Evaluating Induced CCG Parsers on Grounded Semantic Parsing

Yonatan Bisk[1*] **Siva Reddy**[2*] **John Blitzer**[3] **Julia Hockenmaier**[4] **Mark Steedman**[2]
[1]ISI, University of Southern California
[2]ILCC, School of Informatics, University of Edinburgh
[3]Google, Mountain View
[4]Department of Computer Science, University of Illinois at Urbana-Champaign
ybisk@isi.edu, siva.reddy@ed.ac.uk, blitzer@google.com,
juliahmr@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 (**S**emantic **PA**rsing of **DE**clarative **S**entences) containing 93K cloze-style questions paired with answers. We evaluate all our models on this dataset. Our code and data are available at https://github.com/sivareddyg/graph-parser.

1 Introduction

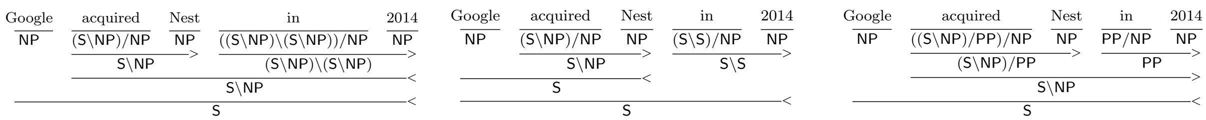
The past several years have seen significant progress in unsupervised grammar induction (Carroll and Charniak, 1992; Yuret, 1998; Klein and Manning, 2004; Spitkovsky et al., 2010; Garrette et al., 2015; Bisk and Hockenmaier, 2015). But how useful are unsupervised syntactic parsers for downstream NLP 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*, $(\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP}$, indicates that it has two arguments, and the prepositional phrase *in 2014* is an adjunct, whereas in (c) the

[*]Equal contribution



(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).

supertag ((S\NP)/PP)/NP indicates *acquired* has three arguments including the prepositional phrase. In (a) and (b), depending on the supertag of *in*, the derivation differs. When trained on labeled treebanks, (a) is preferred. However note that all these derivations could lead to the same semantics (e.g., to the logical form in Equation 1). Without syntactic supervision, there may not be any reason for the parser to prefer one analysis over the other. One procedure to evaluate unsupervised induction methods has been to compare the assigned supertags to treebanked supertags, but this evaluation does not consider that multiple derivations could lead to the same semantics. This problem is also not solved by evaluating syntactic dependencies. Moreover, while many dependency standards agree on the head direction of simple constituents (e.g., noun phrases) they disagree on the most semantically useful ones (e.g., coordination and relative clauses).[1]

3 Our Proposed Evaluation

The above syntax-based evaluation metrics conceal the real performance differences and their effect on downstream tasks. Here we propose an extrinsic evaluation where we evaluate our ability to convert sentences to Freebase logical forms starting via CCG derivations. Our motivation is that most sentences can only have a single realization in Freebase, and any derivation that could lead to this realization is potentially a correct derivation. For example, the Freebase logical form for the example sentence in Figure 1 is shown below, and none of its derivations are penalized if they could result in this logical form.

$$\begin{aligned}&\lambda e.\, \text{business.acquisition}(e)\\ &\wedge \text{acquiring\_company}(e, \textsc{Google})\\ &\wedge \text{company\_acquired}(e, \textsc{Nest})\\ &\wedge \text{date}(e, 2014)\end{aligned} \tag{1}$$

[1]Please see Bisk and Hockenmaier (2013) for more details.

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.

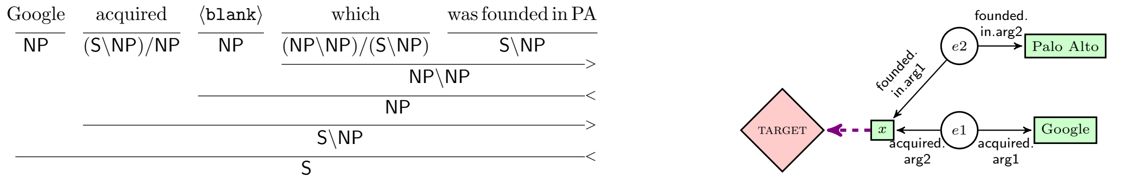
Entity Slot-Filling Task. Given a declarative sentence containing mentions of Freebase entities, we randomly remove one of the mentions to create a blank slot. The task is to fill this slot by translating the declarative sentence into a Freebase query. Consider the following sentence where the entity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

Google acquired ______ which was founded in Palo Alto

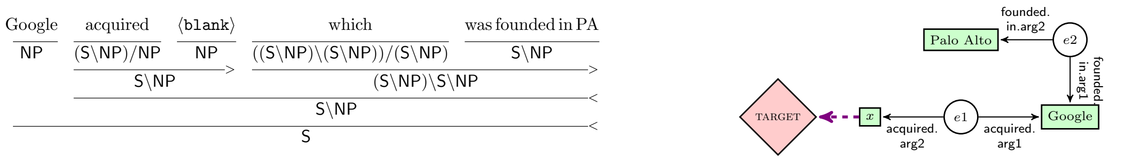
To correctly fill in the blank, one has to query Freebase for the entities acquired by *Google* (constraint 1) and founded in *Palo Alto* (constraint 2). If either of those constraints are not applied, there will be many entities as answers. For each question, we execute a single Freebase query containing 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\NP)/NP of the verb *acquired* is $\lambda f.\lambda g.\lambda e.\exists x.\exists y.\text{acquired}(e) \wedge f(x) \wedge g(y) \wedge arg_1(e,y) \wedge arg_2(e,x)$), 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



$$\lambda e_1.\exists x e_2.\,\text{TARGET}(x) \wedge \text{acquired}(e_1) \wedge arg_1(e_1, \text{Google}) \wedge arg_2(e_1, x) \wedge \text{founded}(e_2) \wedge \mathbf{arg_2}(\mathbf{e_2}, x) \wedge \text{in}(e_2, \text{PaloAlto})$$



$$\lambda e_1.\exists x e_2.\,\text{TARGET}(x) \wedge \text{acquired}(e_1) \wedge arg_1(e_1, \text{Google}) \wedge arg_2(e_1, x) \wedge \text{founded}(e_2) \wedge \mathbf{arg_2}(\mathbf{e_2}, \text{Google}) \wedge \text{in}(e_2, \text{PaloAlto})$$

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.

two syntactic derivations for the same sentence, and the corresponding logical forms and equivalent graph representations derived by GRAPHPARSER (Reddy et al., 2014). The graph representations are possible because GRAPHPARSER assumes access to co-indexations of input CCG categories. We provide co-indexation for all induced categories, including multiple co-indexations when an induced category is ambiguous. For example, $(\mathsf{S}\backslash\mathsf{N})/(\mathsf{S}\backslash\mathsf{N})$ refers to either $(\mathsf{S_x}\backslash\mathsf{N_y})/(\mathsf{S_x}\backslash\mathsf{N_y})$ indicating an auxiliary verb or $(\mathsf{S_x}\backslash\mathsf{N_y})/(\mathsf{S_z}\backslash\mathsf{N_y})$ indicating a control verb. Initially, the predicates in the expression/graph will be based entirely on the surface form of the words in the sentence. This is the "*ungrounded*" semantic representation.

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.

	Sentences	Tokens	Types	Entities
Train	79,247	685,922	69,095	37,606
Dev	4,763	41,102	9,306	4,358
Test	9,309	80,437	15,180	7,431

Table 1: SPADES Corpus Statistics

5 Experiments

5.1 Training and Evaluation Datasets

Our dataset SPADES (**S**emantic **PA**rsing of **DE**clarative **S**entences) 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

	CCGbank (Syntax)		Slot Filling (Semantics)			
	LF1	UF1	2	3	4	Overall
Sentences			∼6K	∼3K	∼600	∼10K
Bag-of-Words	–	–	34.1	28.1	**16.5**	31.4
Syntax: UNSUPERVISED	37.1	64.2	27.7	20.0	**16.8**	24.8
Syntax: SEMI-SUPERVISED-POS	53.0	68.5	30.5	21.8	**18.8**	27.3
Syntax: SEMI-SUPERVISED-WORD	53.5	68.9	30.7	25.5	**18.7**	28.4
Syntax: SUPERVISED	84.2	91.0	32.7	29.0	**20.2**	30.9

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.

explicit by the fact that one third of the entities in our test set are never seen during training, so without a general approach to query creation and execution there is a limit on a system's performance.

5.2 Our Models

We use different CCG parsers varying in the amounts of supervision. For the UNSUPERVISED scenario, we use Bisk and Hockenmaier (2015)'s parser which exploits a small set of universal rules to automatically induce and weight a large set of lexical categories. For the semi-supervised, we explore two options – SEMI-SUPERVISED-WORD and SEMI-SUPERVISED-POS. We use Bisk et al. in both settings but we constrain its lexicon manually rather than inducing it from scratch. In the former, we restrict the top 200 words in English to occur only with the CCG categories that comprise 95% of the occurrences of a word's use in Section 22 of WSJ/CCGbank. In the latter, we restrict the POS tags instead of words. For the SUPERVISED scenario, we use EasyCCG (Lewis and Steedman, 2014) trained on CCGbank.

Finally, in order to further demonstrate the amount of useful information being learned by our parsers, we present a competitive Bag-of-Words baseline, which is a perceptron classifier that performs "semantic parsing" by predicting either a Freebase or a null relation between the empty slot and every other entity in the sentence, using the words in the sentence as features. This naive approach is competitive on simple sentences with only two entities, rivaling even the fully supervised parser, but falters as complexity increases.

5.3 Results and Discussion

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-

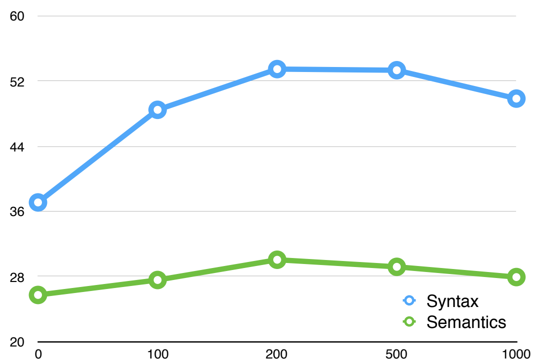


Figure 3: The effect of increasing number of lexical types in SEMI-SUPERVISED-WORD on syntax and semantics. When the lexicon grows past 200 lexical types both syntax and semantics begins to degrade. We also observe there is a correlation between syntactic parsing and semantic parsing performance.

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.

	Error	Example
Prevalent	Incorrect conjunction	*Stockholm, Sweden*
	Appositive	*, a chemist ,*
	Introductory clauses	*In Frankfurt, ...*
	Reduced relatives	*... , established in 1909, ...*
B&H 15	Verb chains	*is also headquartered*
	Possessive	*Anderson 's Foundation*
	PP Attachment	*of the foundation in Vancouver*

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

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

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*, pages 1533–1544, Seattle, Washington, USA, October.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP Model for Inducing Combinatory Categorial Grammars. *Transactions of the Association for Computational Linguistics*, pages 75–88.

Yonatan Bisk and Julia Hockenmaier. 2015. Probing the linguistic strengths and limitations of unsupervised grammar induction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, Beijing,China, July.

Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1240.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338, Atlanta, Georgia, USA, June.

Glenn Carroll and Eugene Charniak. 1992. Two Experiments on Learning Probabilistic Dependency Grammars from Corpora. *Working Notes of the Workshop Statistically-Based NLP Techniques*, pages 1–15, March.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0), June.

Dan Garrette, Chris Dyer, Jason Baldridge, and Noah A Smith. 2015. Weakly-Supervised Grammar-Informed Bayesian CCG Parser Learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1693–1701.

Dan Klein and Christopher D Manning. 2004. Corpus-Based Induction of Syntactic Structure: Models of Dependency and Constituency. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 478–485, Barcelona, Spain, July.

Mike Lewis and Mark Steedman. 2014. A* CCG Parsing with a Supertag-factored Model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 990–1000, Doha, Qatar, October.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics*, pages 1–16, June.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Valentin I Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. 2010. From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 751–759, Los Angeles, California, June.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, September.

Deniz Yuret. 1998. *Discovery of Linguistic Relations Using Lexical Attraction*. Ph.D. thesis, Massachusetts Institute of Technology.

Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher J.C. Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 601–610, Jeju Island, Korea, July.