A* CCG Parsing with a Supertag-factored Model

Mike Lewis
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
mike.lewis@ed.ac.uk

Mark Steedman
School of Informatics
University of Edinburgh
Edinburgh, EH8 9AB, UK
steedman@inf.ed.ac.uk

Abstract

We introduce a new CCG parsing model which is factored on lexical category assignments. Parsing is then simply a deterministic search for the most probable category sequence that supports a CCG derivation. The parser is extremely simple, with a tiny feature set, no POS tagger, and no statistical model of the derivation or dependencies. Formulating the model in this way allows a highly effective heuristic for A* parsing, which makes parsing extremely fast. Compared to the standard C&C CCG parser, our model is more accurate out-of-domain, is four times faster, has higher coverage, and is greatly simplified. We also show that using our parser improves the performance of a state-of-the-art question answering system.

1 Introduction

CCG is a strongly lexicalized grammatical formalism, in which the vast majority of the decisions made during interpretation involve choosing the correct definitions of words. We explore the effect of modelling this explicitly in a parser, by only using a probabilistic model of lexical categories (based on a local context window), rather than modelling the derivation or dependencies.

Existing state-of-the-art CCG parsers use complex pipelines of POS-tagging, supertagging and parsing—each with its own feature sets and parameters (and sources of error)—together with further parameters governing their integration (Clark and Curran, 2007). We show that much simpler models can achieve high performance. Our model predicts lexical categories based on a tiny feature set of word embeddings, capitalization, and 2-character suffixes—with no parsing model beyond a small set of CCG combinators, and no POS-tagger. Simpler models are easier to implement, replicate and extend.

Another goal of our model is to parse CCG optimally and efficiently, without using excessive pruning. CCG's large set of lexical categories, and generalized notion of constituency, mean that sentences can have a huge number of potential parses. Fast existing CCG parsers rely on aggressive pruning—for example, the C&C parser uses a supertagger to dramatically cut the search space considered by the parser. Even the loosest beam setting for their supertagger discards the correct parse for 20% of sentences. The structure of our model allows us to introduce a simple but powerful heuristic for A* parsing, meaning it can parse almost 50 sentences per second exactly, with no beam-search or pruning. Adding very mild pruning increases the speed to 186 sentences per second with minimal loss of accuracy.

Our approach faces two obvious challenges. Firstly, categories are assigned based on a local window, which may not contain the necessary context for resolving some attachment decisions. For example, in *I saw a squirrel 2 weeks ago with a nut*, the model cannot make an informed decision on whether to assign *with* an adverbial or adnominal preposition category, as the crucial words *saw* and *squirrel* fall outside the local context window. Secondly, even if the supertagger makes all lexical category decisions correctly, then the parser can still make erroneous decisions. One example is in coordination-scope ambiguities, such as *clever boys and girls*, where the two interpretations use the same assignment of categories.

We hypothesise that such decisions are relatively rare, and are challenging for any parsing model, so a weak model is unlikely to result in substantially lower accuracy. Our implementation of this model[1], which we call EASYCCG, has high

[1]Available from https://github.com/mikelewis0/easyccg

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 990–1000,
October 25-29, 2014, Doha, Qatar.

accuracy—suggesting that most parsing decisions can be made accurately based on a local context window.

Of course, there are many parsing decisions that can only be made accurately with more complex models. However, exploring the power and limitations of simpler models may help focus future research on the more challenging cases.

2 Background

2.1 Combinatory Categorial Grammar

CCG (Steedman, 2000) is a strongly lexicalized grammatical formalism. Words have categories representing their syntactic role, which are either atomic, or functions from one category to another.

Phrase-structure grammars have a relatively small number of lexical categories types (e.g. POS-tags), and a large set of rules used to build a syntactic analysis of a complete sentence (e.g. an adjective and noun can combine into a noun). In contrast, CCG parsing has many lexical category types (we use 425), but a small set of combinatory rule types (we use 10 binary and 13 unary rule schemata). This means that, aside from the lexicon, the grammar is small enough to be hand-coded—which allows us, in this paper, to confine the entire statistical model to the lexicon.

CCG's generalized notion of constituency means that many derivations are possible for a given a set of lexical categories. However, most of these derivations will be semantically equivalent—for example, deriving the same dependency structures—in which case the actual choice of derivation is unimportant. Such ambiguity is often called *spurious*.

2.2 Existing CCG Parsing Models

The seminal C&C parser is by far the most popular choice of CCG parser (Clark and Curran, 2007). It showed that it was possible to parse to an expressive linguistic formalism with high speed and accuracy. The performance of the parser has enabled large-scale logic-based distributional research (Harrington, 2010; Lewis and Steedman, 2013a; Lewis and Steedman, 2013b; Reddy et al., 2014), and it is a key component of Boxer (Bos, 2008).

The C&C parser uses CKY chart parsing, with a log-linear model to rank parses. The vast number of possible parses means that computing the complete chart is impractical. To resolve this problem, a *supertagger* is first run over the sentence to prune the set of lexical categories considered by the parser for each word. The initial beam outputs an average of just 1.2 categories per word, rather than the 425 possible categories—making the standard CKY parsing algorithm very efficient. If the parser fails to find any analysis of the complete sentence with this set of supertags, the supertagger re-analyses the sentence with a more relaxed beam (*adaptive supertagging*).

2.3 A* Parsing

Klein and Manning (2003a) introduce A* parsing for PCFGs. The parser maintains a chart and an agenda, which is a priority queue of items to add to the chart. The agenda is sorted based on the items' inside probability, and a heuristic upper-bound on the outside probability—to give an upper bound on the probability of the complete parse. The chart is then expanded in best-first order, until a complete parse for the sentence is found.

Klein and Manning calculate an upper bound on the outside probability of a span based on a summary of the context. For example, the summary for the SX heuristic is the category of the span, and the number of words in the sentence before and after the span. The value of the heuristic is the probability of the best possible sentence meeting these restrictions. These probabilities are pre-computed for every non-terminal symbol and for every possible number of preceding and succeeding words, leading to large look-up tables.

Auli and Lopez (2011b) find that A* CCG parsing with this heuristic is very slow. However, they achieve a modest 15% speed improvement over CKY when A* is combined with adaptive supertagging. One reason is that the heuristic estimate is rather coarse, as it deals with the best possible outside context, rather than the actual sentence. We introduce a new heuristic which gives a tighter upper bound on the outside probability.

3 Model

3.1 Lexical Category Model

As input, our parser takes a distribution over all CCG lexical categories for each word in the sentence. These distributions are assigned using Lewis and Steedman (2014)'s semi-supervised supertagging model. The supertagger is a unigram log-linear classifier that uses features of the ± 3 word context window surrounding a word. The

key feature is word embeddings, initialized with the 50-dimensional embeddings trained in Turian et al. (2010), and fine-tuned during supervised training. The model also uses 2-character suffixes and capitalization features.

The use of word embeddings, which are trained on a large unlabelled corpus, allows the supertagger to generalize well to words not present in the labelled data. It does not use a POS-tagger, which avoids problems caused by POS-tagging errors.

Our methods could be applied to any supertagging model, but we find empirically that this model gives higher performance than the C&C supertagger.

3.2 Parsing Model

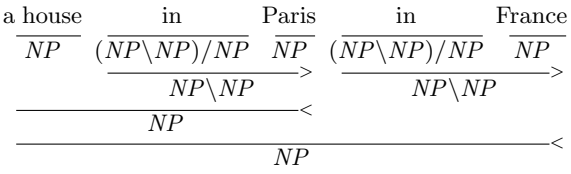
Let a CCG parse y of a sentence S be a list of lexical categories $c_1 \dots c_n$ and a derivation. If we assume all derivations licensed by our grammar are equally likely, and that lexical category assignments are conditionally independent given the sentence, we can compute the optimal parse $\hat{y}$ as:

$\hat{y} = argmax_y \prod_{i=1}^{n} p(c_i|S)$

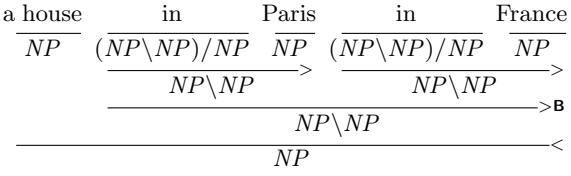
As discussed in Section 2.1, many derivations are possible given a sequence of lexical categories, some of which may be semantically distinct. However, our model will assign all of these an equal score, as they use the same sequence of lexical categories. Therefore we extend our model with a simple deterministic heuristic for ranking parses that use the same lexical categories. Given a set of derivations with equal probability, we output the one maximizing the sums of the length of all arcs in the corresponding dependency tree.

The effect of this heuristic is to prefer non-local attachments in cases of ambiguity, which we found worked better on development data than favouring local attachments. In cases of spurious ambiguity, all parses will have the same value of this heuristic, so one is chosen arbitrarily. For example, one of the parses in Figures 1a and 1b would be selected over the parse in Figure 1c.

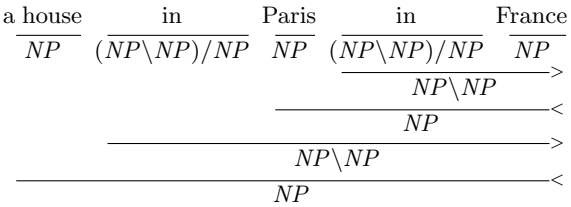
Of course, we could use any function of the parses in place of this heuristic, for example a head-dependency model. However, one aim of this paper is to demonstrate that an extremely simple parsing model can achieve high performance, so we leave more sophisticated alternatives to future work.



(a) A standard derivation of *a house in Paris in France*, with a dependency from *in France* to *house*



(b) A derivation of *a house in Paris in France*, which is spuriously equivalent to Figure 1a. A composition combinator is used to compose the predicates *in Paris* and *in France*, creating a constituent which creates dependencies to its argument from both *in Paris* and *in France*.



(c) A derivation of *a house in Paris in France*, which yields different dependencies to Figures 1a and 1b: here, there is a dependency from *in France* to *Paris*, not *house*.

Figure 1: Three CCG parses of *a house in Paris in France*, given the same set of supertags. The first two are spuriously equivalent, but the third is semantically distinct.

3.3 A* Search

For parsing, we use an A* search for the most-probable complete CCG derivation of a sentence. A key advantage of A* parsing over CKY parsing is that it does not require us to prune the search space first with a supertagger, allowing the parser to consider the complete distribution of 425 categories for each word (in contrast to an average of 3.57 categories per word considered by the C&C parser's most relaxed beam). This is possible because A* only searches for the Viterbi parse of a sentence, rather than building a complete chart with every possible category per word (another alternative, used by Hockenmaier (2003), is to use a highly aggressive beam search in the parser).

In A* parsing, items on the agenda are sorted by their *cost*; the product of their inside probability and an upper bound on their outside probability.

For a span $w_i \dots w_j$ with lexical categories $c_i \dots c_j$ in a sentence $S = w_1 \dots w_n$, the inside probability is simply: $\prod_{k=i}^{j} p(c_k|S)$

The factorization of our model lets us give the following upper-bound on the outside probability:

$$h(w_i \dots w_j) = \prod_{k=1}^{k<i} max_{c_k} p(c_k|S) \times \prod_{k=j+1}^{k \leq n} max_{c_k} p(c_k|S)$$

This heuristic assumes that all words outside the span will take their highest-probability supertag. Because the model is factored on lexical categories, this estimate is clearly an upper bound. As supertagging is over 90% accurate, the upper bound will often be exact, and in Section 4.3 we show empirically that it is extremely efficient. The values of the heuristic can be computed once for each sentence and cached.

To implement the preference for non-local attachment described in Section 3.2, if two agenda items have the same cost, the one with the longer dependencies is preferred.

Intuitively, the parser first attempts to find a parse for the sentence using the 1-best category for each word, by building as complete a chart as possible. If it fails to find a parse for the complete sentence, it adds one more supertag to the chart (choosing the most probable tag not already in the chart), and tries again. This strategy allows the parser to consider an unbounded number of categories for each word, as it does not build a complete chart with all supertags.

3.4 Grammar

Here, we describe the set of combinators and unary rules in the EASYCCG grammar. Because we do not have any probabilistic model of the derivation, all rules can apply with equal probability. This means that some care needs to be taken in designing the grammar to ensure that all the rules are generally applicable. We also try to limit spurious ambiguity, and build derivations which are compatible with the C&C parser's scripts for extracting dependencies (for evaluation). We describe the grammar in detail, to ensure replicability.

Our parser uses the following binary combinators from Steedman (2012): *forward application*, *backward application*, *forward composition*, *backward crossed composition*, *generalized forward composition*, *generalized backward crossed composition*. These combinators are posited to be linguistically universal. The generalized rules are generalized to degree 2. Following Steedman (2000) and Clark and Curran (2007), backward composition is blocked where the argument of the right-hand category is an N or NP. The unhelpful *[nb]* feature is ignored.

Initial	Result	Usage
N	NP	Bare noun phrases
NP	$S/(S\backslash NP)$	Type raising
NP	$(S\backslash NP)/((S\backslash NP)/NP)$	
PP	$(S\backslash NP)/((S\backslash NP)/PP)$	
$S_{pss}\backslash NP$	$NP\backslash NP$	Reduced relative clauses
$S_{ng}\backslash NP$	$NP\backslash NP$	
$S_{adj}\backslash NP$	$NP\backslash NP$	
$S_{to}\backslash NP$	$NP\backslash NP$	
$S_{to}\backslash NP$	$N\backslash N$	
S_{dcl}/NP	$NP\backslash NP$	
$S_{pss}\backslash NP$	S/S	VP Sentence Modifiers
$S_{ng}\backslash NP$	S/S	
$S_{to}\backslash NP$	S/S	

Table 1: Set of unary rules used by the parser.

As in the C&C parser, we add a special *Conjunction* rule:

$$\frac{Y \qquad X}{X\backslash X}>$$

Where $Y \in \{conj, comma, semicolon\}$. We block conjunctions where the right-hand category is type-raised, punctuation, N, or $NP\backslash NP$. This rule (and the restrictions) could be removed by changing CCGBank to analyse conjunctions with $(X\backslash X)/X$ categories.

We also add syntagmatic rules for removing any punctuation to the right, and for removing open-brackets and open-quotes to the left

The grammar also contains 13 unary rules, listed in Table 1. These rules were chosen based on their frequency in the training data, and their clear semantic interpretations.

Following Clark and Curran (2007), we also add a (switchable) constraint that only category combinations that have combined in the training data may combine in the test data. We found that this was necessary for evaluation, as the C&C conversion tool for extracting predicate-argument dependencies had relatively low coverage on the CCG derivations produced by our parser. While this restriction is theoretically inelegant, we found it did increase parsing speed without lowering lexi-

cal category accuracy.

We also use Eisner Normal Form Constraints (Eisner, 1996), and Hockenmaier and Bisk's (2010) Constraint 5, which automatically rule out certain spuriously equivalent derivations, improving parsing speed.

We add a hard constraint that the root category of the sentence must be a declarative sentence, a question, or a noun-phrase.

This grammar is smaller and cleaner than that used by the C&C parser, which uses 32 unary rules (some of which are semantically dubious, such as $S[dcl] \rightarrow NP\backslash NP$), and non-standard binary combinators such as merging two NPs into an NP. The C&C parser also has a large number of special case rules for handling punctuation. Our smaller grammar reduces the grammar constant, eases implementation, and simplifies the job of building downstream semantic parsers such as those of Bos (2008) or Lewis and Steedman (2013a) (which must implement semantic analogs of each syntactic rule).

3.5 Extracting Dependency Structures

The parsing model defined in Section 3.2 requires us to compute unlabelled dependency trees from CCG derivations (to prefer non-local attachments). It is simple to extract an unlabelled dependency tree from a CCG parse, by defining one argument of each binary rule instantiation to be the head. For forward application and (generalized) forward composition, we define the head to be the left argument, unless the left argument is an endocentric head-passing modifier category X/X. We do the inverse for the corresponding 'backward' combinators. For punctuation rules, the head is the argument which is not punctuation, and the head of a *Conjunction* rule is the right-hand argument.

The standard CCG parsing evaluation uses a different concept of dependencies, corresponding to the predicate-argument structure defined by CCGBank. These dependencies capture a deeper information—for example by assigning both *boy* and *girl* as subjects of *talk* in *a boy and a girl talked*. We extract these dependencies using the `generate` program supplied with the C&C parser.

3.6 Pruning

Our parsing model is able to efficiently and optimally search for the best parse. However, we found that over 80% of the run-time of our pipeline was spent during supertagging. Naively, the log-linear model needs to output a probability for each of the 425 categories. This is expensive both in terms of the number of dot products required, and the cost of building the initial priority-queue for the A* parsing agenda. It is also largely unnecessary—for example, periods at the end of sentences always have the same category, but our supertagger calculates a distribution over all possible categories.

Note that the motivation for introducing pruning here is fundamentally different from for the C&C pipeline. The C&C supertagger prunes the the categories so that the parser can build the complete set of derivations given those categories. In contrast, our parser can efficiently search large (or infinite) spaces of categories, but pruning is helpful for making supertagging itself more efficient, and for building the initial agenda.

We therefore implemented the following strategies to improve efficiency:

- Only allowing at most 50 categories per word. The C&C parser takes on average 1.27 tags per word (and an average of 3.57 at its loosest beam setting), so this restriction is a very mild one. Nevertheless, it considerably reduces the potential size of the agenda.

- Using a variable-width beam β which prunes categories less likely than β times the probability of the best category. We set $\beta = 0.00001$, which is two orders-of-magnitude smaller than the equivalent C&C beam. Again, this heuristic is useful for reducing the length of the agenda.

- Using a tag dictionary of possible categories for each word, so that weights are only calculated for a subset of the categories. Unlike the other methods, this approach does affect the probabilities which are calculated, as the normalizing constant is only computed for a subset of the categories. However, the probability mass contained in the pruned categories is small, and it only slightly decreases parsing accuracy. To build the tag dictionary, we parsed 42 million sentences of Wikipedia using our parser, and for all words occurring at least 500 times, we stored the set of observed word-category combinations. When parsing new sentences, these words are only allowed to occur with one of these categories.

Supertagger	Parser	CCGBank				Wikipedia			Bioinfer		
		F1 (cov)	COV	F1 (all)	Time	F1 (cov)	COV	F1 (all)	F1 (cov)	COV	F1 (all)
C&C	C&C	85.47	99.63	85.30	54s	81.19	99.0	80.64	76.08	97.2	74.88
EASYCCG	EASYCCG	83.37	99.96	83.37	13s	81.75	100	81.75	77.24	100	77.24
EASYCCG	C&C	86.14	99.96	86.11	69s	82.46	100	82.46	78.00	99.8	77.88

Table 2: Parsing F1-scores for labelled dependencies across a range of domains. F1 (cov) refers to results on sentences which the parser is able to parse, and F1 (all) gives results over all sentences. For the EASYCCG results, scores are only over parses where the C&C dependency extraction script was successful, which was 99.3% on CCGBank, 99.5% on Wikipedia, and 100% on Bioinfer.

4 Experiments

4.1 Experimental Setup

We trained our model on Sections 02-21 of CCGBank (Hockenmaier and Steedman, 2007), using Section 00 for development. For testing, we used Section 23 of CCGBank, a Wikipedia corpus annotated by Honnibal and Curran (2009), and the Bioinfer corpus of biomedical abstracts (Pyysalo et al., 2007). The latter two are out-of-domain, so are more challenging for the parsers.

We compare the performance of our model against both the C&C parser, and the system described in Lewis and Steedman (2014). This model uses the same supertagger as used in EASYCCG, but uses the C&C parser for parsing, using adaptive supertagging with the default values.

All timing experiments used the same 1.8Ghz AMD machine.

4.2 Parsing Accuracy

Results are shown in Table 2. Our parser performs competitively with a much more complex parsing model, and outperforms the C&C pipeline on both out-of-domain datasets. This result confirms our hypothesis that the majority of parsing decisions can be made accurately with a simple tagging model and a deterministic parser.

We see that the combination of the EASYCCG supertagger and the C&C parser achieves the best accuracy across all domains. This result shows that, unsurprisingly, there is some value to having a statistical model of the dependencies that the parser is evaluated on. However, the difference is not large, particularly out-of-domain, considering that a sophisticated and complex statistical parser is being compared with a deterministic one. Our parser is also far faster than this baseline.

It is interesting that the performance gap is much lower on out-of-domain datasets (2.8 points in domain, but only 0.65-0.75 out-of-domain), suggesting that much of the C&C parser's dependency model is domain specific, and does not generalize well to other domains.

	Speed (sentences/second)		
System	Tagger	Parser	Total
C&C	**343**	52	45
EASYCCG tagger + C&C parser	299	58	49
EASYCCG baseline	56	222	45
+Tag Dictionary	185	217	99
+Max 50 tags/word	238	345	141
+β=0.00001	299	**493**	**186**
EASYCCG — *null heuristic*	300	221	127

Table 3: Effect of our optimizations of parsing speed.

We also briefly experimented using the C&C supertagger (with a beam of $\beta = 10^{-5}$) with the EASYCCG parser. Performance was much worse, with an F-score of 79.63% on the 97.8% of sentences it parsed on CCGBank Section 23. This shows that our model is reliant on the accuracy of the supertagger.

4.3 Parsing Speed

CCG parsers have been used in distributional approaches to semantics (Lewis and Steedman, 2013a; Lewis and Steedman, 2013b), which benefit from large corpora. However, even though the C&C parser is relatively fast, it will still take over 40 CPU-days to parse the Gigaword corpus on our hardware, which is slow enough to be an obstacle to scaling distributional semantics to larger cor-

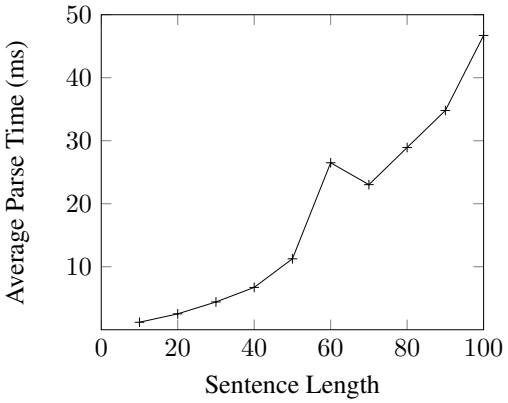


Figure 2: Average parse times in milliseconds, by sentence length.

pora such as ClueWeb. Therefore, it is important to be able to parse sentences at a high speed.

We measured parsing times on Section 23 of CCGBank (after developing against Section 00), using the optimizations described in Section 4.3. We also experimented with the *null heuristic*, which always estimates the outside probability as being 1.0. Times exclude the time taken to load models.

Results are shown in Table 3. The best EASYCCG model is roughly four times faster than the C&C parser[2]. Adding the tag dictionary caused accuracy to drop slightly from 83.46 to 83.37, and meant the parser failed to parse a single sentence in the test set ("*Among its provisions :*") but other changes did not affect accuracy. The pruning in the supertagger improves parsing speed, by limiting the length of the priority queue it builds for the agenda. Of course, we could use a backoff model to ensure full coverage (analogously to adaptive supertagging), but we leave that to future work. Using our A* heuristic doubles the speed of parsing (excluding supertagging).

To better understand the properties of our model, we also investigate how parsing time varies with sentence length. Unlike the cubic CKY algorithm typically used by chart parsers, our A* search potentially takes exponential time in the sentence length. For this experiment, we used the Sections 02-21 of CCGBank. Sentences were divided into bins of width 10, and we calculated the average parsing time for sentences in each bin.

Results are shown in Figure 2, and demonstrate that while parsing is highly efficient for sentences of up to 50 words (over 95% of CCGBank), it scales super-linearly with long sentences. In fact, Section 00 contains a sentence of 249 words, which took 37 *seconds* to parse (3 times longer than the other 1912 sentences put together). In practice, this scaling is unlikely to be problematic, as long sentences are typically filtered when processing large corpora.

4.4 Semantic Parsing

A major motivation for CCG parsing is to exploit its transparent interface to the semantics, allowing syntactic parsers to do much of the work of semantic parsers. Therefore, perhaps the most relevant measure of the performance of a CCG parser is its effect on the accuracy of downstream applications.

We experimented with a supervised version of Reddy et al. (2014)'s model for question-answering on Freebase (i.e. without using Reddy et al.'s lexicon derived from unlabelled text), using the WEBQUESTIONS dataset (Berant et al., 2013)[3]. The model learns to map CCG parses to database queries. We compare the performance of the QA system using both our parser and C&C, taking the 10-best parses from each parser for each sentence. Syntactic question parsing models were trained from the combination of 10 copies of Rimell and Clark (2008)'s question dataset and one copy of the CCGBank

The accuracy of Reddy et al. (2014)'s model varies significantly between iterations of the training data. Rather than tune the number of iterations, we instead measure the accuracy after each iteration. We experimented with the models' 1-best answers, and the oracle accuracy of their 100 best answers. The oracle accuracy gives a better indication of the performance of the parser, by mitigating errors caused by the semantic component.

Results are shown in Figure 3, and demonstrate that using EASYCCG can lead to better downstream performance than the C&C parser. The improvement is particularly large on oracle accuracy, increasing the upper bound on the performance of the semantic parser by around 4 points.

5 Related Work

CCG parsing has been the subject of much research. We have already described the C&C pars-

[2]It is worth noting that the C&C parser code is written in highly-optimized C++, compared to our simple Java implementation. It seems likely that our parser could be made substantially faster with a similar level of engineering effort.

[3]Using the *Business*, *Film* and *People* domains, with 1115 questions for training and 570 for testing.

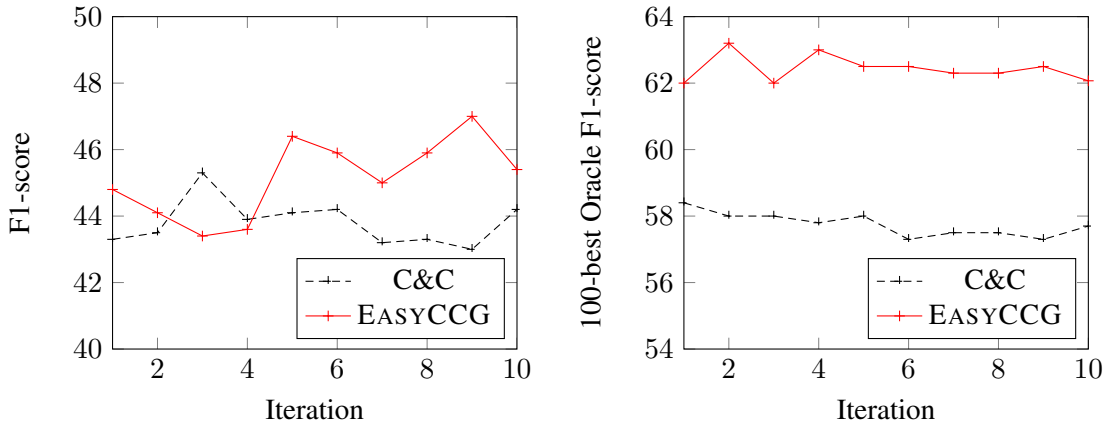


Figure 3: Question Answering accuracy per iteration of Reddy et al. (2014)'s supervised model.

ing model. Kummerfeld et al. (2010) showed that the speed of the C&C parser can be improved with domain-specific self-training—similar improvements may be possible applying this technique to our model. Auli and Lopez (2011a) have achieved the best CCG parsing accuracy, by allowing the parser and supertagger to perform joint inference (though there is a significant speed penalty). Auli and Lopez (2011b) were the first to use A* parsing for CCG, but their system is both much slower and less accurate than ours (due to a different model and a different A* heuristic). Krishnamurthy and Mitchell (2014) show how CCG parsing can be improved by jointly modelling the syntax and semantics. Fowler and Penn (2010) apply the Petrov parser to CCG, making a small improvement in accuracy over the C&C parser, at the cost of a 300-fold speed decrease. Zhang and Clark (2011) and Xu et al. (2014) explored shift-reduce CCG parsing, but despite the use of a linear-time algorithm, parsing speed in practice is significantly slower than the C&C parser.

Parsers based on supertagging models have previously been applied to other strongly lexicalized formalisms, such as to LTAG (Bangalore and Joshi, 1999) and to HPSG (Ninomiya et al., 2006). A major contribution of our work over these is showing that factoring models on lexical categories allows fast and exact A* parsing, without the need for beam search. Our parsing approach could be applied to any strongly lexicalized formalism.

Our work fits into a tradition of attempting to simplify complex models without sacrificing performance. Klein and Manning (2003b) showed that unlexicalized parsers were only slightly less accurate than their lexicalized counterparts. Collobert et al. (2011) showed how a range of NLP tagging tasks could be performed at high accuracy using a small feature set based on vector-space word embeddings. However, the extension of this work to phrase-structure parsing (Collobert, 2011) required a more complex model, and did not match the performance of traditional parsing techniques. We achieve state-of-the-art results using the same feature set and a simpler model by exploiting CCG's lexicalized nature, which makes it more natural to delegate parsing decisions to a tagging model.

Other parsing research has focused on building fast parsers for web-scale processing, typically using dependency grammars (e.g. Nivre (2003)). CCG has some advantages over dependency grammars, such as supporting surface-compositional semantics. The fastest dependency parsers use an *easy-first* strategy, in which edges are added greedily in order of their score, with $\mathcal{O}(nlog(n))$ complexity (Goldberg and Elhadad, 2010; Tratz and Hovy, 2011). This strategy is reminiscent of our A* search, which expands the chart in a best-first order. A* has higher asymptotic complexity, but finds a globally optimal solution.

6 Future Work

We believe that our model opens several interesting directions for future research.

One interesting angle would be to increase the amount of information in CCGBank's lexical entries, to further reduce the search space for the parser. For example, *PP* categories could be distinguished with the relevant preposition as a feature; punctuation and coordination could be given more detailed categories to avoid needing their own combinators, and slashes could be extended

with Baldridge and Kruijff (2003)'s multi-modal extensions to limit over-generation. Honnibal and Curran (2009) show how unary rules can be lexicalized in CCG. Such improvements may improve both the speed and accuracy of our model.

Because our parser is factored on a unigram tagging model, it can be trained from isolated annotated words, and does not require annotated parse trees or full sentences. Reducing the requirements for training data eases the task for human annotators. It may also make the model more amenable to semi-supervised approaches to CCG parsing, which have typically focused on extending the lexicon (Thomforde and Steedman, 2011; Deoskar et al., 2014). Finally, it may make it easier to convert other annotated resources, such as UCCA (Abend and Rappoport, 2013) or AMR (Banarescu et al., 2013), to CCG training data—as only specific words need to be converted, rather than full sentences.

Our model is weak at certain kinds of decisions, e.g. coordination-scope ambiguities or non-local attachments. Incorporating specific models for such decisions may improve accuracy, while still allowing fast and exact search—for example, we intend to try including Coppola et al. (2011)'s model for prepositional phrase attachment.

7 Conclusions

We have shown that a simple, principled, deterministic parser combined with a tagging model can parse an expressive linguistic formalism with high speed and accuracy. Although accuracy is not state-of-the-art on CCGBank, our model gives excellent performance on two out-of-domain datasets, and improves the accuracy of a question-answering system. We have shown that this model allows an efficient heuristic for A* parsing, which makes parsing extremely fast, and may enable logic-based distributional semantics to scale to larger corpora. Our methods are directly applicable to other lexicalized formalisms, such as LTAG, LFG and HPSG.

Acknowledgments

We would like to thank Tejaswini Deoskar, Bharat Ram Ambati, Michael Roth and the anonymous reviewers for helpful feedback on an earlier version of this paper, and Siva Reddy for running the semantic parsing experiments.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of ACL*.

Michael Auli and Adam Lopez. 2011a. A comparison of loopy belief propagation and dual decomposition for integrated CCG supertagging and parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 470–480. Association for Computational Linguistics.

Michael Auli and Adam Lopez. 2011b. Efficient CCG parsing: A* versus adaptive supertagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1577–1585. Association for Computational Linguistics.

Jason Baldridge and Geert-Jan M Kruijff. 2003. Multi-modal combinatory categorial grammar. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1*, pages 211–218. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, Sofia, Bulgaria, August. Association for Computational Linguistics.

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 EMNLP*.

Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In Johan Bos and Rodolfo Delmonte, editors, *Semantics in Text Processing. STEP 2008 Conference Proceedings*, Research in Computational Semantics, pages 277–286. College Publications.

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

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *International Conference on Artificial Intelligence and Statistics*, number EPFL-CONF-192374.

Gregory F Coppola, Alexandra Birch, Tejaswini Deoskar, and Mark Steedman. 2011. Simple semi-supervised learning for prepositional phrase attachment. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 129–139. Association for Computational Linguistics.

Tejaswini Deoskar, Christos Christodoulopoulos, Alexandra Birch, and Mark Steedman. 2014. Generalizing a Strongly Lexicalized Parser using Unlabeled Data. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jason Eisner. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 79–86. Association for Computational Linguistics.

Timothy AD Fowler and Gerald Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Brian Harrington. 2010. A semantic network approach to measuring relatedness. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 356–364. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Julia Hockenmaier. 2003. Data and models for statistical parsing with combinatory categorial grammar.

Matthew Honnibal and James R Curran. 2009. Fully lexicalising CCGbank with hat categories. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1212–1221. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003a. A* parsing: fast exact viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 40–47. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003b. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom M Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. June.

Jonathan K. Kummerfeld, Jessika Roesner, Tim Dawborn, James Haggerty, James R. Curran, and Stephen Clark. 2010. Faster parsing by supertagger adaptation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 345–355. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2013a. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics*, 1:179–192.

Mike Lewis and Mark Steedman. 2013b. Unsupervised induction of cross-lingual semantic relations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 681–692, Seattle, Washington, USA, October. Association for Computational Linguistics.

Mike Lewis and Mark Steedman. 2014. Improved CCG parsing with Semi-supervised Supertagging. *Transactions of the Association for Computational Linguistics (to appear)*.

Takashi Ninomiya, Takuya Matsuzaki, Yoshimasa Tsuruoka, Yusuke Miyao, and Jun'ichi Tsujii. 2006. Extremely lexicalized models for accurate and fast hpsg parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 155–163, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*.

Sampo Pyysalo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. Bioinfer: a corpus for information extraction in the biomedical domain. *BMC bioinformatics*, 8(1):50.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale Semantic Parsing without Question-Answer Pairs. *Transactions of the Association for Computational Linguistics (to appear)*.

Laura Rimell and Stephen Clark. 2008. Adapting a lexicalized-grammar parser to contrasting domains. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 475–484. Association for Computational Linguistics.

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

Mark Steedman. 2012. *Taking Scope: The Natural Semantics of Quantifiers*. MIT Press.

Emily Thomforde and Mark Steedman. 2011. Semi-supervised CCG lexicon extension. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1246–1256. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2011. A fast, effective, non-projective, semantically-enriched parser. In *Proceedings of EMNLP*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Wenduan Xu, Stephen Clark, and Yue Zhang. 2014. Shift-reduce ccg parsing with a dependency model. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*. Association for Computational Linguistics, June.

Yue Zhang and Stephen Clark. 2011. Shift-reduce CCG parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 683–692. Association for Computational Linguistics.