Evaluating Induced CCG Parsers on Grounded Semantic Parsing

Yonatan Bisk\textsuperscript{1}\textsuperscript{*}, Siva Reddy\textsuperscript{2}\textsuperscript{*}, John Blitzer\textsuperscript{3}, Julia Hockenmaier\textsuperscript{4}, Mark Steedman\textsuperscript{2}

\textsuperscript{1}ISI, University of Southern California
\textsuperscript{2}ILCC, School of Informatics, University of Edinburgh
\textsuperscript{3}Google, Mountain View
\textsuperscript{4}Department of Computer Science, University of Illinois at Urbana-Champaign

ybisk@isi.edu, siva.reddy@ed.ac.uk, blitzer@google.com,
jluijahmr@illinois.edu, steedman@inf.ed.ac.uk

Abstract

We compare the effectiveness of four different syntactic CCG parsers for a semantic slot-filling task to explore how much syntactic supervision is required for downstream semantic analysis. This extrinsic, task-based evaluation provides a unique window to explore the strengths and weaknesses of semantics captured by unsupervised grammar induction systems. We release a new Freebase semantic parsing dataset called SPADES (Semantic PArsing of DEclarative Sentences) containing 93K cloze-style questions paired with answers. We evaluate all our models on this dataset. Our code and data are available at \url{https://github.com/sivareddyg/graph-parser}.

1 Introduction

The past several years have seen significant progress in unsupervised grammar induction (Carroll and Charniak, 1992; Yuret, 1998; Klein and Manning, 2004; Spitkovsky et al., 2010; Garrette et al., 2015; Bisk and Hockenmaier, 2015). But how useful are unsupervised syntactic parsers for downstream NLP tasks? What phenomena are they able to capture, and where would additional annotation be required? Instead of standard intrinsic evaluations – attachment scores that depend strongly on the particular annotation styles of the gold treebank – we examine the utility of unsupervised and weakly supervised parsers for semantics. We perform an extrinsic evaluation of unsupervised and weakly supervised CCG parsers on a grounded semantic parsing task that will shed light on the extent to which these systems recover semantic information. We focus on English to perform a direct comparison with supervised parsers (although unsupervised or weakly supervised approaches are likely to be most beneficial for domains or languages where supervised parsers are not available).

Specifically, we evaluate different parsing scenarios with varying amounts of supervision. These are designed to shed light on the question of how well syntactic knowledge correlates with performance on a semantic evaluation. We evaluate the following scenarios (all of which assume POS-tagged input): 1) no supervision; 2) a lexicon containing words mapped to CCG categories; 3) a lexicon containing POS tags mapped to CCG categories; 4) sentences annotated with CCG derivations (i.e., fully supervised). Our evaluation reveals which constructions are problematic for unsupervised parsers (and annotation efforts should focus on). Our results indicate that unsupervised syntax is useful for semantics, while a simple semi-supervised parser outperforms a fully unsupervised approach, and could hence be a viable option for low resource languages.

2 CCG Intrinsic Evaluations

CCG (Steedman, 2000) is a lexicalized formalism in which words are assigned syntactic types, also known as supertags, encoding subcategorization information. Consider the sentence Google acquired Nest in 2014, and its CCG derivations shown in Figure 1.

In (a) and (b), the supertag of acquired, $(S\backslash NP)/NP$, indicates that it has two arguments, and the prepositional phrase in 2014 is an adjunct, whereas in (c) the
Google acquired Nest in 2014

(a) in 2014 modifies acquired Nest
(b) in 2014 modifies Google acquired Nest
c) acquired Google takes the argument in 2014

Figure 1: Example of multiple valid derivations that can be grounded to the same Freebase logical form (Eq. 1) even though they differ dramatically in performance under parsing metrics (5, 4, or 3 “correct” supertags).

Since grammar induction systems are traditionally trained on declarative sentences, we would ideally require declarative sentences paired with Freebase logical forms. But such datasets do not exist in the Freebase semantic parsing literature (Cai and Yates, 2013; Berant et al., 2013). To alleviate this problem, and yet perform Freebase semantic parsing, we propose an entity slot-filling task.

**Entity Slot-Filling Task.** Given a declarative sentence containing mentions of Freebase entities, we randomly remove one of the mentions to create a blank slot. The task is to fill this slot by translating the declarative sentence into a Freebase query. Consider the following sentence where the entity Nest has been removed:

*Google acquired _____ which was founded in Palo Alto*

To correctly fill in the blank, one has to query Freebase for the entities acquired by Google (constraint 1) and founded in Palo Alto (constraint 2). If either of those constraints are not applied, there will be many entities as answers. For each question, we execute a single Freebase query containing all the constraints and retrieve a list of answer entities. From this list, we pick the first entity as our predicted answer, and consider the prediction as correct if the gold answer is the same as the predicted answer.

**4 Sentences to Freebase Logical Forms**

CCG provides a clean interface between syntax and semantics, i.e. each argument of a words syntactic category corresponds to an argument of the lambda expression that defines its semantic interpretation (e.g., the lambda expression corresponding to the category ($S\backslash NP$)/NP of the verb *acquired* is $\lambda f.\lambda g.\lambda e.\exists x.\exists y. acquired(e) \land f(x) \land g(y) \land arg_1(e, y) \land arg_2(e, x))$, and the logical form for the complete sentence can be constructed by composing word level lambda expressions following the syntactic derivation (Bos et al., 2004). In Figure 2 we show
Figure 2: The lexical categories for *which* determine the relative clause attachment and therefore the resulting ungrounded logical form. The top derivation correctly executes a query to retrieve companies founded in Palo Alto and acquired by Google. The bottom incorrectly asserts that Google was founded in Palo Alto.

Our next step is to convert these ungrounded graphs to Freebase graphs.² Like Reddy et al. (2014), we treat this problem as a graph matching problem. Using GRAPHPARSER we retrieve all the Freebase graphs that are isomorphic to the ungrounded graph, and select only the graphs that could correctly predict the blank slot, as candidate graphs. Using these candidate graphs, we train a structured perceptron that learns to rank grounded graphs for a given ungrounded graph.³ We use ungrounded predicate and Freebase predicate alignments as our features.

² Note that there is one-to-one correspondence between Freebase graphs and Freebase logical forms.
³ Please see Section 4.3 of Reddy et al. (2016) for details.

5 Experiments

5.1 Training and Evaluation Datasets

Our dataset SPADES (Semantic PArsing of DEclarative Sentences) is constructed from the declarative sentences collected by Reddy et al. (2014) from CLUEWEB09 (Gabrilovich et al., 2013) based on the following constraints: 1) There exists at least one isomorphic Freebase graph to the ungrounded representation of the input sentence; 2) There are no variable nodes in the ungrounded graph (e.g., Google *acquired a company*) is discarded whereas Google *acquired the company Nest is selected*). We split this data into training (85%), development (5%) and testing (10%) sentences (Table 1). We introduce empty slots into these sentences by randomly removing an entity. SPADES can be downloaded at http://github.com/sivareddyg/graph-parser.

There has been other recent interest in similar datasets for sentence completion (Zweig et al., 2012) and machine reading (Hermann et al., 2015), but unlike other corpora our data is tied directly to Freebase and requires the execution of a semantic parse to correctly predict the missing entity. This is made more

<table>
<thead>
<tr>
<th></th>
<th>Sentences</th>
<th>Tokens</th>
<th>Types</th>
<th>Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>79,247</td>
<td>685,922</td>
<td>69,095</td>
<td>37,606</td>
</tr>
<tr>
<td>Dev</td>
<td>4,763</td>
<td>41,102</td>
<td>9,306</td>
<td>4,358</td>
</tr>
<tr>
<td>Test</td>
<td>9,309</td>
<td>80,437</td>
<td>15,180</td>
<td>7,431</td>
</tr>
</tbody>
</table>

Table 1: SPADES Corpus Statistics
Table 2: Syntactic and semantic evaluation of the parsing models. Left: Simplified labeled F1 and undirected unlabeled F1 on CCGbank, Section 23. Right: Slot filling performance (by number of entities per sentence). Slot-filling results are updated after the camera-ready submission. In the previous version, instead of evaluating if the gold entity is same as the first predicted entity, we mistakenly evaluated if the gold entity is present in the list of predicted answer entities. However, the initial claims are still valid. All other results and discussion are revised.

Our primary focus is a comparison of intrinsic syntactic evaluation with our extrinsic semantic evaluation. To highlight the differences we present Section 23 parsing performance for our four models in Table 2. Dependency performance is evaluated on both the simplified labeled F1 of Bisk and Hockenmaier (2015) and Undirected Unlabeled F1. Despite the supervised parser performing almost twice as well as the semi-supervised parsers on CCGbank LF1 (53.5 vs 84.2), in our semantic evaluation we see a comparatively small gain in performance (28.4 vs 30.9). It is interesting that such weakly supervised models are able to achieve over 90% of the performance of a fully supervised parser. To explore this further, we break down the semantics performance of all our models by the number of entities in a sentence. Each sentence has two, three, or four entities, one of which will be dropped for prediction. The more entities there are in a sentence, the more likely the models are to misanalyze a relation leading to their making the wrong prediction. These results are presented on the right side of Table 2. There are still notable discrepancies in performance, which we analyze more closely in the next section.

Another interesting result is the drop in performance by the Bag-of-Words Model. As the number of entities in the sentence increase, the model weak-
Table 1

<table>
<thead>
<tr>
<th>Annotated Words</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>37.1</td>
<td>25.72</td>
</tr>
<tr>
<td>100</td>
<td>48.49</td>
<td>27.56</td>
</tr>
<tr>
<td>200</td>
<td>53.5</td>
<td>30.05</td>
</tr>
<tr>
<td>500</td>
<td>53.36</td>
<td>29.18</td>
</tr>
<tr>
<td>1000</td>
<td>49.87</td>
<td>27.92</td>
</tr>
</tbody>
</table>

Figure 3: The effect of increasing number of lexical types in **SEMI-SUPERVISED-WORD** on syntax and semantics. When the lexicon grows past 200 lexical types both syntax and semantics begins to degrade. We also observe there is a correlation between syntactic parsing and semantic parsing performance.

ens, performing worse than the unsupervised parser on sentences with four entities. It becomes non-trivial for it to isolate which entities and relations should be used for prediction. This seems to indicate that the unsupervised grammar is capturing more useful syntactic/semantic information than what is available from the words alone. Ensemble systems that incorporate syntax and a Bag-of-Words baseline may yield even better performance.

### 5.4 The Benefits of Annotation

The performance of **SEMI-SUPERVISED-POS** and **SEMI-SUPERVISED-WORD** suggests that when resources are scarce, it is beneficial to create a even a small lexicon of CCG categories. We analyze this further in Figure 3. Here we show how performance changes as a function of the number of labeled lexical types. Our values range from 0 to 1000 lexical types. We see syntactic improvements of 16pts and semantic gains of 4.33pts (16.8%) with 200 words, before performance degrades. It is possible that increasing annotation may only benefit fully supervised models. Finally, when computing the most frequent lexical types we excluded commas. We found a drop in performance when restricting commas to the category , (they are commonly conj in our data). Additional in-domain knowledge might further improve performance.

Table 3: Causes of semantic grounding errors with examples not previously isolated via intrinsic evaluation.

<table>
<thead>
<tr>
<th>Error</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect conjunction</td>
<td><em>Stockholm, Sweden</em></td>
</tr>
<tr>
<td>Appositive</td>
<td><em>a chemist</em></td>
</tr>
<tr>
<td>Introductory clauses</td>
<td><em>In Frankfurt,</em> ...</td>
</tr>
<tr>
<td>Reduced relatives</td>
<td><em>... established in 1909,</em> ...</td>
</tr>
<tr>
<td>Verb chains</td>
<td><em>is also headquartered</em></td>
</tr>
<tr>
<td>Possessive</td>
<td><em>Anderson’s Foundation</em></td>
</tr>
<tr>
<td>PP Attachment</td>
<td><em>of the foundation in Vancouver</em></td>
</tr>
</tbody>
</table>

### 5.5 Common Errors

Bisk and Hockenmaier (2015) performed an in-depth analysis of the types of categories learned and correctly used by their models (the same models as this paper). Their analysis was based on syntactic evaluation against CCGbank. In particular, they found the most egregious “semantic” errors to be the misuse of verb chains, possessives and PP attachment (bottom of Table 3). Since we now have access to a purely semantic evaluation, we can therefore ask whether these errors exist here, and how common they are. We do this analysis in two steps. First, we manually analyzed parses for which the unsupervised model failed to predict the correct semantics, but where the supervised parser succeeded. The top of Table 3 presents several of the most common reasons for failure. These mistakes were more mundane (e.g. incorrect use of a conjunction) than failures to use complex CCG categories or analyze attachments.

Second, we can compare grammatical decisions made by the semi-supervised and unsupervised parsers against EasyCCG on sentences they successfully grounded. Bisk and Hockenmaier (2015) found that their unsupervised parser made mistakes on many very simple categories. We found the same result. When evaluating our parsers against the treebank we found the unsupervised model only correctly predicted transitive verbs 20% of the time and adverbs 39% of the time. In contrast, on our data, we produced the correct transitive category (according to EasyCCG) 65% of the time, and the correct adverb 68% of the time. These correct parsing decisions also lead to improved performance across many other categories (e.g. prepositions). This is likely due to our
corpus containing simpler constructions. In contrast, auxiliary verbs, relative clauses, and commas still proved difficult or harder than in the treebank. This implies that future work should tailor the annotation effort to their specific domain rather than relying on guidance solely from the treebank.

6 Conclusion

Our goal in this paper was to present the first semantic evaluation of induced grammars in order to better understand their utility and strengths. We showed that induced grammars are learning more semantically useful structure than a Bag-of-Words model. Furthermore, we showed how minimal syntactic supervision can provide substantial gains in semantic evaluation. Our ongoing work explores creating a syntax-semantics loop where each benefits the other with no human (annotation) in the loop.

Acknowledgments

This paper is partly based on work that was done when the first and second authors were interns at Google, and on work that that was supported by NSF grant 1053856 to JH, and a Google PhD Fellowship to SR.

References


