Morpho-syntactic Lexical Generalization for CCG Semantic Parsing

Adrienne Wang
Computer Science & Engineering
University of Washington
Seattle, WA
axwang@cs.washington.edu

Tom Kwiatkowski
Allen Institute for AI
Seattle, WA
tomk@allenai.org

Luke Zettlemoyer
Computer Science & Engineering
University of Washington
Seattle, WA
lsz@cs.washington.edu

Abstract

In this paper, we demonstrate that significant performance gains can be achieved in CCG semantic parsing by introducing a linguistically motivated grammar induction scheme. We present a new morpho-syntactic factored lexicon that models systematic variations in morphology, syntax, and semantics across word classes. The grammar uses domain-independent facts about the English language to restrict the number of incorrect parses that must be considered, thereby enabling effective learning from less data. Experiments in benchmark domains match previous models with one quarter of the data and provide new state-of-the-art results with all available data, including up to 45% relative test-error reduction.

1 Introduction

Semantic parsers map sentences to formal representations of their meaning (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Liang et al., 2011). One common approach is to induce a probabilistic CCG grammar, which defines the meanings of individual words and phrases and how to best combine them to analyze complete sentences. There has been recent work developing learning algorithms for CCG semantic parsers (Kwiatkowski et al., 2010; Artzi and Zettlemoyer, 2011) and using them for applications ranging from question answering (Cai and Yates, 2013b; Kwiatkowski et al., 2013) to robot control (Matuszek et al., 2012; Krishnamurthy and Kollar, 2013).

One key learning challenge for this style of learning is to induce the CCG lexicon, which lists possible meanings for each phrase and defines a set of possible parses for each sentence. Previous approaches have either hand-engineered a small set of lexical templates (Zettlemoyer and Collins, 2005, 2007) or automatically learned such templates (Kwiatkowski et al., 2010, 2011). These methods are designed to learn grammars that overgenerate; they produce spurious parses that can complicate parameter estimation.

In this paper, we demonstrate that significant gains can instead be achieved by using a more constrained, linguistically motivated grammar induction scheme. The grammar is restricted by introducing detailed syntactic modeling of a wider range of constructions than considered in previous work, for example introducing explicit treatments of relational nouns, Davidsonian events, and verb tense. We also introduce a new lexical generalization model that abstracts over systematic morphological, syntactic, and semantic alternations within word classes. This includes, for example, the facts that verbs in relative clauses and nominal constructions (e.g., "flights departing NYC" and "departing flights") should be infinitival while verbs in phrases (e.g., "What flights depart at noon?") should have tense. These grammar modeling techniques use universal, domain-independent facts about the English language to restrict word usage to appropriate syntactic contexts, and as such are potentially applicable to any semantic parsing application.

More specifically, we introduce a new *morpho-syntactic*, factored CCG lexicon that imposes our grammar restrictions during learning. Each lexical entry has (1) a word stem, automatically constructed from Wiktionary, with part-of-speech and morphological attributes, (2) a lexeme that is learned and pairs the stem with semantic content that is invariant to syntactic usage, and (3) a lexical template that specifies the remaining syntactic and semantic content. The full set of templates is defined in terms of a small set of base templates and template transformations that model morphological variants such as passivization and nominalization of verbs. This approach allows us to efficiently encode a general grammar for semantic parsing while also eliminating large classes of incorrect analyses considered by previous work.

We perform experiments in two benchmark semantic parsing datasets: GeoQuery (Zelle and Mooney, 1996) and ATIS (Dahl et al., 1994). In both cases, our approach achieves state-of-the-art performance, including a nearly 45% relative error reduction on the ATIS test set. We also show that the gains increase with less data, including matching previous model's performance with less than 25% of the training data. Such gains are particularly practical for semantic parsers; they can greatly reduce the amount of data that is needed for each new application domain.

2 Related Work

Grammar induction methods for CCG semantic parsers have either used hand-engineered lexical templates, e.g. (Zettlemoyer and Collins, 2005, 2007; Artzi and Zettlemoyer, 2011), or algorithms to learn such templates directly from data, e.g. (Kwiatkowski et al., 2010, 2011). Here, we extend the first approach, and show that better lexical generalization provides significant performance gains.

Although CCG is a common choice for semantic parsers, many other formalisms have been studied, including DCS trees (Liang et al., 2011), integer linear programs (Clarke et al., 2010), and synchronous grammars (Wong and Mooney, 2007; Jones et al., 2012; Andreas et al., 2013). All of these approaches build complete meaning representations for individual sentences, but the data we use has also been studied in related work on cross-sentence reasoning (Miller et al., 1996; Zettlemoyer and Collins, 2009) and modeling semantic interpretation as a tagging problem (Tur et al., 2013; Heck et al., 2013). Although we focus on full analysis with CCG, the general idea of using linguistic constraints to improve learning is broadly applicable.

Semantic parsers are also commonly learned from a variety of different types of supervision, including logical forms (Kate and Mooney, 2006; Wong and Mooney, 2007; Muresan, 2011; Kwiatkowski et al., 2012), question-answer pairs (Clarke et al., 2010; Liang et al., 2011), conversational logs (Artzi and Zettlemoyer, 2011), distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013b), sentences paired with system behavior (Goldwasser and Roth, 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013b), and even from database constraints with no explicit semantic supervision (Poon, 2013). We learn from logical forms, but CCG learning algorithms have been developed for each case above, making our techniques applicable.

There has been significant related work that influenced the design of our morpho-syntactic grammars. This includes linguistics studies of relational nouns (Partee and Borschev, 1998; de Bruin and Scha, 1988), Davidsonian events (Davidson, 1967), parsing as abduction (Hobbs et al., 1988), and other more general theories for lexicons (Pustejovsky, 1991) and CCG (Steedman, 2011). It also includes work on using morphology in CCG syntactic parsing (Honnibal et al., 2010) and more broad-coverage semantics in CCG (Bos, 2008; Lewis and Steedman, 2013). However, our work is unique in studying the use of related ideas for semantic parsing.

Finally, there has also been recent progress on semantic parsing against large, open domain databases such as Freebase (Cai and Yates, 2013a; Kwiatkowski et al., 2013; Berant et al., 2013). Unfortuantely, existing Freebase datasets are not a good fit to test our approach because the sentences they include have relatively simple structure and can be interepreted accurately using only factoid lookups with no database joins (Yao and Van Durme, 2014). Our work focuses on learning more syntactically rich models that support compositional reasoning.

3 Background

Lambda Calculus We represent the meanings of sentences, words and phrases with

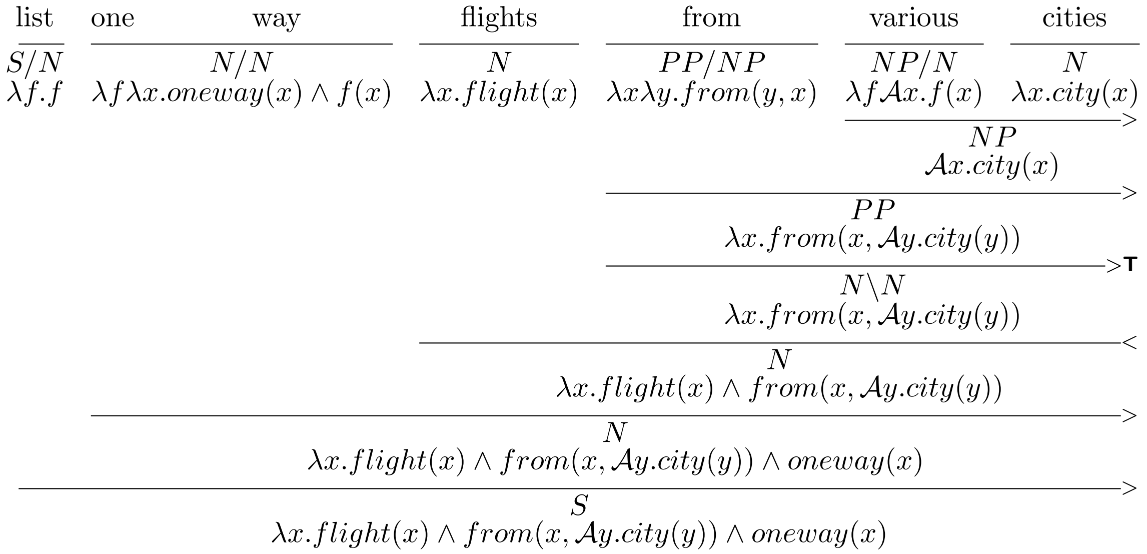

list
one
way
flights
from
various
cities
S/N
$\lambda f.f$
N/N
$\lambda f \lambda x.oneway(x) \wedge f(x)$
N
$\lambda x.flight(x)$
PP/NP
$\lambda x \lambda y.from(y, x)$
NP/N
$\lambda f \mathcal{A}x.f(x)$
N
$\lambda x.city(x)$
>
NP
$\mathcal{A}x.city(x)$
>
PP
$\lambda x.from(x, \mathcal{A}y.city(y))$
>T
$N\backslash N$
$\lambda x.from(x, \mathcal{A}y.city(y))$
<
N
$\lambda x.flight(x) \wedge from(x, \mathcal{A}y.city(y))$
>
N
$\lambda x.flight(x) \wedge from(x, \mathcal{A}y.city(y)) \wedge oneway(x)$
>
S
$\lambda x.flight(x) \wedge from(x, \mathcal{A}y.city(y)) \wedge oneway(x)$

Figure 1: An example CCG parse.

lambda calculus logical expressions. We use a version of the typed lambda calculus (Carpenter, 1997), in which the basic types include entities, events, truth values and numbers. Function types are assigned to lambda expressions. The expression $\lambda x.flight(x)$ with type $\langle e, t\rangle$ takes an entity and returns a truth value, and represents a set of flights.

Combinatory Categorial Grammar CCG (Steedman, 1996, 2000) is a formalism that tightly couples syntax and semantics, and can be used to model a wide range of linguistic phenomena. A traditional CCG grammar includes a lexicon Λ with lexical entries like the following:

$$flights \vdash N : \lambda x.flight(x)$$
$$from \vdash PP/NP : \lambda y.\lambda x.from(x, y)$$
$$cities \vdash N : \lambda x.city(x)$$

where a lexical item $w \vdash X : h$ has words w, syntactic category X, and logical expression h.

CCG uses a small set of *combinatory rules* to jointly build syntactic parses and semantic representations. Two common combinatory rules are forward (>) and backward (<) *application*:

$$X/Y : f \quad Y : g \quad \Rightarrow \quad X : f(g) \qquad (>)$$
$$Y : g \quad X\backslash Y : f \quad \Rightarrow \quad X : f(g) \qquad (<)$$

CCG also includes combinatory rules of forward (> **B**) and backward (< **B**) *composition*:

$$X/Y : f \quad Y/Z : g \Rightarrow X/Z : \lambda x.f(g(x)) \qquad (> \mathbf{B})$$
$$Y\backslash Z : g \quad X\backslash Y : f \Rightarrow X\backslash Z : \lambda x.f(g(x)) \qquad (< \mathbf{B})$$

These rules apply to build syntactic and semantic derivations concurrently.

In this paper, we also implement type raising rules for compact representation of PP (prepositional phrase) and AP (adverbial phrase).

$$PP : g \Rightarrow N\backslash N : \lambda f \lambda x.f(x) \wedge g(x) \qquad (\mathbf{T})$$
$$AP : g \Rightarrow S\backslash S : \lambda f \lambda e.f(e) \wedge g(e) \qquad (\mathbf{T})$$
$$AP : g \Rightarrow S/S : \lambda f \lambda e.f(e) \wedge g(e) \qquad (\mathbf{T})$$

Figure 1 shows an example CCG parse (Steedman, 1996, 2000) where the lexical entries are listed across the top and the output lambda-calculus meaning representation is at the bottom. This meaning is a function (denoted by λx...) that defines a set of flights with certain properties and includes a generalized Skolem constant (Steedman, 2011) ($\mathcal{A}y$...) that performs existential quantification. Following recent work (Artzi and Zettlemoyer, 2013b), we use meaning representations that model a variety of linguistic constructions, for example including Skolem constants for plurals and Davidson quantifiers for events, which we will introduce briefly throughout this paper as they appear.

Weighted CCGs A weighted CCG grammar is defined as $G = (\Lambda, \Theta)$, where Λ is a CCG lexicon and $\Theta \in \mathbb{R}^d$ is a d-dimensional parameter vector, which will be used to rank the parses allowed under Λ.

For a sentence x, G produces a set of candidate parse trees $Y = \mathsf{Y}(x; G)$. Given a feature vector $\Phi \in \mathbb{R}^d$, each parse tree y for sentence x is scored by $\mathsf{S}(y; \Theta) = \theta \cdot \phi(x, y)$. The output logical form $\hat{z}$ is then defined to be at the root of the highest-scoring parse $\hat{y}$:

$$\hat{y} \;=\; \arg\max_{y \in \mathsf{Y}(x;G)} \mathsf{S}(y; \Theta) \qquad (1)$$

We use existing CKY-style parsing algorithms for this computation, implemented with UW SPF (Artzi and Zettlemoyer, 2013a). Section 7 describes the set of features we use in the learned models.

Learning with GENLEX We will also make use of an existing learning algorithm (Zettlemoyer and Collins, 2007) (ZC07). We first briefly review the ZC07 algorithm, and describe our modifications in Section 7.

Given a set of training examples $D = \{(x_i, z_i) : i = 1...n\}$, x_i being the ith sentence and z_i being its annotated logical form, the algorithm learns a set of parameters Θ for the grammar, while also inducing the lexicon Λ.

The ZC07 learning algorithm uses a function $\mathsf{GENLEX}(x, z)$ to define a set of lexical entries that could be used to parse the sentence x to construct the logical form z. For each training example (x, z), $\mathsf{GENLEX}(x, z)$ maps all substrings x to a set of potential lexical entries, generated by exhaustively pairing the logical constants in z using a set of hand-engineered templates. The example is then parsed with this much bigger lexicon and lexical entries from the highest scoring parses are added to Λ. The parameters Θ used to score parses are updated using a perceptron learning algorithm.

4 Morpho-Syntactic Lexicon

This section defines our morpho-syntactic lexical formalism. Table 1 shows examples of how lexemes, templates, and morphological transformations are used to build lexical entries for example verbs. In this section, we formally define each of these components and show how they are used to specify the space of possible lexical entries that can be built for each input word. In the following two sections, we will provide more discussion of the complete sets of templates (Section 5) and transformations (Section 6).

Verb, Noun, Preposition, Pronoun, Adjective, Adverb, Conjunction, Numeral, Symbol, Proper Noun, Interjection, Expression

Table 2: Part-of-Speech types

POS	**A**ttribute	**V**alues
Noun	Number	singular, plural
Verb	Person	first, second, third
Verb	Voice	active, passive
Verb	Tense	present, past
Verb	Aspect	simple, progressive, perfect
Verb	Participle	present participle, past participle
Adj, Adv, Det	Degree of comparison	comparative, superlative

Table 3: Morphological attributes and values.

We build on the factored CCG lexicon introduced by Kwiatkowski et al. (2011) but (a) further generalize lexemes to represent word stems, (b) constrain the use of templates with widely available syntactic information, and (c) efficiently model common morphological variations between related words.

The first step, given an input word w, is to do morphological and part-of-speech analysis with the **morpho-syntactic function** F. F maps a word to a set of possible morpho-syntactic representations, each containing a triple (s, p, m) of word stem s, part-of-speech p and morphological category m. For example, F maps the word *flies* to two possible representations:

$$F(\mathit{flies}) = \{(\mathit{fly}, \text{Noun}, (\mathit{plural})),$$
$$(\mathit{fly}, \text{Verb}, (\mathit{third}, \mathit{singular}, \mathit{simple}, \mathit{present}))\}$$

for the plural noun and present-tense verb senses of the word. F is defined based on the stems, part-of-speech types, and morphological attributes marked for each definition in Wiktionary. [1] The full sets of possible part-of-speech and morphological types required for our domains are shown in Table 2 and Table 3.

Each morpho-syntactic analysis $a \in F(w)$ is then paired with lexemes based on stem match. A **lexeme** $(s, \vec{c})$ pairs a word stem s with a list of logical constants $\vec{c} = [c_1 \ldots c_k]$. Table 1 shows the words 'depart', 'departing', 'departure', which are all assigned the lexeme $(\mathit{depart}, [\mathit{depart}])$. In general, there can

[1] www.wiktionary.com

Word	Lexeme : Base Template	Trans	Lexical entry
depart	$(depart, [depart])$:	I	$depart \vdash S\backslash NP : \lambda x \lambda e.depart(e,x)$
departing		I	$departing \vdash S\backslash NP : \lambda x \lambda e.depart(e,x)$
departing	$\xi \vdash S\backslash NP : \lambda x \lambda e.v_1(e,x)$	f_{pres}	$departing \vdash PP : \lambda x \lambda e.depart(e,x)$
departure		f_{nom}	$departure \vdash N : \lambda x \lambda e.depart(e,x)$
use	$(use, [airline])$:	I	$use \vdash S\backslash NP/NP : \lambda x \lambda y \lambda e.airline(e,y,x)$
using		I	$using \vdash S\backslash NP/NP : \lambda x \lambda y \lambda e.airline(e,y,x)$
using	$\xi \vdash S\backslash NP/NP : \lambda x \lambda y \lambda e.v_1(e,y,x)$	f_{pres}	$using \vdash PP/NP : \lambda x \lambda e.airline(e,y,x)$
use		f_{nom}	$use \vdash N/NP : \lambda x \lambda y \lambda e.airline(e,y,x)$

Trans	Template Transformation
f_{pres}	$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \quad \rightarrow \quad \xi \vdash PP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$
f_{nom}	$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \quad \rightarrow \quad \xi \vdash N/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$

Table 1: Lexical entries constructed by combining a *lexeme*, *base template*, and *transformation* for the intransitive verb 'depart' and the transitive verb 'use'.

be many different lexemes for each stem, that vary in the selection of which logical constants are included.

Given analysis (s, p, m) and lexeme $(s, \vec{c})$, we can use a **lexical template** to construct a **lexical entry**. Each template has the form:

$$\lambda(\xi, \vec{v}).[\xi \vdash X : h_{\vec{v}}]$$

where ξ and $\vec{v}$ are variables that abstract over the words and logical constants that will be used to define a lexical entry with syntax X and templated logical form $h_{\vec{v}}$.

To instantiate a template, ξ is filled with the original word w and the constants in $\vec{c}$ replace the variables $\vec{v}$. For example, the template $\lambda(\xi, \vec{v}).[\xi \vdash S\backslash NP : \lambda x \lambda e.v_1(e,x)]$ could be used with the word 'departing' and the lexeme $(depart, [depart])$ to produce the lexical entry $departing \vdash S\backslash NP : \lambda x \lambda e.depart(e,x)$. When clear from context, we will omit the function signature $\lambda_p(\xi, \vec{v})$. for all templates, as seen in Table 1.

In general, there can be many applicable templates, which we organize as follows. Each final template is defined by applying a **morphological transformation** to one of a small set of possible **base templates**. The pairing is found based on the morphological analysis (s, p, m), where each base template is associated with part-of-speech p and each transformation is indexed by the morphology m. A transformation f_m is a function:

$$\mathsf{f}_m(\lambda_p(\xi, \vec{v}).[\xi \vdash X : h_{\vec{v}}]) = \lambda_p(\xi, \vec{v}).[\xi \vdash X' : h'_{\vec{v}}]$$

that takes the base template as input and produces a new template to model the inflected form specified by m.

For example, both base templates in Table 1 are for verbs. The template $\xi \vdash S\backslash NP : \lambda x \lambda e.v_1(e,x)$ can be translated into three other templates based on the transformations I, f_{pres}, and f_{nom}, depending on the analysis of the original words. These transformations generalize across word type; they can be used for the transitive verb 'use' as well as the intransitive 'depart.' Each resulting template, potentially including the original input if the identity transformation I is available, can then be used to make an output lexical entry, as we described above.

5 Lexical Templates

The templates in our lexicon, as introduced in Section 4, model the syntactic and semantic aspects of lexical entries that are shared within each word class. Previous approaches have also used hand-engineered lexical templates, as described in Section 2, but we differ by (1) using more templates allowing for more fine grained analysis and (2) using word class information to restrict template use, for example ensuring that words which cannot be verbs are never paired with templates designed for verbs. This section describes the templates used during learning, first presenting those designed to model grammatical sentences and then a small second set designed for more elliptical spoken utterances.

Base Forms Table 4 lists the primary template set, where each row shows an example with a sentence illustrating its use. Templates are also grouped by the word classes, including adjectives, adverbs, prepositions, and several types of nouns and verbs. While there is not enough space to discuss each row, it is worth

word class	example usage	base template
Noun phrase	Boston	$\xi \vdash NP : v$
Noun (regular) Noun (relation) Noun (function)	What **flight** is provided by delta? I need **fares** of flights delta **schedule** **size** of California	$\xi \vdash N : \lambda x.v(x)$ $\xi \vdash N/PP : \lambda x \lambda y.v(x,y)$ $\xi \vdash N\backslash(N/N) : \lambda f \lambda x.v(\mathcal{A}y.f(\lambda z.true, y), x)$ $\xi \vdash NP/NP : \lambda x.v(x)$
$V_{intrans}$	What flights **depart** from New York?	$\xi \vdash S\backslash NP : \lambda x \lambda e.v(e,x)$
V_{trans}	Which airlines **serve** Seattle (active verb) What airlines **have** flights (passive verb)	$\xi \vdash S\backslash NP/NP : \lambda x \lambda y \lambda e.v(e,y,x)$ $\xi \vdash S\backslash NP/NP : \lambda x \lambda y \lambda e.v(e,x,y)$
$V_{ditrans}$	They **give** him a book	$\xi \vdash S\backslash NP/NP/NP : \lambda x \lambda y \lambda z \lambda e.v(e,z,y,x)$
$V_{imperson}$	It **costs** $500 to fly to Boston	$\xi \vdash S\backslash NP/NP/NP : \lambda x \lambda y \lambda z \lambda e.v(e,y,x)$
V_{aux}	The flights **have** arrived at Boston **Does** delta provide flights from Seattle?	$\xi \vdash S\backslash NP/(S\backslash NP) : \lambda f.f$ $\xi \vdash S/NP/(S/NP) : \lambda f.f$ $\xi \vdash S/S : \lambda f.f$
V_{copula}	The flights **are** from Boston What flight **is** cheap? Alaska **is** the state with the most rivers	$\xi \vdash S\backslash NP/PP : \lambda f \lambda x.f(x)$ $\xi \vdash S\backslash NP/(N/N) : \lambda f \lambda x.f(\lambda y.true, x)$ $\xi \vdash S\backslash NP/NP : \lambda x \lambda y.equals(y,x)$
Adjective	I need a **one way** flight Boston flights **round trip** How **long** is mississippi?	$\xi \vdash N/N : \lambda f \lambda x.f(x) \wedge v(x)$ $\xi \vdash PP : \lambda x.v(x)$ $\xi \vdash DEG : \lambda x.v(x)$
Preposition	List flights **from** Boston List flights that go **to** Dallas List flights **between** Dallas and Boston What flights leave **between** 8am and 9am?	$\xi \vdash PP/NP : \lambda x \lambda y.v(y,x)$ $\xi \vdash AP/NP : \lambda x \lambda e.v(e,x)$ $\xi \vdash PP/NP/NP : \lambda x \lambda y \lambda z.v_1(z,x) \wedge v_2(z,y)$ $\xi \vdash AP/NP/NP : \lambda x \lambda y \lambda e.v_1(e,x) \wedge v_2(e,y)$
Adverb	Which flight departs **daily**? How **early** does the flight arrive?	$\xi \vdash AP : \lambda e.v(e)$ $\xi \vdash DEG : \lambda x.v(x)$
Determiner	Which airline has **a** flight from Boston?	$\xi \vdash NP/N : \lambda f \mathcal{A}x.f(x)$

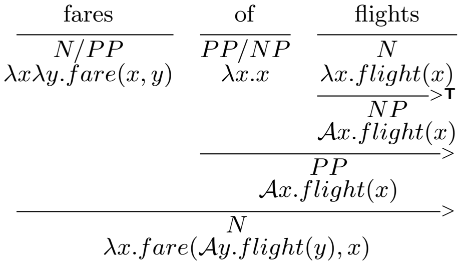
Table 4: Base templates that define different syntactic roles.

type	example usage	base template
$t_{elliptical}$	flights **Newark** to Cleveland flights arriving **2pm** **american airline** from Denver	$\xi \vdash PP : \lambda x.P(x,v)$ $\xi \vdash AP : \lambda e.P(e,v)$ $\xi \vdash N : \lambda x.P(x,v)$
$t_{metonymy}$	List **airlines** from Seattle Shat **airlines** depart from Seattle? **fares** from miami to New York	$\xi \vdash N/PP : \lambda f \lambda x.v(x) \wedge P(\mathcal{A}y.f(y),x))$ $\xi \vdash N/(S\backslash NP) : \lambda f \lambda x.v(x) \wedge P(\mathcal{A}y.f(y),x)$ $\xi \vdash N/PP : \lambda f \lambda x.v(\mathcal{A}y.f(y),x)$

Table 5: Base templates for ungrammatical linguistic phenomena

considering nouns as an illustrative example.

We model nouns as denoting a set of entities that satisfy a given property. Regular nouns are represented using unary predicates. Relational nouns syntactically function as regular nouns but semantically describe sets of entities that have some relationship with a complement (Partee and Borschev, 1998). For example, the relational noun *fare* describes a binary relationship between flights and their price information, as we see in this parse:



This analysis differs from previous approaches (Zettlemoyer and Collins, 2007), where relational nouns were treated as regular nouns and prepositions introduced the binary relationship. The relational noun model reduces lexical ambiguity for the prepositions, which are otherwise highly polysemous.

Adjectives are nominal modifiers that take a noun or a noun phrase as an argument and add properties through conjunction. Prepositions take nominal objects and function as adjectival modifiers for nouns or adverbial modifiers for verbs. Verbs can be subcategorized by their grammatical structures into transitive (V_{trans}), intransitive ($V_{intrans}$), impersonal ($V_{imperson}$), auxiliary (V_{aux}) and copula (V_{copula}). Adverbs are verb modifiers defining aspects like time, rate and duration. The adoption of event semantics allows adverbial modifiers to be represented by predicates and linked by the shared events. Determiners precede nouns or noun phrases and distinguish a reference of the noun. Following the generalized Skolem terms, we model determiners, including indefinite and definite articles, as a $\langle\langle e, t\rangle, e\rangle$ function that selects a unique individual from a $\langle e, t\rangle$-typed function defining a singleton set.

Missing Words The templates presented so far model grammatically correct input. However, in dialogue domains such as ATIS, speakers often omit words. For example, speakers can drop the preposition "from" in "flights from Newark to Cleveland" to create the elliptical utterance "flights Newark to Cleveland". We address this issue with the templates $t_{elliptical}$ illustrated in Table 5. Each of these adds a binary relation P to a lexeme with a single entity typed constant. For our example, the word "Newark" could be assigned the lexical item $Newark \vdash PP : \lambda x.from(x, newark)$ by selecting the first template and $P = from$.

Another common problem is the use of metonymy. In the utterance "What airlines depart from New York?", the word "airlines" is used to reference flight services operated by a specific airline. This is problematic because the word "depart" needs to modify an event of type *flight*. We solve this with the $t_{metonymy}$ templates in Table 5. These introduce a binary predicate P that would, in the case of our example, map airlines on to the flights that they operate.

The templates in Table 5 handle the major cases of missing words seen in our data and are more efficient than the approach taken by (Zettlemoyer and Collins, 2007) who introduced complex type shifting rules and relaxed the grammar to allow every word order.

6 Morphological Transformations

Finally, the morpho-syntactic lexicon introduces morphological transformations, which are functions from base lexical templates to lexical templates that model the syntactic and semantic variation as the word is inflected. These transformations allow us to compactly model, for example, the facts that argument order is reversed when moving from active to passive forms of the same verb, and that the subject can be omitted. To the best of our knowledge, we are the first to study such transformations for semantic parsing.

Table 6 shows the transformations. Each row groups a set of transformations by linguistic category, including singular vs. plural number, active vs. passive voice, and so on, and also includes example sentences where the output templates could be used. Again, for space, we do not detail the motivation for every class, but it is worth looking at some of the alternations for verbs and nouns as our prototypical example.

Some verbs can act as noun modifiers. For example, the present participle "using" modifies "flights" in "flights *using* twa". To model this variation, we use the transformation $\mathsf{f}_{present\_part}$, a mapping that changes the root of the verb signature $S\backslash NP$ to PP:

$$\begin{aligned}\mathsf{f}_{present\_part} : \xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \\ \to \xi \vdash PP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)\end{aligned}$$

where $\mathcal{T} = [\epsilon, NP, NP/NP]$ instantiates this rule for verbs that take different sets of arguments, effectively allowing any verb that is in its finite or -ing form to behave syntactically like a prepositional phrase.

Intransitive present participles can also act as prenominal adjectival modifiers as in "the *departing* flight". We add a second mapping that maps the intransitive category $S\backslash NP$ to the noun modifier N/N.

$$\begin{aligned}\mathsf{f}_{present\_part} : \quad & \xi \vdash S\backslash NP : \lambda x \lambda e.v(e, x) \\ \to \quad & \xi \vdash N/N : \lambda f \lambda x \lambda e.f(x) \wedge v(e, x)\end{aligned}$$

Finally, verbal nouns have meanings derived from actions typically described by verbs but syntactically function as nouns. For example, *landing* in the phrase "landing from jfk" is the gerundive use of the verb *land*. We add the following mapping to $\mathsf{f}_{present\_part}$ and $\mathsf{f}_{nominal}$:

$$\begin{aligned}\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \to \\ \xi \vdash N/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)\end{aligned}$$

with $\mathcal{T}$ from above. This allows for reuse of the same meaning across quite different syntactic constructs, including for example "flights that *depart* from Boston" and "*departure* from Boston."

Template transformations f_m	Example usage
Plural Number (f_{plural})	
I	**flight** → early **flights**
$\xi \vdash N : \lambda x.v(x) \rightarrow \xi \vdash NP : \mathcal{A}x.v(x)$	**city** → flights to **cities**
Singular Number ($\mathsf{f}_{singular}$)	
I	**flight** → **flight**
Possessive ($\mathsf{f}_{possess}$)	
$\xi \vdash NP : v \rightarrow \xi \vdash N/N : \lambda f \lambda x.f(x) \wedge P(x, v)$	**delta** → **delta's** flights
$\xi \vdash N : \lambda x.v(x) \rightarrow \xi \vdash N/N : \lambda f \lambda x.f(x) \wedge P(x, \mathcal{A}y.v(y))$	**airline** → **airline's** flights
Passive Voice ($\mathsf{f}_{passive}$)	
$\xi \vdash \mathcal{Y}/NP : \lambda x_1..x_n \lambda e.v(e, x_1..x_n) \rightarrow \xi \vdash \mathcal{Y}/PP : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**serves** →is **served** by
$\xi \vdash \mathcal{Y}/NP : \lambda x_1..x_n \lambda e.v(e, x_1, .., x_n) \rightarrow \xi \vdash \mathcal{Y} : \lambda x_1..x_{n-1} \lambda e.v(e, x_{n-1}..x_1)$	**name** →city **named** Austin
Present Participle ($\mathsf{f}_{present}$)	
$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash PP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**use** →flights **using** twa
$\xi \vdash S\backslash NP : \lambda x \lambda e.v(e, x) \rightarrow \xi \vdash N/N : \lambda f \lambda x \lambda e.f(x) \wedge v(e, x)$	**arrive** →**arriving** flights
$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash N/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**land** → **landings** at jfk
Past Participle (f_{past})	
$\xi \vdash S\backslash NP/NP : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash PP/PP : \lambda x_1..x_n \lambda e.v(e, x_1..x_n)$	**use** → plane **used** by
Nominalization ($\mathsf{f}_{nominal}$)	
$\xi \vdash S\backslash NP/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1) \rightarrow \xi \vdash N/\mathcal{T} : \lambda x_1..x_n \lambda e.v(e, x_n..x_1)$	**depart** → **departure**
Comparative (f_{comp})	
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash PP/PP : \lambda x \lambda y.v(y) < v(x)$	**short** → **shorter**
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash PP/PP : \lambda x \lambda y.v(y) > v(x)$	**long** → **longer**
Superlative (f_{super})	
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash NP/N : \lambda f.argmin(\lambda x.f(x), \lambda x.v(x))$	**short** → **shortest**
$\xi \vdash DEG : \lambda x.v(x) \rightarrow \xi \vdash NP/N : \lambda f.argmax(\lambda x.f(x), \lambda x.v(x))$	**long** → **longest**

Table 6: Morphological transformations with examples. $\mathcal{T} = [\epsilon, NP, NP/NP]$ and $\mathcal{Y} = [S\backslash NP, S\backslash NP/NP]$ allow a single transformation to generalize across word type.

Nouns can be inflected by number to denote singular and plural forms or by adding an apostrophe to mark a possessive case. The transformation function $\mathsf{f}_{\mathsf{singular}}$ is an identity transformation. Plurals may have different interpretations: one is the generic $\langle e, t \rangle$ set representation, which requires no transformation on the base, or plurals can occur without overt determiners (bare plurals), but semantically imply quantification. We create a plural to singular type shifting rule which implements the $\langle \langle e, t \rangle, e \rangle$ skolem function to select a unique individual from the set. The possessive transformation $\mathsf{f}_{\mathsf{possess}}$ transfers the base template to a noun modifier, and adds a binary predicate P that encodes the relation.

There are also a number of instances of the identity transformation function I, which does not change the base template. Because the semantics we are constructing was designed to answer questions against a static database, it does not need to represent certain phenomena to return the correct answer. This includes more advanced variants of person, tense, aspect, and potentially many others. Ideally, these morphological attributes should add semantic modifiers to the base meaning, for example, tense can constrain the time at which an event occurs. However, none of our domains support such reasoning, so we assign the identity transformation, and leave the exploration of these issues to future work.

7 Learning

One advantage of our morpho-syntactic, factored lexicon is that it can be easily learned with small modifications to existing algorithms (Zettlemoyer and Collins, 2007). We only need to modify the GENLEX procedure that defines the space of possible lexical entries. For each training example (x, z), $\mathsf{GENLEX}(x, z, F)$ first maps each substring in the sentence x into the morphological representation (s, p, c) using F introduced in Section 4. A candidate lexeme set L' is then generated by exhaustively pairing the word stems with all subsets of the logical constants from z. Lexical templates are applied to the lexemes in L' to generate candidate lexical entries for x. Finally, the lexemes that participate in the top scoring correct parse of x are added to the permanent lexicon.

Initialization Following standard practice, we compile an initial lexicon Λ_0, which consists of a list of domain independent lexical items for function words, such as interrogative words and conjunctions. These lexical items are mostly semantically vacuous and serve particular syntactic functions that are not generalizable to other word classes. We also initialize the lexemes with a list of NP entities complied from the database, e.g., (*Boston*, [*bos*]).

Features We use two types of features in the model for discriminating parses. Four *lexical* features are fired on each lexical item: $\phi_{(s,\vec{c})}$ for the lexeme, ϕ_{t_p} for the base template, ϕ_{t_m} for the morphologically modified template, and ϕ_l for the complete lexical item. We also compute the standard *logical expression* features (Zettlemoyer and Collins, 2007) on the root semantics to track the pairwise predicate-argument relations and the cooccuring predicate-predicate relations in conjunctions and disjunctions.

8 Experimental Setup

Data and Metrics We evaluate performance on two benchmark semantic parsing datasets, Geo880 and ATIS. We use the standard data splits, including 600/280 train/test for Geo880 and 4460/480/450 train/develop/test for ATIS. To support the new representations in Section 5, we systematically convert annotations with existential quantifiers, temporal events and relational nouns to new logical forms with equivalent meanings. All systems are evaluated with exact match accuracy, the percentage of fully correct logical forms.

Initialization We assign positive initial weights to the indicator features for entries in the initial lexicon, as defined in Section 7, to encourage their use. The elliptical template and metonymy template features are initialized with negative weights to initially discourage word skipping.

Comparison Systems We compare performance with all recent CCG grammar induction algorithms that work with our datasets. This includes methods that used a limited set of hand-engineered templates for inducing the lexicon, ZC05 (Zettlemoyer and Collins, 2005) and ZC07 (Zettlemoyer and Collins, 2007), and those that learned grammar structure by automatically splitting the labeled logical forms, UBL (Kwiatkowski et al., 2010) and FUBL (Kwiatkowski et al., 2011). We also compare the state-of-the-art for Geo880 (DCS (Liang et al., 2011) and DCS+ which includes an engineered seed lexicon) and ATIS (which is ZC07). Finally, we include results for GUSP (Poon, 2013), a recent unsupervised approach for ATIS.

System	Test
ZC05	79.3
ZC07	86.1
UBL	87.9
FUBL	88.6
DCS	87.9
FULL	90.4
DCS$^+$	91.1

Table 7: Exact-match Geo880 test accuracy.

System	Dev	Test
ZC07	74.4	84.6
UBL	65.6	71.4
FUBL	81.9	82.8
GUSP	-	83.5
TEMP-ONLY	85.5	87.2
FULL	87.5	91.3

Table 8: Exact-match accuracy on the ATIS development and test sets.

System Variants We report results for a complete approach (Full), and variants which use different aspects of the morpho-syntactic lexicon. The TEMP-ONLY variant learned with the templates from Section 5 but, like ZC07, does not use any word class information to restrict their use. The TEMP-POS removes morphology from the lexemes, but includes the word class information from Wiktionary. Finally, we also include DCS$^+$, which initialize a set of words with POS tag JJ, NN, and NNS.

9 Results

Full Models Tables 7 and 8 report the main learning results. Our approach achieves state-of-the-art accuracies on both datasets, demonstrating that our new grammar induction scheme provides a type of linguistically motivated regularization; restricting the algorithm to consider a much smaller hypothesis space allows to learn better models.

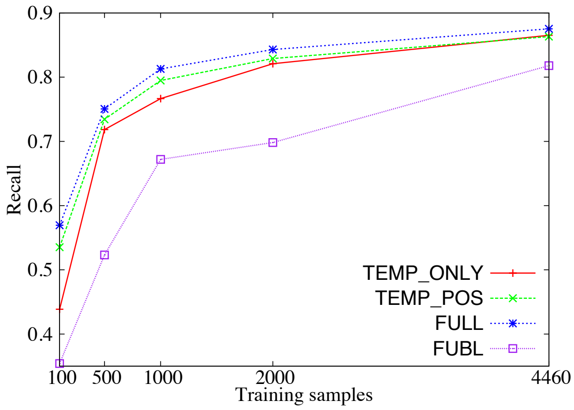


Figure 2: ATIS Learning Curve

On Geo880 the full method edges out the best systems by 2% absolute on the test set, as compared to other systems with no domain-specific lexical initialization. Although DCS requires less supervision, it also uses external signals including a POS tagger.

We see similarly strong results for ATIS, outperforming FUBL on the ATIS development set by 6.8%, and improving the accuracy on the test set by 7.9% over the previous best system ZC07. Unlike FUBL, which excels at the development set but trails ZC07's templated grammar by almost 2 points on the test set, our approach demonstrates consistent improvements on both. Additionally, although the unsupervised model (GUSP) rivals previous approaches, we are able to show that more careful use of supervision open a much wider performance gap.

Learning Curve with Ablations Figure 2 presents a learning curve for the ATIS domain, demonstrating that the learning improvements become even more dramatic for smaller training set sizes. Our model outperforms FUBL by wide margins, matching its final accuracy with only 22% of the total training examples. Our full model also consistently beats the variants with fewer word class restrictions, although by smaller margins. Again, these results further highlight the benefit of importing external syntactic resources and enforcing linguistically motivated constraints during learning.

Learned Lexicon The learned lexicon is also more compact. Table 9 summarizes statistics on unique lexical entries required to parse the ATIS development set. The morpho-syntactic model uses 80.3% of the lexical entries and 63.7% of the lexemes that FUBL needs, while increase performance by nearly 7 points. Upon inspection, our model achieves better lexical decomposition by learning shorter lexical units, for example, the adoption of Davidsonian events allows us to learn unambiguous adverbial modifiers, and the formal modeling of nominalized nouns and relational nouns treats prepositions as syntactic modifiers, instead of being encoded in the semantics. Such restrictions generalize to a much wider variety of syntactic contexts.

System	Lexical Entries	Lexemes
FUBL	1019	721
Our Approach	818	495

Table 9: Lexicon size comparison on the ATIS dev set (460 unique tokens).

10 Summary and Future Work

We demonstrated that significant performance gains can be achieved in CCG semantic parsing by introducing a more constrained, linguistically motivated grammar induction scheme. We introduced a morpho-syntactic factored lexicon that uses domain-independent facts about the English language to restrict the number of incorrect parses that must be considered and demonstrated empirically that it enables effective learning of complete parsers, achieving state-of-the-art performance.

Because our methods are domain independent they should also benefit other semantic parsing applications and other learning algorithms that use different types of supervision, as we hope to verify in future work. We would also like to study how to generalize these gains to languages other than English, by inducing more of the syntactic structure.

Acknowledgements

The research was supported by the NSF (IIS-1115966, IIS-1252835) and the Intel Center for Pervasive Computing at the Univeristy of Washington. The authors thank Robert Gens, Xiao Ling, Xu Miao, Mark Yatskar and the UW NLP group for helpful discussions, and the anonymous reviewers for helpful comments.

References

Andreas, J., Vlachos, A., and Clark, S. (2013). Semantic parsing as machine translation.

Artzi, Y. and Zettlemoyer, L. (2011). Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Artzi, Y. and Zettlemoyer, L. (2013a). UW SPF: The University of Washington Semantic Parsing Framework.

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

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

Bos, J. (2008). Wide-coverage semantic analysis with boxer. In *Proceedings of the Conference on Semantics in Text Processing*.

Cai, Q. and Yates, A. (2013a). Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Cai, Q. and Yates, A. (2013b). Semantic parsing freebase: Towards open-domain semantic parsing. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

Carpenter, B. (1997). *Type-Logical Semantics*. The MITPress.

Chen, D. and Mooney, R. (2011). Learning to interpret natural language navigation instructions from observations. In *Proceedings of the National Conference on Artificial Intelligence*.

Clarke, J., Goldwasser, D., Chang, M., and Roth, D. (2010). Driving semantic parsing from the world's response. In *Proceedings of the Conference on Computational Natural Language Learning*.

Dahl, D. A., Bates, M., Brown, M., Fisher, W., Hunicke-Smith, K., Pallett, D., Pao, C., Rudnicky, A., and Shriberg, E. (1994). Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*.

Davidson, D. (1967). The logical form of action sentences. *Essays on actions and events*, pages 105–148.

de Bruin, J. and Scha, R. (1988). The interpretation of relational nouns. In *Proceedings of the Conference of the Association of Computational Linguistics*, pages 25–32. ACL.

Goldwasser, D. and Roth, D. (2011). Learning from natural instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Heck, L., Hakkani-Tür, D., and Tur, G. (2013). Leveraging knowledge graphs for web-scale unsupervised semantic parsing. In *Proc. of the INTERSPEECH*.

Hobbs, J. R., Stickel, M., Martin, P., and Edwards, D. (1988). Interpretation as abduction. In *Proceedings of the Association for Computational Linguistics*.

Honnibal, M., Kummerfeld, J. K., and Curran, J. R. (2010). Morphological analysis can improve a ccg parser for english. In *Proceedings of the International Conference on Computational Linguistics*.

Jones, B. K., Johnson, M., and Goldwater, S. (2012). Semantic parsing with bayesian tree transducers. In *Proceedings of Association of Computational Linguistics*.

Kate, R. and Mooney, R. (2006). Using string-kernels for learning semantic parsers. In *Proceedings of the Conference of the Association for Computational Linguistics*.

Krishnamurthy, J. and Kollar, T. (2013). Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1(2).

Krishnamurthy, J. and Mitchell, T. (2012). Weakly supervised training of semantic parsers. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Kwiatkowski, T., Choi, E., Artzi, Y., and

Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching.

Kwiatkowski, T., Goldwater, S., Zettlemoyer, L., and Steedman, M. (2012). A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. *Proceedings of the Conference of the European Chapter of the Association of Computational Linguistics.*

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2010). Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., and Steedman, M. (2011). Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Lewis, M. and Steedman, M. (2013). Combined distributional and logical semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Liang, P., Jordan, M., and Klein, D. (2011). Learning dependency-based compositional semantics. In *Proceedings of the Conference of the Association for Computational Linguistics.*

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

Miller, S., Stallard, D., Bobrow, R., and Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *Proceedings Association for Computational Linguistics.*

Muresan, S. (2011). Learning for deep language understanding. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

Partee, B. H. and Borschev, V. (1998). Integrating lexical and formal sematics: Genitives, relational nouns, and type-shifting. In *Proceedings of the Second Tbilisi Symposium on Language, Logic, and Computation.*

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

Pustejovsky, J. (1991). The generative lexicon. volume 17.

Steedman, M. (1996). *Surface Structure and Interpretation.* The MIT Press.

Steedman, M. (2000). *The Syntactic Process.* The MIT Press.

Steedman, M. (2011). *Taking Scope.* The MIT Press.

Tur, G., Deoras, A., and Hakkani-Tur, D. (2013). Semantic parsing using word confusion networks with conditional random fields. In *Proc. of the INTERSPEECH.*

Wong, Y. and Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the Conference of the Association for Computational Linguistics.*

Yao, X. and Van Durme, B. (2014). Information extraction over structured data: Question answering with freebase. In *Association for Computational Linguistics (ACL).*

Zelle, J. and Mooney, R. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence.*

Zettlemoyer, L. and Collins, M. (2005). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence.*

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.*

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