Joint Syntactic and Semantic Parsing with Combinatory Categorial Grammar

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 an approach to training a joint syntactic and semantic parser that combines syntactic training information from CCGbank with semantic training information from a knowledge base via distant supervision. The trained parser produces a full syntactic parse of any sentence, while simultaneously producing logical forms for portions of the sentence that have a semantic representation within the parser's predicate vocabulary. We demonstrate our approach by training a parser whose semantic representation contains 130 predicates from the NELL ontology. A semantic evaluation demonstrates that this parser produces logical forms better than both comparable prior work and a pipelined syntax-then-semantics approach. A syntactic evaluation on CCGbank demonstrates that the parser's dependency F-score is within 2.5% of state-of-the-art.

1 Introduction

Integrating syntactic parsing with semantics has long been a goal of natural language processing and is expected to improve both syntactic and semantic processing. For example, semantics could help predict the differing prepositional phrase attachments in "I caught the butterfly with the net" and "I caught the butterfly with the spots." A joint analysis could also avoid propagating syntactic parsing errors into semantic processing, thereby improving performance.

We suggest that a large populated knowledge base should play a key role in syntactic and semantic parsing: in training the parser, in resolving syntactic ambiguities when the trained parser is applied to new text, and in its output semantic representation. Using semantic information from the knowledge base at training and test time will ideally improve the parser's ability to solve difficult syntactic parsing problems, as in the examples above. A semantic representation tied to a knowledge base allows for powerful inference operations – such as identifying the possible entity referents of a noun phrase – that cannot be performed with shallower representations (e.g., frame semantics (Baker et al., 1998) or a direct conversion of syntax to logic (Bos, 2005)).

This paper presents an approach to training a joint syntactic and semantic parser using a large background knowledge base. Our parser produces a full syntactic parse of every sentence, and furthermore produces logical forms for portions of the sentence that have a semantic representation within the parser's predicate vocabulary. For example, given a phrase like "my favorite town in California," our parser will assign a logical form like $\lambda x.\text{CITY}(x) \wedge \text{LOCATEDIN}(x, \text{CALIFORNIA})$ to the "town in California" portion. Additionally, the parser uses predicate and entity type information during parsing to select a syntactic parse.

Our parser is trained by combining a syntactic parsing task with a distantly-supervised relation extraction task. Syntactic information is provided by CCGbank, a conversion of the Penn Treebank into the CCG formalism (Hockenmaier and Steedman, 2002a). Semantics are learned by training the parser to extract knowledge base relation instances from a corpus of unlabeled sentences, in a distantly-supervised training regime. This approach uses the knowledge base to avoid expensive manual labeling of individual sentence semantics. By optimizing the parser to perform both tasks simultaneously, we train a parser that produces accurate syntactic and semantic analyses.

We demonstrate our approach by training a joint syntactic and semantic parser, which we call ASP. ASP produces a full syntactic analysis of every sentence while simultaneously producing logical forms containing any of 61 category and 69 relation predicates from NELL. Experiments with ASP demonstrate that jointly analyzing syntax and semantics improves semantic parsing performance over comparable prior work and a pipelined syntax-then-semantics approach. ASP's syntactic parsing performance is within 2.5% of state-of-the-art; however, we also find that incorporating semantic information reduces syntactic parsing accuracy by $\sim 0.5\%$.

2 Prior Work

This paper combines two lines of prior work: broad coverage syntactic parsing with CCG and semantic parsing.

Broad coverage syntactic parsing with CCG has produced both resources and successful parsers. These parsers are trained and evaluated using CCGbank (Hockenmaier and Steedman, 2002a), an automatic conversion of the Penn Treebank into the CCG formalism. Several broad coverage parsers have been trained using this resource (Hockenmaier and Steedman, 2002b; Hockenmaier, 2003b). The parsing model in this paper is loosely based on C&C (Clark and Curran, 2007b; Clark and Curran, 2007a), a discriminative log-linear model for statistical parsing. Some work has also attempted to automatically derive logical meaning representations directly from syntactic CCG parses (Bos, 2005; Lewis and Steedman, 2013). However, these approaches to semantics do not ground the text to beliefs in a knowledge base.

Meanwhile, work on semantic parsing has focused on producing semantic parsers for answering simple natural language questions (Zelle and Mooney, 1996; Ge and Mooney, 2005; Wong and Mooney, 2006; Wong and Mooney, 2007; Lu et al., 2008; Kate and Mooney, 2006; Zettlemoyer and Collins, 2005; Kwiatkowski et al., 2011). This line of work has typically used a corpus of sentences with annotated logical forms to train the parser. Recent work has relaxed the requisite supervision conditions (Clarke et al., 2010; Liang et al., 2011), but has still focused on simple questions. Finally, some work has looked at applying semantic parsing to answer queries against large knowledge bases, such as YAGO (Yahya et al., 2012) and Freebase (Cai and Yates, 2013b; Cai and Yates, 2013a; Kwiatkowski et al., 2013; Berant et al., 2013). Although this work considers a larger number (thousands) of predicates than we do, none of these systems are capable of parsing open-domain text. Our approach is most closely related to the distantly-supervised approach of Krishnamurthy and Mitchell (2012).

The parser presented in this paper can be viewed as a combination of both a broad coverage syntactic parser and a semantic parser trained using distant supervision. Combining these two lines of work has synergistic effects – for example, our parser is capable of semantically analyzing conjunctions and relative clauses based on the syntactic annotation of these categories in CCGbank. This synergy gives our parser a richer semantic representation than previous work, while simultaneously enabling broad coverage.

3 Parser Design

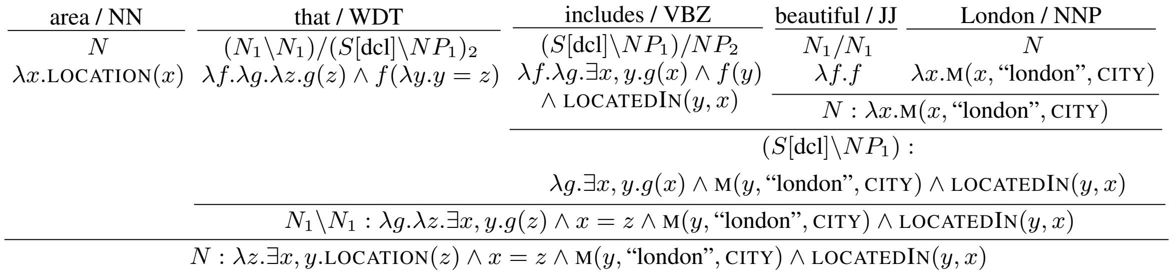
This section describes the Combinatory Categorial Grammar (CCG) parsing model used by ASP. The input to the parser is a part-of-speech tagged sentence, and the output is a syntactic CCG parse tree, along with zero or more logical forms representing the semantics of subspans of the sentence. These logical forms are constructed using category and relation predicates from a broad coverage knowledge base. The parser also outputs a collection of dependency structures summarizing the sentence's predicate-argument structure. Figure 1 illustrates ASP's input/output specification.

3.1 Knowledge Base

The parser uses category and relation predicates from a broad coverage knowledge base both to construct logical forms and to parametrize the parsing model. The knowledge base is assumed to have two kinds of ontological structure: a generalization/subsumption hierarchy and argument type constraints. This paper uses NELL's ontology (Carlson et al., 2010), which, for example, specifies that the category ORGANIZATION is a generalization of SPORTSTEAM, and that both arguments to the LOCATEDIN relation must have type LOCATION. These type constraints are enforced during parsing. Throughout this paper, predicate names are shown in SMALLCAPS.

3.2 Syntax

ASP uses a lexicalized and semantically-typed Combinatory Categorial Grammar (CCG) (Steedman, 1996). Most grammatical information in CCG is encoded in a lexicon Λ, containing entries such as:


area / NN
N
$\lambda x.\text{LOCATION}(x)$
that / WDT
$(N_1\backslash N_1)/(S[\text{dcl}]\backslash NP_1)_2$
$\lambda f.\lambda g.\lambda z.g(z) \wedge f(\lambda y.y = z)$
includes / VBZ
$(S[\text{dcl}]\backslash NP_1)/NP_2$
$\lambda f.\lambda g.\exists x, y.g(x) \wedge f(y) \wedge \text{LOCATEDIN}(y, x)$
beautiful / JJ
N_1/N_1
$\lambda f.f$
London / NNP
N
$\lambda x.\text{M}(x, \text{“london”}, \text{CITY})$
$N : \lambda x.\text{M}(x, \text{“london”}, \text{CITY})$
$(S[\text{dcl}]\backslash NP_1) :$
$\lambda g.\exists x, y.g(x) \wedge \text{M}(y, \text{“london”}, \text{CITY}) \wedge \text{LOCATEDIN}(y, x)$
$N_1\backslash N_1 : \lambda g.\lambda z.\exists x, y.g(z) \wedge x = z \wedge \text{M}(y, \text{“london”}, \text{CITY}) \wedge \text{LOCATEDIN}(y, x)$
$N : \lambda z.\exists x, y.\text{LOCATION}(z) \wedge x = z \wedge \text{M}(y, \text{“london”}, \text{CITY}) \wedge \text{LOCATEDIN}(y, x)$

Head						Argument			
word	POS	semantic type	index	syntactic category	arg. num.	word	POS	semantic type	index
that	WDT	—	1	$(N_1\backslash N_1)/(S\backslash NP_1)_2$	1	area	NN	LOCATION	0
that	WDT	—	1	$(N_1\backslash N_1)/(S\backslash NP_1)_2$	2	includes	VBZ	LOCATEDIN^{-1}	2
includes	VBZ	LOCATEDIN^{-1}	2	$(S[\text{dcl}]\backslash NP_1)/NP_2$	1	area	NN	LOCATION	0
includes	VBZ	LOCATEDIN^{-1}	2	$(S[\text{dcl}]\backslash NP_1)/NP_2$	2	ENTITY:CITY	NNP	CITY	4
beautiful	JJ	—	3	N_1/N_1	1	ENTITY:CITY	NNP	CITY	4

Figure 1: Example input and output for ASP. Given a POS-tagged sentence, the parser produces a CCG syntactic tree and logical form (top), and a collection of dependency structures (bottom).

person	:=	N : PERSON : $\lambda x.\text{PERSON}(x)$
London	:=	N : CITY : $\lambda x.\text{M}(x, \text{“london”}, \text{CITY})$
great	:=	N_1/N_1 : — : $\lambda f.\lambda x.f(x)$
bought	:=	$(S[\text{dcl}]\backslash NP_1)/NP_2$: ACQUIRED : $\lambda f.\lambda g.\exists x, y.f(y) \wedge g(x) \wedge \text{ACQUIRED}(x, y)$

Each lexicon entry maps a word to a syntactic category, semantic type, and logical form. CCG has two kinds of syntactic categories: atomic and functional. Atomic categories include N for noun and S for sentence. Functional categories are functions constructed recursively from atomic categories; these categories are denoted using slashes to separate the category's argument type from its return type. The argument type appears on the right side of the slash, and the return type on the left. The direction of slash determines where the argument must appear – $/$ means an argument on the right, and $\backslash$ means an argument on the left.

Syntactic categories in ASP are annotated with two additional kinds of information. First, atomic categories may have associated syntactic features given in square brackets. These features are used in CCGbank to distinguish variants of atomic syntactic categories, e.g., $S[\text{dcl}]$ denotes a declarative sentence. Second, each category is annotated with head and dependency information using subscripts. These subscripts are used to populate predicate-argument dependencies (described below), and to pass head information using unification. For example, the head of the parse in Figure 1 is "area," due to the coindexing of the argument and return categories in the category $N_1\backslash N_1$.

In addition to the syntactic category, each lexicon entry has a semantic type and a logical form. The semantic type is a category or relation predicate that concisely represents the word's semantics. The semantic type is used to enforce type constraints during parsing and to include semantics in the parser's parametrization. The logical form gives the full semantics of the word in lambda calculus. The parser also allows lexicon entries with the semantic type "—", representing words whose semantics cannot be expressed using predicates from the ontology.

Parsing in CCG combines adjacent categories using a small number of combinators, such as function application:

$$\begin{array}{lll} X/Y : f & Y : g & \Longrightarrow X : f(g) \\ Y : g & X\backslash Y : f & \Longrightarrow X : f(g) \end{array}$$

The first rule states that the category X/Y can be applied to the category Y, returning category X, and that the logical form f is applied to g to produce the logical form for the returned category. Head words and semantic types are also propagated to the returned category based on the annotated head-passing markup.

3.3 Dependency Structures

Parsing a sentence produces a collection of dependency structures which summarize the predicate-argument structure of the sentence. Dependency structures are 10-tuples, of the form:

$<$ head word, head POS, head semantic type, head word index, head word syntactic category, argument number, argument word, argument POS, argument semantic type, argument word index $>$

A dependency structure captures a relationship between a head word and its argument. During parsing, whenever a subscripted argument of a syntactic category is filled, a dependency structure is created between the head of the applied function and its argument. For example, in Figure 1, the first application fills argument 1 of "beautiful" with "London," creating a dependency structure.

3.4 Logical Forms

ASP performs a best-effort semantic analysis of every parsed sentence, producing logical forms for subspans of the sentence when possible. Logical forms are designed so that the meaning of a sentence is a universally- and existentially-quantified conjunction of predicates with partially shared arguments. This representation allows the parser to produce semantic analyses for a reasonable subset of language, including prepositions, verbs, nouns, relative clauses, and conjunctions.

Figure 1 shows a representative sample of a logical form produced by ASP. Generally, the parser produces a lambda calculus statement with several existentially-quantified variables ranging over entities in the knowledge base. The only exception to this rule is conjunctions, which are represented using a scoped universal quantifier over the conjoined predicates. Entity mentions appear in logical forms via a special mention predicate, M, instead of as database constants. For example, "London" appears as $\textsc{m}(x, \text{"london"}, \textsc{city})$, instead of as a constant like LONDON. The meaning of this mention predicate is that x is an entity which can be called "london" and belongs to the CITY category. This representation propagates uncertainty about entity references into the logical form where background knowledge can be used for disambiguation. For example, "London, England" is assigned a logical form that disambiguates "London" to a "London" located in "England."[1]

Lexicon entries without a semantic type are automatically assigned logical forms based on their head passing markup. For example, in Figure 1, the adjective "beautiful" is assigned $\lambda f.f$. This approach allows a logical form to be derived for most sentences, but (somewhat counterintuitively) can lose interesting logical forms from constituent subspans. For example, the preposition "in" has syntactic category $(N_1\backslash N_1)/N_2$, which results in the logical form $\lambda f.\lambda g.g$. This logical form discards any information present in the argument f. We avoid this problem by extracting a logical form from every subtree of the CCG parse.

[1]Specifically, $\lambda x.\exists y.\textsc{citylocatedincountry}(x, y) \wedge \textsc{m}(x, \text{"london"}, \textsc{city}) \wedge \textsc{m}(y, \text{"england"}, \textsc{country})$

3.5 Parametrization

The parser Γ is trained as a discriminative linear model of the following form:

$$\Gamma(\ell, d, t|s; \theta) \quad = \quad \theta^T \phi(d, t, s)$$

Given a parameter vector θ and a sentence s, the parser produces a score for a syntactic parse tree t, a collection of dependency structures d and a logical form ℓ. The score depends on features of the parse produced by the feature function ϕ.

ϕ contains four classes of features: lexicon features, combinator features, dependency features and dependency distance features (Table 1). These features are based on those of C&C (Clark and Curran, 2007b), modified to include semantic types. The features are designed to share syntactic information about a word across its distinct semantic realizations in order to transfer syntactic information from CCGbank to semantic parsing.

The parser also includes a hard type-checking constraint to ensure that logical forms are well-typed. This constraint states that dependency structures with a head semantic type only accept arguments that (1) have a semantic type, and (2) are within the domain/range of the head type.

4 Parameter Estimation

This section describes the training procedure for ASP. Training is performed by minimizing a joint objective function combining a syntactic parsing task and a distantly-supervised relation extraction task. The input training data includes:

1. A collection L of sentences s_i with annotated syntactic trees t_i (e.g., CCGbank).
2. A corpus of sentences S (e.g., Wikipedia).
3. A knowledge base K (e.g., NELL), containing relation instances $r(e_1, e_2) \in K$.
4. A CCG lexicon Λ (see Section 5.2).

Given these resources, the algorithm described in this section produces parameters θ for a semantic parser. Our parameter estimation procedure constructs a joint objective function $O(\theta)$ that decomposes into syntactic and semantic components: $O(\theta) = O_{\text{syn}}(\theta) + O_{\text{sem}}(\theta)$. The syntactic component O_{syn} is a standard syntactic parsing objective constructed using the syntactic resource L. The semantic component O_{sem} is a distantly-supervised relation extraction task based on the semantic constraint from Krishnamurthy and Mitchell (2012). These components are described in more detail in the following sections.

Lexicon features: word, POS $:= X : t : \ell$	
Word/syntactic category	word, X
POS/syntactic category	POS, X
Word semantics	word, X, t

Combinator features: $X\ Y \rightarrow Z$ or $X \rightarrow Z$	
Binary combinator indicator	$X\ Y \rightarrow Z$
Unary combinator indicator	$X \rightarrow Z$
Root syntactic category	Z

Dependency Features: $< h_w, h_p, h_t, h_i, s, n, a_w, a_p, a_t, a_i >$	
Predicate-Argument Indicator	$< h_w, —, h_t, —, s, n, a_w, —, a_t, — >$
Word-Word Indicator	$< h_w, —, —, —, s, n, a_w, —, —, — >$
Predicate-POS Indicator	$< h_w, —, h_t, —, s, n, —, a_p, —, — >$
Word-POS Indicator	$< h_w, —, —, —, s, n, —, a_p, —, — >$
POS-Argument Indicator	$< —, h_p, —, —, s, n, a_w, —, a_t, — >$
POS-Word Indicator	$< —, h_p, —, —, s, n, a_w, —, —, — >$
POS-POS Indicator	$< —, h_p, —, —, s, n, —, a_p, —, — >$

Dependency Distance Features:		
Token distance	$h_w, h_t, —, s, n, d$	$d =$ Number of tokens between h_i and a_i: 0, 1, 2 or more.
Token distance word backoff	$h_w, —, s, n, d$	$d =$ Number of tokens between h_i and a_i: 0, 1, 2 or more.
Token distance POS backoff	$—, —, h_p, s, n, d$	$d =$ Number of tokens between h_i and a_i: 0, 1, 2 or more.

(The above distance features are repeated using the number of intervening verbs and punctuation marks.)

Table 1: Listing of parser feature templates used in the feature function ϕ. Each feature template represents a class of indicator features that fire during parsing when lexicon entries are used, combinators are applied, or dependency structures are instantiated.

4.1 Syntactic Objective

The syntactic objective is the structured perceptron objective instantiated for a syntactic parsing task. This objective encourages the parser to accurately reproduce the syntactic parses in the annotated corpus $L = \{(s_i, t_i)\}_{i=1}^{n}$:

$$O_{\text{syn}}(\theta) = \sum_{i=1}^{n} |\max_{\hat{\ell},\hat{d},\hat{t}} \Gamma(\hat{\ell}, \hat{d}, \hat{t}|s_i; \theta) - \max_{\ell^*,d^*} \Gamma(\ell^*, d^*, t_i|s_i; \theta)|_+$$

The first term in the above expression represents the best CCG parse of the sentence s_i according to the current model. The second term is the best parse of s_i whose syntactic tree equals the true syntactic tree t_i. In the above equation $|\cdot|_+$ denotes the positive part of the expression. Minimizing this objective therefore finds parameters θ that reproduce the annotated syntactic trees.

4.2 Semantic Objective

The semantic objective corresponds to a distantly-supervised relation extraction task that constrains the logical forms produced by the semantic parser. Distant supervision is provided by the following constraint: every relation instance $r(e_1, e_2) \in K$ must be expressed by at least one sentence in $\mathbf{S}_{(e_1,e_2)}$, the set of sentences that mention both e_1 and e_2 (Hoffmann et al., 2011). If this constraint is empirically true and sufficiently constrains the parser's logical forms, then optimizing the semantic objective produces an accurate semantic parser.

A training example in the semantic objective consists of the set of sentences mentioning a pair of entities, $\mathbf{S}_{(e_1,e_2)} = \{s_1, s_2, ...\}$, paired with a binary vector representing the set of relations that the two entities participate in, $\mathbf{y}_{(e_1,e_2)}$. The distant supervision constraint Ψ forces the logical forms predicted for the sentences to entail the relations in $\mathbf{y}_{(e_1,e_2)}$. Ψ is a deterministic OR constraint that checks whether each logical form entails the relation instance $r(e_1, e_2)$, deterministically setting $y_r = 1$ if any logical form entails the instance and $y_r = 0$ otherwise.

Let $(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t})$ represent a collection of semantic parses for the sentences $\mathbf{S} = \mathbf{S}_{(e_1,e_2)}$. Let $\Gamma(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t}|\mathbf{S}; \theta) = \sum_{i=1}^{|\mathbf{S}|} \Gamma(\ell_i, d_i, t_i|s_i; \theta)$ represent the total weight assigned by the parser to a collection of parses for the sentences $\mathbf{S}$. For the pair of entities (e_1, e_2), the semantic objective is:

$$O_{\text{sem}}(\theta) = |\max_{\hat{\boldsymbol{\ell}},\hat{\boldsymbol{d}},\hat{\mathbf{t}}} \Gamma(\hat{\boldsymbol{\ell}}, \hat{\boldsymbol{d}}, \hat{\mathbf{t}}|\mathbf{S}; \theta) - \max_{\boldsymbol{\ell}^*,\boldsymbol{d}^*,\mathbf{t}^*} \big(\Psi(\mathbf{y}_{(e_1,e_2)}, \boldsymbol{\ell}^*, \boldsymbol{d}^*, \mathbf{t}^*) + \Gamma(\boldsymbol{\ell}^*, \boldsymbol{d}^*, \mathbf{t}^*|\mathbf{S}; \theta)\big)|_+$$

4.3 Optimization

Training minimizes the joint objective using the structured perceptron algorithm, which can be viewed as the stochastic subgradient method (Ratliff et al., 2006) applied to the objective $O(\theta)$. We initialize the parameters to zero, i.e., $\theta^0 = 0$. On each iteration, we sample either a syntactic example (s_i, t_i) or a semantic example $(\mathbf{S}_{(e_1,e_2)}, \mathbf{y}_{(e_1,e_2)})$. If a syntactic example is sampled, we apply the following parameter update:

$$\begin{aligned}
\hat{\ell}, \hat{d}, \hat{t} &\leftarrow \arg\max_{\ell,d,t} \Gamma(\ell, d, t|s_i; \theta^t) \\
\ell^*, d^* &\leftarrow \arg\max_{\ell,d} \Gamma(\ell, d, t_i|s_i; \theta^t) \\
\theta^{t+1} &\leftarrow \theta^t + \phi(d^*, t_i, s_i) - \phi(\hat{d}, \hat{t}, s_i)
\end{aligned}$$

This update moves the parameters toward the features of the best parse with the correct syntactic derivation, $\phi(d^*, t_i, s_i)$. If a semantic example is

	Labeled Dependencies			Unlabeled Dependencies			
	P	R	F	P	R	F	Coverage
ASP	85.58	85.31	85.44	91.75	91.46	91.60	99.63
ASP-SYN	86.06	85.84	85.95	92.13	91.89	92.01	99.63
C&C (Clark and Curran, 2007b)	88.34	86.96	87.64	93.74	92.28	93.00	99.63
(Hockenmaier, 2003a)	84.3	84.6	84.4	91.8	92.2	92.0	99.83

Table 2: Syntactic parsing results for Section 23 of CCGbank. Parser performance is measured using precision (P), recall (R) and F-measure (F) of labeled and unlabeled dependencies.

sampled, we instead apply the following update:

$$\begin{aligned}
\hat{\boldsymbol{\ell}}, \hat{\boldsymbol{d}}, \hat{\mathbf{t}} &\leftarrow \arg\max_{\boldsymbol{\ell},\boldsymbol{d},\mathbf{t}} \Gamma(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t} | \mathbf{S}_{(e_1,e_2)}; \theta^t) \\
\boldsymbol{\ell}^*, \boldsymbol{d}^*, \mathbf{t}^* &\leftarrow \arg\max_{\boldsymbol{\ell},\boldsymbol{d},\mathbf{t}} \Gamma(\boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t} | \mathbf{S}_{(e_1,e_2)}; \theta^t) \\
&\qquad + \Psi(\mathbf{y}_{(e_1,e_2)}, \boldsymbol{\ell}, \boldsymbol{d}, \mathbf{t}) \\
\theta^{t+1} &\leftarrow \theta^t + \phi(\boldsymbol{d}^*, \mathbf{t}^*, \mathbf{S}_{(e_1,e_2)}) \\
&\qquad - \phi(\hat{\boldsymbol{d}}, \hat{\mathbf{t}}, \mathbf{S}_{(e_1,e_2)})
\end{aligned}$$

This update moves the parameters toward the features of the best set of parses that satisfy the distant supervision constraint. Training outputs the average of each iteration's parameters, $\bar{\theta} = \frac{1}{n}\sum_{t=1}^{n}\theta^t$. In practice, we train the parser by performing a single pass over the examples in the data set.

All of the maximizations above can be performed exactly using a CKY-style chart parsing algorithm, except for the last one. This maximization is intractable due to the coupling between logical forms in $\boldsymbol{\ell}$ caused by enforcing the distant supervision constraint. We approximate this maximization in two steps. First, we perform a beam search to produce a list of candidate parses for each sentence $s \in \mathbf{S}_{(e_1,e_2)}$. We then extract relation instances from each parse and apply the greedy inference algorithm from Hoffmann et al., (2011) to identify the best set of parses that satisfy the distant supervision constraint. The procedure skips any examples with sentences that cannot be parsed (due to beam search failures) or where the distant supervision constraint cannot be satisfied.

5 Experiments

The experiments below evaluate ASP's syntactic and semantic parsing ability. The parser is trained on CCGbank and a corpus of Wikipedia sentences, using NELL's predicate vocabulary. The syntactic analyses of the trained parser are evaluated against CCGbank, and its logical forms are evaluated on an information extraction task and against an annotated test set of Wikipedia sentences.

5.1 Data Sets

The data sets for the evaluation consist of CCGbank, a corpus of dependency-parsed Wikipedia sentences, and a logical knowledge base derived from NELL and Freebase. Sections 02-21 of CCGbank were used for training, Section 00 for validation, and Section 23 for the final results. The knowledge base's predicate vocabulary is taken from NELL, and its instances are taken from Freebase using a manually-constructed mapping between Freebase and NELL. Using Freebase relation instances produces cleaner training data than NELL's automatically-extracted instances.

Using the relation instances and Wikipedia sentences, we constructed a data set for distantly-supervised relation extraction. We identified mentions of entities in each sentence using simple string matching, then aggregated these sentences by entity pair. 20% of the entity pairs were set aside for validation. In the remaining training data, we downsampled entity pairs that did not participate in at least one relation. We further eliminated sentences containing more than 30 tokens. The resulting training corpus contains 25k entity pairs (half of which participate in a relation), 41k sentences, and 71 distinct relation predicates.

5.2 Grammar Construction

The grammar for ASP contains the annotated lexicon entries and grammar rules in Sections 02-21 of CCGbank, and additional semantic entries produced using a set of dependency parse heuristics.

The lexicon Λ contains all words that occur at least 20 times in CCGbank. Rare words are replaced by their part of speech. The head passing and dependency markup was generated using the rules of the C&C parser (Clark and Curran, 2007b). These lexicon entries are also annotated with logical forms capturing their head passing relationship. For example, the adjective category N_1/N_1 is annotated with the logical form $\lambda f.f$. These entries are all assigned semantic type —.

We augment this lexicon with additional entries

Sentence	**Extracted Logical Form**
St. John, a Mexican-American born in San Francisco, California, her family comes from Zacatecas, Mexico.	$\lambda x.\exists y, z.$M$(x,$ "st. john"$)$ $\wedge$ M$(y,$ "san francisco"$)$ $\wedge$ PERSONBORNINLOCATION(x, y) $\wedge$ CITYLOCATEDINSTATE$(y, z) \wedge$ M$(z,$ "california"$)$
The capital and largest city of Laos is Vientiane and other major cities include Luang Prabang, Savannakhet and Pakse.	$\exists x, y.$M$(x,$ "vientiane"$)$ $\wedge$ CITY(x) $\wedge$ CITYCAPITALOFCOUNTRY$(x, y) \wedge$ M$(y,$ "laos"$)$
Gellar next played a lead role in James Toback 's critically unsuccessful independent "Harvard Man" (2001), where she played the daughter of a mobster.	$\lambda x.\exists y.$M$(y,$ "james toback"$)$ $\wedge$ DIRECTORDIRECTEDMOVIE(y, x) $\wedge$ M$(x,$ "harvard man"$)$

Figure 2: Logical forms produced by ASP for sentences in the information extraction corpus. Each logical form is extracted from the underlined sentence portion.

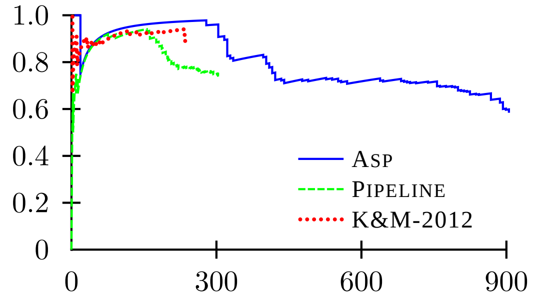


Figure 3: Logical form precision as a function of the expected number of correct extracted logical forms. ASP extracts more correct logical forms because it jointly analyzes syntax and semantics.

mapping words to logical forms with NELL predicates. These entries are instantiated using a set of dependency parse patterns, listed in an online appendix.[2] These patterns are applied to the training corpus, heuristically identifying verbs, prepositions, and possessives that express relations, and nouns that express categories. The patterns also include special cases for forms of "to be." This process generates ~4000 entries (not counting entity names), representing 69 relations and 61 categories from NELL. Section 3.2 shows several lexicon entries generated by this process.

The parser's combinators include function application, composition, and crossed composition, as well as several binary and unary type-changing rules that occur in CCGbank. All combinators were restricted to only apply to categories that combine in Sections 02-21. Finally, the grammar includes a number of heuristically-instantiated binary rules of the form , $N \rightarrow N\backslash N$ that instantiate a relation between adjacent nouns. These rules capture appositives and some other constructions.

5.3 Supertagging

Parsing in practice can be slow because the parser's lexicalized grammar permits a large number of parses for a sentence. We improve parser performance by performing *supertagging* (Bangalore and Joshi, 1999; Clark and Curran, 2004). We trained a logistic regression classifier to predict the syntactic category of each token in a sentence from features of the surrounding tokens and POS tags. Subsequent parsing is restricted to only consider categories whose probability is within a factor of α of the highest-scoring category. The parser uses a backoff strategy, first attempting to parse with the supertags from $\alpha = 0.01$, backing off to $\alpha = 0.001$ if the initial parsing attempt fails.

[2]`http://rtw.ml.cmu.edu/acl2014_asp/`

5.4 Syntactic Evaluation

The syntactic evaluation measures ASP's ability to reproduce the predicate-argument dependencies in CCGbank. As in previous work, our evaluation uses *labeled* and *unlabeled* dependencies. Labeled dependencies are dependency structures with both words and semantic types removed, leaving two word indexes, a syntactic category, and an argument number. Unlabeled dependencies further eliminate the syntactic category and argument number, leaving a pair of word indexes. Performance is measured using precision, recall, and F-measure against the annotated dependency structures in CCGbank. Precision is the fraction of predicted dependencies which are in CCGbank, recall is the fraction of CCGbank dependencies produced by the parser, and F-measure is the harmonic mean of precision and recall.

For comparison, we also trained a syntactic version of our parser, ASP-SYN, using only the CCGbank lexicon and grammar. Comparing against this parser lets us measure the effect of the relation extraction task on syntactic parsing.

Table 2 shows the results of our evaluation. For comparison, we include results for two existing syntactic CCG parsers: C&C, the current state-of-the-art CCG parser (Clark and Curran, 2007b), and the next best system (Hockenmaier, 2003a). Both ASP and ASP-SYN perform reasonably well, within 2.5% of the performance of C&C at the same coverage level. However, ASP-

	Logical Form Accuracy	Extraction Precision	Extraction Recall
ASP	0.28	0.90	0.32
K&M-2012	0.14	1.00	0.06
PIPELINE	0.2	0.63	0.17

Table 3: Logical form accuracy and extraction precision/recall on the annotated test set. The high extraction recall for ASP shows that it produces more complete logical forms than either baseline.

SYN outperforms ASP by around 0.5%, suggesting that ASP's additional semantic knowledge slightly hurts syntactic parsing performance. This performance loss appears to be largely due to poor entity mention detection, as we found that not using entity mention lexicon entries at test time improves ASP's labeled and unlabeled F-scores by 0.3% on Section 00. The knowledge base contains many infrequently-mentioned entities with common names; these entities contribute incorrect semantic type information that confuses the parser.

5.5 Semantic Evaluation

We performed two semantic evaluations to better understand ASP's ability to construct logical forms. The first evaluation emphasizes precision over recall, and the second evaluation accurately measures recall using a manually labeled test set.

5.5.1 Baselines

For comparison, we also trained two baseline models. The first baseline, PIPELINE, is a pipelined syntax-then-semantics approach designed to mimic Boxer (Bos, 2005). This baseline first syntactically parses each sentence using ASP-SYN, then produces a semantic analysis by assigning a logical form to each word. We train this baseline using the semantic objective (Section 4.2) while holding fixed the syntactic parse of each sentence. Note that, unlike Boxer, this baseline learns which logical form to assign each word, and its logical forms contain NELL predicates.

The second baseline, K&M-2012, is the approach of Krishnamurthy and Mitchell (2012), representing the state-of-the-art in distantly-supervised semantic parsing. This approach trains a semantic parser by combining distant semantic supervision with syntactic supervision from dependency parses. The best performing variant of this system also uses dependency parses at test time to constrain the interpretation of test sentences – hence, this system also uses a pipelined syntax-then-semantics approach. To improve comparability, we reimplemented this approach using our parsing model, which has richer features than were used in their paper.

5.5.2 Information Extraction Evaluation

The information extraction evaluation uses each system to extract logical forms from a large corpus of sentences, then measures the fraction of extracted logical forms that are correct. The test set consists of 8.5k sentences sampled from the held-out Wikipedia sentences. Each system was run on this data set, extracting all logical forms from each sentence that entailed at least one category or relation instance. We ranked these extractions using the parser's inside chart score, then manually annotated a sample of 250 logical forms from each system for correctness. Logical forms were marked correct if all category and relation instances entailed by the logical form were expressed by the sentence. Note that a correct logical form need not entail all of the relations expressed by the sentence, reflecting an emphasis on precision over recall. Figure 2 shows some example logical forms produced by ASP in the evaluation.

The annotated sample of logical forms allows us to estimate precision for each system as a function of the number of correct extractions (Figure 3). The number of correct extractions is directly proportional to recall, and was estimated from the total number of extractions and precision at each rank in the sample. All three systems initially have high precision, implying that their extracted logical forms express facts found in the sentence. However, ASP produces 3 times more correct logical forms than either baseline because it jointly analyzes syntax and semantics. The baselines suffer from reduced recall because they depend on receiving an accurate syntactic parse as input; syntactic parsing errors cause these systems to fail.

Examining the incorrect logical forms produced by ASP reveals that incorrect mention detection is by far the most common source of mistakes. Approximately 50% of errors are caused by marking common nouns as entity mentions (e.g., marking "coin" as a COMPANY). These errors occur because the knowledge base contains many infrequently mentioned entities with relatively common names. Another 30% of errors are caused by assigning an incorrect type to a common proper noun (e.g, marking "Bolivia" as a CITY). This analysis suggests that performing entity linking before parsing could significantly reduce errors.

Sentence: *De Niro and Joe Pesci in “Goodfellas”* offered a virtuoso display of the director’s bravura cinematic technique and reestablished, enhanced, and consolidated his reputation.
Annotation:
LF: $\lambda x.\forall p \in \{\lambda d.\textsc{m}(d, \text{“de niro”}), \lambda j.\textsc{m}(j, \text{“joe pesci”})\} \exists y.p(x) \wedge \textsc{starredInMovie}(x, y) \wedge \textsc{m}(y, \text{“goodfellas”})$
Instances: $\textsc{starredInMovie}(\text{de niro}, \text{goodfellas})$, $\textsc{starredInMovie}(\text{joe pesci}, \text{goodfellas})$
Prediction:
LF: $\lambda x.\forall p \in \{\lambda d.\textsc{m}(d, \text{“de niro”}), \lambda j.\textsc{m}(j, \text{“joe pesci”})\} \exists y.p(x) \wedge \textsc{starredInMovie}(x, y) \wedge \textsc{m}(y, \text{“goodfellas”})$
Instances: $\textsc{starredInMovie}(\text{de niro}, \text{goodfellas})$, $\textsc{starredInMovie}(\text{joe pesci}, \text{goodfellas})$
Logical form accuracy: 1 / 1 **Extraction Precision:** 2 / 2 **Extraction Recall:** 2 / 2

Sentence: In addition to the University of Illinois, *Champaign is also home to Parkland College*.
Annotation:
LF: $\exists c, p.\textsc{m}(c, \text{“champaign”}) \wedge \textsc{city}(c) \wedge \textsc{m}(p, \text{“parkland college”}) \wedge \textsc{universityInCity}(p, c)$
Instances: $\textsc{city}(\text{champaign})$, $\textsc{universityInCity}(\text{parkland college}, \text{champaign})$
Prediction:
LF 1: $\lambda x.\exists y \textsc{m}(y, \text{“illinois”}) \wedge \textsc{m}(x, \text{“university”}) \wedge \textsc{cityLocatedInState}(x, y)$
LF 2: $\exists c, p.\textsc{m}(c, \text{“champaign”}) \wedge \textsc{city}(c) \wedge \textsc{m}(p, \text{“parkland college”}) \wedge \textsc{universityInCity}(p, c)$
Instances: $\textsc{city}(\text{champaign})$, $\textsc{universityInCity}(\text{parkland college}, \text{champaign})$,
$\textsc{cityLocatedInState}(\text{university}, \text{illinois})$
Logical form accuracy: 1 / 1 **Extraction Precision:** 2 / 3 **Extraction Recall:** 2 / 2

Figure 4: Two test examples with ASP’s predictions and error calculations. The annotated logical forms are for the italicized sentence spans, while the extracted logical forms are for the underlined spans.

5.5.3 Annotated Sentence Evaluation

A limitation of the previous evaluation is that it does not measure the completeness of predicted logical forms, nor estimate what portion of sentences are left unanalyzed. We conducted a second evaluation to measure these quantities.

The data for this evaluation consists of sentences annotated with logical forms for subspans. We manually annotated Wikipedia sentences from the held-out set with logical forms for the largest subspans for which a logical form existed. To avoid trivial cases, we only annotated logical forms containing at least one category or relation predicate and at least one mention. We also chose not to annotate mentions of entities that are not in the knowledge base, as no system would be able to correctly identify them. The corpus contains 97 sentences with 100 annotated logical forms.

We measured performance using two metrics: logical form accuracy, and extraction precision/recall. Logical form accuracy examines the predicted logical form for the smallest subspan of the sentence containing the annotated span, and marks this prediction correct if it exactly matches the annotation. A limitation of this metric is that it does not assign partial credit to logical forms that are close to, but do not exactly match, the annotation. The extraction metric assigns partial credit by computing the precision and recall of the category and relation instances entailed by the predicted logical form, using those entailed by the annotated logical form as the gold standard. Figure 4 shows the computation of both error metrics on two examples from the test corpus.

Table 3 shows the results of the annotated sentence evaluation. ASP outperforms both baselines in logical form accuracy and extraction recall, suggesting that it produces more complete analyses than either baseline. The extraction precision of 90% suggests that ASP rarely extracts incorrect information. Precision is higher in this evaluation because every sentence in the data set has at least one correct extraction.

6 Discussion

We present an approach to training a joint syntactic and semantic parser. Our parser ASP produces a full syntactic parse of any sentence, while simultaneously producing logical forms for sentence spans that have a semantic representation within its predicate vocabulary. The parser is trained by jointly optimizing performance on a syntactic parsing task and a distantly-supervised relation extraction task. Experimental results demonstrate that jointly analyzing syntax and semantics triples the number of extracted logical forms over approaches that first analyze syntax, then semantics. However, we also find that incorporating semantics slightly reduces syntactic parsing performance. Poor entity mention detection is a major source of error in both cases, suggesting that future work should consider integrating entity linking with joint syntactic and semantic parsing.

Acknowledgments

This work was supported in part by DARPA under award FA8750-13-2-0005. We additionally thank Jamie Callan and Chris Ré’s Hazy group for collecting and processing the Wikipedia corpus.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 1*.

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Johan Bos. 2005. Towards wide-coverage semantic interpretation. In *In Proceedings of Sixth International Workshop on Computational Semantics IWCS-6*.

Qingqing Cai and Alexander Yates. 2013a. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Qingqing Cai and Alexander Yates. 2013b. Semantic Parsing Freebase: Towards Open-domain Semantic Parsing. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM)*.

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. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics*.

Stephen Clark and James R. Curran. 2007a. Perceptron training for a wide-coverage lexicalized-grammar parser. In *Proceedings of the Workshop on Deep Linguistic Processing*.

Stephen Clark and James R. Curran. 2007b. 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*.

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*.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of Third International Conference on Language Resources and Evaluation*.

Julia Hockenmaier and Mark Steedman. 2002b. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*.

Julia Hockenmaier. 2003a. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Julia Hockenmaier. 2003b. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*.

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 and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

Jayant Krishnamurthy and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

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

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*.

Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. 2006. (online) subgradient methods for structured prediction. *Artificial Intelligence and Statistics*.

Mark Steedman. 1996. *Surface Structure and Interpretation*. The MIT Press, Cambridge, MA, USA.

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*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

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*.