Empirically-motivated Generalizations of CCG Semantic Parsing Learning Algorithms

Jesse Glass
Temple University
1801 N Broad Street
Philadelphia, PA 19122, USA
jglassemc2@gmail.com

Alexander Yates
Temple University
1801 N Broad Street
Philadelphia, PA 19122, USA
ayates@gmail.com

Abstract

Learning algorithms for semantic parsing have improved drastically over the past decade, as steady improvements on benchmark datasets have shown. In this paper we investigate whether they can generalize to a novel biomedical dataset that differs in important respects from the traditional geography and air travel benchmark datasets. Empirical results for two state-of-the-art PCCG semantic parsers indicates that learning algorithms are sensitive to the kinds of semantic and syntactic constructions used in a domain. In response, we develop a novel learning algorithm that can produce an effective semantic parser for geography, as well as a much better semantic parser for the biomedical dataset.

1 Introduction

Semantic parsing is the task of converting natural language utterances into formal representations of their meaning. In this paper, we consider in particular a grounded form of semantic parsing, in which the meaning representation language takes its logical constants from a given, fixed ontology. Several recent systems have demonstrated the ability to learn semantic parsers for domains like the GeoQuery database containing geography relations, or the ATIS database of air travel information. In these settings, existing systems can produce correct meaning representations with F1 scores approaching 0.9 (Wong and Mooney, 2007; Kwiatkowski et al., 2011).

These benchmark datasets have supported a diverse and influential line of research into semantic parsing learning algorithms for sophisticated semantic constructions, with continuing advances in accuracy. However, the focus on these datasets leads to a natural question — do other natural datasets have similar syntax and semantics, and if not, can existing algorithms handle the variability in syntax and semantics?

In an effort to investigate and improve the generalization capacity of existing learning algorithms for semantic parsing, we develop a novel, natural experimental setting, and we test whether current semantic parsers generalize to the new setting. For our datset, we use descriptions of clinical trials of experimental drugs in the United States, available from the U.S. National Institutes of Health[1]. Much of the text in the description of these clinical trials can be mapped neatly onto biomedical ontologies, thus permitting grounded semantic analysis. Crucially, the dataset was not designed specifically with semantic parsing or question-answering in mind, and as a result, it provides a natural source for the variety and complexity of utterances that humans use in this domain. As an added benefit, a successful semantic parser in this domain could yield a variety of useful bioinformatics applications by permitting comparisons between and across clinical trials using structured representations of the data, rather than unstructured text.

In this initial investigation of semantic parsing in this context, we ask:

- *Can existing semantic parsing learning algorithms handle the variety and complexity of the clinical trials dataset?* We show that two representative learning algorithms fare poorly on the clinical trials data: the best one achieves a 0.41 F1 in our tests.

- *What types of constructions are the major cause of errors on the clinical trials dataset,*

[1]clinicaltrials.gov

and can semantic parsers be extended to handle them? While this initial investigation does not cover all types of constructions, we identify three important types of constructions that existing learning algorithms do not handle. We propose a new learning algorithm that can handle these types of constructions, and we demonstrate empirically that the new algorithm produces a semantic parser that improves by over 23 points in F1 on the clinical trials dataset compared with existing parsers.

The rest of this paper is organized as follows. The next section provides background information on CCG and semantic parsing. Section 3 describes the text and ontology that form the new clinical trials dataset for semantic parsing, as well as some of the problems that exising approaches have on this dataset. Sections 4 describes our semantic parsing model, and learning and inference algorithms. Section 5 presents our experiments and results, and Section 6 concludes.

2 Background on Semantic Parsing with CCG

Our approach to learning a semantic parser falls into the general framework of context-free Probabilistic Combinatory Categorial Grammars (PCCG) (Zettlemoyer and Collins, 2005) with typed lambda calculus expressions for the semantics. PCCG grammars involve lexical entries, which are weighted unary rewrite rules of the form Syntax : Semantics → Phrase. For example:

Example Lexical Entries

NP : $\texttt{melanoma} \rightarrow$ skin cancer
$S\backslash NP$: $\lambda p \lambda d.\texttt{has condition}(p, d) \rightarrow$ patients with

In addition to lexical rules, PCCG grammars involve weighted binary rewrite rules like the following:

Example CCG Grammar Rules

$X : f(g) \rightarrow X/Y : f \quad Y : g$ (function application)
$X : f(g) \rightarrow Y : g \quad X\backslash Y : f$ (backward application)

These rules apply for any syntactic categories X and Y, and any logical forms f and g. The rules specify mechanisms for deducing syntactic categories for whole phrases based on their constituent parts. They also specify mechanisms for identifying semantics (logical forms) for phrases and sentences based on combinations of the semantics for the constituent parts. Besides function application, other ways to combine the semantics of children typically include conjunction, disjunction, function composition, and substitution, among others. Inference algorithms for PCCG can identify the best parse and logical form for a given sentence using standard dynamic programming algorithms for context-free grammars (Clark and Curran, 2007).

As a baseline in our experiments, we use a learning algorithm for semantic parsing known as Unification Based Learning (UBL) (Kwiatkowski et al., 2010). Source code for UBL is freely available. Its authors found that the semantic parsers it learns achieve results competitive with the state-of-the-art on a variety of standard semantic parsing data sets, including GeoQuery (0.882 F1). UBL uses a log-linear probabilistic model $P(L, T|S)$ over logical forms L and parse tree derivations T, given sentences S. During training, only S and L are observed, and UBL's gradient-based parameter estimation algorithm tries to maximize $\sum_T P(L, T|S)$ over the training dataset. To learn lexicon entries, it adopts a search procedure that involves unification in higher-order logic. The objective of the search procedure is to identify lexical entries for the words in a sentence that, when combined with the lexical entries for other words in the sentence, will produce the observed logical form in the training data. For each training sentence, UBL heuristically explores the space of all possible lexical entries to produce a set of promising candidates, and adds them to the lexicon.

Our second baseline is an extension of this work, called Factored Unification Based Learning (FUBL) (Kwiatkowski et al., 2011). Again, source code is freely available. FUBL factors the lexicon into a set of base lexical entries, and a set of templates that can construct more complex lexical entries from the base entries. This allows for a significantly more compact lexicon, as well as the ability to handle certain linguistic constructions, like ellipsis, that appear frequently in the ATIS dataset and which UBL struggles with. FUBL achieves an F1 of 0.82 on ATIS (compared with 66.3 for UBL), and an F1 of 0.886 on GeoQuery; both results are at or very near the best-reported results for those datasets.

2.1 Previous Work

Many supervised learning frameworks have been applied to the task of learning a semantic parser, including inductive logic programming (Zelle and Mooney, 1996; Thompson and Mooney, 1999; Thompson and Mooney, 2003), support vector machine-based kernel approaches (Kate et al., 2005; Kate and Mooney, 2006; Kate and Mooney, 2007), machine translation-style synchronous grammars (Wong and Mooney, 2007), and context-free grammar-based approaches like probabilistic Combinatory Categorial Grammar (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Zettlemoyer and Collins, 2009; Kwiatkowski et al., 2010; Kwiatkowski et al., 2011; Lu et al., 2008) and discriminative reranking (Ge and Mooney, 2006; Ge and Mooney, 2009). These approaches have yielded steady improvements on standard test sets like GeoQuery. As far as we are aware, such systems have not been tested on domains besides ATIS and GeoQuery.

Because of the complexity involved in building a training dataset for a supervised semantic parser, there has been a recent push towards developing techniques which reduce the annotation cost or the data complexity of the models. Models have been developed which can handle some ambiguity in terms of which logical form is the correct label for each training sentence (Chen et al., 2010; Liang et al., 2009). Another set of approaches have investigated the case where no logical forms are provided, but instead some form of feedback or response from the world is used as evidence for what the correct logical form must have been (Clarke et al., 2010; Liang et al., 2011; Artzi and Zettlemoyer, 2011). Several projects have investigated unsupervised (Goldwasser et al., 2011; Poon, 2013; Krishnamurthy and Mitchell, 2012) and semi-supervised (Yahya et al., 2012; Cai and Yates, 2013) approaches. These techniques tend to handle either the same benchmark domains, or simpler questions over larger ontologies. While such techniques are important, their (unlabeled and labeled) sample complexity is higher than it could be, because the underlying grammars involved are not as general as they could be. Our work investigates techniques that will reduce this sample complexity.

3 The Clinical Trials Dataset

Clinical trials are scientific experiments that measure the effects of a medical procedure, instrument, or product on humans. Since September 2009 in the United States, any clinical trial that is funded by the federal government must make its results publicly available online at clinicaltrials.gov. This site provides a wealth of biomedical text and structured data, which we use to produce a novel test set for semantic parsing.

3.1 The text and ontology

We collected our utterances from a set of 47 random documents from clinicaltrials.gov. Many aspects of each study are reported in structured format; for example, the number of participants who were given a placebo and the number of participants who were given the intervention under consideration are both reported in a table in a standard format. However, certain crucial aspects of each study are reported only in text. Perhaps the most critical aspect of each study that is described only in text is the set of criteria for deciding who will be admitted to the study and who cannot be; these criteria are called inclusion criteria and exclusion criteria. We focus our semantic parsing tests on these criteria because they often form the longest portion of unstructured text for a given clinical trial report; because their meaning can be represented using a concise set of logical constants from a biomedical ontology; and because the criteria have a great deal of significance in the clinical trials domain. For example, these criteria are crucial for understanding why the results of two related studies about the same intervention might differ.

The criteria for a study can be logically represented as a function of candidate test subjects that returns true if they match the study criteria, and false otherwise. We use a variant of lambda calculus over a typed ontology to represent each inclusion and exclusion criterion in our dataset. We randomly collected 803 utterances and manually labeled each using our representation language. 401 were used for training, 109 for development, and 293 for our final tests.

To keep our semantic parsing study simple, we eschewed existing ontologies like UMLS (Bodenreider, 2004) that are large and overly-complex for this problem. We instead developed an ontology of 10 types, 38 relations and functions, and a dictionary of 591 named-entities to build the logical forms. The five most common types and relations in our dataset are listed in Table 1. On average, the logical forms in our dataset involved 3.7 relations per logical form, typically joined with conjunction, implication, or disjunction. If accepted, both the full ontology and dataset will be made publicly available.

3.2 Problems with semantic parsing the clinical trials data

We applied two state-of-the-art learning algorithms for learning PCCG semantic parsers — UBL and its extension, FUBL— to our training

Example Types	Example Functions
`(p)erson`	`t has condition(p, d)`
`(d)isease`	`t complication(d, d)`
`(t)est`	`i result(p, t)`
`(tr)eatment`	`t treated with(p, tr, date)`
`(bo)dy-part`	`t located(d, bo)`

Table 1: Common types and functions in our ontology. In the example functions, `t` indicates boolean type, `i` indicates real values, `p` indicates person, `d` disease, and so on.

patients with acute lymphoma λp . has $\mathtt{condition}(p, \mathtt{acute}(\mathtt{lymphoma}))$
hypertension (*i.e.*, include patients with hypertension) λp . has $\mathtt{condition}(p, \mathtt{hypertension})$
AST > 3 mg (*i.e.*, include patients with a level of the AST enzyme in the blood of greater than 3 mg) λp . $>(\mathtt{result}(p, \mathtt{AST}), \mathtt{unit}(3, \mathtt{mg}))$

Table 2: Example utterances from the clinical trials dataset, and their logical forms. Paraphrases in parentheses do not appear in the actual data.

data and tested the resulting parsers on development data. Results indicate that both systems have difficulty with the clinical trials datasets: FUBL achieves an F1 of 0.413, and UBL of just 0.281.

To help understand why state-of-the-art systems' performance differs so much from performance on benchmark datasets like GeoQuery, we performed an error analysis. Table 3 describes the most common errors we observed. The most common errors occurred on sentences containing coordination constructions, nested function applications, and for UBL, ellipsis, although a long tail of less common errors exists. FUBL manages to handle the elliptical constructions in Clinical Trials well, but not coordination or nested functions. Both systems tend to learn many, overly-specific lexical entries that include too much of the logical form in one lexical entry. For instance, from the coordination example in Table 3, UBL learns a lexical entry for the word "or" that includes the logical form $\lambda p \lambda d1 \lambda d2$. $\mathtt{or}(\mathtt{has}\ \mathtt{condition}(p, d1), \mathtt{has}\ \mathtt{condition}(p, d2))$. While this entry works well when coordinating two diseases or conditions that patients must have, it will not work for coordinations between treatments or dates, or coordinations between diseases that patients should not have. UBL learns over 250 lexical entries for the word "or" from our training dataset of 401 sentences, each one with limited applicability to development sentences.

Based on these observed error types, we next develop novel learning procedures that properly handle coordination, nested function constructions, and ellipsis.

4 Learning to Handle Complex Constructions in Clinical Trials Data

4.1 Model and Inference

We introduce the GLL system for learning a semantic parser that generalizes to both GeoQuery and Clinical Trials data. The semantic parsing model involves a grammar that consists of a fixed set of binary CCG rewrite rules, a learned lexicon Λ, and a new set $\mathbf{T}$ of learned templates for constructing unary type-raising rules. We call these templates for type-raising rules *T-rules*; these are described below in Section 4.4.

Following Kwiatkowsi *et al.*(2010), we assign a probability $P(L, T|S, \vec{\theta}, G)$ to a logical form and parse tree for a sentence licensed by grammar G using a log-linear model with parameters $\vec{\theta}$. We use a set of feature functions $\vec{F}(L, T, S) = (f_1(L, T, S), \ldots, f_K(L, T, S))$, where each f_i counts the number of times that the ith grammar rule is used in the derivation of T and S. The probability of a particular logical form given a sentence and $\vec{\theta}$ is given by:

$$P(L|S, \vec{\theta}, G) = \frac{\sum_T \exp(\vec{\theta} \cdot \vec{F}(L, T, S))}{\sum_{T', L'} \exp(\vec{\theta} \cdot \vec{F}(L', T', S))} \quad (1)$$

where the trees T (and T') are restricted to those that are licensed by G and which produce L (L') as the logical form for the parent node of the tree. Inference is performed using standard dynamic programming algorithms for context-free parsing.

4.2 Learning

The input for the task of learning a semantic parser is a set of sentences $\vec{S}$, where each $S_i \in \vec{S}$ has been labeled with a logical form $L(S_i)$. We assume a fixed set of binary grammar productions, and use the training data to learn lexical entries, T-rules, and parameters. The training objective is to maximize the likelihood of the observed logical

Error Type	Freq.	Example	Description
Nested Funcs.	27%	patients $>$ 18 years of age $\lambda p\ .\ > (\texttt{result}(\texttt{age}, p), \texttt{unit}(18, \texttt{year}))$	Many logical forms involve functions as arguments to other functions or relations.
Ellipsis	26%	diabetes $\lambda p\ .\ \texttt{has condition}(p, \texttt{diabetes})$	Many examples in the inclusion (exclusion) criteria simply list a disease or treatment, with the understanding that a patient p should be included (excluded) if p has the disease or is undergoing the treatment.
Coordination	16%	patient is pregnant or lactating $\lambda p\ .\ \texttt{or}(\texttt{has condition}(p, \texttt{pregnant}),$ $\texttt{has condition}(p, \texttt{lactating}))$	Clinical trials data has more coordination, especially noun phrase and adjective phrase coordination, than GeoQuery.

Table 3: Three common kinds of utterances in the clinical trials development set that caused UBL and FUBL to make errors. Frequency indicates the percentage of all development examples that exhibited that type of construction.

```
Input: set of labeled sentences $\{(S_i, L(S_i))\}$, initial grammar $G_0$, number of iterations $MAX$, learning rate $\alpha$

$\Lambda \leftarrow \emptyset$
$\forall i : \Lambda \leftarrow \Lambda \cup \{S : L(S_i) \rightarrow S_i\}$
$G \leftarrow G_0 \cup \Lambda$
$\vec{\theta} \leftarrow \vec{0}$
For $iteration$ := 1 to $MAX$:
    $TR \leftarrow$ TRLEARN($G$)
    Add dimension $\theta_t$ to $\vec{\theta}$ for $t \in TR - G$
    $G \leftarrow G \cup TR$
    For each sentence $S_i$ :
        $\Lambda \leftarrow$ LEXENTLEARN($S_i, L(S_i), G$)
        Add dimension $\theta_\lambda$ to $\vec{\theta}$ for all $\lambda \in \Lambda - G$
        $G \leftarrow G \cup \Lambda$
        $\vec{\theta} \leftarrow \vec{\theta} + \alpha \nabla_i CLL$
Return $G, \vec{\theta}$
```

Figure 1: The GLL Learning Algorithm. $\nabla_i CLL$ indicates the local gradient of the conditional log likelihood at sentence S_i.

forms, or to find G^* and $\vec{\theta^*}$ such that:

$$G^*, \vec{\theta^*} = \arg\max_{G,\vec{\theta}} \prod_i P(L(S_i)|S_i, \vec{\theta}, G)$$

This is a non-convex optimization problem. We use a greedy optimization procedure that iteratively updates G and $\vec{\theta}$. Figure 1 shows an overview of the full algorithm.

We use stochastic gradient updates to estimate parameters (LeCun et al., 1998). For each example sentence S_i in training, we compute the local gradient of the conditional log likelihood function $CLL = \log P(L(S_i)|S_i, \vec{\theta}, G)$, and update $\vec{\theta}$ by a step in the direction of this local gradient. The partial derivatives for this local gradient are:

$$\begin{aligned}\frac{\partial CLL}{\partial \theta_j} = \ & \mathrm{E}_{P(T|L(S_i),S_i,\vec{\theta},G)}\, f_j(L(S_i), T, S_i) - \\ & \mathrm{E}_{P(T|S_i,\vec{\theta},G)}\, f_j(L(S_i), T, S_i)\end{aligned}$$

4.3 Learning Lexical Entries with Inverse Function Composition

We adopt a greedy approach to learning new lexical entries. We first identify in our current parse any high-scoring lexical entries that cover multiple words, and then look for new lexical rules for the sub-phrases covered by these lexical entries that could combine to create the current parse chart entry using the existing grammar rules. This requires searching through the grammar rules to find children nodes that the nonterminal could be the parent of. In general, this produces an intractably large set, because it requires taking the inverse of function application and function composition for forming the semantics of the nonterminal, and those inverses are intractably large.

Figure 2 shows our algorithm for learning lexical entries, and Figure 3 shows the details of the critical component that generates the semantics of new potential lexical entries. For brevity, we omit the details of how we learn the syntax and mappings from semantics to words or phrases for new lexical entries, but these are borrowed from the existing techniques in UBL. The crucial difference from existing techniques is that the SPLITLEARN algorithm focuses on inverse function composition, while existing techniques focus on inverse

```
Input: training sentence $Sent$, its logical form $L$,
current grammar $G$

Initialize:
  $PC \leftarrow$ parse chart from parsing $Sent$ with $G$
  $splits \leftarrow \emptyset$

For $len := length(Sent)$ to 1:
  For $pos := 0$ to $length(Sent) - len$:
    $e = \arg\max_{entry \in PC[len][pos]} entry.score$
    if $e$'s only derivation is a lexical rule in $G$:
      $(score, \Lambda) \leftarrow$ SPLITLEARN$(e, PC)$
      $splits \leftarrow splits \cup \{(score, \Lambda)\}$

$split^* \leftarrow \arg\max_{split \in splits} split.score$
Return $split^*.\Lambda$
```

Figure 2: LEXENTLEARN Algorithm for learning lexical entries

```
Input: lexical entry $le$, parse chart $PC$

Entries $\leftarrow \emptyset$
For $s \in$ GETSUBEXPR$(le.sem)$:
  $t \leftarrow$ copy of $s$
  $sem' \leftarrow$ copy of $le.sem$
  Apply$[t] \leftarrow \emptyset$
  For $v \in$ FREEVARS$(s) \cap$ REPEATVARS$(sem')$:
    Create variable $v'$, $t \leftarrow t_{v' \text{ sub for } v}$
    Concatenate "$\lambda v'$" onto front of $t$
    Apply$[t] \leftarrow$ Apply$[t] \cup \{v\}$
  For $v \in$ FREEVARS$(t)$:
    Remove "$\lambda v$" from front of $sem'$
    Concatenate "$\lambda v$" onto front of $t$
  Create new variable $w$
  $sub \leftarrow$ "$(w$" + each $a \in$ Apply$[t]$ + ")"
  $sem' \leftarrow sem'_{sub \text{ sub for } s}$
  Concatenate "$\lambda w$" onto front of $sem'$
  Entries $\leftarrow$ Entries $\cup \{t, sem'\}$
  delta$[t]$, delta$[sem'] \leftarrow$ PCSCORE$(t)$ +
                PCSCORE$(sem')$ - PCSCORE$(le)$

$max \leftarrow \max_x$ delta$[x]$
Return $max, \{s \in$ Entries $|$ delta$[s] = max\}$
```

Figure 3: SPLITLEARN Algorithm for generating (the semantics of) new lexical entries.

function application. While *a priori* both techniques are reasonable choices (and both work well on GeoQuery), our empirical results show that inverse function composition can learn the same semantic forms as inverse function application, but in addition can handle nested functions (which are function compositions) and coordination — a form of function composition if one views logical connectives like `or` as boolean functions.

The SPLITLEARN algorithm uses a GETSUBEXPR subroutine to heuristically select only certain subexpressions of the input logical form for computing inverse composition. This is to avoid a combinatorial explosion in the number of learned splits of the input semantics. Mostly we consider any subexpression that forms an argument to some function in $le.sem$, but we take care to also include abstracted versions of these subexpressions, in which some of their arguments are in turn replaced by variables. The subroutine FREEVARS identifies all variables in a logical form that have no quantifier; REPEATVARS identifies all variables that appear at least twice. PCSCORE looks for any entry in the parse chart that has a matching semantics and returns the score of that entry, or 0 if no matches are found. We use PCSCORE to measure the improvement (delta) in the score of the parse if it uses the two new lexical entries, rather than the previous single lexical entry. SPLITLEARN returns the set of lexical entries that tie for the largest improvement in the score of the parse.

Figures 4 and 5 illustrate the difference between SPLITLEARN and lexical entry learning for UBL and FUBL. For both example sentences, there is a point in the learning process where a logical form must be split using inverse function composition in order for useful lexical entries to be learned. At those points, UBL and FUBL split the logical forms using inverse function application, resulting in splits where the semantics of different lexemes are mixed together in the two resulting subexpressions. In Figure 4, all three systems take the logical form $\lambda u.\lambda p. > (\texttt{result}(p, \texttt{bilirubin}), \texttt{unit}(1.5, u))$ and split it by removing some aspect of the final argument, $\texttt{unit}(1.5, u)$, from the full expression. In UBL and FUBL, the term that is left behind in the full expression is something that unifies with $\lambda u.\texttt{unit}(1.5, u)$. In GLL, however, only a variable is left behind, since that variable can be replaced by $\lambda u.\ \texttt{unit}(1.5, u)$ through function composition to obtain the original expression. Thus GLL's split yields one significantly simpler subexpression, which in the end yields simpler lexical entries. In both figures, and in general for most parses we have observed, inverse function composition yields simpler and cleaner subexpressions.

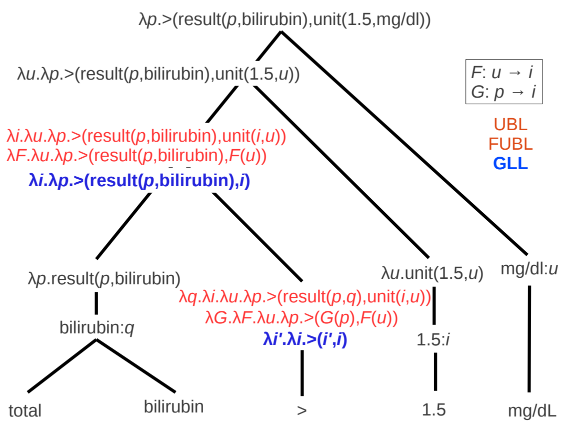
λp.>(result(p,bilirubin),unit(1.5,mg/dl))
λu.λp.>(result(p,bilirubin),unit(1.5,u))
F: u → i
G: p → i
UBL
FUBL
GLL
λi.λu.λp.>(result(p,bilirubin),unit(i,u))
λF.λu.λp.>(result(p,bilirubin),F(u))
λi.λp.>(result(p,bilirubin),i)
λp.result(p,bilirubin)
λq.λi.λu.λp.>(result(p,q),unit(i,u))
λG.λF.λu.λp.>(G(p),F(u))
λi'.λi.>(i',i)
λu.unit(1.5,u)
mg/dl:u
bilirubin:q
1.5:i
total
bilirubin
>
1.5
mg/dL

Figure 4: An example of a sentence with nested-function semantics. GLL's lexical entry learning procedure correctly identifies the most general semantics for the lexeme $>$, while UBL and FUBL learn more specific and complex semantics.

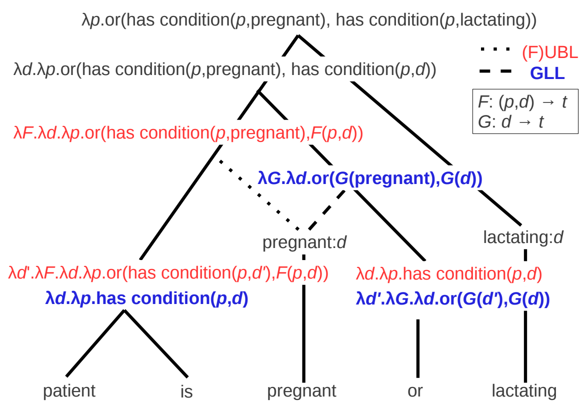
λp.or(has condition(p,pregnant), has condition(p,lactating))
(F)UBL
GLL
λd.λp.or(has condition(p,pregnant), has condition(p,d))
F: (p,d) → t
G: d → t
λF.λd.λp.or(has condition(p,pregnant),F(p,d))
λG.λd.or(G(pregnant),G(d))
pregnant:d
lactating:d
λd'.λF.λd.λp.or(has condition(p,d'),F(p,d))
λd.λp.has condition(p,d)
λd.λp.has condition(p,d)
λd'.λG.λd.or(G(d'),G(d))
patient
is
pregnant
or
lactating

Figure 5: An example of a sentence with coordination semantics. GLL's lexical entry learning procedure correctly identifies the semantics for the lexeme or, while UBL and FUBL learn incorrect semantics.

4.4 Learning T-rules

We use T-rules to handle elliptical constructions. They are essentially a simplification of the factored lexicon used in FUBL that yields very similar results. Each T-rule $\tau \in \mathbf{T}$ is a function of the form λe . **if** $type(e)$ **then return** $Syn : f(e) \rightarrow Syn' : e$, where $type$ is a type from our ontology, Syn and Syn' are two syntactic CCG categories or variables, and f is an arbitrary lambda calculus expression. For example, consider the T-rule $\tau = (\lambda e$. **if** $\texttt{disease}(e)$ **then return** $S\backslash N : \lambda p\ .\ \texttt{has condition}(p, e) \rightarrow N : e)$. When applied to the entity `diabetes`, this T-rule results in an ordinary CCG rule: $S\backslash N : \lambda p\ .\ \texttt{has condition}(p, \texttt{diabetes}) \rightarrow N : \texttt{diabetes}$. Thus each T-rule is a template for constructing unary (type-raising) CCG grammar rules from an entity of the appropriate type.

TRLEARN works by first identifying a set of entity symbols E that appear in multiple lexical entries in the input grammar G. Let the lexical entries for entity $e \in E$ be denoted by $\Lambda(e)$; thus, E consists of all entities where $|\Lambda(e)| \geq 2$. TRLEARN then looks for patterns in each of these sets of lexical entries. If one of the lexical entries in $\Lambda(e)$ has a semantics that consists of just e (for example, the lexical entry $N : \texttt{diabetes} \rightarrow$ diabetes), we create candidate T-rules from every other lexical entry $l' \in \Lambda(e)$ that has the same child, such as $S\backslash N : \lambda p\ .\ \texttt{has condition}(p, \texttt{diabetes}) \rightarrow$ diabetes. From this lexical entry, we create the candidate T-rule $\tau = (\lambda x$. **if** $\texttt{disease}(x)$ **then return** $S\backslash N : \lambda p\ .\ \texttt{has condition}(p, x) \rightarrow N : x)$. In general, the test in the if statement in the T-rule contains a check for the type of entity e. The right-hand side of the implication contains a unary grammar rule whose parent matches the parent of the rule in l', except that entity e has been replaced by a variable x. The child of the grammar rule matches the parent of the basic lexical entry $N : e$, except again that the entity e has been replaced by the variable x.

Having constructed a set of candidate T-rules from this process, TRLEARN must select the ones that will actually be added to the grammar. We use a test of selecting T-rules that cover at least MIN existing grammar rules in the input grammar G. In our implementation, we set $MIN = 2$. When parsing a sentence, the parser checks any parse chart entry for semantics that consist solely of an entity; for any such entry, it looks in a hash-based index for applicable T-rules, applies them to the entity to construct new unary grammar rules, and then applies the unary grammar rules to the parse chart entry to create new nonterminal nodes.

5 Experiments

In our experiments, we test the generality of our learning algorithm by testing its ability to handle both GeoQuery and the Clinical Trials datasets.

5.1 Experimental setup

The clinical trials dataset is described above in Section 3. GeoQuery consists of a database of

System	Precision	Recall	F1
UBL	87.9	88.5	88.2
FUBL	**88.6**	**88.6**	**88.6**
GLL	84.6	86.1	85.5

Table 4: GLL performs comparably to two state-of-the-art learning algorithms for PCCG semantic parsing on the benchmark GeoQuery dataset.

System	Precision	Recall	F1
UBL	20.3	19.9	20.1
FUBL	42.3	39.7	40.8
GLL	**65.3**	**63.2**	**64.1**

Table 5: On the clinical trials dataset, GLL outperforms UBL and FUBL by more than 23 points in F1, for a reduction in error (*i.e.*, 1-F1) of nearly 40% over FUBL.

	Lexicon Size	
System	GeoQuery	Clinical Trials
UBL	5,149	49,635
GLL	4,528	36,112

Table 6: GLL learns a lexicon that is 27% smaller than UBL's lexicon on clinical trials data.

Lexeme	UBL meanings	GLL meanings
$>$	36	2
$<$	28	2
$=$	35	2
and	6	4
or	254	9

Table 7: For certain common and critical lexical items in the clinical trials dataset, GLL learns far fewer (but more general) lexical entries; for the word "or", GLL learns only 3.5% of the entries that UBL learns.

2400 geographical entities, such as nations, rivers, and mountains, as well as 8 geography relations, such as the location of a mountain, and whether one state borders another. The text for semantic parsing consists of a set of 880 geography questions, labeled with a lambda-calculus representation of the sentence's meaning. We follow the procedure described by Kwiatkowski *et al.*. (2010) in splitting these sentences into training, development, and test sentences. This dataset allows us to provide a comparison with other semantic parsers on a well-known dataset. We measured performance based on exact-match of the full logical form, modulo re-ordering of arguments to symmetric relations (like conjunction and disjunction).

5.2 Results and Discussion

Tables 4 and 5 show the results of semantic parsers learned by the UBL, FUBL, and GLL learning algorithms on the GeoQuery and clinical trials datasets, respectively. On the GeoQuery dataset, all three parsers perform very similarly, although GLL's performance is slightly worse. However, on the clinical trials dataset, GLL significantly outperforms both UBL and FUBL in terms of precision, recall, and F1. Of course, there clearly remain many syntactic and semantic constructions that none of these algorithms can currently handle, as all systems perform significantly worse on clinical trials than on GeoQuery.

Tables 6 shows the overall size of UBL's and GLL's learned lexicons, and Table 7 shows the number of learned entries for selected lexical items that appear frequently in the clinical trials corpus. FUBL uses a factored lexicon in which the semantics of a logical form is split across two data structures. As a result, FUBL's lexicon is not directly comparable to the other systems, so for these comparisons we restrict our attention to UBL and GLL. UBL tends to learn far more lexical entries than GLL, particularly for words that appear in multiple sentences. Yet the poorer performance of UBL on clinical trials is an indication that these lexical entries are overly specific.

6 Conclusion

We have introduced the clinical trials dataset, a naturally-occurring set of text where existing learning algorithms for semantic parsing struggle. Our new GLL algorithm uses a novel inverse function composition algorithm to handle coordination and nested function constructions, and pattern learning to handle elliptical constructions. These innovations allow GLL to handle GeoQuery and improve on clinical trials. Many sources of error on clinical trials remain for future research, including long-distance dependencies, attachment ambiguities, and coreference. In addition, further investigation is necessary to test how these algorithms handle additional domains and other types of natural linguistic constructions.

Acknowledgments

This work was supported by National Science Foundation grant 1218692. The authors appreciate the help that Anjan Nepal, Qingqing Cai, Avirup Sil, and Fei Huang provided.

References

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping Semantic Parsers from Conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Research*, 32:D267–D270.

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

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a Multilingual Sportscaster: Using Perceptual Context to Learn Language. *Journal of Artificial Intelligence Research*, 37:397–435.

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

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

Ruifang Ge and Raymond J. Mooney. 2006. Discriminative Reranking for Semantic Parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*.

Ruifang Ge and Raymond J. Mooney. 2009. Learning a Compositional Semantic Parser using an Existing Syntactic Parser. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009)*.

D. Goldwasser, R. Reichart, J. Clarke, and D. Roth. 2011. Confidence driven unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

Rohit J. Kate and Raymond J. Mooney. 2006. Using String-Kernels for Learning Semantic Parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*.

Rohit J. Kate and Raymond J. Mooney. 2007. Semi-Supervised Learning for Semantic Parsing using Support Vector Machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Short Papers (NAACL/HLT-2007)*.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to Transform Natural to Formal Languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly Supervised Training of Semantic Parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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 (EMNLP)*.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

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

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 (EMNLP)*.

Hoifung Poon. 2013. Grounded Unsupervised Semantic Parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

C.A. Thompson and R.J. Mooney. 1999. Automatic construction of semantic lexicons for learning natural language interfaces. In *Proc. 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 487–493.

Cynthia A. Thompson and Raymond J. Mooney. 2003. Acquiring Word-Meaning Mappings for Natural Language Interfaces. *Journal of Artificial Intelligence Research (JAIR)*, 18:1–44.

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 (ACL-2007).*

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 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries using Inductive Logic Programming. In *AAAI/IAAI*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the Twenty First Conference on Uncertainty in Artificial Intelligence (UAI).*

Luke S. Zettlemoyer and Michael Collins. 2007. Online Learning of Relaxed CCG Grammars for Parsing to Logical Form. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL).*

Luke S. Zettlemoyer and Michael Collins. 2009. Learning Context-dependent Mappings from Sentences to Logical Form. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP).*