Introduction to Natural Language Syntax and Parsing
Lecture 7: A CCG Grammar and Treebank for naturally occurring text

Stephen Clark

October 22, 2015

CCG Analyses for Real Text? The examples found in linguistic textbooks and papers can often appear artificial and unlike the sentences encountered in the real world. It is a reasonable question to ask whether the "neat" formal grammar we've seen so far can be applied to the "messy" sentences found on the web or social media, or to sentences which are less messy but contain technical jargon, e.g. from biomedical research papers.

We'll look at examples from three different domains or genres: newspapers, biomedical research papers, and Wikipedia. It's true that these still consist of reasonably well-edited text, so we'll leave open the question of whether a CCG grammar could be developed for e.g. Twitter.

Newspaper Example The sentence on the slide is the famous first sentence from Section 00 of the Penn Treebank. It is immediately clear that, given the lexical categories assigned to the words, the CCG rules we've seen so far will not be able to assemble a spanning analysis.

The first problem is that *Pierre Vinken* is an N, but the verb phrase requires a subject NP. The distinction between N and NP is not clear-cut in CCG, and the two are often conflated, so we'll effectively do the same by introducing a new *unary type-changing rule* which turns an N into an NP. In keeping with CCG convention, the rule is written bottom-up on the slide.

The phrase *61 years old* has the type $S[adj]\backslash NP$, the type of a predicative adjective (since I can say e.g *the man is 61 years old*). However, in this example the phrase is acting as a post-nominal modifier of *Pierre Vinken*. Hence we'll introduce another unary type-changing rule which turns $S[adj]\backslash NP$ into $NP\backslash NP$.

Punctutation is ubiquitous in natural language, and often carries important syntactic information, but is rarely discussed in the NLP literature.[1] Here we'll adopt a simple approach to analysing punctuation, by introducing rules which

[1] One exception is Prof. Briscoe's work on punctuation in the late 1990s.

effectively merge the punctuation mark into a neighbouring constituent. For example, there is the following binary rule instance in the CCG parser: $S . \Rightarrow S$, meaning that an S followed by a period can be replaced with an S.[2] Introducing a similar rule for commas and NPs will suffice for this example sentence.

With all these additional rules in place, the sentence can now be analysed. Note that, out of all the combinatory rule schema, only forward and backward application are necessary for this example. It is possible to use the type-raising and composition rules and still arrive at the correct semantic interpretation — because of CCG's "spurious" ambiguity — but they are not required.

Grammatical Features in CCGbank You may have noticed that many of the S categories in the examples carry grammatical features, such as *dcl* (declarative). The grammar in CCGbank does not make much use of feature structures (in a linguistic, rather than machine learning, sense), unlike, say, a full implementation of an HPSG grammar. However, there is a feature set which distinguishes between different types of sentence and verb phrase, and the CCG parser does contain a unification mechanism to deal with these features. For example, when a verb phrase ($S[dcl]\backslash NP$) is modified by an adverb ($(S\backslash NP)\backslash(S\backslash NP)$), the resulting verb phrase inherits the *dcl* feature. The S categories in the adverbial category effectively carry a variable grammatical feature $[X]$ which gets instantiated when the full categories combine.

The slide lists some of the grammatical features. Julia Hockenmaier's thesis [2] (p.47) contains a full list.

Biomedical Example The main difficulty with analysing biomedical text is the profusion of long and complicated noun phrases. Even linguistic experts have difficulty analysing such noun phrases, which often requires biomedical, as well as linguistic, expertise. For example, is *T cell activation* a kind of *T activation*, or *cell activation*? It's probably *cell activation*, and that's how it has been analysed on the slide. On my RESOURCES webpage there are a number of files that have been manually annotated with CCG lexical categories (by me and Laura Rimell), including 1,000 sentences from the Genia corpus. Note that two versions are provided: one where Laura and I made a best guess at the bracketing for noun phrase cases we weren't sure about, and one where we didn't even try and left the structure flat.

Continuing with the example sentence, the phrase *resulting in enhanced production ...* has the type $S[ng]\backslash NP$; however, in this sentence it is acting as an adverbial modifier of the preceding verb phrase. (Intuitively it's the providing of the signal which results in the enhanced production.) Hence we need another unary type-changing rule, similar to the one used for the newspaper sentence, which turns $S[ng]\backslash NP$ into $(S\backslash NP)\backslash(S\backslash NP)$.

Another common feature of biomedical text is the use of brackets, especially to delimit abbreviations. Similar to punctuation, the CCG parser has some rules

[2] These rules are referred to as rule instances, rather than schema, since they do not contain any variables.

which merge a bracket with a neighbouring constituent; for example, the right bracket after the noun *IL-2* will merge with the noun, to give another noun. But other brackets receive lexical categories. For example, the left bracket before the noun *IL-2* will receive the category $(N\backslash N)/N$ (not shown on the slide), allowing the phrase *interleukin-2 (IL-2)* to become a noun.

Once all these additional rules are in place, and the noun phrases have been identified and analysed, the remaining structure is straightforward, again requiring only forward and backward application.

Wikipedia Example Aside from the punctuation, the notable aspects of the example sentence are the possessive *s*, which receives the category $(NP/N)\backslash NP$, and the compound noun *Alfriston Clergy House* – is it the Alfriston Clergy or the Alfriston House? Otherwise the structure is straightforward, once the lexical categories have been assigned, not even requiring a unary rule in this example (except N changing to NP).

Unary Type-Changing Rules The unary type-changing rules are in some sense against the spirit of CCG, with an emphasis on its lexicalised nature, since these rules are not part of the lexicon and are language-specific. An alternative solution would be to effectively push these rules onto the lexical categories, retaining the fully lexicalised nature of the formalism. The first example on the slide shows what happens to the lexical categories for *once* and *used* when this approach is adopted. Note that we now require additional lexical categories for these words, whereas, with the application of unary type-changing rules, the lexical categories remain the same (i.e. the same as in the canonical construction *Asbestos was once used ...*). Hence the advantage of the unary rules is that, in practice, they lead to a more compact lexicon and reduce the number of possible lexical categories for some of the words.

Real Examples using Composition So far, the real examples we've seen only require function application, with no unbounded dependencies. Do such cases occur at all in real text? The slide shows two example sentences from natural language corpora which contain instances of object extraction, requiring function composition for their analysis. In Rimell et al. [5] we describe the creation of a corpus of naturally occurring sentences which contain unbounded dependencies, across a variety of syntactic constructions, and give statistics for how often such cases occur in corpora. My RESOURCES webpage has a link to the data described in the paper.

Creating a Treebank for CCG In order to build a statistical parser for CCG — following the standard supervised methodology — we need a CCG treebank: gold-standard pairs of sentences and CCG analyses. The sentence analyses are likely to be CCG derivations, but they could be predicate-argument dependencies (in addition to, or instead of, the derivations). The treebank fulfils

two main roles: it provides data for inducing a grammar, and data for training a statistical disambiguation model.

Building a treebank is expensive, requiring significant time and expertise, so rather than build a CCG treebank from scratch it is more desirable to leverage the information in the existing Penn Treebank.

The Penn Treebank The Penn Treebank (PTB) contains analyses in the form of phrase-structure trees, so somehow we need to transform these into CCG analyses. You may think it is just a case of relabeling the nodes in the trees, but there are various reasons why the transduction problem is harder than that. One reason is that, for some constructions, such as various types of coordination, the PTB trees are not even isomorphic to the CCG derivations, and so it's not just a case of relabelling – the tree structures themselves need changing. Hence it was a considerable effort to produce CCGbank, the CCG version of the Penn Treebank (which was achieved by Julia Hockenmaier and Mark Steedman as part of Julia's PhD thesis [2]).

Three types of information are required from the PTB trees to produce CCG derivations: linguistic head information; the argument/adjunct distinction (since CCG lexical categories encode this explicitly); and information regarding traces and extracted arguments so that long-range dependencies can be analysed correctly.

Example PTB Tree (with traces) Most PTB parsers produce phrase-structure trees without the trace information and co-indexing present. However, this information, which can be used to extract the underlying predicate-argument structure, is an important part of the PTB annotation and crucial for deriving the CCG analyses. In the example on the slide, there are two "traces" or "empty elements": NPs 0_1 and 0_2. The idea is that these are not overtly realised in the surface sentence, but in terms of the underlying structure there is both an object of the verb *do* and a subject of *to do*. The object is *what*, and the subject is *I*, encoded by the co-indexing shown in the diagram.

The Basic Transformation Algorithm If we ignore the more difficult long-range dependency examples, the basic translation algorithm from PTB to CCG, at an abstract level, is straightforward, consisting of the three methods given on the slide. Each one is now described in turn.

Determining Constituent Type Three types of constituent need distinguishing: head, complement and adjunct. In fact, this information is not explicitly encoded in the PTB trees, but rules for heuristically recovering it have been around at least since Collins' thesis [1], whose statistical parsing models were defined in terms of heads and complements (e.g. Collins' Model 2 explicitly uses subcategorisation frames, similar to CCG lexical categories).

Appendix A of Collins' thesis gives a list of head-finding rules, and Appendix A of the CCGbank manual [3] also explains how the complement-adjunct distinction is made.

Binarizing the Tree Section 4 from Hockenmaier and Steedman [4] contains an instructive example showing the translation of a PTB tree to a CCG derivation. Section 4.2 shows how the tree is binarized. Binarization is necessary since the nodes in CCG derivation trees contain at most two children, whereas the trees in the PTB are relatively flat, with some nodes having significantly more children than two. In fact, for some constructions, such as compound noun phrases, the PTB doesn't even contain the requisite information to produce the correct analysis, in which case the CCG (sub-)derivation assumes a default right-branching structure.

Assigning Categories Assigning categories can now be performed by distinguishing three cases. Assigning a CCG label to the root node of a derivation tree is performed by a manually-defined mapping; for example a PTB *VP* node is mapped to $S\backslash NP$, and any of $\{ S, SINV, SQ \}$ get mapped to S.

For heads and complements, the category of a complement child is given a CCG label from a manually-defined mapping, similar to the root node; e.g. a PTB *PP* node is also labelled *PP* in the CCG derivation. The category of the head can be determined from the category of the parent node and the relative position and category of the child. For example, if the parent node is S, and the child is an *NP* to the left, then the category of the head will be $S\backslash NP$ (corresponding to a *VP*).

Finally, for heads and adjuncts, the adjunct category essentially has two copies of the parent label, with the direction determined by the relative position of the adjunct. For example, if the parent is $S\backslash NP$ and the adjunct is to the left, then the adjunct category will be $(S\backslash NP)/(S\backslash NP)$.

Long Range Dependencies Perhaps the most interesting part of the translation procedure is how the trace information in the PTB is propagated around the tree, via the co-indexing, to create the correct CCG lexical categories for analysing long-range dependencies. The interested reader is referred to p.57 of the CCGbank manual for a detailed example.

Properties of CCGbank The coverage of the translation algorithm — in terms of how many PTB trees get turned into CCG derivations — is very high: over 99%. One of the striking features of the resulting CCGbank is how many lexical categories there are for some very common words; e.g. *is* and *as* are assigned over 100 different category types!

More Statistics The numbers on the slide are calculated for sections 2-21, traditionally used as training data. Another striking statistic is that, for word *tokens*, the average number of lexical categories is over 19. This number is high

because of the large number of possible categories for many frequent words; for word *types* the average number is lower. There are over 1,200 lexical category types in total, although a large proportion of these occur only once or twice in the training data. Finally, perhaps the most important statistic on this slide is the coverage figure on unseen data. For section 00, 6% of the tokens do not have the correct lexical category in the lexicon: 3.8% because the token is not in the lexicon; and 2.2% because the token is there, but not with the appropriate category.

References

[1] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[2] Julia Hockenmaier. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh, 2003.

[3] Julia Hockenmaier and Mark Steedman. CCGbank: User's manual. Technical Report MS-CIS-05-09, Department of Computer and Information Science, University of Pennsylvania, 2005.

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

[5] Laura Rimell, Stephen Clark, and Mark Steedman. Unbounded dependency recovery for parser evaluation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 813–821, Singapore, 2009.