Building a Semantic Parser Overnight

Yushi Wang[*]
Stanford University
yushiw@cs.stanford.edu

Jonathan Berant[*]
Stanford University
joberant@stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

How do we build a semantic parser in a new domain starting with zero training examples? We introduce a new methodology for this setting: First, we use a simple grammar to generate logical forms paired with canonical utterances. The logical forms are meant to cover the desired set of compositional operators, and the canonical utterances are meant to capture the meaning of the logical forms (although clumsily). We then use crowdsourcing to paraphrase these canonical utterances into natural utterances. The resulting data is used to train the semantic parser. We further study the role of compositionality in the resulting paraphrases. Finally, we test our methodology on seven domains and show that we can build an adequate semantic parser in just a few hours.

1 Introduction

By mapping natural language utterances to executable logical forms, semantic parsers have been useful for a variety of applications requiring precise language understanding (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011; Berant et al., 2013; Kwiatkowski et al., 2013; Artzi and Zettlemoyer, 2013; Kushman and Barzilay, 2013). Previous work has focused on how to train a semantic parser given input utterances, but suppose we wanted to build a semantic parser for a new domain—for example, a natural language interface into a publications database. Since no such interface exists, we do not even have a naturally occurring source of input utterances that we can annotate. So where do we start?

In this paper, we advocate a *functionality-driven process* for rapidly building a semantic parser in a new domain. At a high-level, we seek to minimize the amount of work needed for a new domain by factoring out the domain-general aspects (done by our framework) from the domain-specific ones (done by the builder of the semantic parser). We assume that the builder already has the desired functionality of the semantic parser in mind—e.g., the publications database is set up and the schema is fixed. Figure 1 depicts the functionality-driven process: First, the builder writes a seed lexicon specifying a canonical phrase (*"publication date"*) for

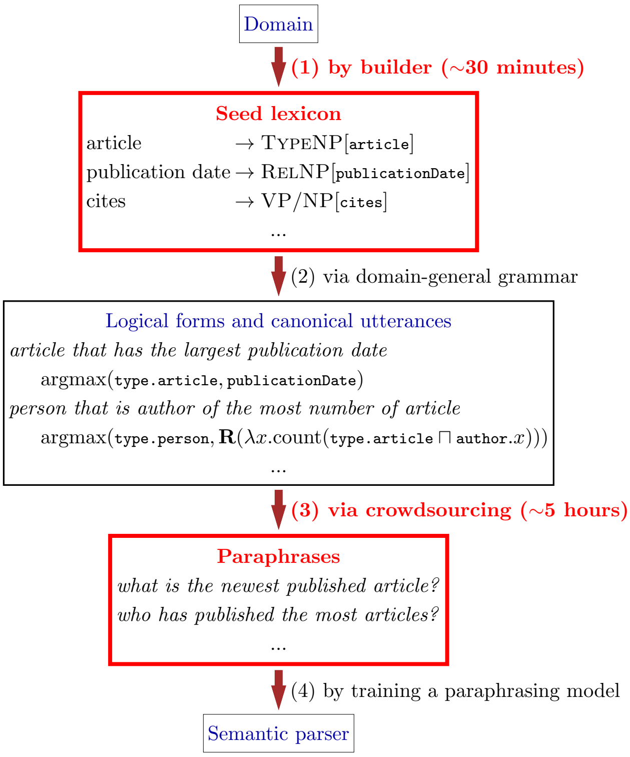
Domain
(1) by builder (~30 minutes)
Seed lexicon
article → TYPENP[article]
publication date → RELNP[publicationDate]
cites → VP/NP[cites]
...
(2) via domain-general grammar
Logical forms and canonical utterances
article that has the largest publication date
argmax(type.article, publicationDate)
person that is author of the most number of article
argmax(type.person, R(λx.count(type.article ⊓ author.x)))
...
(3) via crowdsourcing (~5 hours)
Paraphrases
what is the newest published article?
who has published the most articles?
...
(4) by training a paraphrasing model
Semantic parser

Figure 1: Functionality-driven process for building semantic parsers. The two red boxes are the domain-specific parts provided by the builder of the semantic parser, and the other two are generated by the framework.

[*] Both authors equally contributed to the paper.

*Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics
and the 7th International Joint Conference on Natural Language Processing*, pages 1332–1342,
Beijing, China, July 26-31, 2015. ©2015 Association for Computational Linguistics

each predicate (`publicationDate`). Second, our framework uses a *domain-general grammar*, along with the seed lexicon and the database, to automatically generate a few hundred *canonical utterances* paired with their logical forms (e.g., *"article that has the largest publication date"* and $\arg\max(\texttt{type.article}, \texttt{publicationDate})$). These utterances need not be the most elegant, but they should retain the semantics of the logical forms. Third, the builder leverages crowdsourcing to paraphrase each canonical utterance into a few *natural utterances* (e.g., *"what is the newest published article?"*). Finally, our framework uses this data to train a semantic parser.

Practical advantages. There are two main advantages of our approach: *completeness* and *ease of supervision*. Traditionally, training data is collected in a best-effort manner, which can result in an incomplete coverage of functionality. For example, the WebQuestions dataset (Berant et al., 2013) contains no questions with numeric answers, so any semantic parser trained on that dataset would lack that functionality. These biases are not codified, which results in an idiosyncratic and mysterious user experience, a major drawback of natural language interfaces (Rangel et al., 2014). In contrast, our compact grammar precisely specifies the logical functionality. We enforce completeness by generating canonical utterances that exercise every grammar rule.

In terms of supervision, state-of-the-art semantic parsers are trained from question-answer pairs (Kwiatkowski et al., 2013; Berant and Liang, 2014). Although this is a marked improvement in cost and scalability compared to annotated logical forms, it still requires non-trivial effort: the annotator must (i) understand the question and (ii) figure out the answer, which becomes even harder with compositional utterances. In contrast, our main source of supervision is paraphrases, which only requires (i), not (ii). Such data is thus cheaper and faster to obtain.

Linguistic reflections. The centerpiece of our framework is a domain-general grammar that connects logical forms with canonical utterances. This connection warrants further scrutiny, as the structural mismatch between logic and language is the chief source of difficulty in semantic parsing (Liang et al., 2011; Kwiatkowski et al., 2013; Berant and Liang, 2014).

There are two important questions here. First, is it possible to design a simple grammar that simultaneously generates both logical forms and canonical utterances so that the utterances are understandable by a human? In Section 3, we show how to choose appropriate canonical utterances to maximize alignment with the logical forms.

Second, our grammar can generate an infinite number of canonical utterances. How many do we need for adequate coverage? Certainly, single relations is insufficient: just knowing that *"publication date of X"* paraphrases to *"when X was published"* would offer insufficient information to generalize to *"articles that came after X"* mapping to *"article whose publication date is larger than publication date of X"*. We call this phenomena *sublexical compositionality*—when a short lexical unit (*"came after"*) maps onto a multi-predicate logical form. Our hypothesis is that the sublexical compositional units are small, so we only need to crowdsource a small number of canonical utterances to learn about most of the language variability in the given domain (Section 4).

We applied our functionality-driven process to seven domains, which were chosen to explore particular types of phenomena, such as spatial language, temporal language, and high-arity relations. This resulted in seven new semantic parsing datasets, totaling 12.6K examples. Our approach, which was not tuned on any one domain, was able to obtain an average accuracy of 59% over all domains. On the day of this paper submission, we created an eighth domain and trained a semantic parser overnight.

2 Approach Overview

In our functionality-driven process (Figure 1), there are two parties: the *builder*, who provides domain-specific information, and the *framework*, which provides domain-general information. We assume that the builder has a fixed database w, represented as a set of triples (e_1, p, e_2), where e_1 and e_2 are entities (e.g., `article1`, `2015`) and p is a property (e.g., `publicationDate`). The database w can be queried using lambda DCS logical forms, described further in Section 2.1.

The builder supplies a seed lexicon L, which contains for each database property p (e.g., `publicationDate`) a lexical entry of the form $\langle t \to s[p] \rangle$, where t is a natural language phrase (e.g., *"publication date"*) and s is a syntactic category (e.g., RELNP). In addition, L contains two typical entities for each semantic type in the database (e.g., $\langle \textit{alice} \rightarrow \mathrm{NP}[\texttt{alice}] \rangle$ for the type `person`). The purpose of L is to simply connect each predicate with *some* representation in natural language.

The framework supplies a grammar G, which specifies the modes of composition, both on logical forms and canonical utterances. Formally, G is a set of rules of the form $\langle \alpha_1 \dots \alpha_n \rightarrow s[z] \rangle$, where $\alpha_1 \dots \alpha_n$ is a sequence of tokens or categories, s is a syntactic category and z is the logical form constructed. For example, one rule in G is $\langle \mathrm{RELNP}[r] \; \textit{of} \; \mathrm{NP}[x] \rightarrow \mathrm{NP}[\mathbf{R}(r).x] \rangle$, which constructs z by reversing the binary predicate r and joining it with a the unary predicate x. We use the rules $G \cup L$ to generate a set of (z, c) pairs, where z is a logical form (e.g., $\mathbf{R}(\texttt{publicationDate}).\texttt{article1}$), and c is the corresponding canonical utterance (e.g., *"publication date of article 1"*). The set of (z, c) is denoted by GEN$(G \cup L)$. See Section 3 for details.

Next, the builder (backed by crowdsourcing) paraphrases each canonical utterance c output above into a set of natural utterances $\mathbf{P}(c)$ (e.g., *"when was article 1 published?"*). This defines a set of training examples $\mathcal{D} = \{(x, c, z)\}$, for each $(z, c) \in$ GEN$(G \cup L)$ and $x \in \mathbf{P}(c)$. The crowdsourcing setup is detailed in Section 5.

Finally, the framework trains a semantic parser on $\mathcal{D}$. Our semantic parser is a log-linear distribution $p_\theta(z, c \mid x, w)$ over logical forms and canonical utterances specified by the grammar G. Note that the grammar G will in general not parse x, so the semantic parsing model will be based on paraphrasing, in the spirit of Berant and Liang (2014).

To summarize, (1) the builder produces a seed lexicon L; (2) the framework produces logical forms and canonical utterances GEN$(G \cup L) = \{(z, c)\}$; (3) the builder (via crowdsourcing) uses $\mathbf{P}(\cdot)$ to produce a dataset $\mathcal{D} = \{(x, c, z)\}$; and (4) the framework uses $\mathcal{D}$ to train a semantic parser $p_\theta(z, c \mid x, w)$.

2.1 Lambda DCS

Our logical forms are represented in lambda DCS, a logical language where composition operates on sets rather than truth values. Here we give a brief description; see Liang (2013) for details.

Every logical form z in this paper is either a *unary* (denoting a set of entities) or a *binary* (denoting a set of entity-pairs). In the base case, each entity e (e.g., `2015`) is a unary denoting the singleton set: $[\![e]\!]_w = \{e\}$; and each property p (e.g., `publicationDate`) is a binary denoting all entity-pairs (e_1, e_2) that satisfy the property p. Unaries and binaries can be composed: Given a binary b and unary u, the join $b.u$ denotes all entities e_1 for which there exists an $e_2 \in [\![u]\!]_w$ with $(e_1, e_2) \in [\![b]\!]_w$. For example, `publicationDate.2015` denote entities published in 2015.

The intersection $u_1 \sqcap u_2$, union $u_1 \sqcup u_2$, complement $\neg u$ denote the corresponding set operations on the denotations. We let $\mathbf{R}(b)$ denote the reversal of b: $(e_1, e_2) \in [\![b]\!]_w$ iff $(e_2, e_1) \in [\![\mathbf{R}(b)]\!]_w$. This allows us to define $\mathbf{R}(\texttt{publicationDate}).\texttt{article1}$ as the publication date of article 1. We also include aggregation operations (count(u), sum(u) and average(u, b)), and superlatives (argmax(u, b)).

Finally, we can construct binaries using lambda abstraction: $\lambda x.u$ denotes a set of (e_1, e_2) where $e_1 \in [\![u[x/e_2]]\!]_w$ and $u[x/e_2]$ is the logical form where free occurrences of x are replaced with e_2. For example, $\mathbf{R}(\lambda x.\mathrm{count}(\mathbf{R}(\texttt{cites}).x))$ denotes the set of entities (e_1, e_2), where e_2 is the number of entities that e_1 cites.

3 Generation and canonical compositionality

Our functionality-driven process hinges on having a domain-general grammar that can connect logical forms with canonical utterances compositionally. The motivation is that while it is hard to write a grammar that parses *all* utterances, it is possible to write one that generates *one* canonical utterance for each logical form. To make this explicit:

Assumption 1 (Canonical compositionality) *Using a small grammar, all logical forms expressible in natural language can be realized compositionally based on the logical form.*

Grammar. We target database querying applications, where the parser needs to handle superlatives, comparatives, negation, and coordination. We define a simple grammar that captures these forms of compositionality using canonical utterances in a domain-general way. Figure 2 illustrates a derivation produced by the grammar.

The seed lexicon specified by the builder contains canonical utterances for types, entities, and properties. All types (e.g., `person`) have the syntactic category TYPENP, and all entities (e.g.,

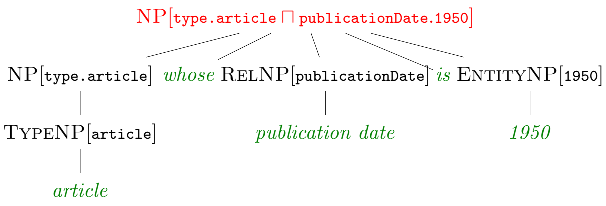

Figure 2: Deriving a logical form z (red) and a canonical utterance c (green) from the grammar G. Each node contains a syntactic category and a logical form, which is generated by applying a rule. Nodes with only leaves as children are produced using the seed lexicon; all other nodes are produced by rules in the domain-general grammar.

alice) are ENTITYNP's. Unary predicates are realized as verb phrases VP (e.g., *"has a private bath"*). The builder can choose to represent binaries as either relational noun phrases (RELNP) or generalized transitive verbs (VP/NP). RELNP's are usually used to describe functional properties (e.g., *"publication date"*), especially numerical properties. VP/NP's include transitive verbs (*"cites"*) but also longer phrases with the same syntactic interface (*"is the president of"*). Table 1 shows the seed lexicon for the SOCIAL domain.

From the seed lexicon, the domain-general grammar (Table 2) constructs noun phrases (NP), verbs phrases (VP), and complementizer phrase (CP), all of which denote unary logical forms. Broadly speaking, the rules **(R1)**–**(R4)**, **(C1)**–**(C4)** take a binary and a noun phrase, and compose them (optionally via comparatives, counting, and negation) to produce a complementizer phrase CP representing a unary (e.g., *"that cites article 1"* or *"that cites more than three article"*). **(G3)** combines these CP's with an NP (e.g., *"article"*). In addition, **(S0)**–**(S4)** handle superlatives (we include argmin in addition to argmax), which take an NP and return the extremum-attaining subset of its denotation. Finally, we support transformations such as join **(T1)** and disjunction **(T4)**, as well as aggregation **(A1)**–**(A2)**.

Rendering utterances for multi-arity predicates was a major challenge. The predicate instances are typically reified in a graph database, akin to a neo-Davidsonian treatment of events: There is an abstract entity with binary predicates relating it to its arguments. For example, in the SOCIAL domain, Alice's education can be represented in the database as five triples:

birthdate	→ RELNP[birthdate]
person|university|field	→ TYPENP[person|···]
company|job title	→ TYPENP[company|···]
student|university|field of study	→ RELNP[student|···]
employee|employer|job title	→ RELNP[employee|···]
start date|end date	→ RELNP[startDate|···]
is friends with	→ VP/NP[friends|···]

Table 1: The seed lexicon for the SOCIAL domain, which specifies for each predicate (e.g., birthdate) a phrase (e.g., *"birthdate"*) that realizes that predicate and its syntactic category (e.g., RELNP).

(e17, student, alice), (e17, university, ucla),
(e17, fieldOfStudy, music),
(e17, startDate, 2005), (e17, endDate, 2009).

All five properties here are represented as RELNP's, with the first one designated as the subject (RELNP_0). We support two ways of querying multi-arity relations: *"student whose university is ucla"* **(T2)** and *"university of student Alice whose start date is 2005"* **(T3)**.

Generating directly from the grammar in Table 2 would result in many uninterpretable canonical utterances. Thus, we perform type checking on the logical forms to rule out *"article that cites 2004"*, and limit the amount of recursion, which keeps the canonical utterances understandable.

Still, the utterances generated by our grammar are not perfectly grammatical; we do not use determiners and make all nouns singular. Nonetheless, AMT workers found most canonical utterances understandable (see Table 3 and Section 5 for details on crowdsourcing). One tip for the builder is to keep the RELNP's and VP/NP's as context-independent as possible; e.g., using *"publication date"* instead of *"date"*. In cases where more context is required, we use parenthetical remarks (e.g., *"number of assists (over a season)"* → RELNP[...]) to pack more context into the confines of the part-of-speech.

Limitations. While our domain-general grammar covers most of the common logical forms in a database querying application, there are several phenomena which are out of scope, notably nested quantification (e.g., *"show me each author's most cited work"*) and anaphora (e.g., *"author who cites herself at least twice"*). Handling these would require a more radical change to the grammar, but is still within scope.

	[glue]	
(G1)	EntityNP[x]	→ NP[x]
(G2)	TypeNP[x]	→ NP[type.x]
(G3)	NP[x] CP[f] (and CP[g])*	→ NP[$x \sqcap f \sqcap g$]
	[simple]	
(R0)	that VP[x]	→ CP[x]
(R1)	whose RelNP[r] Cmp[c] NP[y]	→ CP[$r.c.y$]
	is$\vert$is not$\vert$is smaller than$\vert$is larger than$\vert$is at least$\vert$is at most	→ Cmp[$= \mid \neq \mid < \mid > \mid \leq \mid \geq$]
(R2)	that (not)? VP/NP[r] NP[y]	→ CP[$(\neg)r.y$]
(R3)	that is (not)? RelNP[r] of NP[y]	→ CP[$(\neg)\mathbf{R}(r).y$]
(R4)	that NP[y] (not)? VP/NP[r]	→ CP[$(\neg)(\mathbf{R}(r).y)$]
	[counting]	
(C1)	that has Cnt[c] RelNP[r]	→ CP[$\mathbf{R}(\lambda x.\text{count}(\mathbf{R}(r).x)).c$]
(C2)	that VP/NP[r] Cnt[c] NP[y]	→ CP[$\mathbf{R}(\lambda x.\text{count}(y \sqcap \mathbf{R}(r).x)).c$]
(C3)	that is RelNP[r] of Cnt[c] NP[y]	→ CP[$\mathbf{R}(\lambda x.\text{count}(y \sqcap r.x)).c$]
(C4)	that Cnt[c] NP[y] VP/NP[r]	→ CP[$\mathbf{R}(\lambda x.\text{count}(y \sqcap r.x)).c$]
	(less than$\vert$more than) Num[n]	→ Cnt[$(< . \mid > .)n$]
	[superlatives]	
(S0)	NP[x] that has the largest RelNP[r]	→ NP[$\arg\max(x, r)$]
(S1)	NP[x] that has the most number of RelNP[r]	→ NP[$\arg\max(x, \mathbf{R}(\lambda y.\text{count}(\mathbf{R}(r).y)))$]
(S2)	NP[x] that VP/NP[r] the most number of NP[y]	→ NP[$\arg\max(x, \mathbf{R}(\lambda y.\text{count}(\mathbf{R}(r).y)))$]
(S3)	NP[x] that is RelNP[r] of the most number of NP[y]	→ NP[$\arg\max(x, \mathbf{R}(\lambda z.\text{count}(y \sqcap r.z)))$]
(S4)	NP[x] that the most number of NP[y] VP/NP[r]	→ NP[$\arg\max(x, \mathbf{R}(\lambda z.\text{count}(y \sqcap r.z)))$]
	[transformation]	
(T1)	RelNP[r] of NP[y]	→ NP[$\mathbf{R}(r).y$]
(T2)	RelNP$_0$[h]CP[f] (and CP[g])*	→ NP[$\mathbf{R}(h).(f \sqcap g)$]
(T3)	RelNP[r] of RelNP$_0$[h] NP[x] CP[f] (and CP[g])*	→ NP[$\mathbf{R}(r).(h.x \sqcap f \sqcap g)$]
(T4)	NP[x] or NP[y]	→ NP[$x \sqcup y$]
	[aggregation]	
(A1)	number of NP[x]	→ NP[count(x)]
(A2)	total$\vert$average RelNP[r] of NP[x]	→ NP[sum$\vert$average(x, r)]

Table 2: The domain-general grammar which is combined with the seed lexicon to generate logical forms and canonical utterances that cover the supported logical functionality.

4 Paraphrasing and bounded non-compositionality

While the canonical utterance c is generated compositionally along with the logical form z, natural paraphrases $x \in \mathbf{P}(c)$ generally deviate from this compositional structure. For example, the canonical utterance “NP*[number of* NP*[article* CP*[whose publication date is larger than* NP*[publication date of article 1]]]]”* might get paraphrased to *“How many articles were* ***published after*** *article 1?”*. Here, *“published after”* non-compositionally straddles the inner NP, intuitively responsible for both instances of *“publication date”*. But how non-compositional can paraphrases be? Our framework rests on the assumption that the answer is “not very”:

Assumption 2 (Bounded non-compositionality) *Natural utterances for expressing complex logical forms are compositional with respect to fragments of bounded size.*

In the above example, note that while *“published after”* is non-compositional with respect to the grammar, the rewriting of *“number of”* to *“how many”* is compositional. The upshot of Assumption 2 is that we only need to ask for paraphrases of canonical utterances generated by the grammar up to some small depth to learn about all the non-compositional uses of language, and still be able generalize (compositionally) beyond that.

We now explore the nature of the possible paraphrases. Broadly speaking, most paraphrases involve some sort of *compression*, where the clunky but faithful canonical utterance is smoothed out into graceful prose.

Alternations of single rules. The most basic paraphrase happens at the single phrase level with synonyms (*“block”* to *“brick”*), which preserve the part-of-speech. However, many of our properties are specified using relational noun phrases, which are more naturally realized using prepositions (*“meeting whose attendee is alice ⇒ meeting* ***with*** *alice”*) or verbs (*“author of article 1 ⇒ who* ***wrote*** *article 1”*). If the RelNP is complex, then the argument can become embedded: *“player whose number of points is 15 ⇒ player who* ***scored*** *15* ***points****”*. Superlative and comparative constructions reveal other RelNP-dependent words: *“article that has the largest publication date ⇒* ***newest*** *article”*. When the value of the relation has enough context, then the relation is elided completely: *"housing unit whose housing type is apartment ⇒* ***apartment****"*.

Multi-arity predicates are compressed into a single frame: *"university of student alice whose field of study is music"* becomes *"At which university did Alice* ***study*** *music?"*, where the semantic roles of the verb *"study"* carry the burden of expressing the multiple relations: `student`, `university`, and `fieldOfStudy`. With a different combination of arguments, the natural verb would change: *"Which university did Alice* ***attend****?"*

Sublexical compositionality. The most interesting paraphrases occur across multiple rules, a phenomenon which we called *sublexical compositionality*. The idea is that common, multi-part concepts are compressed to single words or simpler constructions. The simplest compression is a lexical one: *"parent of alice whose gender is female ⇒* ***mother*** *of alice"*. Compression often occurs when we have the same predicate chained twice in a join: *"person that is author of paper whose author is X ⇒ co-author of X"* or *"person whose birthdate is birthdate of X ⇒ person born on the* ***same*** *day as X"*. When two CP's combined via coordination have some similarity, then the coordination can be pushed down (*"meeting whose start time is 3pm and whose end time is 5pm ⇒ meetings between 3pm and 5pm"*) and sometimes even generalized (*"that allows cats and that allows dogs ⇒ that allows pets"*). Sometimes, compression happens due to metonymy, where people stand in for their papers: *"author of article that article whose author is X cites ⇒ who does X cite"*.

5 Crowdsourcing

We tackled seven domains covering various linguistic phenomena. Table 3 lists the domains, their principal phenomena, statistics about their predicates and dataset, and an example from the dataset.

We use Amazon Mechanical Turk (AMT) to paraphrase the canonical utterances generated by the domain-general grammar. In each AMT task, a worker is presented with four canonical utterances and is asked to reformulate them in natural language or state that they are incomprehensible. Each canonical utterance was presented to 10 workers. Over all domains, we collected 18,032 responses. The average time for paraphrasing one utterance was 28 seconds. Paraphrases that share the same canonical utterance are collapsed, while identical paraphrases that have distinct canonical utterances are deleted. This produced a total of 12,602 examples over all domains.

To estimate the level of noise in the data, we manually judged the correctness of 20 examples in each domain, and found that 17% of the utterances were inaccurate. There are two main reasons: lexical ambiguity on our part (*"player that has the least number of team ⇒ player with the lowest jersey number"*), and failure on the worker's part (*"restaurant whose star rating is 3 stars ⇒ hotel which has a 3 star rating"*).

6 Model and Learning

Our semantic parsing model defines a distribution over logical forms given by the domain-general grammar G and additional rules triggered by the input utterance x. Specifically, given an utterance x, we detect numbers, dates, and perform string matching with database entities to recognize named entities. This results in a set of rules $\mathbf{T}(x)$. For example, if x is *"article published in 2015 that cites article 1"*, then $\mathbf{T}(x)$ contains $\langle$*2015* $\rightarrow$ NP[2015]$\rangle$ and $\langle$*article 1* $\rightarrow$ NP[`article1`]$\rangle$. Let L_x be the rules in the seed lexicon L where the entity rules (e.g., $\langle$*alice* $\rightarrow$ NP[`alice`]$\rangle$) are replaced by $\mathbf{T}(x)$.

Our semantic parsing model defines a log-linear distribution over candidate pairs $(z, c) \in \text{GEN}(G \cup L_x)$:

$$p_\theta(z, c \mid x, w) \propto \exp(\phi(c, z, x, w)^\top \theta), \qquad (1)$$

where $\phi(z, c, x, w) \in \mathbb{R}^d$ is a feature vector and $\theta \in \mathbb{R}^d$ is a parameter vector.

To generate candidate logical forms, we use a simple beam search: For each search state, which includes the syntactic category s (e.g., NP) and the depth of the logical form, we generate at most $K = 20$ candidates by applying the rules in Table 2. In practice, the lexical rules $\mathbf{T}(x)$ are applied first, and composition is performed, but not constrained to the utterance. For example, the utterance *"article"* would generate the logical form count(`type.article`). Instead, soft paraphrasing features are used to guide the search. This rather unorthodox approach to semantic parsing can be seen as a generalization of Berant and Liang (2014) and is explained in more detail in Pasupat and Liang (2015).

Training. We train our model by maximizing the regularized log-likelihood $\mathcal{O}(\theta) =$

Domain	# pred.	# ex.	Phenomena	Example
CALENDAR	22	837	temporal language	x: *"Show me meetings after the weekly standup day"* c: *"meeting whose date is at least date of weekly standup"* z: $\texttt{type.meeting} \sqcap \texttt{date.} > \mathbf{R}(\texttt{date}).\texttt{weeklyStandup}$
BLOCKS	19	1995	spatial language	x: *"Select the brick that is to the furthest left."* c: *"block that the most number of block is right of"* z: $\texttt{argmax}(\texttt{type.block}, \mathbf{R}(\lambda x.\texttt{count}(\mathbf{R}(\texttt{right}).x)))$
HOUSING	24	941	measurement units	x: *"Housing that is 800 square feet or bigger?"* c: *"housing unit whose size is at least 800 square feet"* z: $\texttt{type.housingUnit} \sqcap \texttt{area.} > \texttt{.800}$
RESTAURANTS	32	1657	long unary relations	x: *"What restaurant can you eat lunch outside at?"* c: *"restaurant that has outdoor seating and that serves lunch"* z: $\texttt{type.restaurant} \sqcap \texttt{hasOutdoorSeating} \sqcap \texttt{serveslunch}$
PUBLICATIONS	15	801	sublexical compositionality	x: *"Who has co-authored articles with Efron?"* c: *"person that is author of article whose author is efron"* z: $\texttt{type.person} \sqcap \mathbf{R}(\texttt{author}).(\texttt{type.article} \sqcap \texttt{author.efron})$
SOCIAL	45	4419	multi-arity relations	x: *"When did alice start attending brown university?"* c: *"start date of student alice whose university is brown university"* z: $\mathbf{R}(\texttt{date}).(\texttt{student.Alice} \sqcap \texttt{university.Brown})$
BASKETBALL	24	1952	parentheticals	x: *"How many fouls were played by Kobe Bryant in 2004?"* c: *"number of fouls (over a season) of player kobe bryant whose season is 2004"* z: $\text{count}(\mathbf{R}(\texttt{fouls}).(\texttt{player.KobeBryant} \sqcap \texttt{season.2004}))$

Table 3: We experimented on seven domains, covering a variety of phenomena. For each domain, we show the number of predicates, number of examples, and a (c, z) generated by our framework along with a paraphrased utterance x.

Category	**Description**
Basic	#words and bigram matches in (x, c) #words and bigram PPDB matches in (x, c) #unmatched words in x #unmatched words in c size of denotation of z, $(\lvert [\![z]\!]_w \rvert)$ $\text{pos}(x_{0:0})$ conjoined with $\text{type}([\![z]\!]_w)$ #nodes in tree generating z
Lexical	$\forall (i, j) \in \mathcal{A}.\ (x_{i:i}, c_{j:j})$ $\forall (i, j) \in \mathcal{A}.\ (x_{i:i}, c_{j:j+1})$ $\forall (i, j) \in \mathcal{A}.\ (x_{i:i}, c_{j-1:j})$ $\forall (i, j), (i+1, j+1) \in \mathcal{A}.\ (x_{i:i+1}, c_{j:j+1})$ all unaligned words in x and c $(x_{i:j}, c_{i':j'})$ if in phrase table

Table 4: Features for the paraphrasing model. $\text{pos}(x_{i:i})$ is the POS tag; $\text{type}([\![z]\!]_w)$ is a coarse semantic type for the denotation (an entity or a number). $\mathcal{A}$ is a maximum weight alignment between x and c.

$\sum_{(x,c,z) \in \mathcal{D}} \log p_\theta(z, c \mid x, w) - \lambda \|\theta\|_1$. To optimize, we use AdaGrad (Duchi et al., 2010).

Features Table 4 describes the features. Our basic features mainly match words and bigrams in x and c, if they share a lemma or are aligned in the PPDB resource (Ganitkevitch et al., 2013). We count the number of exact matches, PPDB matches, and unmatched words.

To obtain lexical features, we run the Berkeley Aligner (Liang et al., 2006) on the training set and compute conditional probabilities of aligning one word type to another. Based on these probabilities we compute a maximum weight alignment $\mathcal{A}$ between words in x and c. We define features over $\mathcal{A}$ (see Table 4). We also use the word alignments to construct a phrase table by applying the consistent phrase pair heuristic (Och and Ney, 2004). We define an indicator feature for every phrase pair of x and c that appear in the phrase table. Examples from the PUBLICATIONS domain include *fewest–least number* and *by–whose author is*. Note that we do not build a hard lexicon but only use $\mathcal{A}$ and the phrase table to define features, allowing the model to learn useful paraphrases during training. Finally, we define standard features on logical forms and denotations (Berant et al., 2013).

7 Experimental Evaluation

We evaluated our functionality-driven process on the seven domains described in Section 5 and one new domain we describe in Section 7.3. For each domain, we held out a random 20% of the examples as the test set, and performed development on the remaining 80%, further splitting it to a training and development set (80%/20%). We created a database for each domain by randomly generating facts using entities and properties in the domain (with type-checking). We evaluated using accuracy, which is the fraction of examples that yield the correct denotation.

7.1 Domain-specific linguistic variability

Our functionality-driven process is predicated on the fact that each domain exhibits domain-specific

Method	CALENDAR	BLOCKS	HOUSING	RESTAURANTS	PUBLICATIONS	RECIPES	SOCIAL	BASKETBALL	Avg.
FULL	**74.4**	**41.9**	54.0	**75.9**	**59.0**	**70.8**	**48.2**	**46.3**	**58.8**
NOLEX	25.0	35.3	51.9	64.6	50.6	32.3	15.3	19.4	36.8
NOPPDB	73.2	41.4	**54.5**	73.8	56.5	68.1	43.6	44.5	56.9
BASELINE	17.3	27.7	45.9	61.3	46.7	26.3	9.7	15.6	31.3

Table 5: Test set results on all domains and baselines.

phenomena. To corroborate this, we compare our full system to NOLEX, a baseline that omits all lexical features (Table 4), but uses PPDB as a domain-general paraphrasing component. We perform the complementary experiment and compare to NOPPDB, a baseline that omits PPDB match features. We also run BASELINE, where we omit both lexical and PPDB features.

Table 5 presents the results of this experiment. Overall, our framework obtains an average accuracy of 59% across all eight domains. The performance of NOLEX is dramatically lower than FULL, indicating that it is important to learn domain-specific paraphrases using lexical features. The accuracy of NOPPDB is only slightly lower than FULL, showing that most of the required paraphrases can be learned during training. As expected, removing both lexical and PPDB features results in poor performance (BASELINE).

Analysis. We performed error analysis on 10 errors in each domain. Almost 70% of the errors are due to problems in the paraphrasing model, where the canonical utterance has extra material, is missing some content, or results in an incorrect paraphrase. For example, *"restaurants that have waiters and you can sit outside"* is paraphrased to *"restaurant that has waiter service and that takes reservations"*. Another 12.5% result from reordering issues, e.g, we paraphrase *"What venue has fewer than two articles"* to *"article that has less than two venue"*. Inaccurate paraphrases provided by AMT workers account for the rest of the errors.

7.2 Bounded non-compositionality

We hypothesized that we need to obtain paraphrases of canonical utterances corresponding to logical forms of only small depth. We ran the following experiment in the CALENDAR domain to test this claim. First, we define by NP_0, NP_1, and NP_2 the set of utterances generated by an NP that has exactly zero, one, and two NPs embedded in it. We define the training scenario $0 \rightarrow 1$, where we train on examples from NP_0 and test on examples from NP_1; $0 \cup 1 \rightarrow 1$, $0 \cup 1 \rightarrow 2$, and $0 \cup 1 \cup 2 \rightarrow 2$ are defined analogously. Our hypothesis is that generalization on $0 \cup 1 \rightarrow 2$ should be better than for $0 \rightarrow 1$, since NP_1 utterances have non-compositional paraphrases, but training on NP_0 does not expose them.

Scenario	Acc.	Scenario	Acc.
$0 \rightarrow 1$	22.9	$0 \cup 1 \rightarrow 2$	28.1
$0 \cup 1 \rightarrow 1$	85.8	$0 \cup 1 \cup 2 \rightarrow 2$	47.5

Table 6: Test set results in the CALENDAR domain on bounded non-compositionality.

The results in Table 6 verify this hypothesis. The accuracy of $0 \rightarrow 1$ is almost 65% lower than $0 \cup 1 \rightarrow 1$. On the other hand, the accuracy of $0 \cup 1 \rightarrow 2$ is only 19% lower than $0 \cup 1 \cup 2 \rightarrow 2$.

7.3 An overnight experiment

To verify the title of this paper, we attempted to create a semantic parser for a new domain (RECIPES) exactly 24 hours before the submission deadline. Starting at midnight, we created a seed lexicon in less than 30 minutes. Then we generated canonical utterances and allowed AMT workers to provide paraphrases overnight. In the morning, we trained our parser and obtained an accuracy of 70.8% on the test set.

7.4 Testing on independent data

Geo880. To test how our parser generalizes to utterances independent of our framework, we created a semantic parser for the domain of US geography, and tested on the standard 280 test examples from GEO880 (Zelle and Mooney, 1996). We did not use the standard 600 training examples. Our parser obtained 56.4% accuracy, which is substantially lower than state-of-the-art (∼ 90%).

We performed error analysis on 100 random sentences from the development set where accuracy was 60%. We found that the parser learns from the training data to prefer shorter paraphrases, which accounts for 30% of the errors. In most of these cases, the correct logical form is ranked at the top-3 results (accuracy for the top-3 derivations is 73%). GEO880 contains highly compositional utterances, and in 25% of the errors the correct derivation tree exceeds the maximum depth used for our parser. Another 17.5% of the errors are caused by problems in the paraphrasing model. For example, in the utterance *"what is the size of california"*, the model learns that *"size"* corresponds to *"population"* rather than *"area"*. Errors related to reordering and the syntactic structure of the input utterance account for 7.5% of the errors. For example, the utterance *"what is the area of the largest state"* is paraphrased to *"state that has the largest area"*.

Calendar. In Section 7.1, we evaluated on utterances obtained by paraphrasing canonical utterances from the grammar. To examine the coverage of our parser on independently-produced utterances, we asked AMT workers to freely come up with queries. We collected 186 such queries; 5 were spam and discarded. We replaced all entities (people, dates, etc.) with entities from our seed lexicon to avoid focusing on entity detection.

We were able to annotate 52% of the utterances with logical forms from our grammar. We could not annotate 20% of the utterances due to relative time references, such as *"What time is my next meeting?"*. 14% of the utterances were not covered due to binary predicates not in the grammar (*"What is the agenda of the meeting?"*) or missing entities (*"When is Dan's birthday?"*). Another 2% required unsupported calculations (*"How much free time do I have tomorrow?"*), and the rest are out of scope for other reasons (*"When does my Verizon data plan start over?"*).

We evaluated our trained semantic parser on the 95 utterances annotated with logical forms. Our parser obtained an accuracy of 46.3% and oracle accuracy of 84.2%, which measures how often the correct denotation is on the final beam. The large gap shows that there is considerable room for improvement in the paraphrasing model.

8 Related work and discussion

Much of current excitement around semantic parsing emphasizes large knowledge bases such as Freebase (Cai and Yates, 2013; Kwiatkowski et al., 2013; Berant et al., 2013). However, despite the apparent scale, the actual question answering datasets (Free917 and WebQuestions) are limited in compositionality. Moreover, specialized domains with specialized jargon will always exist, e.g., in regular expressions (Kushman and Barzilay, 2013) or grounding to perception (Matuszek et al., 2012; Tellex et al., 2011; Krishnamurthy and Kollar, 2013). Therefore, we believe building a targeted domain-specific semantic parser for a new website or device is a very practical goal.

Recent work has made significant strides in reducing supervision from logical forms (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007) to denotations (Clarke et al., 2010; Liang et al., 2011) and to weaker forms (Artzi and Zettlemoyer, 2011; Reddy et al., 2014). All of these works presuppose having input utterances, which do not exist in a new domain. Our methodology overcomes this hurdle by exploiting a very lightweight form of annotation: paraphrasing.

Paraphrasing has been applied to single-property question answering (Fader et al., 2013) and semantic parsing (Berant and Liang, 2014). We not only use paraphrasing in the semantic parser, but also for data collection.

Table 2 might evoke rule-based systems (Woods et al., 1972; Warren and Pereira, 1982) or controlled natural languages (Schwitter, 2010). However, there is an important distinction: the grammar need only connect a logical form to *one* canonical utterance; it is not used directly for parsing. This relaxation allows the grammar to be much simpler. Our philosophy is to use the simple domain-general grammar to carry the torch just to the point of being understandable by a human, and let the human perform the remaining correction to produce a natural utterance.

In summary, our contributions are two-fold: a new functionality-driven process and an exploration of some of its linguistic implications. We believe that our methodology is a promising way to build semantic parsers, and in future work, we would like to extend it to handle anaphora and nested quantification.

Acknowledgments. We gratefully acknowledge the support of the Google Natural Language Understanding Focused Program and the DARPA Deep Exploration and Filtering of Text (DEFT) Program contract no. FA8750-13-2-0040.

Reproducibility. All code,[1] data, and experiments for this paper are available on the CodaLab platform at `https://www.codalab.org/worksheets/0x269ef752f8c344a28383240f7bb2be9c/`.

[1] Our system uses the SEMPRE toolkit (`http://nlp.stanford.edu/software/sempre`).

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.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

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

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

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

J. Ganitkevitch, B. V. Durme, and C. Callison-Burch. 2013. PPDB: The paraphrase database. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 758–764.

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.

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, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *North American Association for Computational Linguistics (NAACL)*, pages 104–111.

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

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

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

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

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

R. A. P. Rangel, M. A. Aguirre, J. J. Gonzlez, and J. M. Carpio. 2014. Features and pitfalls that users should seek in natural language interfaces to databases. In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*, pages 617–630.

S. Reddy, M. Lapata, and M. Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics (TACL)*, 2(10):377–392.

R. Schwitter. 2010. Controlled natural languages for knowledge representation. In *International Conference on Computational Linguistics (COLING)*, pages 1113–1121.

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)*.

D. Warren and F. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics*, 8:110–122.

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.

W. A. Woods, R. M. Kaplan, and B. N. Webber. 1972. The lunar sciences natural language information system: Final report. Technical report, BBN Report 2378, Bolt Beranek and Newman Inc.

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.