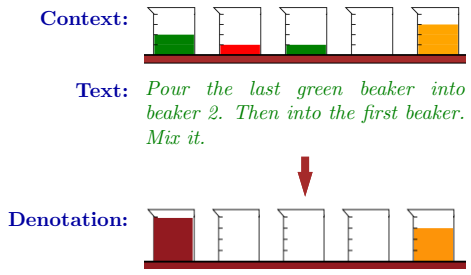


Figure 1: Our task is to learn to map a piece of text in some context to a denotation. An example from the ALCHEMY dataset is shown. In this paper, we ask: what intermediate logical form is suitable for modeling this mapping?

In this paper, we propose *projecting* a full semantic parsing model onto simpler models over equivalence classes of logical form derivations. As illustrated in Figure 2, we consider the following sequence of models:

- **Model A**: our full model that derives logical forms (e.g., in Figure 1, the last utterance maps to `mix(args[1][1])`) compositionally from the text so that spans of the utterance (e.g., "*it*") align to parts of the logical form (e.g., `args[1][1]`, which retrieves an argument from a previous logical form). This is based on standard semantic parsing (e.g., Zettlemoyer and Collins (2005)).
- **Model B**: collapse all derivations with the same logical form; we map utterances to full logical forms, but without an alignment between the utterance and logical forms. This "floating" approach was used in Pasupat and Liang (2015) and Wang et al. (2015).
- **Model C**: further collapse all logical forms whose top-level arguments have the same denotation. In other words, we map utterances

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1456–1465,
Berlin, Germany, August 7-12, 2016.

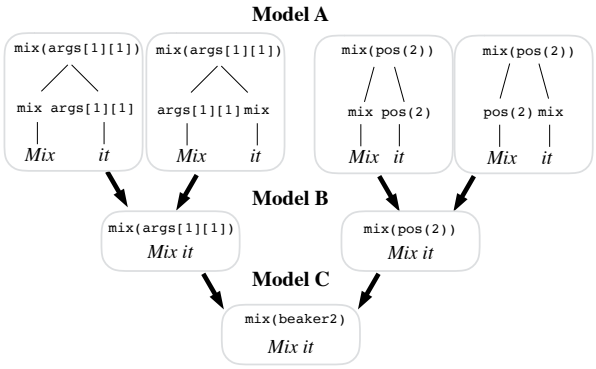


Figure 2: Derivations generated for the last utterance in Figure 1. All derivations above execute to `mix(beaker2)`. Model A generates anchored logical forms (derivations) where words are aligned to predicates, which leads to multiple derivations with the same logical form. Model B discards these alignments, and Model C collapses the arguments of the logical forms to denotations.

to flat logical forms (e.g., `mix(beaker2)`), where the arguments of the top-level predicate are objects in the world. This model is in the spirit of Yao et al. (2014) and Bordes et al. (2014), who directly predicted concrete paths in a knowledge graph for question answering.

Model A excels at credit assignment: the latent derivation explains how parts of the logical form are triggered by parts of the utterance. The price is an unmanageably large search space, given that we do not have a seed lexicon. At the other end, Model C only considers a small set of logical forms, but the mapping from text to the correct logical form is more complex and harder to model.

We collected three new context-dependent semantic parsing datasets using Amazon Mechanical Turk: ALCHEMY (Figure 1), SCENE (Figure 3), and TANGRAMS (Figure 4). Along the way, we develop a new parser which processes utterances left-to-right but can construct logical forms without an explicit alignment.

Our empirical findings are as follows: First, Model C is surprisingly effective, mostly surpassing the other two given bounded computational resources (a fixed beam size). Second, on a synthetic dataset, with infinite beam, Model A outperforms the other two models. Third, we can bootstrap up to Model A from the projected models with finite beam.

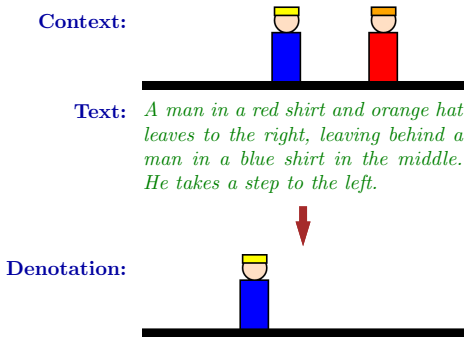


Figure 3: SCENE dataset: Each person has a shirt of some color and a hat of some color. They enter, leave, move around on a stage, and trade hats.

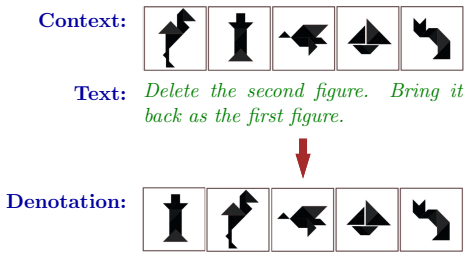


Figure 4: TANGRAMS dataset: One can add figures, remove figures, and swap the position of figures. All the figures slide to the left.

2 Task

In this section, we formalize the task and describe the new datasets we created for the task.

2.1 Setup

First, we will define the context-dependent semantic parsing task. Define w_0 as the initial world state, which consists of a set of entities (beakers in ALCHEMY) and properties (location, color(s), and amount filled). The text $\mathbf{x}$ is a sequence of utterances $x_1, \ldots, x_L$. For each utterance x_i (e.g., "*mix*"), we have a latent logical form z_i (e.g., `mix(args[1][2])`). Define the context $c_i = (w_0, z_{1:i-1})$ to include the initial world state w_0 and the history of past logical forms $z_{1:i-1}$. Each logical form z_i is executed on the context c_i to produce the next state: $w_i = \text{Exec}(c_i, z_i)$ for each $i = 1, \ldots, L$. Overloading notation, we write $w_L = \text{Exec}(w_0, \mathbf{z})$, where $\mathbf{z} = (z_1, \ldots, z_L)$.

The learning problem is: given a set of training examples $\{(w_0, \mathbf{x}, w_L)\}$, learn a mapping from the text $\mathbf{x}$ to logical forms $\mathbf{z} = (z_1, \ldots, z_L)$ that produces the correct final state ($w_L = \text{Exec}(w_0, \mathbf{z})$).

Dataset	# examples	# train	# test	words/example	utterances
SCENE	4402	3363	1039	56.2	"*then one more*", "*he moves back*"
ALCHEMY	4560	3661	899	39.9	"*mix*", "*throw the rest out*"
TANGRAMS	4989	4189	800	27.2	"*undo*", "*replace it*", "*take it away*"

Table 1: We collected three datasets. The number of examples, train/test split, number of tokens per example, along with interesting phenomena are shown for each dataset.

2.2 Datasets

We created three new context-dependent datasets, ALCHEMY, SCENE, and TANGRAMS (see Table 1 for a summary), which aim to capture a diverse set of context-dependent linguistic phenomena such as ellipsis (e.g., "*mix*" in ALCHEMY), anaphora on entities (e.g., "*he*" in SCENE), and anaphora on actions (e.g., "*repeat step 3*", "*bring it back*" in TANGRAMS).

For each dataset, we have a set of properties and actions. In ALCHEMY, properties are `color`, and `amount`; actions are `pour`, `drain`, and `mix`. In SCENE, properties are `hat-color` and `shirt-color`; actions are `enter`, `leave`, `move`, and `trade-hats`. In TANGRAMS, there is one property (`shape`), and actions are `add`, `remove`, and `swap`. In addition, we include the position property (`pos`) in each dataset. Each example has $L = 5$ utterances, each denoting some transformation of the world state.

Our datasets are unique in that they are grounded to a world state and have rich linguistic context-dependence. In the context-dependent ATIS dataset (Dahl et al., 1994) used by Zettlemoyer and Collins (2009), logical forms of utterances depend on previous logical forms, though there is no world state and the linguistic phenomena is limited to nominal references. In the map navigation dataset (Chen and Mooney, 2011), used by Artzi and Zettlemoyer (2013), utterances only reference the current world state. Vlachos and Clark (2014) released a corpus of annotated dialogues, which has interesting linguistic context-dependence, but there is no world state.

Data collection. Our strategy was to automatically generate sequences of world states and ask Amazon Mechanical Turk (AMT) workers to describe the successive transformations. Specifically, we started with a random world state w_0. For each $i = 1, \ldots, L$, we sample a valid action and argument (e.g., `pour(beaker1, beaker2)`). To encourage context-dependent descriptions, we upweight recently used actions and arguments (e.g., the next action is more like to be `drain(beaker2)` rather than `drain(beaker5)`). Next, we presented an AMT worker with states $w_0, \ldots, w_L$ and asked the worker to write a description in between each pair of successive states.

In initial experiments, we found it rather non-trivial to obtain interesting linguistic context-dependence in these micro-domains: often a context-independent utterance such as "*beaker 2*" is just clearer and not much longer than a possibly ambiguous "*it*". We modified the domains to encourage more context. For example, in SCENE, we removed any visual indication of absolute position and allowed people to only move next to other people. This way, workers would say "*to the left of the man in the red hat*" rather than "*to position 2*".

3 Model

We now describe Model A, our full context-dependent semantic parsing model. First, let $\mathcal{Z}$ denote the set of candidate logical forms (e.g., `pour(color(green),color(red))`). Each logical form consists of a top-level action with arguments, which are either primitive values (`green`, `3`, etc.), or composed via selection and superlative operations. See Table 2 for a full description. One notable feature of the logical forms is the context dependency: for example, given some context $(w_0, z_{1:4})$, the predicate `actions[2]` refers to the action of z_2 and `args[2][1]` refers to first argument of z_2.[1]

We use the term *anchored logical forms* (a.k.a. derivations) to refer to logical forms augmented with alignments between sub-logical forms of z_i and spans of the utterance x_i. In the example above, `color(green)` might align with "*green beaker*" from Figure 1; see Figure 2 for another example.

[1]These special predicates play the role of references in Zettlemoyer and Collins (2009). They perform context-independent parsing and resolve references, whereas we resolve them jointly while parsing.

Property[p] Value[v]	$\Rightarrow$ Set[$p(v)$]	all entities whose property p is v
Set[s] Property[p]	$\Rightarrow$ Value[argmin/argmax(s, p)]	element in s with smallest/largest p
Set[s] Int[i]	$\Rightarrow$ Value[$s[i]$]	i-th element of s
Action[a] Value[v_1] Value[v_2]	$\Rightarrow$ Root[$a(v_1, v_2)$]	top-level action applied to arguments v_1, v_2

Table 2: Grammar that defines the space of candidate logical forms. Values include numbers, colors, as well as special tokens `args`$[i][j]$ (for all $i \in \{1, \ldots, L\}$ and $j \in \{1, 2\}$) that refer to the j-th argument used in the i-th logical form. Actions include the fixed domain-specific set plus special tokens `actions`$[i]$ (for all $i \in \{1, \ldots, L\}$), which refers to the i-th action in the context.

	Derivation condition	Example
(F1)	z_i contains predicate r	(z_i contains predicate `pour`, "*pour*")
(F2)	property p of $z_i.b_j$ is y	(`color` of arg 1 is `green`, "*green*")
(F3)	action $z_i.a$ is a and property p of $z_i.y_j$ is y	(action is `pour` and `pos` of arg 2 is `2`, "*pour, 2*")
(F4)	properties p of $z_i.v_1$ is y and p' of $z_i.v_2$ is y'	(`color` of arg 1 is `green` and `pos` of arg 2 is `2`, "*first green, 2*")
(F5)	arg $z_i.v_j$ is one of z_{i-1}'s args	(arg reused, "*it*")
(F6)	action $z_i.a = z_{i-1}.a$	(action reused, "*pour*")
(F7)	properties p of $z_i.y_j$ is y and p' of $z_{i-1}.y_k$ is y'	(`pos` of arg 1 is `2` and `pos` of prev. arg 2 is `2`, "*then*")
(F8)	$t_1 < s_2$	spans don't overlap

Table 3: Features $\phi(x_i, c_i, z_i)$ for Model A: The left hand side describes conditions under which the system fires indicator features, and right hand side shows sample features for each condition. For each derivation condition (F1)–(F7), we conjoin the condition with the span of the utterance that the referenced actions and arguments align to. For condition (F8), we just fire the indicator by itself.

Log-linear model. We place a conditional distribution over anchored logical forms $z_i \in \mathcal{Z}$ given an utterance x_i and context $c_i = (w_0, z_{1:i-1})$, which consists of the initial world state w_0 and the history of past logical forms $z_{1:i-1}$. We use a standard log-linear model:

$$p_\theta(z_i \mid x_i, c_i) \propto \exp(\phi(x_i, c_i, z_i) \cdot \theta), \quad (1)$$

where ϕ is the feature mapping and θ is the parameter vector (to be learned). Chaining these distributions together, we get a distribution over a sequence of logical forms $\mathbf{z} = (z_1, \ldots, z_L)$ given the whole text $\mathbf{x}$:

$$p_\theta(\mathbf{z} \mid \mathbf{x}, w_0) = \prod_{i=1}^{L} p_\theta(z_i \mid x_i, (w_0, z_{1:i-1})). \quad (2)$$

Features. Our feature mapping ϕ consists of two types of indicators:

1. For each derivation, we fire features based on the structure of the logical form/spans.

2. For each span s (e.g., "*green beaker*") aligned to a sub-logical form z (e.g., `color(green)`), we fire features on unigrams, bigrams, and trigrams inside s conjoined with various conditions of z.

The exact features given in Table 3, references the first two utterances of Figure 1 and the associated logical forms below:

$x_1 =$ "*Pour the last green beaker into beaker 2.*"

$z_1 =$ `pour(argmin(color(green),pos),pos(2))`

$x_2 =$ "*Then into the first beaker.*"

$z_2 =$ `actions[1](args[1][2],pos(3))`.

We describe the notation we use for Table 3, restricting our discussion to actions that have two or fewer arguments. Our featurization scheme, however, generalizes to an arbitrary number of arguments. Given a logical form z_i, let $z_i.a$ be its action and $(z_i.b_1, z_i.b_2)$ be its arguments (e.g., `color(green)`). The first and second arguments are anchored over spans $[s_1, t_1]$ and $[s_2, t_2]$, respectively. Each argument $z_i.b_j$ has a corresponding value $z_i.v_j$ (e.g., `beaker1`), obtained by executing $z_i.b_j$ on the context c_i. Finally, let $j, k \in \{1, 2\}$ be indices of the arguments. For example, we would label the constituent parts of z_1 (defined above) as follows:

- $z_1.a =$ `pour`
- $z_1.b_1 =$ `argmin(color(green),pos)`
- $z_1.v_1 =$ `beaker3`
- $z_1.b_2 =$ `pos(2)`
- $z_1.v_2 =$ `beaker2`

4 Left-to-right parsing

We describe a new parser suitable for learning from denotations in the context-dependent setting. Like a shift-reduce parser, we proceed left to right, but each *shift* operation advances an entire utterance rather than one word. We then sit on the utterance for a while, performing a sequence of *build* operations, which either combine two logical forms on the stack (like the reduce operation) or generate fresh logical forms, similar to what is done in the floating parser of Pasupat and Liang (2015).

Our parser has two desirable properties: First, proceeding left-to-right allows us to build and score logical forms z_i that depend on the world state w_{i-1}, which is a function of the previous logical forms. Note that w_{i-1} is a random variable in our setting, whereas it is fixed in Zettlemoyer and Collins (2009). Second, the *build* operation allows us the flexibility to handle ellipsis (e.g., "*Mix.*") and anaphora on full logical forms (e.g., "*Do it again.*"), where there's not a clear alignment between the words and the predicates generated.

The parser transitions through a sequence of hypotheses. Each hypothesis is $h = (i, b, \sigma)$, where i is the index of the current utterance, where b is the number of predicates constructed on utterance x_i, and σ is a stack (list) of logical forms. The stack includes both the previous logical forms $z_{1:i-1}$ and fragments of logical forms built on the current utterance. When processing a particular hypothesis, the parser can choose to perform either the shift or build operation:

Shift: The parser moves to the next utterance by incrementing the utterance index i and resetting b, which transitions a hypothesis from (i, b, σ) to $(i + 1, 0, \sigma)$.

Build: The parser creates a new logical form by combining zero or more logical forms on the stack. There are four types of build operations:

1. Create a predicate out of thin air (e.g., `args[1][1]` in Figure 5). This is useful when the utterance does not explicitly reference the arguments or action. For example, in Figure 5, we are able to generate the logical form `args[1][1]` in the presence of ellipsis.

2. Create a predicate anchored to some span of the utterance (e.g., `actions[1]` anchored to "*Repeat*"). This allows us to do credit assignment and capture which part of the utterance explains which part of the logical form.

3. Pop z from the stack σ and push z' onto σ, where z' is created by applying a rule in Table 2 to z.

4. Pop z, z' from the stack σ and push z'' onto σ, where z'' is created by applying a rule in Table 2 to z, z' (e.g., `actions[1](args[1][1])` by the top-level root rule).

The build step stops once a maximum number of predicates B have been constructed or when the top-level rule is applied.

We have so far described the search space over logical forms. In practice, we keep a beam of the K hypotheses with the highest score under the current log-linear model.

5 Model Projections

Model A is ambitious, as it tries to learn from scratch how each word aligns to part of the logical form. For example, when Model A parses "*Mix it*", one derivation will correctly align "*Mix*" to `mix`, but others will align "*Mix*" to `args[1][1]`, "*Mix*" to `pos(2)`, and so on (Figure 2).

As we do not assume a seed lexicon that could map "*Mix*" to `mix`, the set of anchored logical forms is exponentially large. For example, parsing just the first sentence of Figure 1 would generate 1,216,140 intermediate anchored logical forms.

How can we reduce the search space? The key is that the space of logical forms is *much smaller* than the space of anchored logical forms. Even though both grow exponentially, dealing directly with logical forms allows us to generate `pour` without the combinatorial choice over alignments. We thus define Model B over the space of these logical forms. Figure 2 shows that the two anchored logical forms, which are treated differently in Model A are collapsed in Model B. This dramatically reduces the search space; parsing the first sentence of Figure 1 generates 7,047 intermediate logical forms.

We can go further and notice that many compositional logical forms reduce to the same flat logical form if we evaluate all the arguments. For example, in Figure 2, `mix(args[1][1])` and `mix(pos(2))` are equivalent to `mix(beaker2)`. We define Model C to be the

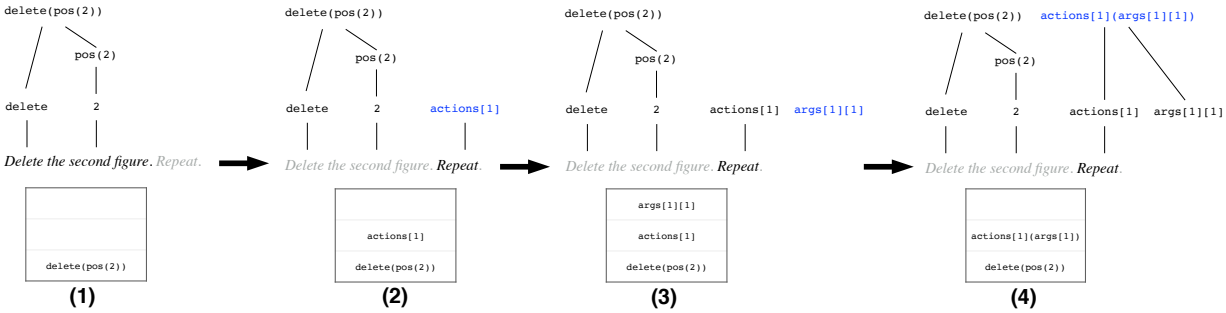


Figure 5: Suppose we have already constructed `delete(pos(2))` for "*Delete the second figure.*" Continuing, we shift the utterance "*Repeat*". Then, we build `action[1]` aligned to the word "*Repeat.*" followed by `args[1][1]`, which is unaligned. Finally, we combine the two logical forms.

space of these flat logical forms which consist of a top-level action plus primitive arguments. Using Model C, parsing the first sentence of Figure 1 generates only 349 intermediate logical forms.

Note that in the context-dependent setting, the number of flat logical forms (Model C) still increases exponentially with the number of utterances, but it is an overwhelming improvement over Model A. Furthermore, unlike other forms of relaxation, we are still generating logical forms that can express any denotation as before. The gains from Model B to Model C hinge on the fact that in our world, the number of denotations is much smaller than the number of logical forms.

Projecting the features. While we have defined the space over logical forms for Models B and C, we still need to define a distribution over these spaces to to complete the picture. To do this, we propose projecting the features of the log-linear model (1). Define $\Pi_{\text{A}\to\text{B}}$ to be a map from a anchored logical form z^{A} (e.g., `mix(pos(2))` aligned to "*mix*") to an unanchored one z^{B} (e.g., `mix(pos(2))`), and define $\Pi_{\text{B}\to\text{C}}$ to be a map from z^{B} to the flat logical form z^{C} (e.g., `mix(beaker2)`).

We construct a log-linear model for Model B by constructing features $\phi(z^{\text{B}})$ (omitting the dependence on x_i, c_i for convenience) based on the Model A features $\phi(z^{\text{A}})$. Specifically, $\phi(z^{\text{B}})$ is the component-wise maximum of $\phi(z^{\text{A}})$ over all z^{A} that project down to z^{B}; $\phi(z^{\text{C}})$ is defined similarly:

$$\phi(z^{\text{B}}) \stackrel{\text{def}}{=} \max\{\phi(z^{\text{A}}) : \Pi_{\text{A}\to\text{B}}(z^{\text{A}}) = z^{\text{B}}\}, \quad (3)$$

$$\phi(z^{\text{C}}) \stackrel{\text{def}}{=} \max\{\phi(z^{\text{B}}) : \Pi_{\text{B}\to\text{C}}(z^{\text{B}}) = z^{\text{C}}\}. \quad (4)$$

Concretely, Model B's features include indicator features over LF conditions in Table 3 conjoined with every n-gram of the entire utterance, as there is no alignment. This is similar to the model of Pasupat and Liang (2015). Note that most of the derivation conditions (F2)–(F7) already depend on properties of the denotations of the arguments, so in Model C, we can directly reason over the space of flat logical forms z^{C} (e.g., `mix(beaker2)`) rather than explicitly computing the max over more complex logical forms z^{B} (e.g., `mix(color(red))`).

Expressivity. In going from Model A to Model C, we gain in computational efficiency, but we lose in modeling expressivity. For example, for "*second green beaker*" in Figure 1, instead of predicting `color(green)[2]`, we would have to predict `beaker3`, which is not easily explained by the words "*second green beaker*" using the simple features in Table 3.

At the same time, we found that simple features can actually *simulate* some logical forms. For example, `color(green)` can be explained by the feature that looks at the `color` property of `beaker3`. Nailing `color(green)[2]`, however, is not easy. Surprisingly, Model C can use a conjunction of features to express superlatives (e.g., `argmax(color(red),pos)`) by using one feature that places more mass on selecting objects that are red and another feature that places more mass on objects that have a greater position value.

6 Experiments

Our experiments aim to explore the computation-expressivity tradeoff in going from Model A to Model B to Model C. We would expect that under the computational constraint of a finite beam size, Model A will be hurt the most, but with an

Dataset	Model	3-acc	3-ora	5-acc	5-ora
ALCHEMY	B	0.189	0.258	0.037	0.055
	C	**0.568**	**0.925**	**0.523**	**0.809**
SCENE	B	0.068	0.118	0.017	0.031
	C	**0.232**	**0.431**	**0.147**	**0.253**
TANGRAMS	B	**0.649**	**0.910**	**0.276**	0.513
	C	0.567	0.899	0.272	**0.698**

Table 4: Test set accuracy and oracle accuracy for examples containing $L = 3$ and $L = 5$ utterances. Model C surpasses Model B in both accuracy and oracle on ALCHEMY and SCENE, whereas Model B does better in TANGRAMS.

infinite beam, Model A should perform better.

We evaluate all models on *accuracy*, the fraction of examples that a model predicts correctly. A predicted logical form z is deemed to be correct for an example $(w_0, \mathbf{x}, w_L)$ if the predicted logical form $\mathbf{z}$ executes to the correct final world state w_L. We also measure the *oracle accuracy*, which is the fraction of examples where at least one $\mathbf{z}$ on the beam executes to w_L. All experiments train for 6 iterations using AdaGrad (Duchi et al., 2010) and L_1 regularization with a coefficient of 0.001.

6.1 Real data experiments

Setup. We use a beam size of 500 within each utterance, and prune to the top 5 between utterances. For the first two iterations, Models B and C train on only the first utterance of each example ($L = 1$). In the remaining iterations, the models train on two utterance examples. We then evaluate on examples with $L = 1, \ldots, 5$, which tests our models ability to extrapolate to longer texts.

Accuracy with finite beam. We compare models B and C on the three real datasets for both $L = 3$ and $L = 5$ utterances (Model A was too expensive to use). Table 4 shows that on 5 utterance examples, the flatter Model C achieves an average accuracy of 20% higher than the more compositional Model B. Similarly, the average oracle accuracy is 39% higher. This suggests that (i) the correct logical form often falls off the beam for Model B due to a larger search space, and (ii) the expressivity of Model C is sufficient in many cases.

On the other hand, Model B outperforms Model C on the TANGRAMS dataset. This happens for two reasons. The TANGRAMS dataset has the smallest search space, since all of the utterances refer to objects using position only. Additionally, many utterances reference logical forms that Model C is unable to express, such as "*repeat the first step*", or "*add it back*".

Figure 6 shows how the models perform as the number of utterances per example varies. When the search space is small (fewer number of utterances), Model B outperforms or is competitive with Model C. However, as the search space increases (tighter computational constraints), Model C does increasingly better.

Overall, both models perform worse as L increases, since to predict the final world state w_L correctly, a model essentially needs to predict an entire sequence of logical forms $z_1, \ldots, z_L$, and errors cascade. Furthermore, for larger L, the utterances tend to have richer context-dependence.

6.2 Artificial data experiments

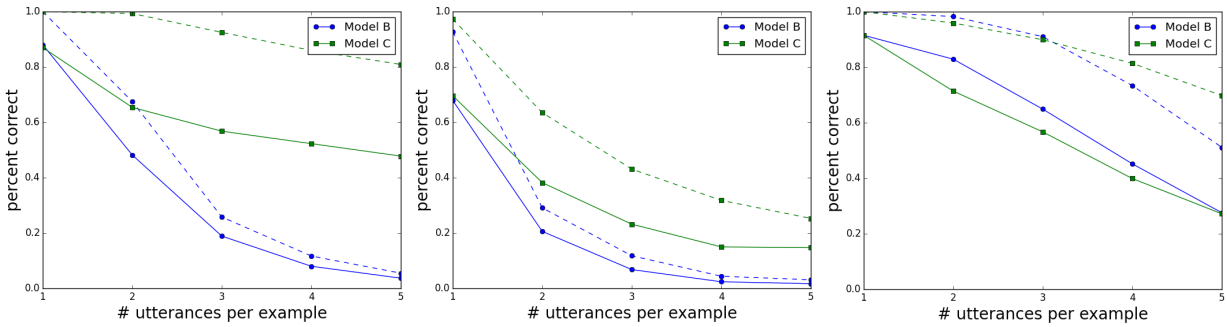
Setup. Due to the large search space, running model A on real data is impractical. In order feasibly evaluate Model A, we constructed an artificial dataset. The worlds are created using the procedure described in Section 2.2. We use a simple template to generate utterances (e.g., "*drain 1 from the 2 green beaker*").

To reduce the search space for Model A, we only allow actions (e.g., `drain`) to align to verbs and property values (e.g., `green`) to align to adjectives. Using these linguistic constraints provides a slightly optimistic assessment of Model A's performance.

We train on a dataset of 500 training examples and evaluate on 500 test examples. We repeat this procedure for varying beam sizes, from 40 to 260. The model only uses features (F1) through (F3).

Accuracy under infinite beam. Since Model A is more expressive, we would expect it to be more powerful when we have no computational constraints. Figure 7 shows that this is indeed the case: When the beam size is greater than 250, all models attain an oracle of 1, and Model A outperforms Model B, which performs similarly to Model C. This is because the alignments provide a powerful signal for constructing the logical forms. Without alignments, Models B and C learn noisier features, and accuracy suffers accordingly.

Bootstrapping. Model A performs the best with unconstrained computation, and Model C performs the best with constrained computation. Is there some way to bridge the two? Even though



(a) ALCHEMY (b) SCENE (c) TANGRAMS

Figure 6: Test results on our three datasets as we vary the number of utterances. The solid lines are the accuracy, and the dashed line are the oracles: With finite beam, Model C significantly outperforms Model B on ALCHEMY and SCENE, but is slightly worse on TANGRAMS.

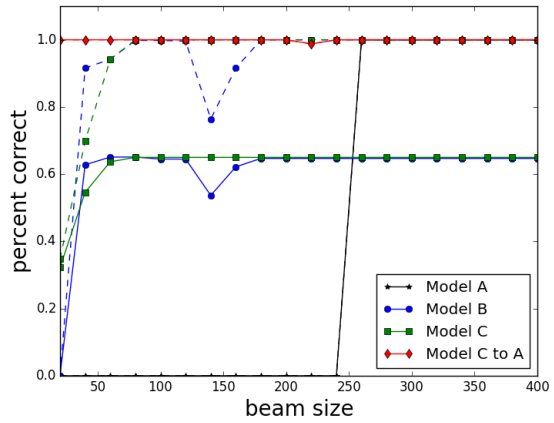


Figure 7: Test results on our artificial dataset with varying beam sizes. The solid lines are the accuracies, and the dashed line are the oracle accuracies. Model A is unable to learn anything with beam size < 240. However, for beam sizes larger than 240, Model A attains 100% accuracy. Model C does better than Models A and B when the beam size is small < 40, but otherwise performs comparably to Model B. Bootstrapping Model A using Model C parameters outperforms all of the other models and attains 100% even with smaller beams.

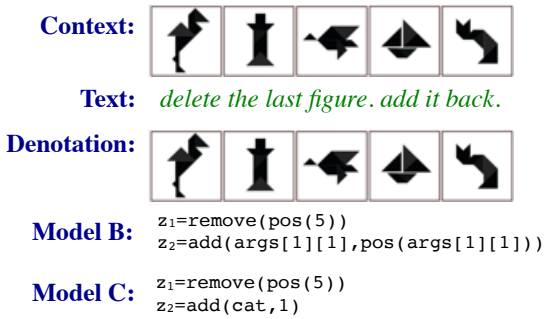


Figure 8: Predicted logical forms for this text: The logical form `add` takes a figure and position as input. Model B predicts the correct logical form. Model C does not understand that "*back*" refers to position 5, and adds the cat figure to position 1.

model	beam	action	argument	context	noise
B	0.47	0.03	0.17	0.23	0.04
C	0.15	0.03	0.25	0.5	0.07

Table 5: Percentage of errors for Model B and C: Model B suffers predominantly from computation constraints, while Model C suffers predominantly from a lack of expressivity.

Model C has limited expressivity, it can still learn to associate words like "*green*" with their corresponding predicate `green`. These should be useful for Model A too.

To operationalize this, we first train Model C and use the parameters to initialize model A. Then we train Model A. Figure 7 shows that although Model A and C predict different logical forms, the initialization allows Model C to A to perform well in constrained beam settings. This bootstrapping works here because Model C is a *projection* of Model A, and thus they share the same features.

6.3 Error Analysis

We randomly sampled 20 incorrect predictions on 3 utterance examples from each of the three real datasets for Model B and Model C. We categorized each prediction error into one of the following categories: (i) logical forms falling off the beam; (ii) choosing the wrong action (e.g., mapping "*drain*" to `pour`); (iii) choosing the wrong

argument due to misunderstanding the description (e.g., mapping "*third beaker*" to `pos(1)`); (iv) choosing the wrong action or argument due to misunderstanding of context (see Figure 8); (v) noise in the dataset. Table 5 shows the fraction of each error category.

7 Related Work and Discussion

Context-dependent semantic parsing. Utterances can depend on either linguistic context or world state context. Zettlemoyer and Collins (2009) developed a model that handles references to previous logical forms; Artzi and Zettlemoyer (2013) developed a model that handles references to the current world state. Our system considers both types of context, handling linguistic phenomena such as ellipsis and anaphora that reference both previous world states and logical forms.

Logical form generation. Traditional semantic parsers generate logical forms by aligning each part of the logical form to the utterance (Zelle and Mooney, 1996; Wong and Mooney, 2007; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011). In general, such systems rely on a lexicon, which can be hand-engineered, extracted (Cai and Yates, 2013; Berant et al., 2013), or automatically learned from annotated logical forms (Kwiatkowski et al., 2010; Chen, 2012).

Recent work on learning from denotations has moved away from anchored logical forms. Pasupat and Liang (2014) and Wang et al. (2015) proposed generating logical forms without alignments, similar to our Model B. Yao et al. (2014) and Bordes et al. (2014) have explored predicting paths in a knowledge graph directly, which is similar to the flat logical forms of Model C.

Relaxation and bootstrapping. The idea of first training a simpler model in order to work up to a more complex one has been explored other contexts. In the unsupervised learning of generative models, bootstrapping can help escape local optima and provide helpful regularization (Och and Ney, 2003; Liang et al., 2009). When it is difficult to even find one logical form that reaches the denotation, one can use the relaxation technique of Steinhardt and Liang (2015).

Recall that projecting from Model A to C creates a more computationally tractable model at the cost of expressivity. However, this is because Model C used a linear model. One might imagine that a non-linear model would be able to recuperate some of the loss of expressivity. Indeed, Neelakantan et al. (2016) use recurrent neural networks attempt to perform logical operations. One could go one step further and bypass logical forms altogether, performing all the logical reasoning in a continuous space (Bowman et al., 2014; Weston et al., 2015; Guu et al., 2015; Reed and de Freitas, 2016). This certainly avoids the combinatorial explosion of logical forms in Model A, but could also present additional optimization challenges. It would be worth exploring this avenue to completely understand the computation-expressivity tradeoff.

Reproducibility

Our code, data, and experiments are available on CodaLab at `https://worksheets.codalab.org/worksheets/0xad3fc9f52f514e849b282a105b1e3f02/`.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. The third author is supported by a Microsoft Research Faculty Fellowship.

References

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

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

A. Bordes, S. Chopra, and J. Weston. 2014. Question answering with subgraph embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. R. Bowman, C. Potts, and C. D. Manning. 2014. Can recursive neural tensor networks learn logical reasoning? In *International Conference on Learning Representations (ICLR)*.

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

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

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

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

D. A. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Workshop on Human Language Technology*, pages 43–48.

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

K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In *Empirical Methods in Natural Language Processing (EMNLP)*.

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. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 91–99.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv*.

A. Neelakantan, Q. V. Le, and I. Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *International Conference on Learning Representations (ICLR)*.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

P. Pasupat and P. Liang. 2014. Zero-shot entity extraction from web pages. In *Association for Computational Linguistics (ACL)*.

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

S. Reed and N. de Freitas. 2016. Neural programmer-interpreters. In *International Conference on Learning Representations (ICLR)*.

J. Steinhardt and P. Liang. 2015. Learning with relaxed supervision. In *Advances in Neural Information Processing Systems (NIPS)*.

A. Vlachos and S. Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics (TACL)*, 2:547–559.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

J. Weston, A. Bordes, S. Chopra, and T. Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv*.

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.

X. Yao, J. Berant, and B. Van-Durme. 2014. Freebase QA: Information extraction or semantic parsing. In *Workshop on Semantic parsing*.

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.

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

L. S. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.