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.

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 using no annotated logical forms. Both datasets on Freebase names for readability (/people/person becomes Person).

2 Setup

2.1 Knowledge base

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

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

2.2 Logical forms

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

The chief motivation of $\lambda$-DCS is to produce logical forms that are simpler than lambda calculus forms. For example, $\lambda x.\exists a.p_1(x, a) \land \exists b.p_2(a, b) \land p_3(b, e)$ is expressed compactly in $\lambda$-DCS as $p_1.p_2.p_3.e$. Like DCS (Liang et al., 2011), $\lambda$-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 $\lambda$-DCS.

Each logical form in simple $\lambda$-DCS is either a unary (which denotes a subset of $E$) or a binary (which denotes a subset of $E \times E$). The basic $\lambda$-DCS logical forms $z$ and their denotations $[z]_K$ are defined recursively as follows:

- **Unary base case**: If $e \in E$ is an entity (e.g., Seattle), then $e$ is a unary logical form with $[e]_K = \{e\}$.
- **Binary base case**: If $p \in P$ is a property (e.g., PlaceOfBirth), then $p$ is a binary logical form with $[p]_K = \{(e_1, e_2) : (e_1, p, e_2) \in K\}$.\(^2\)
- **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: $[b.u]_K = \{e_1 \in E : \exists e_2. (e_1, e_2) \in [b]_K \land e_2 \in [u]_K\}$.

\(^1\)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., $\lambda x.\text{Performance.Actor.}x$), 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 \( u_1 \cap u_2 \) (e.g., \( \text{ Profession.Scientist} \cap \text{ PlaceOfBirth.Seattle} \)) denotes set intersection: \( [u_1 \cap u_2]_K = [u_1]_K \cap [u_2]_K \).

• Aggregation: If \( u \) is a unary, then \( \text{count}(u) \) denotes the cardinality: \( \left| [\text{count}(u)]_K \right| = \{|[u]_K|\} \).

As a final example, “number of dramas starring Tom Cruise” in lambda calculus would be represented as \( \text{count}(\lambda x. \text{Genre}(x, \text{Drama}) \land \exists y. \text{Performance}(x, y) \land \text{Actor}(y, \text{TomCruise})) \); in \( \lambda\)-DCS, it is simply \( \text{count}((\text{Genre.Drama} \cap \text{Performance.Actor.TomCruise})) \).

It is useful to think of the knowledge base \( K \) as a directed graph in which entities are nodes and properties are labels on the edges. Then simple \( \lambda\)-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 \( \text{PeopleBornHere} \)). 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 \( z_1 \cap z_2 \), join \( z_1 \cdot z_2 \), aggregation \( z_1(z_2) \) (e.g., if \( z_1 = \text{count} \), or bridging \( z_1 \cdot p.z_2 \) for any property \( p \in \mathcal{P} \) (explained more in Section 3.2).

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.

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 \( x \): 

\[
p_h(d \mid x) = \frac{\exp(\phi(x,d)^\top \theta)}{\sum_{d' \in D(x)} \exp(\phi(x,d')^\top \theta)},
\]

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 \( ([d.z]_K = y_i) \), summing over the latent derivation \( d \). Formally, our training objective is

\[
\mathcal{O}(\theta) = \sum_{i=1}^n \log \sum_{d \in D(x) : [d.z]_K = y_i} p_h(d \mid x_i).
\]

Section 4 describes an approximation of this objective that we maximize to choose parameters \( \theta \).

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
 grew up in [Person, Location]
born in [Person, Date]
moved to [Person, Place]
born in [Person, Date]
moved to [Person, Place]
Figure 3: We construct a bipartite graph over phrases \( R_1 \) and predicates \( R_2 \). 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 \( R_1 \) (e.g., “born in”[Person, Location]) and predicates \( R_2 \) (e.g., PlaceOfBirth). For each \( r \in R_1 \cup R_2 \), we create its extension \( \mathcal{F}(r) \), which is a set of co-occurring entity-pairs (e.g., \( \mathcal{F} (“born in”)[Person, Location] = \{(BarackObama, Honolulu), \ldots \} \)). 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 a\[the\] \( x \) IN”, where \( \text{IN} \) is a preposition, then we add \( e_1 \) to \( \mathcal{F}(x) \). For (Honolulu, “is a city in”, Hawaii), we extract \( x = “\text{city}” \) and add Honolulu to \( \mathcal{F}(“\text{city}”) \). 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 \( \text{Type} \cdot t \) and \( \text{Profession} \cdot t \) for all (abstract) entities \( t \in E \) (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 \( r \in R_2 \) to be its denotation, that is, the corresponding set of entities or entity pairs.

Lexicon construction Given typed phrases \( R_1 \), logical predicates \( R_2 \), 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.
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 $r_1$ ($r_2 = \arg \max_x |\mathcal{F}(r_1) \cap \mathcal{F}(r_2)|$)

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
<td>Log of # entity pairs that occur with the phrase $r_1$ ($</td>
</tr>
<tr>
<td></td>
<td>Log of # entity pairs that occur with the logical predicate $r_2$ ($</td>
</tr>
<tr>
<td></td>
<td>Log of # entity pairs that occur with both $r_1$ and $r_2$ ($</td>
</tr>
<tr>
<td></td>
<td>Whether $r_2$ is the best match for $r_1$ ($r_2 = \arg \max_x</td>
</tr>
</tbody>
</table>

| 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 $(t_1, t_2)$.

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 $z_1 \cap b. z_2$ 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, $\lambda x. \exists e. \text{Marriage}(x, e) \wedge \text{Spouse}(e, \text{TomCruise})$. 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
Table 2: Three examples of the bridging operation. The bridging binary predicate $b$ is in boldface.

<table>
<thead>
<tr>
<th>#</th>
<th>Form 1</th>
<th>Form 2</th>
<th>Bridging</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type.FormOfGovernment</td>
<td>Chile</td>
<td>Type.FormOfGovernment $\sqcap$ GovernmentTypeOf.Chile</td>
</tr>
<tr>
<td>2</td>
<td>X-Men</td>
<td>ComicBookCoverPriceOf.X-Men</td>
<td>Marriage.(Spouse.TomCruise $\sqcap$ StartDate.2006)</td>
</tr>
<tr>
<td>3</td>
<td>Marriage.Spouse.TomCruise</td>
<td>2006</td>
<td></td>
</tr>
</tbody>
</table>

constructs a logical form $p_1.(p_2.z_i \sqcap b.z)$ 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 $D(x)$ and is used to approximate the sum in the objective function $O(\theta)$ in (1). In experiments on WebQUESTIONS, $D(x)$ contained 197 derivations on average.

We write the approximate objective as $O(\theta; \tilde{\theta}) = \sum_i \log \sum_{d \in D(x_i; \tilde{\theta})} p(d \mid x_i; \theta)$ to explicitly show dependence on the parameters $\tilde{\theta}$ used for beam search. We optimize the objective by initializing $\theta_0$ to 0 and applying Adagrad (stochastic gradient ascent with per-feature adaptive step size control) (Duchi et al., 2010), so that $\theta_{t+1}$ is set based on taking a stochastic approximation of

$$\frac{\partial O(\theta_i; \theta_t)}{\partial \theta_i} \mid_{\theta = \theta_t}.$$

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th># examples</th>
<th># word types</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeoQuery</td>
<td>880</td>
<td>279</td>
</tr>
<tr>
<td>ATIS</td>
<td>5,418</td>
<td>936</td>
</tr>
<tr>
<td>FREE917</td>
<td>917</td>
<td>2,036</td>
</tr>
<tr>
<td>WEBQUESTIONS</td>
<td>5,810</td>
<td>4,526</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>System</th>
<th>FREE917</th>
<th>WebQ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGNMENT</td>
<td>38.0</td>
<td>30.6</td>
</tr>
<tr>
<td>BRIDGING</td>
<td>66.9</td>
<td>21.2</td>
</tr>
<tr>
<td>ALIGNMENT+BRIDGING</td>
<td>71.3</td>
<td>32.9</td>
</tr>
</tbody>
</table>

For questions 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 F1 measure, comparing the predicted set of entities to the annotated set of entities.

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.
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.

<table>
<thead>
<tr>
<th>System</th>
<th>FREE917</th>
<th>WebQ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FULL</td>
<td>71.3</td>
<td>32.9</td>
</tr>
<tr>
<td>-POS</td>
<td>70.5</td>
<td>28.9</td>
</tr>
<tr>
<td>-DENOTATION</td>
<td>58.6</td>
<td>28.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>System</th>
<th>FREE917</th>
<th>WebQ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIGNMENT</td>
<td>71.3</td>
<td>32.9</td>
</tr>
<tr>
<td>LEXICALIZED</td>
<td>68.5</td>
<td>34.2</td>
</tr>
<tr>
<td>LEXICALIZED+ALIGNMENT</td>
<td>69.0</td>
<td>36.4</td>
</tr>
</tbody>
</table>

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 NNP “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 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 \( \hat{D}(x) \) depends on the parameters \( \theta \). In the initial stages of learning, \( \theta \) is far from optimal, so good derivations are likely to fall below the k-best cutoff of internal parser beams. As a result, \( \hat{D}(x) \) contains few derivations with the correct answer. Still, placing these few derivations on the beam allows the training procedure to bootstrap \( \theta \) into a good solution. Figure 4 illustrates this improvement in \( \hat{D}(x) \) across early training iterations.

Smaller choices of \( k \) yield a coarser approxima-
tion 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, as including a good derivation in $\hat{D}(x)$ 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., $\lambda x.\text{Marriage.Spouse.(Gender.Female}\land x)$). 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.

---

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


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


