Weakly Supervised Training of Semantic Parsers

Jayant Krishnamurthy
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
jayantk@cs.cmu.edu

Tom M. Mitchell
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
tom.mitchell@cmu.edu

Abstract

We present a method for training a semantic parser using only a knowledge base and an unlabeled text corpus, without any individually annotated sentences. Our key observation is that multiple forms of weak supervision can be combined to train an accurate semantic parser: *semantic supervision* from a knowledge base, and *syntactic supervision* from dependency-parsed sentences. We apply our approach to train a semantic parser that uses 77 relations from Freebase in its knowledge representation. This semantic parser extracts instances of binary relations with state-of-the-art accuracy, while simultaneously recovering much richer semantic structures, such as conjunctions of multiple relations with partially shared arguments. We demonstrate recovery of this richer structure by extracting logical forms from natural language queries against Freebase. On this task, the trained semantic parser achieves 80% precision and 56% recall, despite never having seen an annotated logical form.

1 Introduction

Semantic parsing converts natural language statements into logical forms in a meaning representation language. For example, the phrase "town in California" might be represented as $\lambda x.\text{CITY}(x) \wedge \text{LOCATEDIN}(x, \text{CALIFORNIA})$, where CITY, LOCATEDIN and CALIFORNIA are predicates and entities from a knowledge base. The expressivity and utility of semantic parsing is derived from this meaning representation, which is essentially a program that is directly executable by a computer. In this sense, broad coverage semantic parsing is the goal of natural language understanding.

Unfortunately, due to data annotation constraints, modern semantic parsers only operate in narrow domains. The best performing semantic parsers are trained using extensive manual annotation: typically, a number of sentences must be annotated with their desired logical form. Although other forms of supervision exist (Clarke et al., 2010; Liang et al., 2011), these methods similarly require annotations for individual sentences. More automated training methods are required to produce semantic parsers with richer meaning representations.

This paper presents an algorithm for training a semantic parser without per-sentence annotations. Instead, our approach exploits two easily-obtainable sources of supervision: a large knowledge base and (automatically) dependency-parsed sentences. The semantic parser is trained to identify relation instances from the knowledge base while simultaneously producing parses that syntactically agree with the dependency parses. Combining these two sources of supervision allows us to train an accurate semantic parser *for any knowledge base* without annotated training data.

We demonstrate our approach by training a Combinatory Categorial Grammar (CCG) (Steedman, 1996) that parses sentences into logical forms containing any of 77 relations from Freebase. Our training data consists of relation instances from Freebase and automatically dependency-parsed sentences from a web corpus. The trained semantic parser extracts binary relations with state-of-the-art performance, while recovering considerably richer semantic structure. We demonstrate recovery of this semantic structure using natural language queries

$$\dfrac{\dfrac{\text{town}}{N : \lambda x.\textsc{City}(x)}\text{Lex} \quad \dfrac{\dfrac{\text{in}}{(N\backslash N)/N : \lambda f.\lambda g.\lambda x.\exists y.f(y) \wedge g(x) \wedge \textsc{LocatedIn}(x,y)}\text{Lex} \quad \dfrac{\text{California}}{N : \lambda x.x = \textsc{California}}\text{Lex}}{N\backslash N : \lambda g.\lambda x.\exists y.y = \textsc{California} \wedge g(x) \wedge \textsc{LocatedIn}(x,y)}>}{N : \lambda x.\exists y.y = \textsc{California} \wedge \textsc{City}(x) \wedge \textsc{LocatedIn}(x,y)}<$$

Figure 1: An example parse of "town in California" using the example CCG lexicon. The first stage in parsing retrieves a category from each word from the lexicon, represented by the "Lex" entries. The second stage applies CCG combination rules, in this case both forms of function application, to combine these categories into a semantic parse.

against Freebase. Our weakly-supervised semantic parser predicts the correct logical form for 56% of queries, despite never seeing a labeled logical form.

This paper is structured as follows. We first provide some background information on CCG and the structure of a knowledge base in Section 2. Section 3 formulates the weakly supervised training problem for semantic parsers and presents our algorithm. Section 4 describes how we applied our algorithm to construct a semantic parser for Freebase, and Section 5 presents our results. We conclude with related work and discussion.

2 Background

2.1 Combinatory Categorial Grammar

Combinatory Categorial grammar (CCG) is a linguistic formalism that represents both the syntax and semantics of language (Steedman, 1996). CCG is a lexicalized formalism that encodes all grammatical information in a lexicon Λ. This lexicon contains syntactic and semantic categories for each word. A lexicon may include entries such as:

$$\begin{aligned} \text{town} &:= N : \lambda x.\textsc{City}(x) \\ \text{California} &:= N : \lambda x.x = \textsc{California} \\ \text{in} &:= (N\backslash N)/N : \lambda f.\lambda g.\lambda x. \\ &\quad \exists y.f(y) \wedge g(x) \wedge \textsc{LocatedIn}(x,y) \end{aligned}$$

Each entry of the lexicon $w := s : l$ maps a word or short phrase w to a syntactic category s and a logical form l. Syntactic categories s may be atomic (N) or complex ($N\backslash N$). Logical forms l are lambda calculus expressions constructed using predicates from a knowledge base. These logical forms combine during parsing to form a complete logical form for the parsed text.

Parses are constructed by combining adjacent categories using several combination rules, such as forward ($>$) and backward ($<$) application:

$$\begin{aligned} X/Y : f \quad Y : g &\implies X : f(g) \quad (>) \\ Y : g \quad X\backslash Y : f &\implies X : f(g) \quad (<) \end{aligned}$$

These rules mean that the complex category X/Y ($X\backslash Y$) behaves like a function which accepts an argument of type Y on its right (left) and returns a value of type X. Parsing amounts to sequentially applying these two rules, as shown in Figure 1. The result of parsing is an ordered pair, containing both a syntactic parse tree and an associated logical form. We refer to such an ordered pair as a semantic parse, or by using the letter ℓ.

Given a lexicon, there may be multiple semantic parses ℓ for a given phrase $\mathbf{w}$. Like context-free grammars (CFGs), CCGs can be extended to represent a probability distribution over parses $P(\ell|\mathbf{w};\theta)$ where θ is a parameter vector.

2.2 Knowledge Base

The main input to our system is a propositional knowledge base $K = (E, R, C, \Delta)$, containing entities E, categories C, relations R and relation instances Δ. Categories and relations are predicates which operate on entities and return truth values; categories $c \in C$ are one-place predicates ($\textsc{City}(e)$) and relations $r \in R$ are two-place predicates ($\textsc{LocatedIn}(e_1, e_2)$). Entities $e \in E$ represent real-world entities and have a set of known text names. For example, CALIFORNIA is an entity whose text names include "California" and "CA." Relation instances $r(e_1, e_2) \in \Delta$ are facts asserted by the knowledge base, such as $\textsc{LocatedIn}(\textsc{Sacramento}, \textsc{California})$. Examples of such knowledge bases include Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010), and YAGO (Suchanek et al., 2007).

The knowledge base influences the semantic parser in two ways. First, CCG logical forms are constructed by combining categories, relations and entities from the knowledge base with logical connectives; hence, the predicates in the knowledge base determine the expressivity of the parser's semantic representation. Second, the known relation

instances $r(e_1, e_2) \in \Delta$ are used as weak supervision to train the semantic parser.

3 Weakly Supervised Semantic Parsing

We define *weakly supervised semantic parsing* as the following learning problem.

Input:

1. A knowledge base $K = (E, R, C, \Delta)$, as defined above.
2. A corpus of dependency-parsed sentences S.
3. A CCG lexicon Λ that produces logical forms containing predicates from K. Section 4.1 describes an approach to generate this lexicon.
4. A procedure for identifying mentions of entities from K in sentences from S. (e.g., simple string matching).

Output:

1. Parameters θ for the CCG that produce correct semantic parses ℓ for sentences $s \in S$.

This problem is ill-posed without additional assumptions: since the correct logical form for a sentence is never observed, there is no *a priori* reason to prefer one semantic parse to another. Our training algorithm makes two assumptions about correct semantic parses, which are encoded as weak supervision constraints. These constraints make learning possible by adding an inductive bias:

1. Every relation instance $r(e_1, e_2) \in \Delta$ is expressed by at least one sentence in S (Riedel et al., 2010; Hoffmann et al., 2011).

2. The correct semantic parse of a sentence s contains a subset of the syntactic dependencies contained in a dependency parse of s.

Our weakly supervised training uses these constraints as a proxy for labeled semantic parses. The training algorithm has two steps. First, the algorithm constructs a graphical model that contains both the semantic parser and constant factors encoding the above two constraints. This graphical model is then used to estimate parameters θ for the semantic parser, essentially optimizing θ to produce parses that satisfy the weak supervision constraints. If our assumptions are correct and sufficiently constrain the parameter space, then this procedure will identify parameters for an accurate semantic parser.

3.1 Encoding the Weak Supervision Constraints

The first step of training constructs a graphical model containing the semantic parser and two weak supervision constraints. However, the first weak supervision constraint couples the semantic parses for *every* sentence $s \in S$. Such coupling would result in an undesirably large graphical model. We therefore modify this constraint to enforce that every relation $r(e_1, e_2)$ is expressed at least once in $S_{(e_1,e_2)} \subseteq S$, the subset of sentences which mention both e_1 and e_2. These mentions are detected using the provided mention-identification procedure.

Figure 2 depicts the graphical model constructed for training. The semantic constraint couples the extractions for all sentences $S_{(e_1,e_2)}$, so the graphical model is instantiated once per (e_1, e_2) tuple. The model has 4 types of random variables and values: $S_i = s_i$ represents a sentence, $L_i = \ell_i$ represents a semantic parse, $Z_i = z_i$ represents the satisfaction of the syntactic constraint and $Y_r = y_r$ represents the truth value of relation r. S_i, L_i and Z_i are replicated once for each sentence $s \in S_{(e_1,e_2)}$, while Y_r is replicated once for each relation type r in the knowledge base (all $r \in R$).

For each entity pair (e_1, e_2), this graphical model defines a conditional distribution over $\mathbf{L}, \mathbf{Y}, \mathbf{Z}$ given $\mathbf{S}$. This distribution factorizes as:

$$p(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z}, \mathbf{L} = \boldsymbol{\ell} | \mathbf{S} = \mathbf{s}; \theta) = \frac{1}{Z_{\mathbf{s}}} \prod_r \Psi(y_r, \boldsymbol{\ell}) \prod_i \Phi(z_i, \ell_i, s_i) \Gamma(s_i, \ell_i; \theta)$$

The factorization contains three replicated factors. Γ represents the semantic parser, which is parametrized by θ and produces a semantic parse ℓ_i for each sentence s_i. Ψ and Φ are deterministic factors representing the two weak supervision constraints. We now describe each factor in more detail.

Semantic Parser

The factor Γ represents the semantic parser, which is a log-linear probabilistic CCG using the input lexicon Λ. Given a sentence s and parameters θ, the parser defines an unnormalized probability distribution over semantic parses ℓ, each of which includes both a syntactic CCG parse tree and logical form.

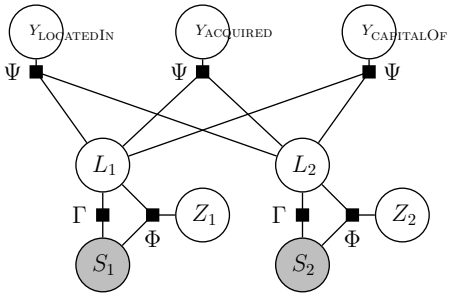

Figure 2: Factor graph containing the semantic parser Γ and weak supervision constraints Ψ and Φ, instantiated for an (e_1, e_2) tuple occurring in 2 sentences S_1 and S_2, with corresponding semantic parses L_1 and L_2. The knowledge base contains 3 relations, represented by the Y variables.

Let $f(\ell, s)$ represent a feature function mapping semantic parses to vectors of feature values[1]. The factor Γ is then defined as:

$$\Gamma(s, \ell; \theta) \quad = \quad \exp\{\theta^T f(\ell, s)\}$$

If the features $f(\ell, s)$ factorize according to the structure of the CCG parse tree, it is possible to perform exact inference using a CKY-style dynamic programming algorithm. However, other aspects of the graphical model preclude exact inference, so we perform approximate inference using beam search. Inference is explained in more detail in Section 3.2.

Semantic Constraint

The semantic constraint states that, given an entity tuple (e_1, e_2), every relation instance $r(e_1, e_2) \in \Delta$ must be expressed somewhere in $S_{(e_1,e_2)}$. Furthermore, no semantic parse can express a relation instance which is not in the knowledge base. This constraint is identical to the multiple deterministic-OR constraint used by Hoffmann et al. (2011) to train a sentential relation extractor.

The graphical model contains a semantic constraint factor Ψ and one binary variable Y_r for each relation r in the knowledge base. Y_r represents whether $r(e_1, e_2)$ is expressed by any sentence in $S_{(e_1,e_2)}$. The Ψ factor determines whether each semantic parse in $\boldsymbol{\ell}$ extracts a relation between e_1 and e_2. It then aggregates these sentence-level extractions using a deterministic OR: if any sentence extracts $r(e_1, e_2)$ then $Y_r = 1$. Otherwise, $Y_r = 0$.

[1]Section 4.3 describes the features used by our semantic parser for Freebase.

$$\Psi(Y_r, \boldsymbol{\ell}) =$$
$$1 \text{ if } Y_r = 1 \wedge \exists i.\textsc{Extracts}(\ell_i, r, e_1, e_2)$$
$$1 \text{ if } Y_r = 0 \wedge \nexists i.\textsc{Extracts}(\ell_i, r, e_1, e_2)$$
$$0 \text{ otherwise}$$

The EXTRACTS function determines the relation instances that are asserted by a semantic parse ℓ. $\textsc{Extracts}(\ell, r, e_1, e_2)$ is true if ℓ asserts the relation $r(e_1, e_2)$ and false otherwise. This function essentially converts the semantic parser into a sentential relation extractor, and its implementation may depend on the types of logical connectives included in the lexicon Λ. Logical forms in our Freebase semantic parser consist of conjunctions of predicates from the knowledge base; we therefore define $\textsc{Extracts}(\ell, r, e_1, e_2)$ as true if ℓ's logical form contains the clauses $r(x, y)$, $x = e_1$ and $y = e_2$.

Syntactic Constraint

A problem with the semantic constraint is that it admits a large number of ungrammatical parses. The syntactic constraint penalizes ungrammatical parses by encouraging the semantic parser to produce parse trees that agree with a dependency parse of the same sentence. Specifically, the syntactic constraint requires the predicate-argument structure of the CCG parse to agree with the predicate-argument structure of the dependency parse.

Agreement is defined as a function of each CCG rule application in ℓ. In the parse tree ℓ, each rule application combines two subtrees, ℓ_h and ℓ_c, into a single tree spanning a larger portion of the sentence. A rule application is consistent with a dependency parse t if the head words of ℓ_h and ℓ_c have a dependency edge between them in t. $\textsc{Agree}(\ell, t)$ is true if and only if every rule application in ℓ is consistent with t. This syntactic constraint is encoded in the graphical model by the Φ factors and $\mathbf{Z}$ variables:

$$\Phi(z, \ell, s) \quad = \quad 1 \text{ if } z = \textsc{Agree}(\ell, \textsc{DepParse}(s))$$
$$0 \text{ otherwise}$$

3.2 Parameter Estimation

To train the model, a single training example is constructed for every tuple of entities (e_1, e_2). The input to the model is $\mathbf{s} = S_{(e_1,e_2)}$, the set of sentences

containing e_1 and e_2. The weak supervision variables, $\mathbf{y}, \mathbf{z}$, are the output of the model. $\mathbf{y}$ is constructed by setting $y_r = 1$ if $r(e_1, e_2) \in \Delta$, and 0 otherwise. This setting trains the semantic parser to extract every true relation instance between (e_1, e_2) from some sentence in $S_{(e_1,e_2)}$, while simultaneously avoiding incorrect instances. Finally, $\mathbf{z} = \mathbf{1}$, to encourage agreement between the semantic and dependency parses. The training data for the model is therefore a collection, $\{(\mathbf{s}^j, \mathbf{y}^j, \mathbf{z}^j)\}_{j=1}^n$, where j indexes entity tuples (e_1, e_2).

Training optimizes the semantic parser parameters θ to predict $\mathbf{Y} = \mathbf{y}^j, \mathbf{Z} = \mathbf{z}^j$ given $\mathbf{S} = \mathbf{s}^j$. The parameters θ are estimated by running the structured perceptron algorithm (Collins, 2002) on the training data defined above. The structured perceptron algorithm iteratively applies a simple update rule for each example $(\mathbf{s}^j, \mathbf{y}^j, \mathbf{z}^j)$ in the training data:

$$\begin{aligned}
\boldsymbol{\ell}^{predicted} &\leftarrow \arg\max_{\boldsymbol{\ell}} \max_{\mathbf{y},\mathbf{z}} p(\boldsymbol{\ell}, \mathbf{y}, \mathbf{z} | \mathbf{s}^j; \theta^t) \\
\boldsymbol{\ell}^{actual} &\leftarrow \arg\max_{\boldsymbol{\ell}} p(\boldsymbol{\ell} | \mathbf{y}^j, \mathbf{z}^j, \mathbf{s}^j; \theta^t) \\
\theta^{t+1} &\leftarrow \theta^t + \sum_i f(\ell_i^{actual}, s_i) \\
&\quad - \sum_i f(\ell_i^{predicted}, s_i)
\end{aligned}$$

Each iteration of training requires solving two maximization problems. The first maximization, $\max_{\boldsymbol{\ell},\mathbf{y},\mathbf{z}} p(\boldsymbol{\ell}, \mathbf{y}, \mathbf{z} | \mathbf{s}; \theta^t)$, is straightforward because $\mathbf{y}$ and $\mathbf{z}$ are deterministic functions of $\boldsymbol{\ell}$. Therefore, it is solved by finding the maximum probability assignment $\boldsymbol{\ell}$, then choosing values for $\mathbf{y}$ and $\mathbf{z}$ that satisfy the weak supervision constraints.

The second maximization, $\max_{\boldsymbol{\ell}} p(\boldsymbol{\ell} | \mathbf{y}, \mathbf{z}, \mathbf{s}; \theta^t)$, is more challenging. When $\mathbf{y}$ and $\mathbf{z}$ are given, the inference procedure must restrict its search to the parses $\boldsymbol{\ell}$ which satisfy these weak supervision constraints. The original formulation of the Ψ factors permitted tractable inference (Hoffmann et al., 2011), but the EXTRACTS function and the Φ factors preclude efficient inference. We approximate this maximization using beam search over CCG parses ℓ. For each sentence s, we perform a beam search to produce $k = 300$ possible semantic parses. We then check the value of Φ for each generated parse and eliminate parses which do not satisfy this syntactic constraint. Finally, we apply EXTRACTS to each parse, then use the greedy approximate inference procedure from Hoffmann et al. (2011) for the Ψ factors.

4 Building a Grammar for Freebase

We apply the training algorithm from the previous section to produce a semantic parser for a subset of Freebase. This section describes details of the grammar we construct for this task, including the construction of the lexicon Λ, some extensions to the CCG parser, and the features used during training. In this section, we assume access to a knowledge base $K = (E, C, R, \Delta)$, a corpus of dependency-parsed sentences S and a procedure for identifying mentions of entities in sentences.

4.1 Constructing the Lexicon Λ

The first step in constructing the semantic parser is defining a lexicon Λ. We construct Λ by applying simple dependency-parse-based heuristics to sentences in the training corpus. The resulting lexicon Λ captures a variety of linguistic phenomena, including verbs, common nouns ("city"), noun compounds ("California city") and prepositional modifiers ("city in California").

The first step in lexicon construction is to use the mention identification procedure to identify all mentions of entities in the sentences S. This process results in (e_1, e_2, s) triples, consisting of sentences with two entity mentions. The dependency path between e_1 and e_2 in s is then matched against the dependency parse patterns in Table 1. Each matched pattern adds one or more lexical entries to Λ

Each pattern in Table 1 has a corresponding lexical category template, which is a CCG lexical category containing parameters e, c and r that are chosen at initialization time. Given the triple (e_1, e_2, s), relations r are chosen such that $r(e_1, e_2) \in \Delta$, and categories c are chosen such that $c(e_1) \in \Delta$ or $c(e_2) \in \Delta$. The template is then instantiated with every combination of these e, c and r values.

After instantiating lexical categories for each sentence in S, we prune infrequent lexical categories to improve parser efficiency. This pruning step is required because the common noun pattern generates a large number of lexical categories, the majority of which are incorrect. Therefore, we eliminate all common noun categories instantiated by fewer than

Part of Speech	Dependency Parse Pattern	Lexical Category Template
Proper Noun	(name of entity e) Sacramento	w := $N : \lambda x.x = e$ Sacramento := $N : \lambda x.x = \text{SACRAMENTO}$
Common Noun	$e_1 \xrightarrow{SBJ}$ [is, are, was, ...] $\xleftarrow{OBJ}$ w Sacramento is the capital	w := $N : \lambda x.c(x)$ capital := $N : \lambda x.\text{CITY}(x)$
Noun Modifier	$e_1 \xleftarrow{NMOD} e_2$ Sacramento, California	Type change $N : \lambda x.c(x)$ to $N\|N : \lambda f.\lambda x.\exists y.c(x) \wedge f(y) \wedge r(x,y)$ $N : \lambda x.\text{CITY}(x)$ to $N\|N : \lambda f.\lambda x.\exists y.\text{CITY}(x) \wedge f(y) \wedge \text{LOCATEDIN}(x,y)$
Preposition	$e_1 \xleftarrow{NMOD}$ w $\xleftarrow{PMOD} e_2$ Sacramento in California $e_1 \xrightarrow{SBJ}$ VB* $\xleftarrow{ADV}$ w $\xleftarrow{PMOD} e_2$ Sacramento is located in California	w := $(N\backslash N)/N : \lambda f.\lambda g.\lambda x.\exists y.f(y) \wedge g(x) \wedge r(x,y)$ in := $(N\backslash N)/N : \lambda f.\lambda g.\lambda x.\exists y.f(y) \wedge g(x) \wedge \text{LOCATEDIN}(x,y)$ w := $PP/N : \lambda f.\lambda x.f(x)$ in := $PP/N : \lambda f.\lambda x.f(x)$
Verb	$e_1 \xrightarrow{SBJ}$ w* $\xleftarrow{OBJ} e_2$ Sacramento governs California $e_1 \xrightarrow{SBJ}$ w* $\xleftarrow{ADV}$ [IN,TO] $\xleftarrow{PMOD} e_2$ Sacramento is located in California $e_1 \xleftarrow{NMOD}$ w* $\xleftarrow{ADV}$ [IN,TO] $\xleftarrow{PMOD} e_2$ Sacramento located in California	w* := $(S\backslash N)/N : \lambda f.\lambda g.\exists x,y.f(y) \wedge g(x) \wedge r(x,y)$ governs := $(S\backslash N)/N : \lambda f.\lambda g.\exists x,y.f(y) \wedge g(x) \wedge \text{LOCATEDIN}(x,y)$ w* := $(S\backslash N)/PP : \lambda f.\lambda g.\exists x,y.f(y) \wedge g(x) \wedge r(x,y)$ is located := $(S\backslash N)/PP : \lambda f.\lambda g.\exists x,y.f(y) \wedge g(x) \wedge \text{LOCATEDIN}(x,y)$ w* := $(N\backslash N)/PP : \lambda f.\lambda g.\lambda y.f(y) \wedge g(x) \wedge r(x,y)$ located := $(N\backslash N)/PP : \lambda f.\lambda g.\lambda y.f(y) \wedge g(x) \wedge \text{LOCATEDIN}(x,y)$
Forms of "to be"	(none)	w* := $(S\backslash N)/N : \lambda f.\lambda g.\exists x.g(x) \wedge f(x)$

Table 1: Dependency parse patterns used to instantiate lexical categories for the semantic parser lexicon Λ. Each pattern is followed by an example phrase that instantiates it. An * indicates a position that may be filled by multiple consecutive words in the sentence. e_1 and e_2 are the entities identified in the sentence, r represents a relation where $r(e_1, e_2)$, and c represents a category where $c(e_1)$. Each template may be instantiated with multiple values for the variables e, c, r.

5 sentences in S. The other rules are less fertile, so we do not need to prune their output.

In addition to these categories, the grammar includes type-changing rules from N to $N|N$. These rules capture noun compounds by allowing nouns to become functions from nouns to nouns. There are several such type-changing rules since the resulting category includes a hidden relation r between the noun and its modifier (see Table 1). As with lexical categories, the set of type changing rules included in the grammar is determined by matching dependency parse patterns to the training data. Similar rules for noun compounds are used in other CCG parsers (Clark and Curran, 2007).

The instantiated lexicon represents the semantics of words and phrases as conjunctions of predicates from the knowledge base, possibly including existentially quantified variables and λ expressions. The syntactic types N and PP are semantically represented as functions from entities to truth values (e.g., $\lambda x.\text{CITY}(x)$), while sentences S are statements with no λ terms, such as $\exists x, y.x = \text{CALIFORNIA} \wedge \text{CITY}(y) \wedge \text{LOCATEDIN}(x, y)$. Variables in the semantic representation (x, y) range over entities from the knowledge base. Intuitively, the N and PP categories represent sets of entities, while sentences represent assertions about the world.

4.2 Extensions to CCG

The semantic parser is trained using sentences from a web corpus, which contains many out-of-domain words. As a consequence, many of the words encountered during training cannot be represented using the vocabulary of predicates from the knowledge base. To handle these extraneous words, we allow the CCG parser to skip words while parsing a sentence. During parsing, the parser first decides whether to retrieve a lexical category for each word in the sentence. The sentence is then parsed as if only the retrieved lexical categories existed.

4.3 Features

The features $f(\ell, s)$ for our probabilistic CCG contain two sets of features. The first set contains lexical features, which count the number of times each lexical entry is used in ℓ. The second set contains rule application features, which count the number of times each combination rule is applied to each possible set of arguments. An argument is defined by its syntactic and semantic category, and in some cases by the lexical entry which created it. We lex-

icalize arguments for prepositional phrases PP and common nouns (initialized by the second rule in Table 1). This lexicalization allows the parser to distinguish between prepositional phrases headed by different prepositions, as well as between different common nouns. All other types are distinguished solely by syntactic and semantic category.

5 Evaluation

In this section, we evaluate the performance of a semantic parser for Freebase, trained using our weakly-supervised algorithm. Empirical comparison is somewhat difficult because the most comparable previous work – weakly-supervised relation extraction – uses a shallower semantic representation. Our evaluation therefore has two components: (1) a binary relation extraction task, to demonstrate that the trained semantic parser extracts instances of binary relations with performance comparable to other state-of-the-art systems, and (2) a natural language database query task, to demonstrate the parser's ability to extract more complex logical forms than binary relation instances, such as logical expressions involving conjunctions of multiple categories and relations with partially shared arguments.

5.1 Corpus Construction

Our experiments use a subset of 77 relations[2] from Freebase[3] as the knowledge base and a corpus of web sentences. We constructed the sentence corpus by first sampling sentences from a web crawl and parsing them with MaltParser (Nivre et al., 2006). Long sentences tended to have noisy parses while also rarely expressing relations, so we discarded sentences longer than 10 words. Entities were identified by performing a simple string match between canonical entity names in Freebase and proper noun phrases identified by the parser. In cases where a single noun phrase matched multiple entities, we selected the entity participating in the most relations. The resulting corpus contains 2.5 million (e_1, e_2, s) triples, from which we reserved 10% for validation and 10% for testing. The validation set was used to estimate performance during algorithm development, while the test set was used to generate the final experimental results. All triples for each (e_1, e_2) tuple were placed in the same set.

Relation Name	Relation Instances	Sentences
CITYLOCATEDINSTATE	2951	13422
CITYLOCATEDINCOUNTRY	1696	7904
CITYOFPERSONBIRTH	397	440
COMPANIESHEADQUARTEREDHERE	326	432
MUSICARTISTMUSICIAN	251	291
CITYUNIVERSITIES	239	338
CITYCAPITALOFCOUNTRY	123	2529
HASHUSBAND	103	367
PARENTOFPERSON	85	356
HASSPOUSE	81	461

Table 2: Occurrence statistics for the 10 most frequent relations in the training data. "Relation Instances" shows the number of entity tuples (e_1, e_2) that appear as positive examples for each relation, and "Sentences" shows the total number of sentences in which these tuples appear.

Approximately 1% of the resulting (e_1, e_2, s) triples are positive examples, meaning there exists some relation r where $r(e_1, e_2) \in \Delta$[4]. To improve training efficiency and prediction performance, we subsample 5% of the negative examples for training, producing a training set of 125k sentences with 27k positive examples. The validation and test sets retain the original positive/negative ratio. Table 2 shows some statistics of the most frequent relations in the test set.

5.2 Relation Extraction

The first experiment measures the semantic parser's ability to extract relations from sentences in our web corpus. We compare our semantic parser to MULTIR (Hoffmann et al., 2011), which is a state-of-the-art weakly supervised relation extractor. This method uses the same weak supervision constraint and parameter estimation procedure, but replaces the semantic parser by a linear classifier. The features for this classifier include the dependency path between the entity mentions, the type of each mention, and the intervening context (Mintz et al., 2009).

Both the semantic parser and MULTIR were trained by running 5 iterations of the structured per-

[2]These relations are defined by a set of MQL queries and potentially traverse multiple relation links.

[3]http://www.freebase.com

[4]Note that the positive/negative ratio was much lower without the length filter or entity disambiguation, which is partly why filtering was performed.

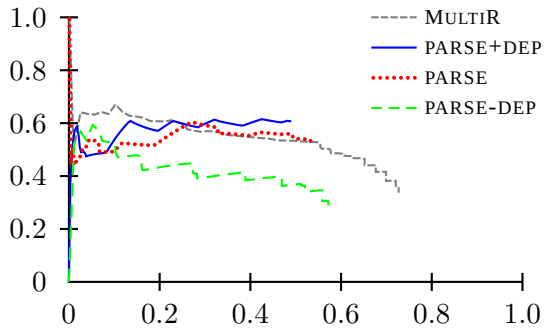

Figure 3: Aggregate precision as a function of recall, for MULTIR (Hoffman et al., 2011) and our three semantic parser variants.

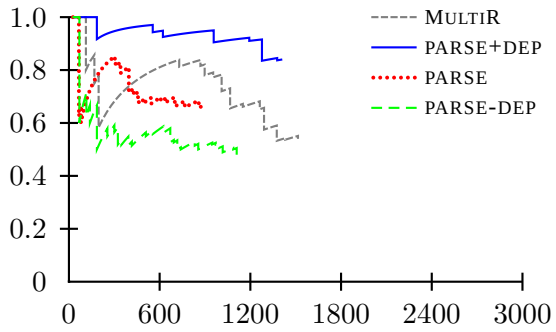

Figure 4: Sentential precision as a function of the expected number of correct extractions for MULTIR (Hoffman et al., 2011) and our three semantic parser variants.

ceptron algorithm[5]. At test time, both models predicted a relation $r \in R$ or NONE for each (e_1, e_2, s) triple in the test set. The parser parses the sentence without considering the entities marked in the sentence, then applies the EXTRACTS function defined in Section 3.1 to identify a relation between e_1 and e_2. We compare three versions of the semantic parser: PARSE, which is the basic semantic parser, PARSE+DEP which additionally observes the correct dependency parse at test time, and PARSE-DEP which is trained without the syntactic constraint. Note that MULTIR uses the sentence's dependency parse to construct its feature vector.

Our evaluation considers two performance measures: aggregate and sentential precision/recall. Aggregate precision takes the union of all extracted relation instances $r(e_1, e_2)$ from the test corpus and compares these instances to Freebase. To produce a precision/recall curve, each extracted instance $r(e_1, e_2)$ is assigned the maximum score over all sentences which extracted it. This metric is easy to compute, but may be inaccurate due to inaccuracies and missing relations in Freebase.

Sentential precision computes the precision of extractions on individual (e_1, e_2, s) tuples. This metric is evaluated by manually sampling and evaluating 100 test sentences from which a relation was extracted per model. Unfortunately, it is difficult to compute recall for this metric, since the true number of sentences expressing relations is unknown. We instead report precision as a function of the expected number of correct extractions, which is directly proportional to recall.

Figure 3 displays aggregate precision/recall and Figure 4 displays sentential precision/recall for all 4 models. Generally, PARSE behaves like MULTIR with somewhat lower recall. In the sentential evaluation, PARSE+DEP outperforms both PARSE and MULTIR. The difference between PARSE+DEP's aggregate and sentential precision stems from the fact that PARSE+DEP extracts each relation instance from more sentences than either MULTIR or PARSE. PARSE-DEP has the worst performance in both evaluations, suggesting the importance of syntactic supervision. Precision in the aggregate experiment is low partially due to examples with incorrect entity disambiguation.

We found that the skewed distribution of relation types hides interesting differences between the models. Therefore, we include Figure 5 comparing our syntactically-supervised parsers to MULTIR, ignoring the two most frequent relations (which together

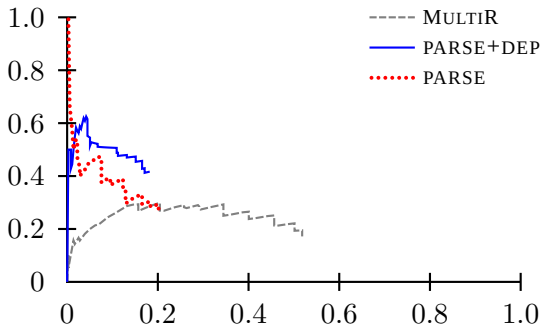

Figure 5: Aggregate precision as a function of recall, ignoring the two most frequent relations, CITYLOCATEDINSTATE and CITYLOCATEDINCOUNTRY.

[5]The structured perceptron algorithm does not converge to a parameter estimate, and we empirically found that performance did not improve beyond 5 iterations.

make up over half of all relation instances). Both PARSE and PARSE+DEP are considerably more precise than MULTIR on these less frequent relations because their compositional meaning representation shares parameter strength between relations. For example, the semantic parsers learn that "in" often combines with a city to form a prepositional phrase; the parsers can apply this knowledge to identify city arguments of any relation. However, MULTIR is capable of higher recall, since its dependency parse features can represent syntactic dependencies that cannot be represented by our semantic parsers. This limitation is a consequence of our heuristic lexicon initialization procedure, and could be rectified by a more flexible initialization procedure.

5.3 Natural Language Database Queries

The second experiment measures our trained parser's ability to correctly translate natural language queries into logical queries against Freebase.

To avoid biasing the evaluation, we constructed a test corpus of natural language queries in a data-driven fashion. We searched the test data for sentences with two related entities separated by an "is a" expression. The portion of the sentence before the "is a" expression was discarded and the remainder retained as a candidate query. For example "Jesse is an author from Austin, Texas," was converted into the candidate query "author from Austin, Texas." Each candidate query was then annotated with a logical form using categories and relations from the knowledge base; candidate queries without satisfactory logical forms were discarded. We annotated 50 validation and 50 test queries in this fashion. The validation set was used to estimate performance during algorithm development and the test set was used to generate the final results. Example queries with their annotated logical forms are shown in Table 3.

Table 4 displays the results of the query evaluation. For this evaluation, we forced the parser to include every word of the query in the parse. Precision is the percentage of successfully parsed queries for which the correct logical form was predicted. Recall is the percentage of all queries for which the correct logical form was predicted. This evaluation demonstrates that the semantic parser successfully interprets common nouns and identifies multiple relations with shared arguments. The performance difference between PARSE and PARSE-DEP also demonstrates the benefit of including syntactic supervision.

Example Query	Logical Form
capital of Russia	$\lambda x.\text{CITYCAPITALOFCOUNTRY}(x, \text{RUSSIA})$
wife of Abraham	$\lambda x.\text{HASHUSBAND}(x, \text{ABRAHAM})$
vocalist from London, England	$\lambda x.\text{MUSICIAN}(x) \wedge$ $\text{PERSONBORNIN}(x, \text{LONDON}) \wedge$ $\text{CITYINCOUNTRY}(\text{LONDON}, \text{ENGLAND})$
home of ConocoPhillips in Canada	$\lambda x.\text{HEADQUARTERS}(\text{CONOCOPHILLIPS}, x)$ $\wedge\text{CITYINCOUNTRY}(x, \text{CANADA})$

Table 3: Example natural language queries and their correct annotated logical form.

	Precision	Recall
PARSE	0.80	0.56
PARSE-DEP	0.45	0.32

Table 4: Precision and recall for predicting logical forms of natural language queries against Freebase. The table compares PARSE, trained with syntactic supervision to PARSE-DEP, trained without syntactic supervision.

Examining the system output, we find two major sources of error. The first is missing lexical categories for uncommon words (e.g., "ex-guitarist"), which negatively impact recall by making some queries unparsable. The second is difficulty distinguishing between relations with similar type signatures, such as CITYLOCATEDINCOUNTRY and CITYCAPITALOFCOUNTRY.

6 Related Work

There are many approaches to supervised semantic parsing, including inductive logic programming (Zelle and Mooney, 1996), probabilistic and synchronous grammars (Ge and Mooney, 2005; Wong and Mooney, 2006; Wong and Mooney, 2007; Lu et al., 2008), and automatically learned transformation rules (Kate et al., 2005). This work most closely follows the work on semantic parsing using CCG (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010). These supervised systems are all trained with annotated sentence/logical form pairs; hence these approaches are labor intensive and do not scale to broad domains with large numbers of predicates.

Several recent papers have attempted to reduce the amount of human supervision required to train

a semantic parser. One line of work eliminates the need for an annotated logical form, instead using only the correct answer for a database query (Liang et al., 2011) or even a binary correct/incorrect signal (Clarke et al., 2010). This type of feedback may be easier to obtain than full logical forms, but still requires individually annotated sentences. Other approaches are completely unsupervised, but do not tie the language to an existing meaning representation (Poon and Domingos, 2009). It is also possible to self-train a semantic parser without any labeled data (Goldwasser et al., 2011). However, this approach does not perform as well as more supervised approaches, since the parser's self-training predictions are not constrained by the correct logical form.

Recent research has produced several weakly supervised relation extractors (Craven and Kumlien, 1999; Mintz et al., 2009; Wu and Weld, 2010; Riedel et al., 2010; Hoffmann et al., 2011). These systems scale up to hundreds of predicates, but have much shallower semantic representations than semantic parsers. For example, these systems cannot be directly used to respond to natural language queries. This work extends weakly supervised relation extraction to produce richer semantic structure, using only slightly more supervision in the form of dependency parses.

7 Discussion

This paper presents a method for training a semantic parser using only a knowledge base and a corpus of unlabeled sentences. Our key observation is that multiple forms of weak supervision can be combined to train an accurate semantic parser: *semantic supervision* from a knowledge base of facts, and *syntactic supervision* in the form of a standard dependency parser. We presented an algorithm for training a semantic parser in the form of a probabilistic Combinatory Categorial Grammar, using these two types of weak supervision. We used this algorithm to train a semantic parser for an ontology of 77 Freebase predicates, using Freebase itself as the weak semantic supervision.

Experimental results show that our trained semantic parser extracts binary relations as well as a state-of-the-art weakly supervised relation extractor (Hoffmann et al., 2011). Further experiments tested our trained parser's ability to extract more complex meanings from sentences, including logical forms involving conjunctions of multiple relation and category predicates with shared arguments (e.g., $\lambda x.\text{MUSICIAN}(x) \wedge \text{PERSONBORNIN}(x, \text{LONDON}) \wedge \text{CITYINCOUNTRY}(\text{LONDON}, \text{ENGLAND})$). To test this capability, we applied the trained parser to natural language queries against Freebase. The semantic parser correctly interpreted 56% of these queries, despite the broad domain and never having seen an annotated logical form. Together, these two experimental analyses suggest that the combination of syntactic and semantic weak supervision is indeed a sufficient basis for training semantic parsers for a diverse range of corpora and predicate ontologies.

One limitation of our method is the reliance on hand-built dependency parse patterns for lexicon initialization. Although these patterns capture a variety of linguistic phenomena, they require manual engineering and may miss important relations. An area for future work is developing an automated way to produce this lexicon, perhaps by extending the recent work on automatic lexicon generation (Kwiatkowski et al., 2010) to the weakly supervised setting. Such an algorithm seems especially important if one wishes to model phenomena such as adjectives, which are difficult to initialize heuristically without generating large numbers of lexical entries.

An elegant aspect of semantic parsing is that it is easily extensible to include more complex linguistic phenomena, such as quantification and events (multi-argument relations). In the future, we plan to increase the expressivity of our parser's meaning representation to capture more linguistic and semantic phenomena. In this fashion, we can make progress toward broad coverage semantic parsing, and thus natural language understanding.

Acknowledgments

This research has been supported in part by DARPA under contract number FA8750-09-C-0179, and by a grant from Google. Additionally, we thank Yahoo! for use of their M45 cluster. We also gratefully acknowledge the contributions of our colleagues on the NELL project, Justin Betteridge for collecting the Freebase relations, Jamie Callan and colleagues for the web crawl, and Thomas Kollar and Matt Gardner for helpful comments on earlier drafts of this paper.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke S. Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon. Association for Computational Linguistics.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European conference on Machine learning and Knowledge Discovery in Databases*.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA. ACM.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th*

Annual Meeting of the Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*.

Luke S. Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.