Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 5220–5233
Marseille, 20-25 June 2022

Development of a Multilingual CCG Treebank via Universal Dependencies Conversion

Tu-Anh Tran, Yusuke Miyao
The University of Tokyo
7-3-1 Hongo, Bunkyo, Tokyo, Japan
{anh, yusuke}@is.s.u-tokyo.ac.jp

Abstract

This paper introduces an algorithm to convert Universal Dependencies (UD) treebanks to Combinatory Categorial Grammar (CCG) treebanks. As CCG encodes almost all grammatical information into the lexicon, obtaining a high-quality CCG derivation from a dependency tree is a challenging task. Our algorithm relies on hand-crafted rules to assign categories to constituents, and a non-statistical parser to derive full CCG parses given the assigned categories. To evaluate our converted treebanks, we perform lexical, sentential, and syntactic rule coverage analysis, as well as CCG parsing experiments. Finally, we discuss how our method handles complex constructions, and propose possible future extensions.

Keywords: treebank, combinatory categorial grammar, universal dependencies

1. Introduction

Combinatory Categorial Grammar (Steedman, 2000) is a lexicalized grammar formalism that can capture both syntactic and semantic information, while allowing fast and efficient parsing. Derived syntactic structures and semantic representations can be used for various downstream tasks without task-specific training data, such as question answering (Clark et al., 2004), relation extraction (Krishnamurthy and Mitchell, 2012), and recognizing textual entailment (Martínez-Gómez et al., 2017). The English CCGbank (Hockenmaier and Steedman, 2007), one of the first available treebanks for CCG, plays an important role in the development of many wide-coverage CCG parsers for English. Having a similar resource for other languages and domains accelerates NLP research, in particular on resource-scarce languages/domains where one cannot rely on massive training data needed for training large neural network models (Peters et al., 2018; Devlin et al., 2019). Multilingual CCG resources also contribute to cross-linguistic research on syntactic/semantic theories and multilingual CCG parsing.

Since manual annotation is expensive, conversion from a source treebank is a preferable approach. Besides English, independent works have been done in the past to create CCG treebanks for several languages from source treebanks of different grammar formalisms, such as for German (Hockenmaier, 2006), Italian (Bos et al., 2009), Chinese (Tse and Curran, 2010), Japanese (Uematsu et al., 2013), and Hindi (Ambati et al., 2018). Such works often involve conversion rules that are specific to the languages and treebanks being converted, making the process difficult to adapt and generalize to others.

In this paper, we propose a method to create a multilingual collection of CCG treebanks by converting from dependency treebanks. To minimize the need for language-specific conversion rules, we select the Universal Dependencies (Nivre et al., 2016) as our source treebanks. UD, as of v2.9, contains over 200 treebanks in 122 languages that follow cross-linguistically consistent annotation guidelines.[1] Our goal is to develop a universal set of hand-crafted rules that can be applied to a wide range of languages in UD, while sacrificing as little as possible the conversion quality and coverage of each converted treebank. Converted CCG treebanks can be used directly to train multilingual CCG parsers as we demonstrate in the experiments, while one can also use our resource as a starting point to further improve the quality of each treebank by adding language-specific conversion rules. Our work thus opens up a new research direction for the development of CCG resources, parsers, and semantic analysis that uses them.

A high-level overview of our conversion process is illustrated in Figure 1. Our algorithm consists of two phases, each corresponding to a single pass through a whole treebank in UD:

Phase 1:

1. Binarize dependency trees based on a pre-defined obliqueness hierarchy to match the binary structures of CCG derivations.
2. Apply hand-crafted rules that assign CCG categories to noun phrases, modifiers, function words, and punctuation marks.
3. Apply CCG's combinatory rules to infer the categories of unassigned constituents.

Phase 2:

1. Gather votes from assigned categories in Phase 1 to decide the most common word order of a treebank.
2. Based on the most common word order, reassign CCG categories to predicates.
3. Apply a non-statistical CCG parser to generate candidate parses given currently assigned categories.
4. Filter out parses that do not match our binarized

[1] `https://universaldependencies.org/`.

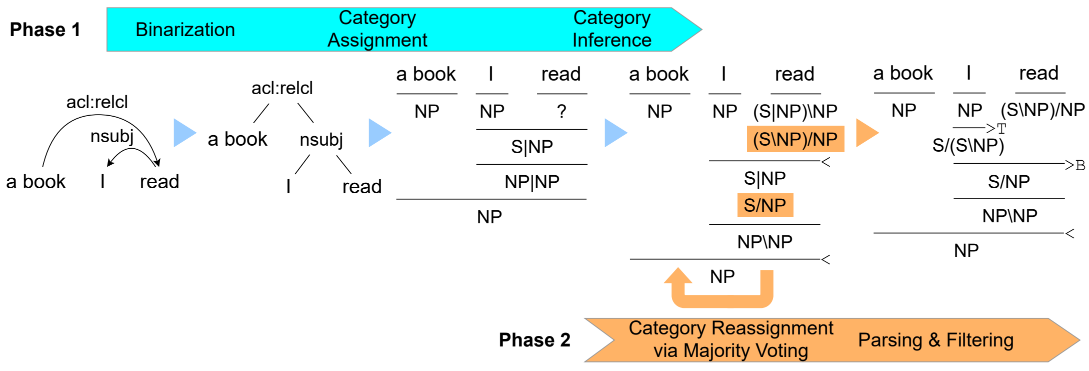


Figure 1: Overview of our proposed approach for converting a UD tree to its equivalent CCG derivation.

Forward Application (>)	X/Y Y	⇒ X
Backward Application (<)	Y X\Y	⇒ X
Forward Composition (>**B**)	X/Y Y/Z	⇒ X/Z
Backward Composition (<**B**)	Y\Z X\Y	⇒ X\Z
Forward Crossed Composition (>$\mathbf{B}_\times$)	X/Y Y\Z	⇒ X\Z
Backward Crossed Composition (<$\mathbf{B}_\times$)	Y/Z X\Y	⇒ X/Z
Forward Type-raising (>**T**)	X	⇒ T/(T\X)
Backward Type-raising (<**T**)	X	⇒ T\(T/X)

Table 1: Basic CCG combinatory rules.

trees and hand-crafted rules in Phase 1.

Section 3 discusses each of the steps above in more detail. Overall, Phase 1 identifies and assigns initial categories to constituents, while Phase 2 corrects and finalizes obtained CCG parses.

We evaluate the effectiveness of our algorithm by performing coverage analysis and parsing experiments on the converted treebanks. Analysis results on a subset of 21 treebanks of different 21 languages, as well as discussion on the strengths and limitations of our algorithm, are presented in Section 5.

2. Background

2.1. Combinatory Categorial Grammar

CCG is a strongly lexicalized grammar formalism, in which words are assigned syntactic *categories* that govern how they interact with other constituents. There are two types of categories: *atomic categories*, such as S (for sentences) or NP (for noun phrases), and *complex categories*, which are in the form of X/Y or $X\backslash Y$, with X and Y being categories themselves. X/Y (or $X\backslash Y$) takes an argument Y to the right (or left), and yields a result X. In this case, X/Y is also called a *functor* category, and Y is called an *argument* category. For example, an intransitive verb in English typically receives category $S\backslash NP$, which means it takes a noun phrase argument to its left and returns a sentence as a result.

CCG also contains a set of rules that defines how categories can combine with each other. Table 1 shows a list of basic combinatory rules used in CCG. In addition, non-combinatory rules such as unary and binary type-changing rules are often included (e.g. $S\backslash NP \Rightarrow NP\backslash NP$), as they have been shown to alleviate the problem of category proliferation during treebank conversion (Hockenmaier and Steedman, 2002).

2.2. Universal Dependencies

UD is a project to create cross-linguistically consistent dependency annotation guidelines. As of v2.9, there are 217 treebanks in 122 languages. One main difference between UD and other dependency grammars is its treatment of function words. To achieve better parallelism among annotations of different languages, function words are treated as dependents of content words (Nivre et al., 2016). UD is being actively developed, with adjustments to dependency definitions and new features such as Enhanced Dependencies (Nivre et al., 2020). The current version of UD consists of 37 universal dependency relations, 17 universal part-of-speech (POS) tags, and 24 universal features.

2.3. Related Work

The English CCGbank (Hockenmaier and Steedman, 2007) is one of the pioneering works to create a treebank for CCG, by converting from the Penn Treebank (Marcus et al., 1993). From then on, there have been multiple works to create CCG treebanks for German (Hockenmaier, 2006), Italian (Bos et al., 2009), Chinese (Tse and Curran, 2010), Japanese (Uematsu et al., 2013), and Hindi (Ambati et al., 2018). For works that involve converting from a dependency treebank, a common approach is to first convert to constituency trees, binarize the constituency trees, then apply conversion rules to the binarized trees. Due to a large number of cross-serial dependencies in the Hindi dependency treebank, Ambati et al. (2018) diverge from this approach by first extracting a CCG lexicon from the dependency treebank, then using a non-statistical CCG parser to attain CCG derivations. In general, all previous works involve conversion methods that are specific to the languages and treebanks being converted, making it difficult to generalize to others. Moreover, source treebanks for German, Italian, and Japanese also contain additional information regarding phrase structures (German), or predicate-argument structures (Italian, Japanese), which

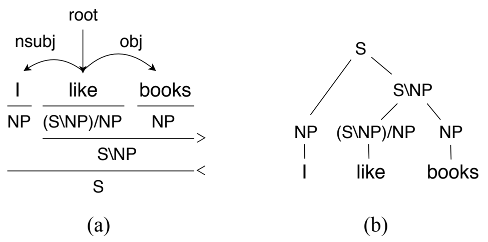


Figure 2: (a) is a standard CCG representation, with a dependency tree shown on top. (b) is an equivalent constituent structure.

help alleviate certain ambiguities, such as argument-adjunct distinction. This distinction, or lack thereof, is a big obstacle when converting UD treebanks to CCG derivations.

Recently, Yoshikawa et al. (2019) propose a neural network-based model to automatically convert dependency trees to CCG derivations for parser domain adaptation. However, their method requires an existing CCG parser for fine-tuning, which is not available for most languages in UD. Evang and Bos (2016) propose an annotation projection approach to induce CCG via parallel corpora; however, the relatively small number of parallel corpora available compared to UD limits its range of applicability. Reddy et al. (2017) introduce an interface that converts UD dependency trees to logical forms. Compared to their work, our conversion to CCG allows more flexibility in the types of semantic representations that could be derived, such as first-order logic neo-Davidsonian representations (Bos et al., 2004), or higher-order logic representations (Mineshima et al., 2015), while also retains the syntactic information encoded in UD. Additionally, we also perform larger-scale experiments and analysis on more languages. Our binarization method takes inspiration from their work.

3. The Conversion Process

A simple, typical CCG derivation is illustrated in Figure 2. To obtain a unique and complete derivation from a dependency tree, we need to:

1. Identify constituents of the dependency tree,
2. Identify the category of each constituent,
3. Identify the combinatory rules to be applied.

The constituent structure of a CCG derivation can be represented in the form of a binary tree (Figure 2(b)). Since dependency trees are structurally different, a binarization step is required. As the binarized trees also represent the constituent structures of the sentences being converted, thus answering requirement (1), an *obliqueness hierarchy* is necessary to impose a correct traversal order during binarization of the dependency trees. The details of this step are explained in Section 3.1.1.

To meet requirement (2), we design hand-crafted rules (Section 3.1.2) that assign categories to noun phrases, modifiers, function words, and punctuation marks. Unassigned constituents are later inferred using CCG's

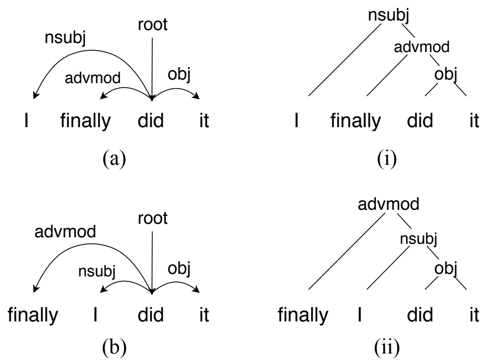


Figure 3: (a) and (b) show two sentences with a slight difference in word order. Without position information, both (a) and (b) would be binarized into (i) according to the obliqueness hierarchy (`obj` > `advmod` > `nsubj`). However, (i) leads to an invalid combination for (b), as "finally" cannot combine with "did it" due to being non-adjacent. (ii) shows the correct binarization for (b) when the condition for words' positions is applied, as it puts `nsubj` before `advmod` in the traversal order.

application rules in Phase 1 (Section 3.1), or through the results of non-statistical parsing in Phase 2 (Section 3.2). Finally, the parser also gives us a set of most probable combinatory rules applicable to our trees, which we can use to satisfy requirement (3).

Preprocessing: We ignore most dependency subtypes, such as `obl:tmod`, as these labels are not used consistently across treebanks of different languages. We also remove quotation marks from UD trees, following Hockenmaier and Steedman (2007).

3.1. Phase 1

3.1.1. Binarization

We binarize dependency trees using a modified version of the binarization method proposed by Reddy et al. (2017). The method traverses the dependency trees recursively from top to bottom, and builds binarized trees by gradually adding subtrees in the order it traverses. Since a binarized tree decides which constituents combine with each other, their method depends on an *obliqueness hierarchy* to traverse in an order that can lead to syntactically sound combinations. An obliqueness hierarchy is an ordering of grammatical relations based on their obliqueness; the hierarchy they use is similar to that defined in grammar formalisms such as Head-driven Phrase Structure Grammar (HPSG, (Pollard and Sag, 1994)). However, their method is designed to extract logical forms, and thus does not take into account the position of each constituent in a sentence. This can lead to invalid CCG combinations, as combinatory rules in CCG are only applied to string-adjacent entities. We adapt Reddy et al. (2017)'s method to our task by adding a position-based condition: (1) for dependents of the same distance to the head, traverse in the order of the obliqueness hierarchy; (2) for dependents of different distances to the head, traverse closer dependents first. Here, "distance" is measured by the number of siblings between a dependent and its head (Figure 3).

3.1.2. Category Assignment Rules

This section describes our hand-crafted rules for category assignment. Similar to previous works on CCG induction (Bisk and Hockenmaier, 2012), we assume two atomic categories S and NP for our target grammar. Categories are assigned to internal nodes of the binarized trees obtained in the previous steps. Due to the varied word-order tendencies of different languages, we set the default slash direction of complex categories to "|", which can either take value "/" or "\". This value is either decided through heuristic rules based on relative positions of functors and arguments, or through majority voting in Phase 2. The rules below do not depend on one another, and can be applied in any order.

Root: We determine the category of a whole sentence through the `root` of the dependency tree. A sentence is assigned category NP if:

- The `root` has one of the following UPOS tags: `NOUN`, `NUM`, `PRON`, `PROPN`, `SYM`,
- The `root` does not have any nominal subject, clausal subject, or expletive children.

The sentence is assigned category $S|NP$ if:

- The `root` does not have one of the following POS tags: `NOUN`, `NUM`, `PRON`, `PROPN`, `SYM`,
- The `root` does not have any nominal subject, clausal subject, or expletive children.

Otherwise, the sentence is assigned category `S`.

Punctuation: We follow Hockenmaier and Steedman (2007) and set the category of each punctuation mark to be the punctuation mark itself. Exceptions include dashes, parentheses, and variants of open and closing brackets in different languages (e.g., "【】" in Japanese, "《》" in Japanese, Chinese, and Korean). These punctuation marks are treated like normal constituents and carry standard CCG categories.

Adnominal clause: An adnominal clause (`acl`) modifies a nominal, and thus generally has category $NP|NP$. If an adnominal clause is not marked by any markers (`mark`), we apply a type-changing rule to change its original category to $NP|NP$ (Figure 4). The original category of an adnominal clause excluding markers is set to S if it has a clausal or a nominal subject, and $S|NP$ otherwise.

Relative clause: A relative clause is tagged as a subtype of an adnominal clause in UD (`acl:relcl`), but it requires a separate rule to produce a correct CCG derivation:

- The relative pronoun (identified through feature `PronType=Rel`) is assigned category $(NP|NP)|(S|NP)$, as it takes a sentence missing a subject or an object as an argument, and yields a nominal modifier.

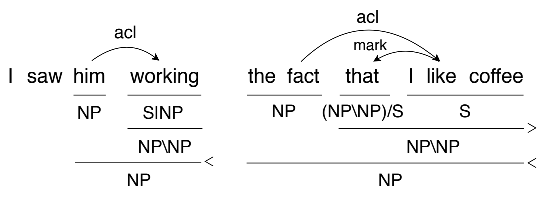


Figure 4: On the left is an example of a unary type-changing rule for `acl`. The slash direction of $NP|NP$ is by default "|", but can be inferred to be "\" based on the adjective clause's relative position to its head. On the right is an example of an adnominal clause with a marker "that", which absorbs category S of "I like coffee" and changes it to $NP|NP$.

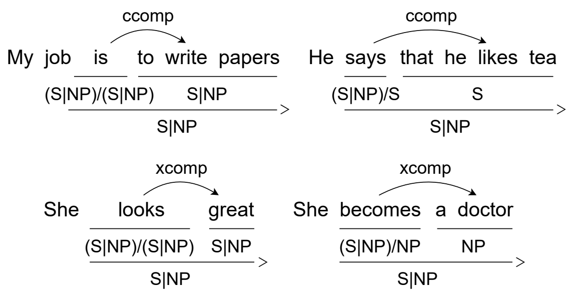


Figure 5: Examples of our rules for `ccomp/xcomp`.

- If a relative clause does not have a relative pronoun, its original category is set to $S|NP$, and is type-changed to $NP|NP$.
- In the case of an interrogative pronoun, the constituent consisting of the interrogative pronoun and its head is assigned category $(NP|NP)|(S|NP)$.

Relative clauses require additional handling in Phase 2 of the algorithm, which we describe in Section 3.2.

Adverbial clause: Similarly, an adverbial clause `advcl` usually has category $(S|NP)|(S|NP)$, as it modifies a verb or a predicate. If an adverbial clause does not have any markers (`mark`), we apply a type-changing rule to change its original category to $(S|NP)|(S|NP)$. We set the original category of an adverbial clause excluding markers to S if it has a clausal or a nominal subject, and $S|NP$ otherwise. An adverbial clause can also appear in sentential modifier locations, in which case its category would be $S|S$.

Clausal complement: We assign category S to a clausal complement (`ccomp`) if it has a subject, and category $S|NP$ if it does not. An open clausal complement (`xcomp`) is assigned category `NP` if its head element has one of the following UPOS tags: `NOUN`, `NUM`, `PRON`, `PROPN`, `SYM`. Otherwise, it is also assigned category $S|NP$ (Figure 5).

Clausal subject: We only apply rules for a clausal subject (`csubj`) if it has another subject within. In this case, if a clausal subject is marked by a marker (`mark`), it is assigned category S. Otherwise, it is assigned cate-

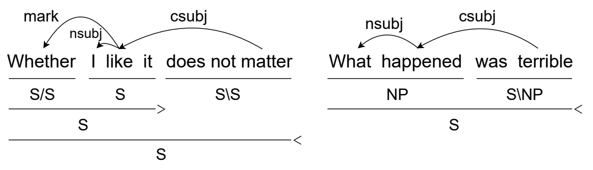


Figure 6: Examples of our rule applied to `csubj`.

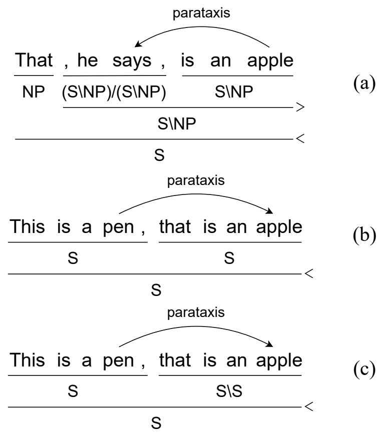

parataxis
That , he says , is an apple
NP (S\NP)/(S\NP) S\NP
S\NP
S
(a)
parataxis
This is a pen , that is an apple
S S
S
(b)
parataxis
This is a pen , that is an apple
S S\S
S
(c)

Figure 7: Analysis (a) shows an example of our rule for parataxis. Analysis (b) illustrates the correct coordination-like derivation for side-by-side sentences, while analysis (c) shows our compromise.

gory NP (Figure 6). In other cases, clausal subjects are treated like normal core arguments, and their categories are inferred through the category inference step.

Parataxis: The UD guidelines detail five different constructions where parataxis can appear: side-by-side sentences, reported speech, news article bylines, interjected clauses, and tag questions. Unfortunately, this makes it difficult to define a common rule for all the constructions. For example, side-by-side ("run-on") sentences and reported speech have different CCG analyses, but they are hard to distinguish just by their dependency structures. We decide to treat the dependent constituent as a modifier to the head constituent (Figure 7).

Noun phrase: Category NP is assigned to tokens that have one of the UPOS tags: `NOUN`, `NUM`, `PRON`, `PROPN`, `SYM`, or non-noun tokens with accompanying determiners that act as nominal subjects or objects, *if* they do not modify any other constituents. Otherwise, their categories are inferred through the category inference step.

Vocative/dislocated/discourse/overridden disfluency elements: Since these elements are optional to the grammar and meaning of a sentence, we treat them as modifiers to their head. As a result, they carry category $X|X$, where X is the category of their head.

3.1.3. Category Inference

Our rules described in Section 3.1.2 assign categories to only a subset of constituents. As a result, there are bound to be unassigned categories. In these situations, we follow CCG's forward and backward application rules to infer the missing categories from existing ones. To make category inference possible, we need to identify how constituents should be combined, and thus identify the functor/argument role of each constituent.

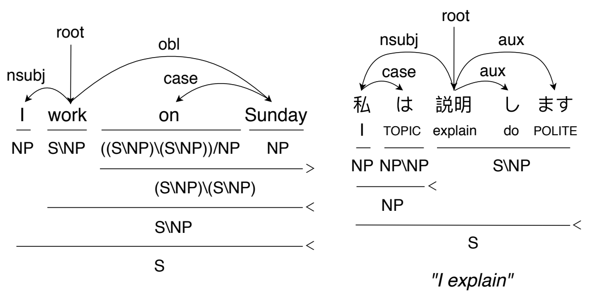


Figure 8: Examples of situations where categories for case markers may differ. On the left, case marker "on" has a category of the form $X|Y$, while on the right, case marker "は" (topic marker) has a category of the form $X|X$ to preserve the category of its head "私" ("I").

Our rules for identifying functors and arguments are designed around the relationship between heads, arguments, and modifiers. Specifically:

1. We set the head of a head-argument relation (`nsubj`, `csubj`, `obj`, `iobj`, `xcomp`, `ccomp`, `expl`) as a functor, and its dependent as an argument.
2. We set the head of a head-modifier relation (the rest of the UD relations, with the exception of `conj`, `cc`, and `punct`) as an argument, and its dependent as a functor.

In general, a functor category in case (1) has the form $X|Y$, where X and Y are usually different categories. This means that it takes one category as input and outputs a different category. Transitive verbs ($(S\backslash NP)/NP$) are one example. In case (2), a functor category usually has the form $X|X$, meaning it inputs and outputs the same category. Nominal modifiers or multi-word expressions ($NP|NP$) are typical cases. Given a CCG combination with the same result and argument category, we can easily infer the functor category. One exception to rule (2) is case markers (`case`). A case marker can have the form $X|Y$ if its head is a modifier to another constituent, and the form $X|X$ if its head is an argument to another constituent (Figure 8).

The category inference step is run top-down, and is repeated until no more categories can be inferred. There are two situations where additional logic is required:

Punctuation: As mentioned in Section 3.1.2, dashes, parentheses, and other brackets follow the same CCG combinatory rules as normal constituents. Other punctuation marks follow a separate rule (e.g. , $X \Rightarrow X$), similar to the English CCGbank.

Coordination: We use the following non-combinatory rules for coordination, also similar to the English CCGbank. The order of the categories on the left hand side does not matter.

$$\begin{aligned} conj \quad X &\Rightarrow X[conj] \\ , \quad X &\Rightarrow X[conj] \\ X[conj] \quad X &\Rightarrow X \end{aligned}$$

3.2. Phase 2

3.2.1. Word Order Majority Voting

In Phase 1, we can deduce the slash direction in each category through relative positions between functors and arguments. However, it is not guaranteed that all cases are covered, as shown in the example of "read" and "I read" in Figure 1. To handle these situations, we keep track of the frequency of each category assigned in Phase 1, and apply majority voting to determine the most common word order of a treebank. For example, in Japanese_GSDLUW, category $(S\backslash NP)\backslash NP$ for transitive verbs is assigned 6.9 times more frequently than the next most popular category $(S/NP)\backslash NP$ after Phase 1, which can be explained by the SOV order of Japanese. Word order majority voting is particularly useful when we need to determine the categories of predicates in the cases of long-distance dependency through extraction. In those cases, core arguments are often extracted from their canonical positions; using majority voting ensures the correct categories are assigned. We discuss the detailed procedure in the next sections.

3.2.2. Category Reassignment for Predicates

Categories of certain predicates can be correctly determined through CCG's application rules alone, such as the category $(S\backslash NP)/NP$ of *like* in Figure 2. Our algorithm relies on an assumption that basic clause structures like Figure 2 are more common in UD treebanks than other constructions where word order may be changed (e.g., through extraction), and thus the categories of predicates in these cases represent the most common word order of a treebank. In this step, when we detect a core argument of a predicate not appearing in its most common position, we reassign an appropriate majority-voted category to the predicate, overwriting its existing category inferred in Phase 1.

3.2.3. Candidate Parse Generation

Given all the categories assigned in Phase 1 and reassigned in Phase 2 so far, we apply a non-statistical parser to generate a set of candidate parses. We use a custom implementation of NLTK's CCG parser (Bird and Loper, 2004), in which we add the forward crossed composition rule that is missing from the official implementation. NLTK's CCG parser generates parses only from categories, without taking into account surface words, and thus can be applied universally. Using a parser allows us to efficiently explore possible parses, while making sure that the final parse is valid.

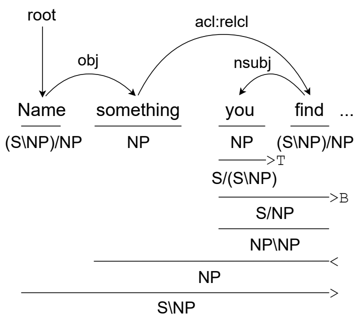


Figure 9: Analysis of a sentence with a relative clause.

3.2.4. Candidate Parse Filtering

Since the parser is non-statistical, it generates a large number of candidate parses, which need to be filtered. We exclude all parses that do not match the binarized trees and manually assigned categories described in Phase 1, and give preference to the parse that uses the smallest number of composition and type-raising rules.

4. Discussion

In this section, we discuss how our algorithm can be used to handle certain cases of long-distance dependency and its limitations. We provide additional examples in several languages in the Appendix.

4.1. Case Study: Relative Clauses

A relative clause typically misses a subject or an object as an argument. In Phase 1, we assign the category of a relative clause to be $S|NP$. The slash direction of this category is determined based on which argument is missing, and the canonical position of the missing argument. In the example in Figure 9, we infer from the dependency structure and the majority-voted word order that:

- *find* is a predicate with two arguments, one of which is missing,
- Since the predicate *find* already has a nominal subject, it is missing an object argument,
- Since in English, an object is commonly to the right of its verb, we set the category of the relative clause to S/NP.

The category of *find* is assigned $(S\backslash NP)/NP$, which is the most common category for a verbal predicate with two arguments in English UD treebanks. Given these categories, we apply the NLTK's CCG parser to the subtree that spans the relative clause *you find* to get the desired derivation.

4.2. Case Study: Wh-questions

In some wh-questions in English UD treebanks, wh-words are annotated as objects of the main predicates of the wh-question. Similarly, for the example in Figure 10, we can infer that:

- *call* is a predicate with three arguments (one subjects and two objects),

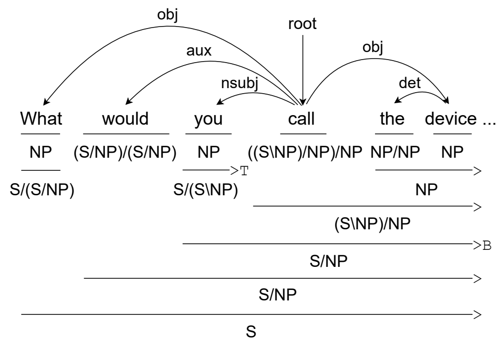


Figure 10: Analysis of a wh-question.

- Since in English, objects are commonly to the right of their verb (through majority voting), *What* is an extracted element and not in its canonical position,
- As a result, the category of the phrase after *What* should be S/NP to reflect that an argument to the right is being extracted.

The category of *call* is assigned $((S\backslash NP)/NP)/NP$, which is the most common category for a verbal predicate with three arguments in English. After applying the NLTK's CCG parser, we arrive at our final derivation.

4.3. Limitations

Our procedure described in Phase 2 is most effective when there is a sign of a missing argument (e.g, in relative clauses), or when an extracted argument is explicitly tagged in the dependency tree (e.g., in wh-questions, sentences with topicalization). In other cases such as tough movement, where there is no explicit dependency between the subject of the main verb and the object of the embedded verb, our method cannot yet be reliably applied. In addition, the use of majority voting is more suitable for languages with a dominant word order. Unlike the Penn Treebank, there are no traces in UD to identify the canonical position of an extracted or elided element. Languages with a considerable level of word order freedom such as Dutch or German will require more information than what UD currently provides.

We conjecture that predicate-argument structures are essential to handle both of these problems. In future work, we plan to investigate the viability of using additional data sources, such as the Enhanced UD (Nivre et al., 2020), the Universal Proposition Bank (Akbik et al., 2015), or integrating language-specific rules to further improve our conversion.

Additionally, dependency trees with the `orphan` relation are currently not converted. In UD, `orphan` is usually used in elliptical constructions where a predicate is elided. In English, for example, `orphan` is used to to handle gapping and stripping; however, these constructions cannot be analyzed with standard CCG combinatory rules (Steedman, 2000).

Finally, dependency trees with crossing arcs present a challenge for binarization. Certain treebanks, such as Ancient Greek and Latin treebanks, have a high number of sentences with crossing dependencies, which lead to significantly lower conversion rates. These sentences are also not being converted by our algorithm, and will be another focus of our future work.

5. Evaluation

For the following experiments, treebanks with their surface stripped off, or with more than 20% of their sentences containing dependency label `dep` or UPOS tag `X`, are excluded, as we depend on the surface for our punctuation rules, and treebanks having too many `dep` or `X` suggest an underlying problem with their annotation quality.[2] In addition, we also exclude treebanks without a proper `train/test` split, as it is necessary for our evaluation. Treebanks of languages tagged as having "No dominant order" on The World Atlas of Language Structures (WALS, (Dryer, 2013)) are also excluded.

To assess the conversion quality, we conduct lexical, sentential, and syntactic rule coverage analyses on the converted treebanks, which are commonly used metrics for evaluating induced grammars (Hockenmaier and Steedman, 2007; Tse and Curran, 2010; Uematsu et al., 2013). CCG parsing experiments are also performed on treebanks with more than 10,000 complete derivations in the training set. For languages that have more than one such treebank, we choose the largest treebank available. Figure 11 summarizes our conversion and parsing results on 21 treebanks of 21 languages. Complete conversion statistics on 101 treebanks of 59 languages tested are reported in Table 2 of the Appendix.

5.1. Conversion Rate and Coverage

Conversion rate: A conversion rate of a treebank measures the percentage of its sentences that are fully converted to CCG derivations. We observe better than 80% conversion rates for 62 treebanks (out of 101) of 35 languages (out of 59). Most conversion errors can be attributed to cross-serial dependencies, dependency relation `dep`, and UPOS tag `X`. The abundance of `dep` and `X` suggests lower annotation quality of some treebanks in UD, but it also means that conversion rates can be further increased by improving the source treebanks.

Lexical coverage: We treat the converted `train` set of each treebank as the gold standard, and the `dev` and `test` sets as unseen data. Lexical coverage measures how well the gold lexicon covers the categories in unseen data. Standard treatment of rare words is applied; tokens that appear less than five times are replaced by "UNK". Unassigned categories are not included in the analysis. We achieve over 90% lexical coverage on 79 treebanks of 48 languages (Table 2, Appendix).

Sentential coverage: Sentential coverage measures the percentage of sentences in unseen data that can be fully assigned with categories from the gold lexicon.

[2]In UD, `dep` and `X` are only used when it is impossible to assign a more precise label, or when there are problems with the conversion/parsing software.

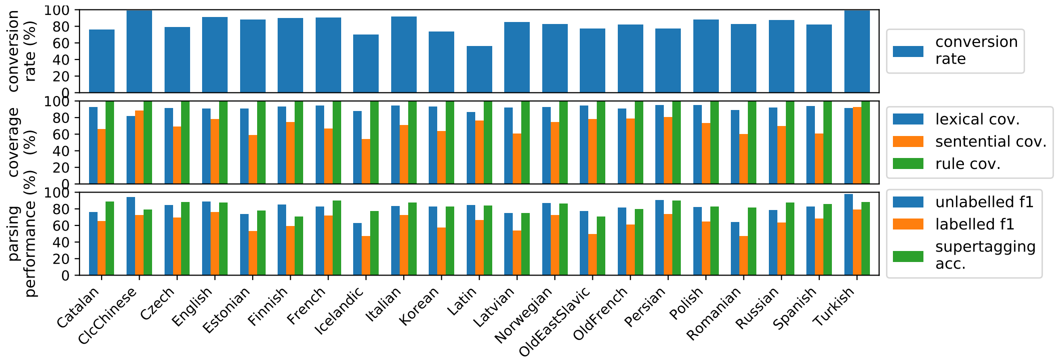


Figure 11: Conversion statistics and CCG parsing results on 21 treebanks of 21 languages, sorted by alphabetical order. Detailed numbers are reported in Table 3 of the Appendix.

We use fully converted sentences in the dev and test sets for sentential coverage analysis. The majority of our converted treebanks achieve between 55% and 70% coverage. In reality, we observe that most sentences in the dev and test sets contain only a small number of tokens not covered by the gold lexicon. This explains the high lexical coverage but average sentential coverage, and also suggests that sentential coverage can greatly benefit from minor manual correction.

Syntactic rule coverage: Syntactic rule coverage on unseen data is measured by calculating the percentage of CCG rule instantiations in dev and test sets that exist in the train set. We are able to achieve near-perfect coverage for all languages.

Parsing performance: We use an off-the-shelf CCG parser *depccg* (Yoshikawa et al., 2017) on 21 treebanks with more than 10,000 sentences in the training set. We run the training script for 20 epochs on each treebank, keeping all other default hyper-parameter settings. No pre-trained language model is used. Parsing performance is evaluated on the test split of each treebank. While a standard evaluation metric for CCG parsing is based on the recovery of dependencies in predicate-argument structures, such information is not trivial to obtain from UD. As a result, we choose a more traditional metric, PARSEVAL (Black et al., 1991). We achieve over 80% unlabelled PARSEVAL F1 and supertagging accuracy on most tested treebanks, suggesting the learnability of our induced grammars, as well as the viability of obtaining a good CCG parser for many languages from the converted treebanks.

5.2. Manual Evaluation

We randomly sampled and manually checked 100 sentences from each test set of the obtained English-EWT, Japanese-GSDLUW, and Vietnamese-VTB treebanks, languages that we have expert speakers of.

English: We found 6 cases (out of 25) of incorrect binarization of coordination structures. These are caused by UD assigning a single label conj to all types of coordination. We speculate that introducing a separate label for each type (coordination between verb phrases/noun phrases/sentences/etc.) would lead to better results. Another group of binarization errors involves yes-no questions (5 cases) and embedded questions (one case). Since in UD, auxiliary verbs, such as *did* in *did you do it?*, are dependents of the main verb, our algorithm fails to binarize *did you* as a constituent. We believe these situations are best handled as an English-specific problem, thus necessitating a specific binarization rule for English. Finally, we found one error involving a sentence with inversion, caused by the lack of a proper category assignment rule for this phenomenon.

Japanese: There are two main areas where our induced grammar diverges from previous work (Uematsu et al., 2013). First, we assign $S\backslash NP$ to sentences without overt subjects, while Uematsu et al. (2013) accept them as S. Second, we assign $(S\backslash NP)\backslash(S\backslash NP)$ to auxiliary verbs (e.g, ます, た, etc.), while they assign $S\backslash S$. We consider these to be simply variations in our grammars. Lastly, we noticed some annotation artifacts related to markers (mark; e.g., て,から, etc.)[3] that led to several cases with non-standard derivations.

Vietnamese: We found 4 cases (out of 16) of incorrect binarization of coordination structures (similar to English), and 10 cases of incorrect categories assigned due to annotation errors (e.g., copulas tagged as conjunctions, head noun phrases in zero copula constructions tagged as compound nouns, etc.). These cases reaffirm that the quality of our obtained derivations is strongly tied to the quality of the source treebanks.

6. Conclusion

We introduced a rule-based algorithm to create CCG treebanks from UD. We believe the CCG derivations obtained from our algorithm can serve as a starting point for CCG treebank development and parsing research in many languages, from which further improvements can be made by applying additional language-specific rules or manual fine-tuning to the converted treebanks.

7. Acknowledgements

This work was supported by JST, CREST Grant Number JPMJCR2114, Japan.

[3]Markers are typically used to mark subordinate clauses, but in Japanese they are used in many different constructions.

8. Bibliographical References

Akbik, A., Chiticariu, L., Danilevsky, M., Li, Y., Vaithyanathan, S., and Zhu, H. (2015). Generating high quality proposition Banks for multilingual semantic role labeling. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 397–407, Beijing, China, July. Association for Computational Linguistics.

Ambati, B. R., Deoskar, T., and Steedman, M. (2018). Hindi CCGbank: CCG Treebank from the Hindi Dependency Treebank. *Language Resources and Evaluation*, 52:67–100.

Bird, S. and Loper, E. (2004). NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July. Association for Computational Linguistics.

Bisk, Y. and Hockenmaier, J. (2012). Simple robust grammar induction with combinatory categorial grammars. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*.

Bos, J., Clark, S., Steedman, M., Curran, J. R., and Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 1240–1246, Geneva, Switzerland, aug 23–aug 27. COLING.

Bos, J., Bosco, C., and Mazzei, A. (2009). Converting a Dependency Treebank to a Categorial Grammar Treebank for Italian. In *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*, pages 27–38, Milan, Italy.

Clark, S., Steedman, M., and Curran, J. R. (2004). Object-extraction and question-parsing using CCG. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 111–118, Barcelona, Spain, July. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Dryer, M. S. (2013). Order of demonstrative and noun. In Matthew S. Dryer et al., editors, *The World Atlas of Language Structures Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Evang, K. and Bos, J. (2016). Cross-lingual learning of an open-domain semantic parser. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 579–588, Osaka, Japan, December. The COLING 2016 Organizing Committee.

Hockenmaier, J. and Steedman, M. (2002). Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain, May. European Language Resources Association (ELRA).

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

Hockenmaier, J. (2006). Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 505–512, Sydney, Australia, July. Association for Computational Linguistics.

Krishnamurthy, J. and Mitchell, T. (2012). Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765, Jeju Island, Korea, July. Association for Computational Linguistics.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Martínez-Gómez, P., Mineshima, K., Miyao, Y., and Bekki, D. (2017). On-demand injection of lexical knowledge for recognising textual entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 710–720, Valencia, Spain, April. Association for Computational Linguistics.

Mineshima, K., Martínez-Gómez, P., Miyao, Y., and Bekki, D. (2015). Higher-order logical inference with compositional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2055–2061, Lisbon, Portugal, September. Association for Computational Linguistics.

Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., and Zeman, D. (2016). Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and*

Evaluation (LREC'16), pages 1659–1666, Portorož, Slovenia, May. European Language Resources Association (ELRA).

Nivre, J., de Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F., and Zeman, D. (2020). Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France, May. European Language Resources Association.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June. Association for Computational Linguistics.

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Reddy, S., Täckström, O., Petrov, S., Steedman, M., and Lapata, M. (2017). Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 89–101, Copenhagen, Denmark, September. Association for Computational Linguistics.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA.

Tse, D. and Curran, J. R. (2010). Chinese CCGbank: extracting CCG derivations from the Penn Chinese treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1083–1091, Beijing, China, August. Coling 2010 Organizing Committee.

Uematsu, S., Matsuzaki, T., Hanaoka, H., Miyao, Y., and Mima, H. (2013). Integrating multiple dependency corpora for inducing wide-coverage Japanese CCG resources. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1042–1051, Sofia, Bulgaria, August. Association for Computational Linguistics.

Yoshikawa, M., Noji, H., and Matsumoto, Y. (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.

Yoshikawa, M., Noji, H., Mineshima, K., and Bekki, D. (2019). Automatic generation of high quality CCGbanks for parser domain adaptation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 129–139, Florence, Italy, July. Association for Computational Linguistics.

A. Appendix

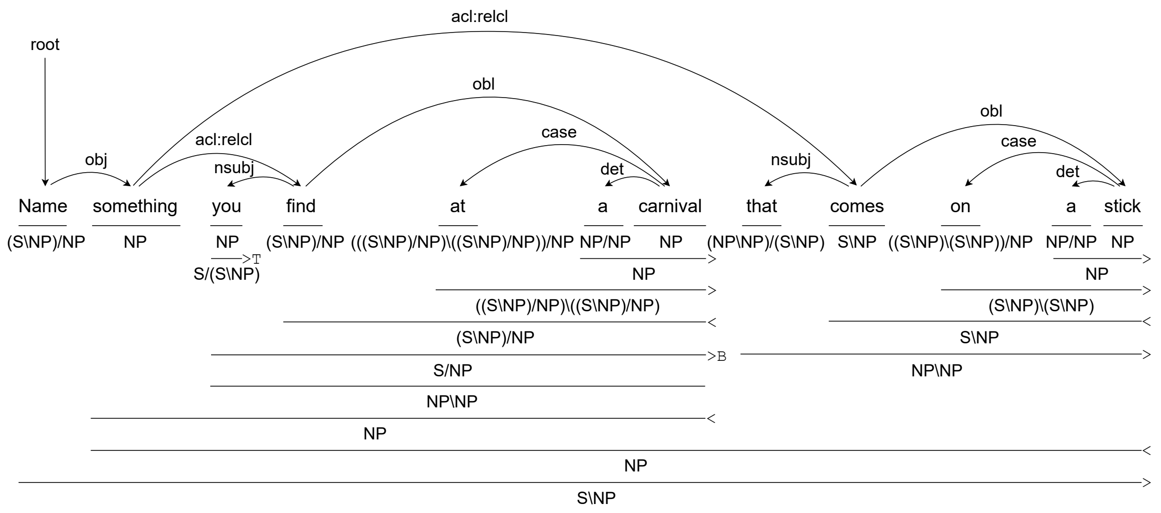
Treebank	Conversion Rate		Statistics							Coverage (dev)			Coverage (test)		
	#Sent.	%Converted	%Cross.	#Tok.	#Cat.	#Cat.>1	#Cat./Tok.	#Rules	#U. Rules	%Lex.	%Sent.	%Rule	%Lex.	%Sent.	%Rule
Ancient_Greek	30999	46.72	48.69	27051	651	393	1.8	125738	1554	-	-	-	-	-	-
-PROIEL	17080	55.55	37.52	16531	506	314	1.71	80496	1093	89.12	57.88	92.66	90.49	68.39	99.35
-Perseus	13919	35.89	62.38	13733	419	256	1.48	45242	1002	90.72	40.68	90.35	92.40	54.79	93.08
Basque-BDT	8993	63.84	31.52	17899	423	261	1.74	65982	940	93.60	65.91	99.02	93.50	67.07	98.83
Bulgarian-BTB	11138	91.97	3.06	27463	393	253	1.6	130764	795	95.13	73.02	99.55	95.54	76.83	99.69
Buryat-BDT	927	88.13	8.20	3850	140	83	1.35	8218	305	-	-	-	82.19	39.38	71.61
Catalan-AnCora	16678	75.90	5.72	29539	805	483	2.08	368016	1698	92.12	60.80	99.76	92.41	65.95	99.65
Chinese	9994	78.96	1.13	25643	534	525	2.5	197074	1037	-	-	-	-	-	-
-GSD	4997	78.25	2.24	16754	527	323	1.9	97453	1028	90.67	44.10	99.43	90.26	46.77	99.53
-GSDSimp	4997	79.67	0.02	17001	529	329	1.9	99621	1028	90.64	44.64	99.44	90.31	46.58	99.54
Classical_Chinese-Kyoto	58301	99.07	0.00	8418	298	213	3.99	238296	598	81.44	87.82	99.87	81.59	88.08	99.87
Coptic-Scriptorium	2010	71.39	13.63	2243	265	174	2.79	32841	419	85.71	52.94	99.05	84.54	52.96	99.23
Croatian-SET	9010	79.84	8.45	32536	678	431	1.71	146864	1367	93.98	56.37	99.52	93.61	58.86	99.48
Czech	125382	78.50	11.72	156917	2171	1325	2.49	1472203	5432	-	-	-	-	-	-
-CAC	24709	73.42	12.71	55079	1080	651	1.63	311171	2551	92.78	57.11	96.05	92.13	56.91	95.72
-FicTree	12760	82.30	11.40	25136	654	383	1.68	116131	1503	92.92	73.23	99.17	92.87	72.19	99.16
-PDT	87913	79.37	11.49	123050	1870	1131	2.1	1044901	4512	91.15	68.07	99.75	91.42	68.74	99.77
Danish-DDT	5512	67.54	21.35	13097	773	412	1.71	52719	1442	91.68	57.44	98.49	91.01	54.39	98.49
English	36783	90.10	4.00	33759	793	507	3.27	476868	1669	-	-	-	-	-	-
-Atis	5432	94.92	1.84	904	154	113	3.92	53933	299	76.33	38.40	91.25	82.73	44.96	92.25
-EWT	16621	90.99	3.35	21402	546	363	2.12	205488	1049	90.59	77.31	99.75	90.49	77.97	99.84
-GUM	7397	87.48	4.77	14323	401	246	1.96	104351	832	90.98	67.93	99.58	91.82	67.38	99.60
-LinES	5243	86.80	8.07	9679	415	256	2.19	73288	868	90.66	63.94	99.24	90.89	65.38	99.58
-ParTUT	2090	88.13	1.82	6910	286	191	1.75	39808	503	93.02	57.75	99.41	92.46	55.17	99.56
Estonian	36835	87.52	3.39	86223	1151	705	1.89	424593	2845	-	-	-	-	-	-
-EDT	30972	87.93	3.22	78915	1073	661	1.74	368749	2667	91.99	61.41	99.65	90.92	58.53	99.56
-EWT	5863	85.33	4.30	15172	475	288	1.7	55844	1075	91.47	62.64	98.86	91.01	55.88	98.19
Faroese-FarPaHC	1621	66.38	0.19	2765	398	229	2.48	21675	826	86.51	29.10	96.88	88.00	35.64	97.11
Finnish	33859	88.60	7.00	76833	857	550	1.83	291318	2198	-	-	-	-	-	-
-FTB	18723	90.07	7.70	42162	433	298	1.53	123936	1096	93.69	76.99	99.43	93.39	74.22	99.58
-TDT	15136	86.79	6.14	50580	745	454	1.51	167382	1766	93.44	65.86	99.53	92.94	64.14	99.49
French	25425	88.48	4.79	46404	880	532	2.2	470378	1833	-	-	-	-	-	-
-GSD	16341	90.42	4.35	41700	555	358	1.59	344372	1105	95.03	73.10	99.78	94.26	66.48	99.62
-ParTUT	1020	81.76	5.10	3632	234	139	1.65	19914	388	92.62	59.60	98.92	94.02	62.63	98.77
-ParisStories	1755	79.72	7.69	2302	379	180	2.2	19427	756	-	-	-	85.66	54.52	97.02
-Rhapsodie	3210	82.99	7.94	3706	355	178	2.14	28315	734	90.85	72.23	98.30	90.01	71.30	98.03
-Sequoia	3099	91.06	2.13	9124	339	213	1.85	58350	610	93.87	68.90	99.43	94.52	67.15	99.46
Galician-TreeGal	1000	72.80	11.20	4067	226	131	1.52	14831	428	-	-	-	95.29	59.86	98.57
Gothic-PROIEL	5401	72.43	17.57	6145	324	192	2.01	29655	643	90.56	67.76	97.84	92.31	75.07	98.45
Icelandic	50957	68.22	0.37	50489	1735	1062	2.46	585974	3975	-	-	-	-	-	-
-IcePaHC	44029	69.86	0.37	47398	1692	1000	2.28	518640	3852	85.49	53.27	99.43	87.90	53.71	99.39
-Modern	6928	57.82	0.38	6205	437	436	2.6	67334	920	93.33	82.88	99.37	92.98	77.10	99.39
Indonesian	6628	90.65	2.28	21189	572	343	2.09	132122	1161	-	-	-	-	-	-
-CSUI	1030	93.69	3.11	4650	245	155	2.03	27339	453	-	-	-	89.78	36.86	99.05
-GSD	5598	90.09	2.13	19098	512	297	1.83	104783	1048	92.30	58.13	99.50	92.18	59.76	99.56
Irish-IDT	4910	50.12	15.05	9267	402	235	1.69	42775	777	90.40	55.27	98.72	90.80	58.82	97.88
Italian	27768	88.51	2.16	40646	1037	636	2.72	530315	2018	-	-	-	-	-	-
-ISDT	14167	91.72	1.36	27459	640	397	1.71	248554	1229	93.90	69.03	99.65	94.43	70.55	99.66
-ParTUT	2090	89.19	2.01	8126	316	192	1.57	45740	553	93.72	60.00	99.25	95.17	66.67	99.56
-TWITTIRO	1424	72.75	1.05	5412	252	144	1.56	22104	539	90.74	37.62	92.01	86.11	15.24	86.65
-VIT	10087	86.08	3.48	22364	771	455	1.89	213917	1418	89.80	46.91	99.77	90.91	59.78	99.55
Japanese	16200	86.63	0.31	35732	268	218	1.95	282609	489	-	-	-	-	-	-
-GSD	8100	84.37	0.32	19053	245	174	1.78	154068	439	94.97	71.46	99.89	94.30	68.90	99.93
-GSDLUW	8100	88.89	0.31	27653	207	136	1.29	128541	379	97.48	85.49	99.87	97.23	83.26	99.92
Kazakh-KTB	1078	85.44	7.33	3871	138	80	1.26	8301	312	-	-	-	93.06	69.20	88.33
Korean-Kaist	27363	73.72	21.70	73414	841	518	1.5	258013	1876	94.04	68.66	99.68	93.18	63.53	99.68
Kurmanji-MG	754	64.32	8.49	2015	146	81	1.36	5751	268	-	-	-	86.68	19.62	74.32
Latin	58405	58.71	33.45	35695	1016	613	2.41	403753	2429	-	-	-	-	-	-
-ITTB	26977	56.39	36.27	13554	704	424	2.31	196987	1533	87.93	75.77	99.70	86.62	76.29	99.58
-LLCT	9023	68.34	28.86	5610	346	241	1.98	102918	793	92.46	85.77	99.54	86.77	76.01	97.20
-PROIEL	18411	60.55	28.39	16913	433	280	1.74	77968	981	91.81	74.18	99.30	92.52	77.17	99.43
-Perseus	2273	49.05	48.13	4613	185	115	1.32	9865	448	-	-	-	96.10	73.75	97.14
-UDante	1721	37.94	48.17	4924	344	196	1.63	16015	759	92.01	30.46	93.20	90.45	28.33	96.79
Latvian-LVTB	15984	84.82	6.76	45783	717	435	1.73	206297	1759	92.20	61.11	99.51	91.78	60.63	99.53

Treebank	Conversion Rate		Statistics							Coverage (dev)			Coverage (test)		
	#Sent.	%Converted	%Cross.	#Tok.	#Cat.	#Cat.>1	#Cat./Tok.	#Rules	#U. Rules	%Lex.	%Sent.	%Rule	%Lex.	%Sent.	%Rule
Lithuanian-HSE	263	76.43	14.07	1790	163	83	1.38	3755	382	91.92	41.46	90.30	92.89	43.18	92.30
Livvi-KKPP	125	82.40	12.80	657	70	45	1.18	1152	175	–	–	–	78.79	24.42	63.34
Maltese-MUDT	2074	85.68	3.86	7409	243	166	1.78	34154	502	93.40	50.14	99.28	93.31	51.02	99.17
Marathi-UFAL	466	89.27	4.08	873	71	46	1.52	3070	150	93.49	67.50	99.14	85.43	55.81	93.51
North_Sami-Giella	3122	91.86	4.39	7375	212	136	1.45	22808	487	–	–	–	94.11	63.43	97.85
Norwegian	37619	81.57	7.54	51928	1055	623	2.02	437745	1978	–	–	–	–	–	–
-Bokmaal	20044	82.73	7.39	30564	777	447	1.78	225173	1462	92.31	74.06	99.62	92.27	74.12	99.71
-Nynorsk	17575	80.26	7.71	28554	792	455	1.77	212572	1453	92.12	72.25	99.62	92.85	74.22	99.69
Old_Church_Slavonic-PROIEL	6338	75.23	16.31	7229	351	221	1.96	33453	725	92.08	72.83	98.60	88.35	54.17	82.05
Old_East_Slavic	18003	76.43	16.27	27463	574	353	1.59	102856	1307	–	–	–	–	–	–
-RNC	1059	59.87	33.43	4720	236	136	1.36	13358	542	–	–	–	95.02	41.50	94.35
-TOROT	16944	77.47	15.20	23716	453	290	1.57	89498	960	93.65	81.00	98.69	94.17	78.13	99.31
Old_French-SRCMF	18029	82.00	15.19	15432	464	302	2.08	134058	1016	89.43	76.95	99.76	90.42	78.41	99.72
Persian	35104	74.62	12.72	31261	811	481	2.64	434328	1672	–	–	–	–	–	–
-PerDT	29107	77.28	14.22	27923	565	361	2.08	356480	1165	94.41	79.19	99.86	94.64	80.68	99.86
-Seraji	5997	61.73	5.45	10842	528	286	2.25	77848	1012	90.30	50.12	99.53	89.10	46.11	99.54
Polish	39398	92.35	3.79	71292	853	538	2	398067	1955	–	–	–	–	–	–
-LFG	17246	98.12	0.64	32490	371	249	1.47	114768	655	95.87	85.39	99.69	95.52	85.17	99.65
-PDB	22152	87.85	6.25	58264	799	497	1.63	283299	1862	94.87	73.50	99.69	94.84	73.46	99.63
Portuguese	21376	81.39	6.79	41524	831	505	2.29	405680	1786	–	–	–	–	–	–
-Bosque	9357	80.05	8.87	23796	551	331	1.73	163300	1092	94.50	68.11	99.75	93.85	68.66	99.68
-GSD	12019	82.44	5.18	29284	624	385	1.84	242380	1378	93.59	64.90	99.79	94.45	67.66	99.71
Romanian	40430	82.15	7.30	66174	1359	848	2.25	708959	2966	–	–	–	–	–	–
-Nonstandard	26225	82.85	5.43	31603	1133	687	2.14	436327	2434	89.88	70.17	99.82	88.72	60.15	98.97
-RRT	9524	82.58	8.86	30574	656	440	1.72	171229	1396	93.59	56.66	99.60	93.29	52.73	98.83
-SiMoNERo	4681	77.38	14.61	15708	414	271	1.85	101403	786	92.31	54.18	99.58	93.06	56.61	99.66
Russian	110237	88.02	6.67	165650	1506	951	2.24	1470807	3644	–	–	–	–	–	–
-GSD	5030	90.02	6.12	27722	397	258	1.44	82498	976	95.59	65.38	99.28	95.65	63.81	99.50
-SynTagRus	87336	87.75	6.82	141089	1383	888	1.97	1239947	3354	92.29	70.87	99.85	92.04	69.60	99.86
-Taiga	17871	88.77	6.13	35730	553	329	1.49	148362	1289	94.04	72.41	99.67	93.47	72.51	99.56
Sanskrit-Vedic	3997	74.43	23.42	5348	220	139	1.57	15812	478	–	–	–	93.44	77.92	97.66
Scottish_Gaelic-ARCOSG	4402	64.88	6.97	5240	351	226	2.15	36533	639	87.79	65.62	99.04	85.06	58.56	98.78
Serbian-SET	4384	85.99	3.24	17723	396	257	1.67	79051	823	95.00	62.96	99.51	95.05	64.30	99.41
Slovak-SNK	10604	91.72	3.27	26390	440	289	1.45	88163	996	96.31	73.55	99.08	95.89	74.31	99.18
Slovenian	11188	84.07	9.86	31682	504	336	1.68	126362	1007	–	–	–	–	–	–
-SSJ	8000	82.30	12.00	29330	440	293	1.5	105572	858	93.65	53.79	96.60	93.65	58.07	96.64
-SST	3188	88.52	4.49	4951	265	174	1.84	20790	487	–	–	–	90.37	69.77	98.50
Spanish	33675	80.34	5.69	61453	1202	761	2.35	713429	2487	–	–	–	–	–	–
-AnCora	17662	79.01	5.54	36039	907	592	2.07	394852	1867	91.85	56.16	99.79	91.75	58.63	99.76
-GSD	16013	81.82	5.85	42224	779	471	1.66	318577	1535	93.62	63.92	99.75	93.90	60.29	99.78
Swedish	11269	89.08	4.21	24301	774	455	2.15	149457	1434	–	–	–	–	–	–
-LinES	5243	87.43	5.61	13219	577	336	1.91	70473	1078	91.36	57.19	98.79	91.72	61.35	99.23
-Talbanken	6026	90.51	2.99	15017	525	321	1.8	78984	915	88.99	49.32	98.96	91.44	59.27	98.97
Tamil-TTB	600	97.67	1.67	3508	230	140	1.61	9289	456	89.86	48.10	95.42	90.50	37.93	95.99
Telugu-MTG	1328	99.77	0.15	2046	78	53	1.42	5676	157	97.49	93.13	98.63	96.09	89.66	99.05
Turkish	78334	94.46	2.27	104863	1086	659	2.37	613545	2682	–	–	–	–	–	–
-Atis	5432	98.73	1.09	2140	161	107	3.46	43998	290	80.75	71.78	99.66	80.68	76.38	99.60
-BOUN	9761	87.88	3.34	33251	567	345	1.57	101677	1447	92.31	63.15	99.16	92.51	63.19	99.24
-FrameNet	2698	97.48	0.26	8211	137	81	1.31	17948	244	96.18	84.08	99.55	95.69	79.70	98.86
-IMST	5635	86.57	6.35	15992	437	264	1.66	44429	1088	92.94	65.31	98.45	92.49	67.98	98.17
-Kenet	18687	94.32	2.25	46922	468	305	1.63	166437	1191	92.55	64.20	99.64	92.29	62.93	99.73
-Penn	16396	93.65	3.08	35687	633	389	1.79	163056	1310	91.07	62.99	99.69	87.68	57.83	99.52
-Tourism	19725	99.20	0.51	4843	151	108	2.31	76000	329	92.89	94.20	99.83	91.50	92.71	99.92
Turkish_German-SAGT	2184	75.87	13.42	5605	343	202	2.08	25365	704	91.91	38.60	95.86	92.05	37.17	96.94
Ukrainian-IU	7060	87.08	7.69	28980	507	312	1.5	96269	1206	94.85	62.41	99.36	94.81	65.67	99.34
Upper_Sorbian-UFAL	646	81.89	11.30	3747	161	112	1.27	8284	347	–	–	–	88.15	20.59	83.13
Uyghur-UDT	3456	88.60	4.98	10731	238	146	1.62	36262	679	95.01	68.49	98.45	94.70	65.38	98.53
Welsh-CCG	2111	70.82	1.80	5210	208	138	1.95	24355	373	93.84	64.79	98.94	95.36	58.71	98.87
Western_Armenian-ArmTDP	5026	78.19	7.72	17476	555	311	1.58	64885	1247	92.83	56.53	98.55	92.27	63.89	99.09
Wolof-WTB	2107	82.82	2.99	5182	361	207	2.16	33791	688	88.46	44.79	98.59	88.48	44.83	98.28

Table 2: Conversion results on 101 treebanks of 59 languages in UD v2.9. Column names from left to right: (1) Treebank, (2) Number of sentences, (3) Conversion rate, (4) Percentage of sentences with crossing dependencies, (5) Number of distinct tokens, (6) Number of distinct categories, (7) Number of distinct categories that appear more than once, (8) Average number of categories per token, (9) Number of CCG rule instantiations, (10) Number of unique CCG rules, (11) Lexical coverage on `dev`, (12) Sentential coverage on `dev`, (13) Syntactic rule coverage on `dev`, (14) Lexical coverage on `test`, (15) Sentential coverage on `test`, (16) Syntactic rule coverage on `test`.

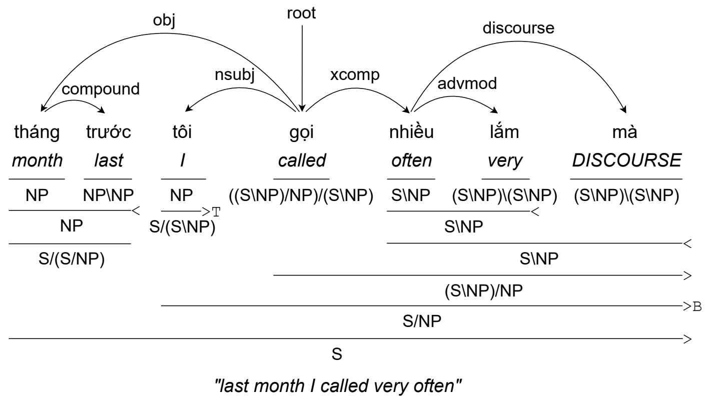
Treebank	#Train	#Test	PARSEVAL Unlabelled			PARSEVAL Labelled			Supertagging
	samples	samples	%Precision	%Recall	%F1	%Precision	%Recall	%F1	accuracy (%)
Catalan-AnCora	10002	1389	92.77	64.53	76.11	79.85	55.55	65.52	88.9
Classical_Chinese-Kyoto	47471	4814	94.52	94.00	94.26	72.81	72.41	72.61	79.4
Czech-PDT	54435	8019	93.38	77.80	84.88	76.83	64.00	69.83	88.1
English-EWT	11303	1956	92.75	85.34	88.89	79.47	73.12	76.16	87.7
Estonian-EDT	21698	2790	89.87	62.34	73.62	64.77	44.93	53.06	78.0
Finnish-FTB	13491	1699	88.60	82.15	85.25	61.66	57.17	59.33	70.7
French-GSD	13073	361	93.11	74.61	82.84	81.10	64.98	72.15	89.8
Icelandic-IcePaHC	24414	3392	91.47	48.12	63.06	68.99	36.30	47.57	77.4
Italian-ISDT	12049	438	91.74	76.62	83.50	79.64	66.51	72.49	87.9
Korean-Kaist	16875	1763	90.96	76.32	83.00	63.31	53.12	57.77	82.9
Latin-ITTB	12712	1261	93.57	77.10	84.54	73.44	60.51	66.35	83.9
Latvian-LVTB	10071	1867	89.26	65.04	75.25	63.88	46.55	53.85	74.8
Norwegian-Bokmaal	12974	1592	93.45	81.78	87.23	78.09	68.34	72.89	86.7
Old_East_Slavic-TOROT	10352	1358	89.30	68.50	77.53	56.98	43.71	49.47	70.9
Old_French-SRCMF	11563	1681	93.62	72.61	81.79	69.77	54.11	60.95	80.0
Persian-PerDT	20240	1144	94.69	86.88	90.62	77.37	70.99	74.04	90.3
Polish-PDB	15562	1952	92.33	74.40	82.40	72.51	58.42	64.71	82.6
Romanian-Nonstandard	19870	916	90.68	49.47	64.02	66.60	36.33	47.02	81.9
Russian-SynTagRus	61116	7691	92.41	68.29	78.54	75.06	55.47	63.79	87.7
Spanish-GSD	11623	340	90.63	76.01	82.68	75.15	63.03	68.56	85.9
Turkish-Tourism	15231	2180	98.05	97.73	97.89	79.15	78.89	79.02	88.4

Table 3: CCG parsing performance measured on the converted test sets of 21 treebanks of 21 languages that satisfy the criteria described in the Evaluation section.


root
obj
acl:relcl
nsubj
acl:relcl
obl
case
det
nsubj
obl
case
det
Name something you find at a carnival that comes on a stick
(S\NP)/NP NP NP (S\NP)/NP (((S\NP)/NP)\((S\NP)/NP))/NP NP/NP NP (NP\NP)/(S\NP) S\NP ((S\NP)\(S\NP))/NP NP/NP NP
S/(S\NP)
NP
((S\NP)/NP)\((S\NP)/NP)
(S\NP)/NP
S/NP
NP\NP
NP
NP
S\NP
(S\NP)\(S\NP)
S\NP
NP\NP

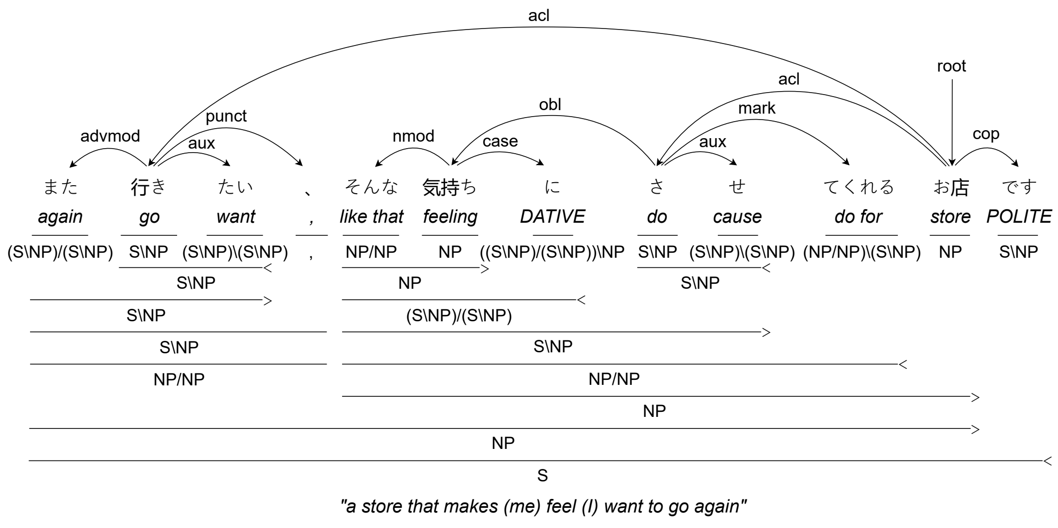
(English_EWT: `sent_id = answers-20111107163942AA08rP5_ans-0001`)

Figure 12: An example of our obtained CCG derivation for a sentence with relative clauses in English.


obj
root
discourse
compound
nsubj
xcomp
advmod
tháng trước tôi gọi nhiều lắm mà
month last I called often very DISCOURSE
NP NP\NP NP ((S\NP)/NP)/(S\NP) S\NP (S\NP)\(S\NP) (S\NP)\(S\NP)
NP
S/(S\NP)
S\NP
S/(S/NP)
S\NP
(S\NP)/NP
S/NP
S
"last month I called very often"

(Vietnamese_VTB: `sent_id = train-s1091`)

Figure 13: An example of our obtained CCG derivation for a sentence with topicalization in Vietnamese.


acl
root
acl
obl
mark
advmod
punct
aux
nmod
case
aux
cop
また 行き たい 、 そんな 気持ち に さ せ てくれる お店 です
again go want , like that feeling DATIVE do cause do for store POLITE
(S\NP)/(S\NP) S\NP (S\NP)\(S\NP) , NP/NP NP ((S\NP)/(S\NP))\NP S\NP (S\NP)\(S\NP) (NP/NP)\(S\NP) NP S\NP
S\NP
NP
S\NP
S\NP
(S\NP)/(S\NP)
S\NP
S\NP
NP/NP
NP/NP
NP
NP
S
"a store that makes (me) feel (I) want to go again"

(Japanese_GSDLUW: `sent_id = train-s2`)

Figure 14: An example of our obtained CCG derivation for a sentence with relative clauses in Japanese.