Induction of Linguistic Structure with Combinatory Categorial Grammars

Yonatan Bisk and Julia Hockenmaier
Department of Computer Science
University of Illinois at Urbana-Champaign
201 N Goodwin Ave. Urbana IL, 61801
{bisk1,juliahmr}@illinois.edu

Abstract

Our system consists of a simple, EM-based induction algorithm (Bisk and Hockenmaier, 2012), which induces a language-specific Combinatory Categorial grammar (CCG) and lexicon based on a small number of linguistic principles, e.g. that verbs may be the roots of sentences and can take nouns as arguments.

1 Introduction

Much of the recent work on grammar induction has focused on the development of sophisticated statistical models that incorporate expressive priors (Cohen and Smith, 2010) or linguistic universals (Naseem et al., 2010; Boonkwan and Steedman, 2011) that have all been shown to be very helpful. But, with some notable exceptions, such as (Cohn et al., 2011), the question of what underlying linguistic representation to use has received considerably less attention. Our induction algorithm is based on Combinatory Categorial Grammar (Steedman, 2000), a linguistically expressive, lexicalized grammar formalism which associates words with rich syntactic categories that capture language-specific facts about basic word order and subcategorization. While Boonkwan and Steedman (2011) have shown that linguists can easily devise a language-specific inventory of such categories that allows a parser to achieve high performance in the absence of annotated training data, our algorithm automatically discovers the set of categories it requires to parse the sentences in the training data.

2 Combinatory Categorial Grammar (CCG)

The set of CCG categories is built recursively from two atomic types, S (sentence) and N (noun). Complex types are of the form X/Y or X\Y, and represent functions which combine with an argument of type Y to yield a constituent of type X as result. The slash indicates whether the Y precedes (\) or follows (/) the functor. An English lexicon should contain categories such as S\N and (S\N)/N for verbs: both transitive and intransitive verbs subcategorize for a preceding subject, and the transitive verb additionally takes an object to its right. In this manner, the argument slots of lexical categories also define word-word dependencies between heads and their arguments (Clark and Hockenmaier, 2002; Hockenmaier and Steedman, 2007). Modifiers are generally of the form X|X: in English, pre-nominal adjectives are N/N, whereas adverbs may be (N/N)/(N/N), S/S, or S\S, and prepositions can have categories such as (N\N)/N or (S\S)/N. That is, CCG assumes that the direction of the corresponding dependency goes from the modifier to the head. This discrepancy between CCG and most other analyses can easily be removed under the assumption that all categories of the form X|X are modifiers whose dependencies should be reversed when comparing against other frameworks.

Adjacent constituents can be combined according to a small, universal set of combinatory rules. For the purposes of this work we restrict ourselves to function application and B^1 composition:

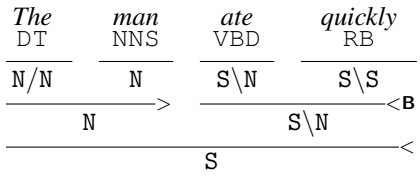
$$X/Y \quad Y \quad \Rightarrow X \quad (>)$$

NAACL-HLT Workshop on the Induction of Linguistic Structure, pages 90–95,
Montréal, Canada, June 3-8, 2012.

Y	X\Y	⇒ X	(<)
X/Y	Y$|_i$Z	⇒ X$|_i$Z	($B^1_>$)
Y$|_i$Z	X\Y	⇒ X$|_i$Z	($B^1_<$)

Here the slash variable $|_i$ can be instantiated with either the forward or backward slash.

These rules allow derivations (parses) such as:



CCG also has unary type-raising rules of the form

X	⇒ T/(T\X)	(>T)
X	⇒ T\(T/X)	(<T)

We only allow nouns to be type-raised, and impose the restriction that the argument T\N (or T/N) of the type-raised category has to already be present in the lexicon of the language.

This restricted set of combinatory rules provides sufficient power for reasonable parse accuracy but does not allow us to capture non-projective (crossing) dependencies.

Coordination is handled by a ternary rule

X	conj	X	⇒ X	(>)

which we binarize as:

X	X[conj]	⇒ X	(< &)
conj	X	⇒ X[conj]	(> &)

Punctuation, when present, can be absorbed by rules of the form

X	Pct	⇒ X	(< p)
Pct	X	⇒ X	(> p)

The iterative combination of these categories resulting in S or N is considered a successful parse. In order to avoid spurious ambiguities, we restrict our derivations to be normal-form (Hockenmaier and Bisk, 2010).

3 An algorithm for unsupervised CCG induction

We now describe our induction algorithm, which consists of two stages: category induction (creation of the grammar), followed by parameter estimation for the probability model.

3.1 Category induction

We assume there are two atomic categories, N (nouns or noun phrases) and S (sentences), a special conjunction category conj, and a special start symbol TOP. We assume that all strings we encounter are either nouns or sentences:

$$\texttt{N} \Rightarrow \texttt{TOP} \qquad \texttt{S} \Rightarrow \texttt{TOP}$$

We also assume that we can group POS-tags into four groups: nominal tags, verbal tags, conjunctions, and others. This allows us to create an initial lexicon $\mathcal{L}^{(0)}$, which only contains entries for atomic categories, e.g. for the English Penn Treebank tag set (Marcus et al., 1993):

$$\begin{aligned} \texttt{N} &: \{\texttt{NN}, \texttt{NNS}, \texttt{NNP}, \texttt{PRP}, \texttt{DT}\} \\ \texttt{S} &: \{\texttt{MD}, \texttt{VB}, \texttt{VBZ}, \texttt{VBG}, \texttt{VBN}, \texttt{VBD}\} \\ \texttt{conj} &: \{\texttt{CC}\} \end{aligned}$$

We force any string that contains one or more verbs (besides VBG in English), to be parsed with the S ⇒ TOP rule.

Since the initial lexicon would only allow us to parse single word utterances (or coordinations thereof), we need to induce complex functor categories. The lexicon entries for atomic categories remain, but all POS-tags, including nouns and conjunctions, will be able to acquire complex categories during induction. We impose the following constraints on the lexical categories we induce:

1. Nouns (N) do not take any arguments.
2. The heads of sentences (S|...) and modifiers (X|X, (X|X)|(X|X)) may take N or S as arguments.
3. Sentences (S) may only take nouns (N) as arguments.
   (We assume S\S and S/S are modifiers).
4. Modifiers (X/X or X\X) can be modified by categories of the form (X/X)|(X/X) or (X\X)|(X\X).
5. The maximal arity of any lexical category is 3.
6. Since (S\N)/N is completely equivalent to (S/N)\N, we only allow the former category.

Induction is an iterative process. At each stage, we aim to parse all sentences S_i in our training corpus $\mathcal{D} = \{S_1, ..., S_D\}$ with the current lexicon

$\mathcal{L}^{(t)}$. In order to parse a sentence $S = w_0...w_n$, all words $w_i \in S$ need to have lexical categories that allow a complete parse (resulting in a constituent TOP that spans the entire sentence). Initially, only some words will have lexical categories:

The	*man*	*ate*	*quickly*
DT	NNS	VBD	RB
-	N	S	-

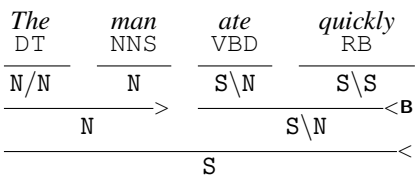
We assume that any word may modify adjacent constituents:

The	*man*	*ate*	*quickly*
DT	NNS	VBD	RB
N/N	N, S/S	S, N\N	S\S

We also assume that any word that previously had a category other than N (which we postulate does not take any arguments) can take any adjacent non-modifier category as argument, leading us here to introduce S\N for the verb:

The	*man*	*ate*	*quickly*
DT	NNS	VBD	RB
N/N	N, S/S	S, N\N, S\N	S\S

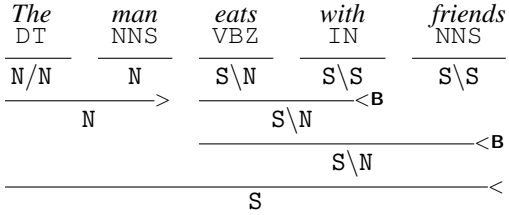
With these categories, we obtain the correct parse:



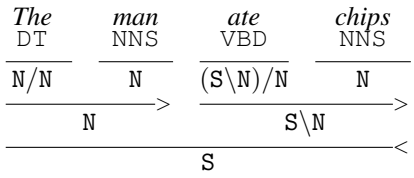
We then update the lexicon with all new tag-category pairs that have been found, excluding those that did not lead to a successful parse:

N/N : {DT} S\N : {VBD, VBZ} S\S : {RB, NNS, IN}

The first stage of induction can only introduce functors of arity 1, but many words, such as prepositions or transitive verbs, require more complex categories, leading us to complete, but incorrect parses such as



During the second iteration, we can discover additional simple, as well as more complex, categories. We now discover transitive verb categories:



The second stage also introduces a large number of complex modifiers of the form (X/X)|(X/X) or (X\X)|(X\X), e.g.:

The	*man*	*ate*	*very*	*quickly*
DT	NNS	VBD	RB	RB
N/N, (S/S)/(S/S)	N, S/S (N\N)/(N\N) (N/N)\(N/N)	S, N\N, S\N (S/S)\(S/S) (S\S)/(S\S)	S\S, (S\S)/(S\S) (N\N)\(N\N)	S\S, (S\S)\(S\S)

The final induction step takes adjacent constituents that can be derived from the existing lexicon into account. This allows us to induce (S\S)/N for IN, since we can combine *a* and *friend* to N.

3.2 Parameter estimation

After constructing the lexicon, we parse the training corpus, and use the Inside-Outside algorithm (Lari and Young, 1991), a variant of the Expectation-Maximization algorithm for probabilistic context-free grammars, to estimate model parameters. We use the baseline model of Hockenmaier and Steedman (2002), which is a simple generative model that is equivalent to an unlexicalized PCFG. In a CFG, the set of terminals and non-terminals is disjoint, but in CCG, not every category can be lexical. Since this model is also the basis of a lexicalized model that captures dependencies, it distinguishes between lexical expansions (which produce words), unary expansions (which are the result of type-raising or the TOP rules), binary expansions where the head is the left child, and binary expansions whose head is the right child. Each tree is generated top-down from the start category TOP. For each (parent) node, first its expansion type $exp \in \{\text{Lex}, \text{Unary}, \text{Left}, \text{Right}\}$ is generated. Based on the expansion type, the model then produces either the word w, or the category of the head child (H), and, possibly the category of the non-head sister category (S):

Lexical $p_e(\exp{=}\text{Lex} \mid \text{P}) \times p_w(w \mid \text{P}, \exp{=}\text{Lex})$

Unary $p_e(\exp{=}\text{Unary} \mid \text{P}) \times p_\text{H}(\text{H} \mid \text{P}, \exp{=}\text{Unary})$

Left $p_e(\exp{=}\text{Left} \mid \text{P}) \times p_\text{H}(\text{H} \mid \text{P}, \exp{=}\text{Left})$
$\times\, p_S(\text{S} \mid \text{P}, \text{H}, \exp{=}\text{Left})$

Right $p_e(\exp{=}\text{Right} \mid \text{P}) \times p_\text{H}(\text{H} \mid \text{P}, \exp{=}\text{Right})$
$\times\, p_S(\text{S} \mid \text{P}, \text{H}, \exp{=}\text{Right})$

3.3 Dependency generation

We use the following regime for generating dependencies from the resulting CCG derivations:

1. Arguments `Y` are dependents of their heads `X|Y`
2. Modifiers `X|X` are dependents of their heads `X` or `X|Y`.
3. The head of the entire string is a dependent of the root node (0)
4. Following the CoNLL-07 shared task representation (Johansson and Nugues, 2007), we analyze coordinations ($\texttt{X}_1$ `conj` $\texttt{X}_2$) as creating a dependency from the first conjunct, $\texttt{X}_1$, to the conjunction `conj`, and from `conj` to the second conjunct $\texttt{X}_2$.

In the case of parse failures we return a right-branching dependency tree.

3.4 Training details

The data provided includes fine, coarse and universal part-of-speech tags. Additionally, the data was split into train, test and development sets though the organizers encouraged merging the data for training. Finally, while punctuation was present, it was not evaluated but potentially provided an additional source of signal during training and test. We chose from among these options and maximum sentence length based on performance on the development set. We primarily focused on training with shorter sentences but grew the dataset if necessary or if, as is the case in Arabic, there was very little short sentence data. Our final training settings were:

Language	Tags	Max Len	Punc
Arabic	Fine	40	✓
Basque	Coarse	20	
Childes	Fine	20	✓
Czech	Fine	10	
Danish	Fine	20	✓
Dutch	Fine	10	✓
Slovene	Fine	10	✓
Swedish	Fine	15	
PTB	Fine	10	
Portuguese	Fine	10	

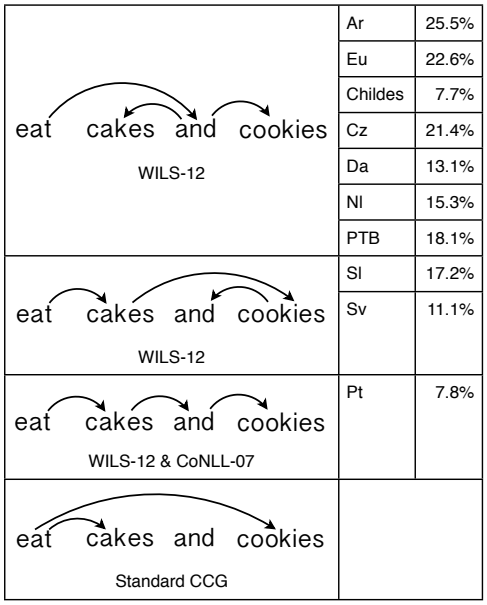
In the case of Czech, we only trained on the test-set because the data set was so large and the results from randomly downsampling the merged dataset were equivalent to simply using the previously defined test-set.

3.5 Future directions

Since our current system is so simple, there is ample space for future work. We plan to investigate the effect of more complex statistical models and priors that have been shown to be helpful in dependency grammar-based systems. We also wish to relax the assumption that we know in advance which part-of-speech tags are nouns, verbs, or conjunctions.

4 Final observations regarding evaluation

Although the analysis of constructions involving basic head-argument and head-modifier dependencies is generally uncontroversial, many common constructions allow a number of plausible analyses. This makes it very difficult to evaluate and compare different unsupervised approaches for grammar induction. The corpora used in this workshop also assume different conventions for a number of constructions. Figure 1 shows the three different types of analysis for coordination adopted by the corpora used in this shared task (as well as the standard CCG analysis). The numbers to the side indicate for each corpus what percentage of our system's error rate is due to missed dependencies within coordinated structures (i.e between a conjunction and a conjunct, or between two conjuncts). It is important to note that the way in which we extract dependencies from coordinations is somewhat arbitrary (and completely independent of the underlying probability model, which currently captures no explicit de-

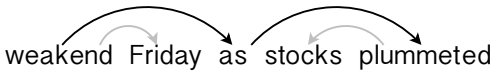


Ar	25.5%
Eu	22.6%
Childes	7.7%
Cz	21.4%
Da	13.1%
Nl	15.3%
PTB	18.1%
Sl	17.2%
Sv	11.1%
Pt	7.8%

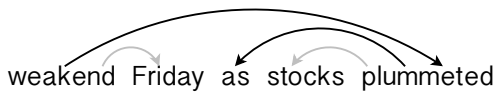
Figure 1: Different analyses of coordination in the various corpora used in this shared task. Our system adopts the CoNLL-07 convention, instead of the standard CCG analysis. For the development set of each corpus, we also indicate what percentage of the errors our system makes is due to missed coordination dependencies.

pendencies). These systematic differences of analysis are also reflected in our final results. The only exception is the Childes corpus, where coordination is significantly rarer.

However, this is a general problem. There are many other constructions for which no agreed-upon standard exists. For example, the Wall Street Journal data used in this shared task assumes a dependency between the verb of the main clause and the verb of a subordinate clause, whereas the CoNLL-07 analysis stipulates a dependency between the main verb and the subordinating conjunction:



(a) CoNLL-07



(b) WILS-12

We therefore believe that much further work is required to address the problems surrounding evaluation and comparison of unsupervised induction systems adequately. Even if the community cannot agree on a single gold standard, systems should not be penalized for producing one kind of linguistically plausible analysis over another. The systematic divergences that arise with coordination for our approach are relatively easy to fix, since we only need to change the way in which we read off dependencies. But this points to a deeper underlying problem that affects the entire field.

Acknowledgements

This research is supported by the National Science Foundation through CAREER award 1053856 and award 0803603.

References

Yonatan Bisk and Julia Hockenmaier. 2012. Simple Robust Grammar Induction with Combinatory Categorial Grammars. In *Association for the Advancement of Artificial Intelligence*.

Prachya Boonkwan and Mark Steedman. 2011. Grammar Induction from Text Using Small Syntactic Prototypes. In *International Joint Conference on Natural Language Processing*, pages 438–446, November.

Stephen Clark and Julia Hockenmaier. 2002. Evaluating a wide-coverage CCG parser. In *Proceedings of the LREC Beyond PARSEVAL workshop*, page 2002, Las Palmas, Spain.

S. B. Cohen and N. A. Smith. 2010. Covariance in unsupervised learning of probabilistic grammars. *Journal of Machine Learning Research*, 11:3017–3051.

Trevor Cohn, Phil Blunsom, and Sharon Goldwater. 2011. Inducing tree-substitution grammars. *Journal of Machine Learning Research*, pages 3053–3096, November.

Julia Hockenmaier and Yonatan Bisk. 2010. Normal-form parsing for Combinatory Categorial Grammars with generalized composition and type-raising. In *COLING*.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Association for Computational Linguistics*, pages 335–342.

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

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of NODALIDA 2007*, Tartu, Estonia.

K Lari and S Young. 1991. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 5(3):237–257, January.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Tahira Naseem, Harr Chen, Regina Barzilay, and Mark Johnson. 2010. Using universal linguistic knowledge to guide grammar induction. In *Empirical Methods in Natural Language Processing*, pages 1234–1244, October.

Mark Steedman. 2000. The syntactic process. *MIT Press*, January.