Semantic Parsing on Freebase from Question-Answer Pairs

Jonathan Berant **Andrew Chou** **Roy Frostig** **Percy Liang**
Computer Science Department, Stanford University
{joberant,akchou}@stanford.edu {rf,pliang}@cs.stanford.edu

Abstract

In this paper, we train a semantic parser that scales up to Freebase. Instead of relying on annotated logical forms, which is especially expensive to obtain at large scale, we learn from question-answer pairs. The main challenge in this setting is narrowing down the huge number of possible logical predicates for a given question. We tackle this problem in two ways: First, we build a coarse mapping from phrases to predicates using a knowledge base and a large text corpus. Second, we use a *bridging* operation to generate additional predicates based on neighboring predicates. On the dataset of Cai and Yates (2013), despite not having annotated logical forms, our system outperforms their state-of-the-art parser. Additionally, we collected a more realistic and challenging dataset of question-answer pairs and improves over a natural baseline.

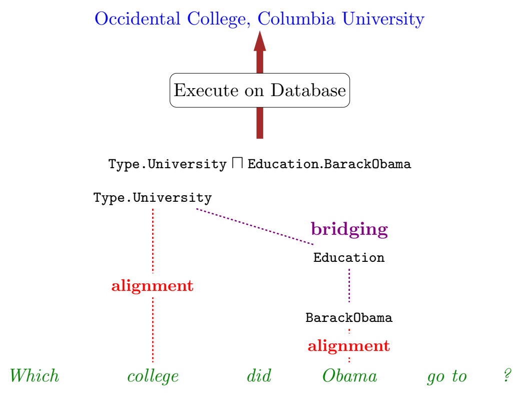


Figure 1: Our task is to map questions to answers via latent logical forms. To narrow down the space of logical predicates, we use a (i) coarse *alignment* based on Freebase and a text corpus and (ii) a *bridging* operation that generates predicates compatible with neighboring predicates.

1 Introduction

We focus on the problem of semantic parsing natural language utterances into logical forms that can be executed to produce denotations. Traditional semantic parsers (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Kwiatkowski et al., 2010) have two limitations: (i) they require annotated logical forms as supervision, and (ii) they operate in limited domains with a small number of logical predicates. Recent developments aim to lift these limitations, either by reducing the amount of supervision (Clarke et al., 2010; Liang et al., 2011; Goldwasser et al., 2011; Artzi and Zettlemoyer, 2011) or by increasing the number of logical predicates (Cai and Yates, 2013). The goal of this paper is to do both: learn a semantic parser without annotated logical forms that scales to the large number of predicates on Freebase.

At the lexical level, a major challenge in semantic parsing is mapping natural language phrases (e.g., *"attend"*) to logical predicates (e.g., `Education`). While limited-domain semantic parsers are able to learn the lexicon from per-example supervision (Kwiatkowski et al., 2011; Liang et al., 2011), at large scale they have inadequate coverage (Cai and Yates, 2013). Previous work on semantic parsing on Freebase uses a combination of manual rules (Yahya et al., 2012; Unger et al., 2012), distant supervision (Krishnamurthy and Mitchell, 2012), and schema matching (Cai and Yates, 2013). We use a large amount of web text and a knowledge base to build a coarse alignment between phrases and predicates—an approach similar in spirit to Cai and Yates (2013).

However, this alignment only allows us to generate a subset of the desired predicates. Aligning light verbs (e.g., *"go"*) and prepositions is not very informative due to polysemy, and rare predicates (e.g., *"cover price"*) are difficult to cover even given a large corpus. To improve coverage, we propose a new *bridging* operation that generates predicates based on adjacent predicates rather than on words.

At the compositional level, a semantic parser must combine the predicates into a coherent logical form. Previous work based on CCG requires manually specifying combination rules (Krishnamurthy and Mitchell, 2012) or inducing the rules from annotated logical forms (Kwiatkowski et al., 2010; Cai and Yates, 2013). We instead define a few simple composition rules which over-generate and then use model features to simulate soft rules and categories. In particular, we use POS tag features and features on the denotations of the predicted logical forms.

We experimented with two question answering datasets on Freebase. First, on the dataset of Cai and Yates (2013), we showed that our system outperforms their state-of-the-art system 62% to 59%, despite using no annotated logical forms. Second, we collected a new realistic dataset of questions by performing a breadth-first search using the Google Suggest API; these questions are then answered by Amazon Mechanical Turk workers. Although this dataset is much more challenging and noisy, we are still able to achieve 31.4% accuracy, a 4.5% absolute improvement over a natural baseline. Both datasets as well as the source code for SEMPRE, our semantic parser, are publicly released and can be downloaded from `http://nlp.stanford.edu/software/sempre/`.

2 Setup

Problem Statement Our task is as follows: Given (i) a knowledge base $\mathcal{K}$, and (ii) a training set of question-answer pairs $\{(x_i, y_i)\}_{i=1}^n$, output a semantic parser that maps new questions x to answers y via latent logical forms z and the knowledge base $\mathcal{K}$.

2.1 Knowledge base

Let $\mathcal{E}$ denote a set of *entities* (e.g., `BarackObama`), and let $\mathcal{P}$ denote a set of *properties* (e.g., `PlaceOfBirth`). A *knowledge base* $\mathcal{K}$ is a set of *assertions* $(e_1, p, e_2) \in \mathcal{E} \times \mathcal{P} \times \mathcal{E}$ (e.g., (`BarackObama`, `PlaceOfBirth`, `Honolulu`)).

We use the Freebase knowledge base (Google, 2013), which has 41M non-numeric entities, 19K properties, and 596M assertions.[1]

2.2 Logical forms

To query the knowledge base, we use a logical language called Lambda Dependency-Based Compositional Semantics (λ-DCS)—see Liang (2013) for details. For the purposes of this paper, we use a restricted subset called *simple λ-DCS*, which we will define below for the sake of completeness.

The chief motivation of λ-DCS is to produce logical forms that are simpler than lambda calculus forms. For example, $\lambda x.\exists a.p_1(x, a) \wedge \exists b.p_2(a, b) \wedge p_3(b, e)$ is expressed compactly in λ-DCS as $p_1.p_2.p_3.e$. Like DCS (Liang et al., 2011), λ-DCS makes existential quantification implicit, thereby reducing the number of variables. Variables are only used for anaphora and building composite binary predicates; these do not appear in simple λ-DCS.

Each logical form in simple λ-DCS is either a *unary* (which denotes a subset of $\mathcal{E}$) or a *binary* (which denotes a subset of $\mathcal{E} \times \mathcal{E}$). The basic λ-DCS logical forms z and their denotations $[\![z]\!]_{\mathcal{K}}$ are defined recursively as follows:

- Unary base case: If $e \in \mathcal{E}$ is an entity (e.g., `Seattle`), then e is a unary logical form with $[\![z]\!]_{\mathcal{K}} = \{e\}$.
- Binary base case: If $p \in \mathcal{P}$ is a property (e.g., `PlaceOfBirth`), then p is a binary logical form with $[\![p]\!]_{\mathcal{K}} = \{(e_1, e_2) : (e_1, p, e_2) \in \mathcal{K}\}$.[2]
- Join: If b is a binary and u is a unary, then $b.u$ (e.g., `PlaceOfBirth.Seattle`) is a unary denoting a join and project: $[\![b.u]\!]_{\mathcal{K}} = \{e_1 \in \mathcal{E} : \exists e_2.(e_1, e_2) \in [\![b]\!]_{\mathcal{K}} \wedge e_2 \in [\![u]\!]_{\mathcal{K}}\}$.

[1]In this paper, we condense Freebase names for readability (`/people/person` becomes `Person`).

[2]Binaries can be also built out of lambda abstractions (e.g., $\lambda x.$`Performance.Actor.`x), but as these constructions are not central to this paper, we defer to (Liang, 2013).

- Intersection: If u_1 and u_2 are both unaries, then $u_1 \sqcap u_2$ (e.g., $\texttt{Profession.Scientist} \sqcap \texttt{PlaceOfBirth.Seattle}$) denotes set intersection: $[\![u_1 \sqcap u_2]\!]_\mathcal{K} = [\![u_1]\!]_\mathcal{K} \cap [\![u_2]\!]_\mathcal{K}$.
- Aggregation: If u is a unary, then $\texttt{count}(u)$ denotes the cardinality: $[\![\texttt{count}(u)]\!]_\mathcal{K} = \{|[\![u]\!]_\mathcal{K}|\}$.

As a final example, *"number of dramas starring Tom Cruise"* in lambda calculus would be represented as $\texttt{count}(\lambda x.\texttt{Genre}(x, \texttt{Drama}) \wedge \exists y.\texttt{Performance}(x, y) \wedge \texttt{Actor}(y, \texttt{TomCruise}))$; in λ-DCS, it is simply $\texttt{count}(\texttt{Genre.Drama} \sqcap \texttt{Performance.Actor.TomCruise})$.

It is useful to think of the knowledge base $\mathcal{K}$ as a directed graph in which entities are nodes and properties are labels on the edges. Then simple λ-DCS unary logical forms are tree-like graph patterns which pick out a subset of the nodes.

2.3 Framework

Given an utterance x, our semantic parser constructs a distribution over possible derivations $D(x)$. Each *derivation* $d \in D(x)$ is a tree specifying the application of a set of combination rules that culminates in the logical form $d.z$ at the root of the tree—see Figure 2 for an example.

Composition Derivations are constructed recursively based on (i) a lexicon mapping natural language phrases to knowledge base predicates, and (ii) a small set of composition rules.

More specifically, we build a set of derivations for each span of the utterance. We first use the lexicon to generate single-predicate derivations for any matching span (e.g., *"born"* maps to $\texttt{PeopleBornHere}$). Then, given any logical form z_1 that has been constructed over the span $[i_1 : j_1]$ and z_2 over a non-overlapping span $[i_2 : j_2]$, we generate the following logical forms over the enclosing span $[\min(i_1, i_2) : \max(j_1, j_2)]$: intersection $z_1 \sqcap z_2$, join $z_1.z_2$, aggregation $z_1(z_2)$ (e.g., if $z_1 = \text{count}$), or bridging $z_1 \sqcap p.z_2$ for any property $p \in \mathcal{P}$ (explained more in Section 3.2).[3]

Note that the construction of derivations $D(x)$ allows us to skip any words, and in general heavily over-generates. We instead rely on features and learning to guide us away from the bad derivations.

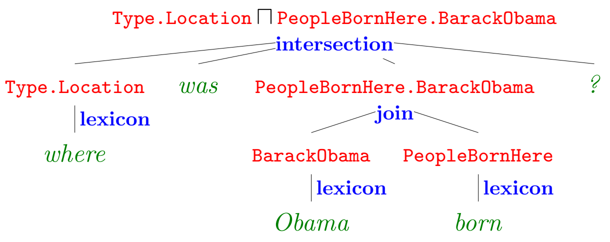


Figure 2: An example of a derivation d of the utterance *"Where was Obama born?"* and its sub-derivations, each labeled with composition rule (in blue) and logical form (in red). The derivation d skips the words *"was"* and *"?"*.

Modeling Following Zettlemoyer and Collins (2005) and Liang et al. (2011), we define a discriminative log-linear model over derivations $d \in D(x)$ given utterances x: $p_\theta(d \mid x) = \frac{\exp\{\phi(x,d)^\top \theta\}}{\sum_{d' \in D(x)} \exp\{\phi(x,d')^\top \theta\}}$, where $\phi(x, d)$ is a feature vector extracted from the utterance and the derivation, and $\theta \in \mathbb{R}^b$ is the vector of parameters to be learned. As our training data consists only of question-answer pairs (x_i, y_i), we maximize the log-likelihood of the correct answer ($[\![d.z]\!]_\mathcal{K} = y_i$), summing over the latent derivation d. Formally, our training objective is

$$\mathcal{O}(\theta) = \sum_{i=1}^{n} \log \sum_{d \in D(x) : [\![d.z]\!]_\mathcal{K} = y_i} p_\theta(d \mid x_i). \quad (1)$$

Section 4 describes an approximation of this objective that we maximize to choose parameters θ.

3 Approach

Our knowledge base has more than 19,000 properties, so a major challenge is generating a manageable set of predicates for an utterance. We propose two strategies for doing this. First (Section 3.1), we construct a lexicon that maps natural language phrases to logical predicates by aligning a large text corpus to Freebase, reminiscent of Cai and Yates (2013). Second, we generate logical predicates compatible with neighboring predicates using the bridging operation (Section 3.2). Bridging is crucial when aligning phrases is difficult or even impossible. The derivations produced by combining these predicates

[3]We also discard logical forms are incompatible according to the Freebase types (e.g., $\texttt{Profession.Politician} \sqcap \texttt{Type.City}$ would be rejected).

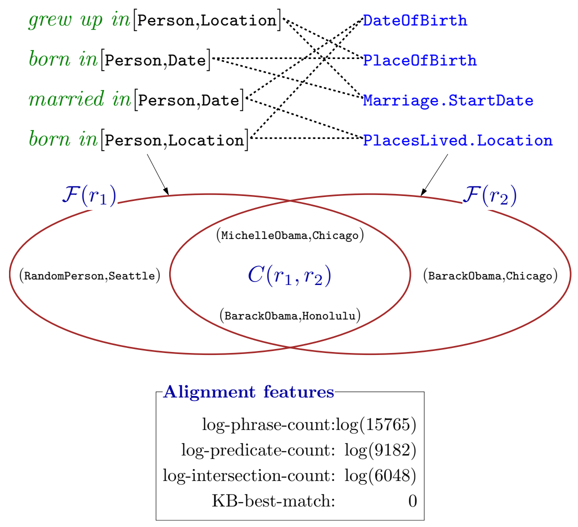


Figure 3: We construct a bipartite graph over phrases $\mathcal{R}_1$ and predicates $\mathcal{R}_2$. Each edge (r_1, r_2) is associated with alignment features.

are scored using features that capture lexical, syntactic and semantic regularities (Section 3.3).

3.1 Alignment

We now discuss the construction of a *lexicon* $\mathcal{L}$, which is a mapping from natural language phrases to logical predicates accompanied by a set of features. Specifically, for a phrase w (e.g., *"born in"*), $\mathcal{L}(w)$ is a set of entries (z, s), where z is a predicate and s is the set of features. A lexicon is constructed by alignment of a large text corpus to the knowledge base (KB). Intuitively, a phrase and a predicate align if they co-occur with many of the same entities.

Here is a summary of our alignment procedure: We construct a set of typed[4] phrases $\mathcal{R}_1$ (e.g., *"born in"*[Person,Location]) and predicates $\mathcal{R}_2$ (e.g., PlaceOfBirth). For each $r \in \mathcal{R}_1 \cup \mathcal{R}_2$, we create its *extension* $\mathcal{F}(r)$, which is a set of co-occurring entity-pairs (e.g., $\mathcal{F}($*"born in"*[Person,Location]$) = \{($BarackObama, Honolulu$), \dots\}$. The lexicon is generated based on the overlap $\mathcal{F}(r_1) \cap \mathcal{F}(r_2)$, for $r_1 \in R_1$ and $r_2 \in R_2$.

Typed phrases 15 million triples (e_1, r, e_2) (e.g., (*"Obama"*, *"was also born in"*, *"August 1961"*)) were extracted from ClueWeb09 using the ReVerb open IE system (Fader et al., 2011). Lin et al. (2012) released a subset of these triples[5] where they were able to substitute the subject arguments with KB entities. We downloaded their dataset and heuristically replaced object arguments with KB entities by walking on the Freebase graph from subject KB entities and performing simple string matching. In addition, we normalized dates with SUTime (Chang and Manning, 2012).

We lemmatize and normalize each text phrase $r \in R_1$ and augment it with a type signature $[t_1, t_2]$ to deal with polysemy (*"born in"* could either map to PlaceOfBirth or DateOfBirth). We add an entity pair (e_1, e_2) to the extension of $\mathcal{F}(r[t_1, t_2])$ if the (Freebase) type of e_1 (e_2) is t_1 (t_2). For example, (BarackObama, 1961) is added to $\mathcal{F}($*"born in"*[Person, Date]$)$. We perform a similar procedure that uses a Hearst-like pattern (Hearst, 1992) to map phrases to unary predicates. If a text phrase $r \in R_1$ matches the pattern "(is|was a|the) x IN", where IN is a preposition, then we add e_1 to $\mathcal{F}(x)$. For (Honolulu, *"is a city in"*, Hawaii), we extract $x =$ *"city"* and add Honolulu to $\mathcal{F}($*"city"*$)$. From the initial 15M triples, we extracted 55,081 typed binary phrases (9,456 untyped) and 6,299 unary phrases.

Logical predicates Binary logical predicates contain (i) all KB properties[6] and (ii) concatenations of two properties $p_1.p_2$ if the intermediate type represents an event (e.g., the *married to* relation is represented by Marriage.Spouse). For unary predicates, we consider all logical forms Type.t and Profession.t for all (abstract) entities $t \in \mathcal{E}$ (e.g. Type.Book and Profession.Author). The types of logical predicates considered during alignment is restricted in this paper, but automatic induction of more compositional logical predicates is an interesting direction. Finally, we define the extension of a logical predicate $r \in \mathcal{R}_2$ to be its denotation, that is, the corresponding set of entities or entity pairs.

Lexicon construction Given typed phrases $\mathcal{R}_1$, logical predicates $\mathcal{R}_2$, and their extensions $\mathcal{F}$, we now generate the lexicon. It is useful to think of a

[4]Freebase associates each entity with a set of types using the Type property.

[5]http://knowitall.cs.washington.edu/linked_extractions/

[6]We filter properties from the domains user and base.

Category	Description
Alignment	Log of # entity pairs that occur with the phrase r_1 ($\|\mathcal{F}(r_1)\|$) Log of # entity pairs that occur with the logical predicate r_2 ($\|\mathcal{F}(r_2)\|$) Log of # entity pairs that occur with both r_1 and r_2 ($\|\mathcal{F}(r_1) \cap \mathcal{F}(r_2)\|$) Whether r_2 is the best match for r_1 ($r_2 = \arg\max_r \|\mathcal{F}(r_1) \cap \mathcal{F}(r)\|$)
Lexicalized	Conjunction of phrase w and predicate z
Text similarity	Phrase r_1 is equal/prefix/suffix of s_2 Phrase overlap of r_1 and s_2
Bridging	Log of # entity pairs that occur with bridging predicate b ($\|\mathcal{F}(b)\|$) Kind of bridging (# unaries involved) The binary b injected
Composition	# of intersect/join/bridging operations POS tags in join/bridging and skipped words Size of denotation of logical form

Table 1: Full set of features. For the alignment and text similarity, r_1 is a phrase, r_2 is a predicate with Freebase name s_2, and b is a binary predicate with type signature (t_1, t_2).

bipartite graph with left nodes $\mathcal{R}_1$ and right nodes $\mathcal{R}_2$ (Figure 3). We add an edge (r_1, r_2) if (i) the type signatures of r_1 and r_2 match[7] and (ii) their extensions have non-empty overlap ($\mathcal{F}(r_1) \cap \mathcal{F}(r_2) \neq \emptyset$). Our final graph contains 109K edges for binary predicates and 294K edges for unary predicates.

Naturally, non-zero overlap by no means guarantees that r_1 should map to r_2. In our noisy data, even *"born in"* and `Marriage.EndDate` co-occur 4 times. Rather than thresholding based on some criterion, we compute a set of features, which are used by the model downstream in conjunction with other sources of information.

We compute three types of features (Table 1). *Alignment* features are unlexicalized and measure association based on argument overlap. *Lexicalized* features are standard conjunctions of the phrase w and the logical form z. *Text similarity* features compare the (untyped) phrase (e.g., *"born"*) to the Freebase name of the logical predicate (e.g., *"People born here"*): Given the phrase r_1 and the Freebase name s_2 of the predicate r_2, we compute string similarity features such as whether r_1 and s_2 are equal, as well as some other measures of token overlap.

[7]Each Freebase property has a designated type signature, which can be extended to composite predicates, e.g., sig(`Marriage.StartDate`) = (`Person`, `Date`).

3.2 Bridging

While alignment can cover many predicates, it is unreliable for cases where the predicates are expressed weakly or implicitly. For example, in *"What government does Chile have?"*, the predicate is expressed by the light verb *have*, in *"What actors are in Top Gun?"*, it is expressed by a highly ambiguous preposition, and in *"What is Italy money?"* [sic], it is omitted altogether. Since natural language doesn't offer much help here, let us turn elsewhere for guidance. Recall that at this point our main goal is to generate a manageable set of candidate logical forms to be scored by the log-linear model.

In the first example, suppose the phrases *"Chile"* and *"government"* are parsed as `Chile` and `Type.FormOfGovernment`, respectively, and we hypothesize a connecting binary. The two predicates impose strong type constraints on that binary, so we can afford to generate all the binary predicates that type check (see Table 2). More formally, given two unaries z_1 and z_2 with types t_1 and t_2, we generate a logical form $z_1 \sqcap b.z_2$ for each binary b whose type signature is (t_1, t_2). Figure 1 visualizes bridging of the unaries `Type.University` and `Obama`.

Now consider the example *"What is the cover price of X-men?"* Here, the binary `ComicBookCoverPrice` is expressed explicitly, but is not in our lexicon since the language use is rare. To handle this, we allow bridging to generate a binary based on a single unary; in this case, based on the unary `X-Men` (Table 2), we generate several binaries including `ComicBookCoverPrice`. Generically, given a unary z with type t, we construct a logical form $b.z$ for any predicate b with type $(*, t)$.

Finally, consider the question *"Who did Tom Cruise marry in 2006?"*. Suppose we parse the phrase *"Tom Cruise marry"* into `Marriage.Spouse.TomCruise`, or more explicitly, $\lambda x.\exists e.\texttt{Marriage}(x, e) \wedge \texttt{Spouse}(e, \texttt{TomCruise})$. Here, the neo-Davidsonian event variable e is an intermediate quantity, but needs to be further modified (in this case, by the temporal modifier *2006*). To handle this, we apply bridging to a unary and the intermediate event (see Table 2). Generically, given a logical form $p_1.p_2.z'$ where p_2 has type $(t_1, *)$, and a unary z with type t, bridging injects z and

#	Form 1	Form 2	Bridging
1	`Type.FormOfGovernment`	`Chile`	`Type.FormOfGovernment` ⊓ **`GovernmentTypeOf`**`.Chile`
2		`X-Men`	**`ComicBookCoverPriceOf`**`.X-Men`
3	`Marriage.Spouse.TomCruise`	`2006`	`Marriage.(Spouse.TomCruise` ⊓ **`StartDate`**`.2006)`

Table 2: Three examples of the bridging operation. The bridging binary predicate b is in boldface.

constructs a logical form $p_1.(p_2.z' \sqcap b.z)$ for each logical predicate b with type (t_1, t).

In each of the three examples, bridging generates a binary predicate based on neighboring logical predicates rather than on explicit lexical material. In a way, our *bridging* operation shares with bridging anaphora (Clark, 1975) the idea of establishing a novel relation between distinct parts of a sentence. Naturally, we need features to distinguish between the generated predicates, or decide whether bridging is even appropriate at all. Given a binary b, features include the log of the predicate count $\log |\mathcal{F}(b)|$, indicators for the kind of bridging, an indicator on the binary b for injections (Table 1). In addition, we add all text similarity features by comparing the Freebase name of b with content words in the question.

3.3 Composition

So far, we have mainly focused on the generation of predicates. We now discuss three classes of features pertaining to their composition.

Rule features Each derivation d is the result of applying some number of intersection, join, and bridging operations. To control this number, we define indicator features on each of these counts. This is in contrast to the norm of having a single feature whose value is equal to the count, which can only represent one-sided preferences for having more or fewer of a given operation. Indicator features stabilize the model, preferring derivations with a well-balanced inventory of operations.

Part-of-speech tag features To guide the composition of predicates, we use POS tags in two ways. First, we introduce features indicating when a word of a given POS tag is skipped, which could capture the fact that skipping auxiliaries is generally acceptable, while skipping proper nouns is not. Second, we introduce features on the POS tags involved in a composition, inspired by dependency parsing (McDonald et al., 2005). Specifically, when we combine logical forms z_1 and z_2 via a join or bridging, we include a feature on the POS tag of (the first word spanned by) z_1 conjoined with the POS tag corresponding to z_2. Rather than using head-modifier information from dependency trees (Branavan et al., 2012; Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013; Poon, 2013), we can learn the appropriate relationships tailored for downstream accuracy. For example, the phrase *"located"* is aligned to the predicate `ContainedBy`. POS features can detect that if *"located"* precedes a noun phrase (*"What is located in Beijing?"*), then the noun phrase is the object of the predicate, and if it follows the noun phrase (*"Where is Beijing located?"*), then it is in subject position.

Note that our three operations (intersection, join, and bridging) are quite permissive, and we rely on features, which encode soft, overlapping rules. In contrast, CCG-based methods (Kwiatkowski et al., 2010; Kwiatkowski et al., 2011) encode the combination preferences structurally in non-overlapping rules; these could be emulated with features with weights clamped to $-\infty$.

Denotation features While it is clear that learning from denotations rather than logical forms is a drawback since it provides less information, it is less obvious that working with denotations actually gives us additional information. Specifically, we include four features indicating whether the denotation of the predicted logical form has size 0, 1, 2, or at least 3. This feature encodes presupposition constraints in a soft way: when people ask a question, usually there is an answer and it is often unique. This allows us to favor logical forms with this property.

4 Experiments

We now evaluate our semantic parser empirically. In Section 4.1, we compare our approach to Cai and Yates (2013) on their recently released dataset (henceforth, FREE917) and present results on a new dataset that we collected (henceforth, WebQuestions). In Section 4.2, we provide detailed experiments to provide additional insight on our system.

Setup We implemented a standard beam-based bottom-up parser which stores the k-best derivations for each span. We use $k = 500$ for all our experiments on Free917 and $k = 200$ on WebQuestions. The root beam yields the candidate set $\tilde{D}(x)$ and is used to approximate the sum in the objective function $\mathcal{O}(\theta)$ in (1). In experiments on WebQuestions, $\tilde{D}(x)$ contained 197 derivations on average.

We write the approximate objective as $\mathcal{O}(\theta; \tilde{\theta}) = \sum_i \log \sum_{d \in \tilde{D}(x_i; \tilde{\theta}) : [\![d.z]\!]_{\mathcal{K}} = y_i} p(d \mid x_i; \theta)$ to explicitly show dependence on the parameters $\tilde{\theta}$ used for beam search. We optimize the objective by initializing θ_0 to 0 and applying AdaGrad (stochastic gradient ascent with per-feature adaptive step size control) (Duchi et al., 2010), so that θ_{t+1} is set based on taking a stochastic approximation of $\frac{\partial \mathcal{O}(\theta; \theta_t)}{\partial \theta}\big|_{\theta = \theta_t}$. We make six passes over the training examples.

We used POS tagging and named-entity recognition to restrict what phrases in the utterance could be mapped by the lexicon. Entities must be named entities, proper nouns or a sequence of at least two tokens. Unaries must be a sequence of nouns, and binaries must be either a content word, or a verb followed by either a noun phrase or a particle. In addition, we used 17 hand-written rules to map question words such as *"where"* and *"how many"* to logical forms such as `Type.Location` and `Count`.

To compute denotations, we convert a logical form z into a SPARQL query and execute it on our copy of Freebase using the Virtuoso engine. On WebQuestions, a full run over the training examples involves approximately 600,000 queries. For evaluation, we predict the answer from the derivation with highest probability.

4.1 Main results

4.1.1 Free917

Cai and Yates (2013) created a dataset consisting of 917 questions involving 635 Freebase relations, annotated with lambda calculus forms. We converted all 917 questions into simple λ-DCS, executed them on Freebase and used the resulting answers to train and evaluate. To map phrases to Freebase entities we used the manually-created entity lexicon used by Cai and Yates (2013), which contains 1,100 entries. Because entity disambiguation is a challenging problem in semantic parsing, the entity lexicon simplifies the problem.

Following Cai and Yates (2013), we held out 30% of the examples for the final test, and performed all development on the remaining 70%. During development, we split the data and used 512 examples (80%) for training and the remaining 129 (20%) for validation. All reported development numbers are averaged across 3 random splits. We evaluated using accuracy, the fraction of examples where the predicted answer exactly matched the correct answer.

Our main empirical result is that our system, which was trained only on question-answer pairs, obtained 62% accuracy on the test set, outperforming the 59% accuracy reported by Cai and Yates (2013), who trained on full logical forms.

4.1.2 WebQuestions

Dataset collection Because Free917 requires logical forms, it is difficult to scale up due to the required expertise of annotating logical forms. We therefore created a new dataset, WebQuestions, of question-answer pairs obtained from non-experts.

To collect this dataset, we used the Google Suggest API to obtain questions that begin with a wh-word and contain exactly one entity. We started with the question *"Where was Barack Obama born?"* and performed a breadth-first search over questions (nodes), using the Google Suggest API supplying the edges of the graph. Specifically, we queried the question excluding the entity, the phrase before the entity, or the phrase after it; each query generates 5 candidate questions, which are added to the queue. We iterated until 1M questions were visited; a random 100K were submitted to Amazon Mechanical Turk (AMT).

The AMT task requested that workers answer the question using only the Freebase page of the questions' entity, or otherwise mark it as unanswerable by Freebase. The answer was restricted to be one of the possible entities, values, or list of entities on the page. As this list was long, we allowed the user to filter the list by typing. We paid the workers \$0.03 per question. Out of 100K questions, 6,642 were annotated identically by at least two AMT workers.

We again held out a 35% random subset of the

Dataset	# examples	# word types
GeoQuery	880	279
ATIS	5,418	936
FREE917	917	2,036
WEBQUESTIONS	5,810	4,525

Table 3: Statistics on various semantic parsing datasets. Our new dataset, WEBQUESTIONS, is much larger than FREE917 and much more lexically diverse than ATIS.

questions for the final test, and performed all development on the remaining 65%, which was further divided into an 80%–20% split for training and validation. To map entities, we built a Lucene index over the 41M Freebase entities.

Table 3 provides some statistics about the new questions. One major difference in the datasets is the distribution of questions: FREE917 starts from Freebase properties and solicits questions about these properties; these questions tend to be tailored to the properties. WEBQUESTIONS starts from questions completely independent of Freebase, and therefore the questions tend to be more natural and varied. For example, for the Freebase property `ComicGenre`, FREE917 contains the question *"What genre is Doonesbury?"*, while WEBQUESTIONS for the property `MusicGenre` contains *"What music did Beethoven compose?"*.

The number of word types in WEBQUESTIONS is larger than in datasets such as ATIS and GeoQuery (Table 3), making lexical mapping much more challenging. On the other hand, in terms of structural complexity WEBQUESTIONS is simpler and many questions contain a unary, a binary and an entity.

In some questions, the answer provided by AMT workers is only roughly accurate, because workers are restricted to selecting answers from the Freebase page. For example, the answer given by workers to the question *"What is James Madison most famous for?"* is *"President of the United States"* rather than *"Authoring the Bill of Rights"*.

Results AMT workers sometimes provide partial answers, e.g., the answer to *"What movies does Taylor Lautner play in?"* is a set of 17 entities, out of which only 10 appear on the Freebase page. We therefore allow partial credit and score an answer using the F_1 measure, comparing the predicted set of entities to the annotated set of entities.

System	FREE917	WebQ.
ALIGNMENT	38.0	30.6
BRIDGING	66.9	21.2
ALIGNMENT+BRIDGING	**71.3**	**32.9**

Table 4: Accuracies on the development set under different schemes of binary predicate generation. In ALIGNMENT, binaries are generated only via the alignment lexicon. In BRIDGING, binaries are generated through the bridging operation only. ALIGNMENT+BRIDGING corresponds to the full system.

As a baseline, we omit from our system the main contributions presented in this paper—that is, we disallow bridging, and remove denotation and alignment features. The accuracy on the test set of this system is 26.9%, whereas our full system obtains 31.4%, a significant improvement.

Note that the number of possible derivations for questions in WEBQUESTIONS is quite large. In the question *"What kind of system of government does the United States have?"* the phrase *"United States"* maps to 231 entities in our lexicon, the verb *"have"* maps to 203 binaries, and the phrases *"kind"*, *"system"*, and *"government"* all map to many different unary and binary predicates. Parsing correctly involves skipping some words, mapping other words to predicates, while resolving many ambiguities in the way that the various predicates can combine.

4.2 Detailed analysis

We now delve deeper to explore the contributions of the various components of our system. All ablation results reported next were run on the development set (over 3 random splits).

Generation of binary predicates Recall that our system has two mechanisms for suggesting binaries: from the alignment lexicon or via the bridging operation. Table 4 shows accuracies when only one or both is used. Interestingly, alignment alone is better than bridging alone on WEBQUESTIONS, whereas for FREE917, it is the opposite. The reason for this is that FREE917 contains questions on rare predicates. These are often missing from the lexicon, but tend to have distinctive types and hence can be predicted from neighboring predicates. In contrast, WEBQUESTIONS contains questions that are commonly searched for and focuses on popular predicates, therefore exhibiting larger lexical variation.

System	FREE917	WebQ.
FULL	**71.3**	**32.9**
-POS	70.5	28.9
-DENOTATION	58.6	28.0

Table 5: Accuracies on the development set with features removed. POS and DENOTATION refer to the POS tag and denotation features from Section 3.3.

System	FREE917	WebQ.
ALIGNMENT	**71.3**	32.9
LEXICALIZED	68.5	34.2
LEXICALIZED+ALIGNMENT	69.0	**36.4**

Table 6: Accuracies on the development set using either unlexicalized alignment features (ALIGNMENT) or lexicalized features (LEXICALIZED).

For instance, when training without an alignment lexicon, the system errs on *"When did Nathan Smith die?"*. Bridging suggests binaries that are compatible with the common types `Person` and `Datetime`, and the binary `PlaceOfBirth` is chosen. On the other hand, without bridging, the system errs on *"In which comic book issue did Kitty Pryde first appear?"*, which refers to the rare predicate `ComicBookFirstAppearance`. With bridging, the parser can identify the correct binary, by linking the types `ComicBook` and `ComicBookCharacter`. On both datasets, best performance is achieved by combining the two sources of information.

Overall, running on WEBQUESTIONS, the parser constructs derivations that contain about 12,000 distinct binary predicates.

Feature variations Table 5 shows the results of feature ablation studies. Accuracy drops when POS tag features are omitted, e.g., in the question *"What number is Kevin Youkilis on the Boston Red Sox"* the parser happily skips the NNPs *"Kevin Youkilis"* and returns the numbers of all players on the Boston Red Sox. A significant loss is incurred without denotation features, largely due to the parser returning logical forms with empty denotations. For instance, the question *"How many people were at the 2006 FIFA world cup final?"* is answered with a logical form containing the property `PeopleInvolved` rather than `SoccerMatchAttendance`, resulting in an empty denotation.

Next we study the impact of lexicalized versus

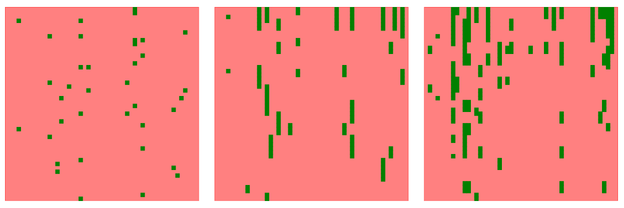
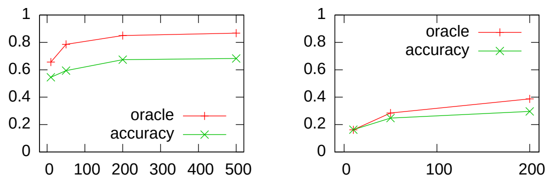


Figure 4: Beam of candidate derivations $\tilde{D}(x)$ for 50 WEBQUESTIONS examples. In each matrix, columns correspond to examples and rows correspond to beam position (ranked by decreasing model score). Green cells mark the positions of derivations with correct denotations. Note that both the number of good derivations and their positions improve as θ is optimized.



(a) FREE917 (b) WEBQUESTIONS

Figure 5: Accuracy and oracle as beam size k increases.

unlexicalized features (Table 6). In the large WEBQUESTIONS dataset, lexicalized features helped, and so we added those features to our model when running on the test set. In FREE917 lexicalized features result in overfitting due to the small number of training examples. Thus, we ran our final parser on the test set without lexicalized features.

Effect of beam size An intrinsic challenge in semantic parsing is to handle the exponentially large set of possible derivations. We rely heavily on the k-best beam approximation in the parser keeping good derivations that lead to the correct answer. Recall that the set of candidate derivations $\tilde{D}(x)$ depends on the parameters θ. In the initial stages of learning, θ is far from optimal, so good derivations are likely to fall below the k-best cutoff of internal parser beams. As a result, $\tilde{D}(x)$ contains few derivations with the correct answer. Still, placing these few derivations on the beam allows the training procedure to bootstrap θ into a good solution. Figure 4 illustrates this improvement in $\tilde{D}(x)$ across early training iterations.

Smaller choices of k yield a coarser approximation in beam search. As we increase k (Figure 5), we see a tapering improvement in accuracy. We also see a widening gap between accuracy and oracle score,[8] as including a good derivation in $\tilde{D}(x)$ is made easier but the learning problem is made more difficult.

Error analysis The accuracy on WEBQUESTIONS is much lower than on FREE917. We analyzed WEBQUESTIONS examples and found several main causes of error: (i) Disambiguating entities in WEBQUESTIONS is much harder because the entity lexicon has 41M entities. For example, given *"Where did the battle of New Orleans start?"* the system identifies *"New Orleans"* as the target entity rather than its surrounding noun phrase. Recall that all FREE917 experiments used a carefully chosen entity lexicon. (ii) Bridging can often fail when the question's entity is compatible with many binaries. For example, in *"What did Charles Babbage make?"*, the system chooses a wrong binary compatible with the type `Person`. (iii) The system sometimes incorrectly draws verbs from subordinate clauses. For example, in *"Where did Walt Disney live before he died?"* it returns the place of death of Walt Disney, ignoring the matrix verb *live*.

5 Discussion

Our work intersects with two strands of work. The first involves learning models of semantics guided by denotations or interactions with the world. Besides semantic parsing for querying databases (Popescu et al., 2003; Clarke et al., 2010; Liang et al., 2011), previous work has looked at interpreting natural language for performing programming tasks (Kushman and Barzilay, 2013; Lei et al., 2013), playing computer games (Branavan et al., 2010; Branavan et al., 2011), following navigational instructions (Chen, 2012; Artzi and Zettlemoyer, 2013), and interacting in the real world via perception (Matuszek et al., 2012; Tellex et al., 2011; Krishnamurthy and Kollar, 2013). Our system uses denotations rather than logical forms as a training signal, but also benefits from denotation features, which becomes possible in the grounded setting.

The second body of work involves connecting natural language and open-domain databases. Several works perform relation extraction using distant supervision from a knowledge base (Riedel et al., 2010; Carlson et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012). While similar in spirit to our alignment procedure for building the lexicon, one difference is that relation extraction cares about facts, aggregating over phrases, whereas a lexicon concerns specific phrases, thus aggregating over facts. On the question answering side, recent methods have made progress in building semantic parsers for the open domain, but still require a fair amount of manual effort (Yahya et al., 2012; Unger et al., 2012; Cai and Yates, 2013). Our system reduces the amount of supervision and has a more extensive evaluation on a new dataset.

Finally, although Freebase has thousands of properties, open information extraction (Banko et al., 2007; Fader et al., 2011; Masaum et al., 2012) and associated question answering systems (Fader et al., 2013) work over an even larger open-ended set of properties. The drawback of this regime is that the noise and the difficulty in canonicalization make it hard to perform reliable composition, thereby nullifying one of the key benefits of semantic parsing. An interesting midpoint involves keeping the structured knowledge base but augmenting the predicates, for example using random walks (Lao et al., 2011) or Markov logic (Zhang et al., 2012). This would allow us to map atomic words (e.g., *"wife"*) to composite predicates (e.g., $\lambda x.\texttt{Marriage.Spouse.}(\texttt{Gender.Female} \sqcap x)$). Learning these composite predicates would drastically increase the possible space of logical forms, but we believe that the methods proposed in this paper—alignment via distant supervision and bridging—can provide some traction on this problem.

Acknowledgments

We would like to thank Thomas Lin, Mausam and Oren Etzioni for providing us with open IE triples that are partially-linked to Freebase, and also Arun Chaganty for helpful comments. The authors gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0040.

[8]Oracle score is the fraction of examples for which $\tilde{D}(x)$ contains any derivation with the correct denotation.

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.

M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction from the web. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2670–2676.

S. Branavan, L. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*, pages 1268–1277.

S. Branavan, D. Silver, and R. Barzilay. 2011. Learning to win by reading manuals in a Monte-Carlo framework. In *Association for Computational Linguistics (ACL)*, pages 268–277.

S. Branavan, N. Kushman, T. Lei, and R. Barzilay. 2012. Learning high-level planning from text. In *Association for Computational Linguistics (ACL)*, pages 126–135.

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

A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr, and T. M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

A. X. Chang and C. Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *Language Resources and Evaluation (LREC)*, pages 3735–3740.

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

H. H. Clark. 1975. Bridging. In *Workshop on theoretical issues in natural language processing*, pages 169–174.

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.

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

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Fader, L. Zettlemoyer, and O. Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Association for Computational Linguistics (ACL)*.

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

Google. 2013. Freebase data dumps (2013-06-09). `https://developers.google.com/freebase/data`.

M. A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Interational Conference on Computational linguistics*, pages 539–545.

R. Hoffmann, C. Zhang, X. Ling, L. S. Zettlemoyer, and D. S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Association for Computational Linguistics (ACL)*, pages 541–550.

J. Krishnamurthy and T. Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics (TACL)*, 1:193–206.

J. Krishnamurthy and T. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 754–765.

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. 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.

N. Lao, T. Mitchell, and W. W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Lei, F. Long, R. Barzilay, and M. Rinard. 2013. From natural language specifications to program input parsers. In *Association for Computational Linguistics (ACL)*.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *As-*

sociation for Computational Linguistics (ACL), pages 590–599.

P. Liang. 2013. Lambda dependency-based compositional semantics. Technical report, ArXiv.

T. Lin, Mausam, and O. Etzioni. 2012. Entity linking at web scale. In *Knowledge Extraction Workshop (AKBC-WEKEX)*.

Masaum, M. Schmitz, R. Bart, S. Soderland, and O. Etzioni. 2012. Open language learning for information extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 523–534.

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)*.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Association for Computational Linguistics (ACL)*, pages 91–98.

H. Poon. 2013. Grounded unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

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.

S. Riedel, L. Yao, and A. McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 148–163.

M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 455–465.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

C. Unger, L. Bhmann, J. Lehmann, A. Ngonga, D. Gerber, and P. Cimiano. 2012. Template-based question answering over RDF data. In *World Wide Web (WWW)*, pages 639–648.

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. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. 2012. Natural language questions for the web of data. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 379–390.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. 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.

C. Zhang, R. Hoffmann, and D. S. Weld. 2012. Ontological smoothing for relation extraction with minimal supervision. In *Association for the Advancement of Artificial Intelligence (AAAI)*.