Configurable Dependency Tree Extraction from CCG Derivations

Kilian Evang
Heinrich Heine University Düsseldorf
Germany
kilian.evang@gmail.com

Abstract

We revisit the problem of extracting dependency structures from the derivation structures of Combinatory Categorial Grammar (CCG). Previous approaches are often restricted to a narrow subset of CCG or support only one flavor of dependency tree. Our approach is more general and easily configurable, so that multiple styles of dependency tree can be obtained. In an initial case study, we show promising results for converting English, German, Italian, and Dutch CCG derivations from the Parallel Meaning Bank into (unlabeled) UD-style dependency trees.

1 Introduction

In a world of heterogeneous linguistically annotated resources, the need often arises to convert annotations from one format into another. The purpose may be to extract features, make data available as input for tools that were not designed for it, or to compare heterogeneous tool outputs on an equal footing. For example, sentences annotated with derivations of Combinatory Categorial Grammar (CCG) (Steedman, 2001) must often be converted to dependency graphs or trees. Figure 1 shows an example.

Clark et al. (2002) define a conversion from derivations to dependency graphs (containing both local and long-range dependencies) by annotating every lexical category that occurs in the English CCGbank (Hockenmaier and Steedman, 2007), specifying the bilexical dependency (or dependencies) that each argument category gives rise to. This annotated inventory of categories and variations thereof have since widely been used to train and evaluate CCG parsers, e.g., Clark and Curran (2007), Zhang and Clark (2011), Lewis and Steedman (2014), Stanojević and Steedman (2019). The obvious drawback of this approach is that each category has to be annotated manually, and adapting the scheme to other languages, other flavors of CCG, or other flavors of dependency graph, is thus labor-intensive.

A simple alternative is to just extract a dependency for each argument category, with the word corresponding to the argument as the dependent, and the word corresponding to the result category as the head. This is, e.g., used for parser training and evaluation for English by Lewis and Steedman (2014) and for English and Japanese by Yoshikawa et al. (2017). Koller and Kuhlmann (2009) define a similar algorithm operating on derivations rather than lexical categories, which however only supports pure first-order CCGs. The main drawback is that only one flavor of dependency tree is supported, in which modifiers and function words such as determiners end up as heads of what they modify due to the way

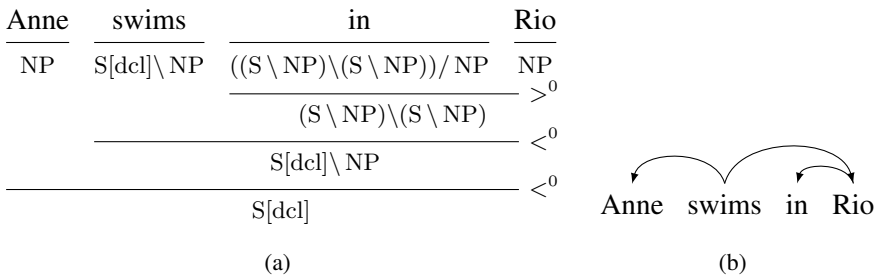

Figure 1: A CCG derivation and corresponding unlabeled UD dependency tree

Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020), pages 87–93
Barcelona, Spain (Online), December 13, 2020

CCG uses functional categories for them. Bisk and Hockenmaier (2013) use a simple algorithm with some tricks to accomodate a variety of different dependency treebank formats, but it only supports a heavily simplified version of CCG with two basic categories, S and N.

In this paper we present a novel approach to extracting dependencies from CCG derivations. It is similar to Clark et al.'s approach in that the lexical categories, augmented with dependency information for each argument category, serve as the basis for the extraction. However, we do not annotate each possible lexical category manually, but rather define a limited and mostly language-independent set of rules to do so automatically, which can be adapted to different dependency flavors easily. Like the other approaches mentioned, we limit ourselves to local dependencies. As a result, the resulting structures are guaranteed to be trees. As a case study, we experiment with converting English, German, Italian, and Dutch derivations from the Parallel Meaning Bank (Abzianidze et al., 2017) to (unlabeled) dependency trees compatible with the annotation guidelines of Basic Universal Dependencies (UD) (Nivre et al., 2016).

2 CCG-to-Dependency Conversion by Argument Category Augmentation

In a CCG derivation, each word is assigned a syntactic *category* that encodes its combinatory potential in terms of syntactic arguments. For example, an English transitive verb like *sees* in *Mary sees John* has the category $(\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP}$, meaning that it combines with an (object) NP to the right and then with a (subject) NP on the left, and that the resulting constituent is a sentence (S):

(1)

Mary	sees	John
NP	$(\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP}$	NP

Categories with slashes (i.e., with arguments) are called *functional*, others are called *basic*. We call the basic category that remains after stripping all arguments from a functional category its *top category*, e.g., S is the top category of $(\mathrm{S} \setminus \mathrm{NP})/\mathrm{NP}$.

The first step in our conversion algorithm is to augment each basic category with a pair of variables X, D, where X represents its identity and D will later be bound to a marker (F or A) that indicates the direction of the dependency:

(2)

Mary	sees	John
$\mathrm{NP}_{A,V}$	$(\mathrm{S}_{B,W} \setminus \mathrm{NP}_{C,X})/\mathrm{NP}_{D,Y}$	$\mathrm{NP}_{E,Z}$

We then unify argument categories of functors with the categories of their arguments (this can be done by traversing the derivation):

(3)

Mary	sees	John
$\mathrm{NP}_{A,V}$	$(\mathrm{S}_{B,W} \setminus \mathrm{NP}_{A,V})/\mathrm{NP}_{D,Y}$	$\mathrm{NP}_{D,Y}$

We now bind the identity variable of the top category of each word to its token number:

(4)

Mary	sees	John
$\mathrm{NP}_{1,V}$	$(\mathrm{S}_{2,W} \setminus \mathrm{NP}_{1,V})/\mathrm{NP}_{3,Y}$	$\mathrm{NP}_{3,Y}$

In the next step, each argument category gets a dependency direction marker. In our example, there are two argument categories, $\mathrm{NP}_{1,V}$ and $\mathrm{NP}_{3,Y}$, both arguments of the functor category $(\mathrm{S}_{2,W} \setminus \mathrm{NP}_{1,V})/\mathrm{NP}_{3,Y}$. In this case, the functor-argument relation of CCG corresponds exactly to the head-dependent relation of most dependency grammars. We indicate this by binding the top category of every argument category to F, meaning that the functor is the head:

(5)

Mary	sees	John
$\mathrm{NP}_{1,\mathrm{F}}$	$(\mathrm{S}_{2,W} \setminus \mathrm{NP}_{1,\mathrm{F}})/\mathrm{NP}_{3,\mathrm{F}}$	$\mathrm{NP}_{3,\mathrm{F}}$

But the functor is not always the head. Consider the following example:

(6)

Anne	swims	in	Rio
$\mathrm{NP}_{1,R}$	$\mathrm{S}_{2,S} \setminus \mathrm{NP}_{E,X}$	$((\mathrm{S}_{3,U} \setminus \mathrm{NP}_{1,R}) \backslash (\mathrm{S}_{2,S} \setminus \mathrm{NP}_{E,X}))/\mathrm{NP}_{4,Y}$	$\mathrm{NP}_{4,Y}$

Remaining category	Head word	New dependency
	3	
$((\mathrm{S}_{3,U} \setminus \mathrm{NP}_{1,\mathrm{F}}) \backslash (\mathrm{S}_{2,\mathrm{A}} \setminus \mathrm{NP}_{E,X})) / \mathrm{NP}_{4,\mathrm{A}}$	4	$\langle 4, 3 \rangle$
$(\mathrm{S}_{3,U} \setminus \mathrm{NP}_{1,\mathrm{F}}) \backslash (\mathrm{S}_{2,\mathrm{A}} \setminus \mathrm{NP}_{E,X})$	2	$\langle 2, 4 \rangle$
$\mathrm{S}_{3,U} \setminus \mathrm{NP}_{1,\mathrm{F}}$	2	$\langle 2, 1 \rangle$
$\mathrm{S}_{3,U}$	2	

Table 1: Conversion of the category of *in* in (7) into dependencies

Under this CCG analysis, the preposition *in* is a functor that takes two arguments: first, the NP *Rio*, and second, the VP that it modifies (*swims*). However, in a UD tree, *in* would not be the head of either *Rio* or *swims*. Instead, the preposition should become a dependent of its object (*Rio*), which in turn becomes a dependent of the verb that it modifies (*swims*). Thus, the dependency direction markers of the arguments of *in* are bound to A, indicating that the argument is the head. The other argument categories are still marked F:

(7)	Anne	swims	in	Rio
	$\mathrm{NP}_{1,\mathrm{F}}$	$\mathrm{S}_{2,\mathrm{A}} \setminus \mathrm{NP}_{E,X}$	$((\mathrm{S}_{3,U} \setminus \mathrm{NP}_{1,\mathrm{F}}) \backslash (\mathrm{S}_{2,\mathrm{A}} \setminus \mathrm{NP}_{E,X})) / \mathrm{NP}_{4,\mathrm{A}}$	$\mathrm{NP}_{4,\mathrm{A}}$

Unfortunately, it is not obvious from this representation between which pairs of words dependencies exist. For example, the presence of $\mathrm{NP}_{1,\mathrm{F}}$ as an argument category in the category of *in* might suggest that *Anne* depends on *in* when it should depend on *swims*. To correctly convert categories into sets of dependencies, we need to process arguments from outermost to innermost and take into account that the head word we need to attach dependents to changes every time we encounter an A-type dependency.

The process is illustrated in Table 1 for *in*. We start out with word 3 (*in*) itself as the head word, but already the first argument is A-type, so the argument head, word 4, becomes the new head word, and we generate a dependency from it to the former head, word 3. The second argument is A-type as well, so the argument head, word 2, becomes the new head word, and we generate a dependency from it to the former head, word 4. The third argument is F-type, so we generate a dependency from the current head, word 2, to the argument head, word 1. Word 2 remains the final head associated with *in*. Note that when we speak of "argument heads", we mean the final heads associated with argument categories (more properly: with the words whose categories have the same top category), found by recursively processing them.

3 Identifying Modifiers and Function Words

Which arguments to mark with F, and which with A, depends on the style of dependency tree one wishes to obtain. Marking every argument with F would yield very modifier-centric and function-word-centric trees due to CCG's treatment of adjectives, adverbs, adpositions, conjunctions, etc., as functors. Most dependency grammars, however, prefer a modifiee-centric, and to varying degrees also content-word-centric, style. Universal Dependencies (Nivre et al., 2016) is an example of an especially content-word-centric style, treating function words as dependents of content words whenever possible. SUD (Gerdes et al., 2018) is a variant of UD that differs mainly in being less content-word-centric. Non-UD treebanks show considerable variation as to which function words they treat as heads and which as dependents, cf. Gelling et al. (2012).

Our solution is to mark arguments with F by default, but to define a list of rules that identify modifier and function word categories whose outermost argument category should be marked with A. We group the rules into different modules (one for modifiers, one for determiners, etc.) that can be turned on and off individually, so various dependency styles are supported. We base our rules on the CCG category inventory of the Parallel Meaning Bank (PMB) (Abzianidze et al., 2017), a large quadrilingual corpus of sentences annotated with CCG derivations. Our rule inventory is shown in Table 2, where X, Y match anything and $|$ matches both $/$ and $\backslash$.

Not in all cases are categories sufficient for identifying function words, so we use the PMB annotations

Module	Symbols	Semtags	Category
Coordinating conjunctions		NIL QUE GRP COO	$(X \backslash X)/X$
Modifiers			$X \mid X$
Adjective copulas	be		$(\mathrm{S[X]} \backslash \mathrm{NP}) \mid (\mathrm{S[adj]} \backslash \mathrm{NP})$ $(\mathrm{S[q]}/(\mathrm{S[adj]} \backslash \mathrm{NP}))/\mathrm{NP}*$
Noun copulas	be		$(\mathrm{S}[X] \backslash \mathrm{NP}) \mid \mathrm{NP}$ $(\mathrm{S[q]}/\mathrm{NP})/\mathrm{NP}*$
Adposition copulas	be		$(\mathrm{S}[X] \backslash \mathrm{NP}) \mid \mathrm{PP}$ $(\mathrm{S[q]}/\mathrm{PP})/\mathrm{NP}*$
Auxiliaries and modals		NOW PST FUT PRG PFT NEC POS NIL	$(\mathrm{S}[X] \backslash \mathrm{NP}) \mid (\mathrm{S}[Y] \backslash \mathrm{NP})$ $(\mathrm{S[q]}/(\mathrm{S}[X] \backslash \mathrm{NP}))/\mathrm{NP}*$
Adpositions			$\mathrm{PP} \mid \mathrm{NP}$ $(X \mid X) \mid Y$
Determiners			NP/N $\mathrm{NP}/(\mathrm{N}/\mathrm{PP})$
Possessive suffix			$(\mathrm{NP}/(\mathrm{N}/\mathrm{PP})) \backslash \mathrm{NP}$
Subordinating conjunctions			$(\mathrm{S} \mid \mathrm{S}) \mid \mathrm{S}[X]$ $(\mathrm{S} \mid \mathrm{S}) \mid (\mathrm{S} \mid \mathrm{NP})$ $((\mathrm{S} \backslash \mathrm{NP}) \mid (\mathrm{S} \backslash \mathrm{NP})) \mid \mathrm{S}[X]$ $((\mathrm{S} \backslash \mathrm{NP}) \mid (\mathrm{S} \backslash \mathrm{NP})) \mid (\mathrm{S}[X] \mid \mathrm{NP})$ $((\mathrm{S}/\mathrm{NP}) \mid (\mathrm{S}/\mathrm{NP})) \mid \mathrm{S}[X]$ $((\mathrm{S}/\mathrm{NP}) \mid (\mathrm{S}/\mathrm{NP})) \mid (\mathrm{S}[X] \mid \mathrm{NP})$
Complementizers			$\mathrm{S[em]}/\mathrm{S[dcl]}$ $(\mathrm{S[to]} \backslash \mathrm{NP})/(\mathrm{S[b]} \backslash \mathrm{NP})$
Relativizers			$(\mathrm{N} \backslash \mathrm{N})/(\mathrm{S[dcl]} \mid \mathrm{NP})$ $(\mathrm{NP} \backslash \mathrm{NP})/(\mathrm{S[dcl]} \mid \mathrm{NP})$ $((\mathrm{N} \backslash \mathrm{N})/(\mathrm{S[dcl]} \mid \mathrm{NP}))/\mathrm{N}$ $((\mathrm{NP} \backslash \mathrm{NP})/(\mathrm{S[dcl]} \mid \mathrm{NP}))/\mathrm{N}$
Fronted *wh*-words			$\mathrm{S[wq]}/(\mathrm{S[q]}/(\mathrm{S[adj]}/\mathrm{NP}))$ $\mathrm{S[wq]}/X$ $(\mathrm{S[wq]}/X)/Y$

Table 2: Simplified rule inventory

of symbols (language-independent generalization of lemmas) and semantic tags in some rules. Copulas, auxiliaries, and modals in verb-first (question) sentences have the problem that our algorithm attaches them to the subject, rather than the complement, due to the inverted order of arguments. We mark them specially and reattach them automatically in postprocessing. The affected categories are marked with an asterisk. The rule inventory shown here is simplified for space reasons. The full inventory is released with our data and code.

4 Case Study: Converting the PMB to UD-style Dependencies[1]

To initially develop and test our conversion algorithm, we annotated parts 00 (for development) and 01 (for testing) of the Parallel Meaning Bank, version 3.0.0. We only used sentences whose annotation status was marked "gold". One trained annotator annotated the sentences with dependency trees following the Basic UD guidelines, but without labels, indicating only the head for each word. We then converted the PMB CCG derivations to dependency trees using the algorithm presented above, with all modules turned on, and computed the unlabeled attachment score when comparing to the manual annotation. The results, shown in Table 3, are consistently above 90%, suggesting that our conversion basically works, although some discrepancies remain.

We performed a manual error analysis to see what kinds of discrepancies were encountered. In a few cases, the discrepancies have nothing to do with our algorithm but are the result of plain annotation errors in the PMB or attachment ambiguity that the PMB's annotators and our annotator resolved differently. Unlike UD, the PMB has no strict rules for attaching punctuation, so it sometimes ends up modifying a different constituent from the one that the UD guidelines mandate. There are various relatively rare categories and constructions that are not yet correctly identified by our rule inventory (sometimes due to arguably inconsistent annotation in the PMB), such as English tag questions, embedded questions,

[1]Our data and code is available at `https://github.com/texttheater/pmb2tsv`.

	Development				Testing			
	English	German	Italian	Dutch	English	German	Italian	Dutch
Statistics								
Sentences	383	100	73	37	208	35	27	13
Mean length	6.0	5.53	5.1	5.7	6.4	5.4	5.6	5.1
Scores								
UAS	.982	.993	.936	.921	.977	.937	.976	1
Error analysis								
PMB annotation error	1	1						
Attachment ambiguity	3				2	1		
Punctuation	4				1			
Tag question	2							
Embedded question					1			
Date					1			
Subordinating conjunction					1			
Possessive suffix					1	1		
Copula/auxiliary/modal				1	1		1	
Expletive nominal			2	1				
Question verb		3	1					
Inverted copula	1							
Pseudo-copula	1	1	1	2				
Non-local dependencies	4							

Table 3: Evaluation results and error analysis

certain date expressions, certain subordinating conjunctions, certain instances of the German and English possessive suffix, certain copulas, auxiliaries, and modals, certain expletive nominals, certain verbs in questions, and inverted copulas that have their subject on the right rather than the left. In the PMB analyses of expressions like Dutch *gelijk hebben* ("to be right") or German *Hunger haben* ("to be hungry"), the verb is curiously tagged like a copula, matching the English translation, which also leads to discrepancies with our analyses. Finally, there are some dependencies that our algorithm cannot extract correctly because they are mediated non-locally via third categories in the PMB annotation. For example, here, the existential *is* mediates the dependency between *person* and *ginger* [PMB 00/0055]:

(8)

There	is	no	person	cutting	some	ginger
$\mathrm{NP}[thr]$	$((\mathrm{S[dcl]}\backslash\mathrm{NP})/(\mathrm{S[ng]}\backslash\mathrm{NP}))/\mathrm{NP}$	NP/N	N	$(\mathrm{S[ng]}\backslash\mathrm{NP})/\mathrm{NP}$	NP/N	N

And here, the emphatic *how* mediates the dependency between *slow* and *are* [PMB 00/0778]:

(9)

How	slow	you	are
$(\mathrm{S[dcl]}/(\mathrm{S[dcl]}/\mathrm{S[adj]}\backslash\mathrm{NP}))/\mathrm{S[adj]}\backslash\mathrm{NP}$	$\mathrm{S[adj]}\backslash\mathrm{NP}$	NP	$(\mathrm{S[dcl]}\backslash\mathrm{NP})/(\mathrm{S[adj]}\backslash\mathrm{NP})$

5 Conclusions

We have presented a general and configurable algorithm to convert CCG derivations into dependency trees, together with a rule inventory for the Parallel Meaning Bank, achieving promising accuracy for conversion to unlabeled Basic UD on English, German, Italian, and Dutch. Compared to the method of Clark et al. (2002), the need for manually marking up all lexical category types in a CCG treebank is reduced because we only have to list the categories of modifiers and function words. However, if our approach was to be extended to dependency schemes with nonlocal dependencies, such as Enhanced UD (Schuster and Manning, 2016), or if labeled dependencies were desired, markup of content words would be required as well. And as our error analysis shows, at least in the PMB, there is a long tail of categories that cannot be discovered by looking at a small sample. Still, the modularity of our approach enables it to support different dependency treebank formats. The work has many possible applications, including feature extraction from CCG parses and cross-framework and cross-language parser evaluation.

Acknowledgments

The author would like to thank the anonymous reviewers for helpful feedback. This research was carried out within the TreeGraSP project, funded by a Consolidator Grant of the European Research Council (ERC).

References

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The Parallel Meaning Bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 242–247, Valencia, Spain, April. Association for Computational Linguistics.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP model for inducing Combinatory Categorial Grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

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

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 327–334, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Douwe Gelling, Trevor Cohn, Phil Blunsom, and João Graça. 2012. The PASCAL challenge on grammar induction. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 64–80, Montréal, Canada, June. Association for Computational Linguistics.

Kim Gerdes, Bruno Guillaume, Sylvain Kahane, and Guy Perrier. 2018. SUD or surface-syntactic universal dependencies: An annotation scheme near-isomorphic to UD. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 66–74, Brussels, Belgium, November. 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.

Alexander Koller and Marco Kuhlmann. 2009. Dependency trees and the strong generative capacity of CCG. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 460–468, Athens, Greece, March. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2371–2378, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Miloš Stanojević and Mark Steedman. 2019. CCG parsing algorithm with incremental tree rotation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 228–239, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Mark Steedman. 2001. *The Syntactic Process*. The MIT Press.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 277–287, Vancouver, Canada, July. Association for Computational Linguistics.

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*, pages 683–692, Portland, Oregon, USA, June. Association for Computational Linguistics.