Chart Parsing for Combinatory Categorial Grammars

Graeme Gange

May 27, 2008

Outline

Why Categorial Grammars?

Most modern parsing methods involve marking each word in a sentence with Part-of-Speech, and parsing according to derivation rules described by a context-free grammar.

Why Categorial Grammars?

Most modern parsing methods involve marking each word in a sentence with Part-of-Speech, and parsing according to derivation rules described by a context-free grammar.
A number of problems:

- Derivation rules are somewhat arbitrary, and poorly represent the natural behaviour of language.
- Context-free languages are not sufficiently expressive – there are many structural phenomena for which they cannot account.

Combinatory Categorial Grammars

A CCG is defined by two things:

- A set of mappings between words and categories (the lexicon)
- A set of rules for manipulating categories (the combinators)

Combinatory Categorial Grammars

A CCG is defined by two things:

- A set of mappings between words and categories (the lexicon)
- A set of rules for manipulating categories (the combinators)

Unlike in a CFG, where the derivation rules are derived from a set of productions, the rules for a CCG are selected from a fixed set of combinators. This set is dependent only on the language being parsed (and the linguistic phenomena being accounted for).

Linguistic Categories

A category associated with a word (or partial derivation) is one of two things: a primitive category, or a function application.

Linguistic Categories

A category associated with a word (or partial derivation) is one of two things: a primitive category, or a function application.

Primitive Categories

Just like POS-tags, but far fewer. Examples are S, NP, NN,...

Linguistic Categories

A category associated with a word (or partial derivation) is one of two things: a primitive category, or a function application.

Primitive Categories

Just like POS-tags, but far fewer. Examples are S, NP, NN,...

Functional Categories

Somewhat more complex. Words are represented as functions that take other categories as arguments to produce a primitive category.

Example: Determiner – NP/N

Basic Combinators

All operations on functional categories are derived from a fixed set of combinators (based on their namesakes from combinatory logic):

Basic Combinators

All operations on functional categories are derived from a fixed set of combinators (based on their namesakes from combinatory logic):

Function Application

The most basic operation: takes a function and an argument, and produces the result.

Example:

$(S\backslash NP)/NP\ NP \Rightarrow S\backslash NP\ (>)$

$NP\ S\backslash NP \Rightarrow S\ (<)$

Basic Combinators

All operations on functional categories are derived from a fixed set of combinators (based on their namesakes from combinatory logic):

Function Application

The most basic operation: takes a function and an argument, and produces the result.

Example:

$(S\backslash NP)/NP\ NP \Rightarrow S\backslash NP\ (>)$

$NP\ S\backslash NP \Rightarrow S\ (<)$

Function Composition (B)

Definition: $X/Y\ Y/Z \Rightarrow X/Z$

Used for combining functions where the output from one forms the input for another.

Example: $(S\backslash NP)/VP\ VP/NP \Rightarrow (S\backslash NP)/NP\ (> B)$

(Thus, 'might eat' becomes a tensed transitive verb)

Other Combinators

Two other combinators are used in CCGs: Substitution and type–raising.

Other Combinators

Two other combinators are used in CCGs: Substitution and type–raising.

Type-raising (T)

Definition: $X \Rightarrow W/(W\backslash X)$ $(> T)$

Other Combinators

Two other combinators are used in CCGs: Substitution and type–raising.

Type-raising (T)

Definition: $X \Rightarrow W/(W\backslash X)$ $(> T)$

There are a number of restrictions on the application of T, however. First, the category W must be a primitive type (S,NP,...). Second, it can only occur when a category of the appropriate type is adjacent.

Other Combinators

Two other combinators are used in CCGs: Substitution and type–raising.

Type-raising (T)

Definition: $X \Rightarrow W/(W\backslash X)$ $(> T)$

There are a number of restrictions on the application of T, however. First, the category W must be a primitive type (S,NP,...). Second, it can only occur when a category of the appropriate type is adjacent.

Substitution (S)

Definition: $(X/Y)/Z\ Y/Z \Rightarrow X/Z$ $(> S)$

This is more commonly seen in its backward permuting variation:

$Y/Z\ (X\backslash Y)/Z \Rightarrow X/Z$ $(< Sx)$

As with T, all the variations of the S combinator are restricted such that the category Z must be primitive.

Other Combinators

Two other combinators are used in CCGs: Substitution and type–raising.

Type-raising (T)

Definition: $X \Rightarrow W/(W\backslash X)$ $(> T)$

There are a number of restrictions on the application of T, however. First, the category W must be a primitive type (S,NP,...). Second, it can only occur when a category of the appropriate type is adjacent.

Substitution (S)

Definition: $(X/Y)/Z\ Y/Z \Rightarrow X/Z$ $(> S)$

This is more commonly seen in its backward permuting variation:

$Y/Z\ (X\backslash Y)/Z \Rightarrow X/Z$ $(< Sx)$

As with T, all the variations of the S combinator are restricted such that the category Z must be primitive.

For English, these are the only substitution combinators that are used. Also needed for some linguistic phenomena is the backward crossed version of the B combinator, $< Bx$.

Outline

Top-down and Recursive Descent Parsing

Any grammar formalism, no matter how powerful, is somewhat useless without a way to parse it.

Top-down and Recursive Descent Parsing

Any grammar formalism, no matter how powerful, is somewhat useless without a way to parse it.
Unfortunately, CCGs are unsuited to top–down parsing, which rather limits the suitability of the simplest parsers, such as recursive descent parsers. Instead, we must turn to more complex parsing, such as chart parsers.

Chart Parsing

Bottom-up parsing is well suited to CCGs. Each of the combinators (excluding T, which must be handled slightly differently) effectively operate by merging pairs of categories into a single result.

Chart Parsing

Bottom-up parsing is well suited to CCGs. Each of the combinators (excluding T, which must be handled slightly differently) effectively operate by merging pairs of categories into a single result.
In fact, the Cocke-Young-Kasami (CYK) parsing algorithm is singularly appropriate for parsing CCGs.
The CYK parsing algorithm acts by systematically generating all possible edges for gradually increasing intervals, until the interval is the entire sentence to be parsed.

Extensions

There are two modifications generally made to this process in order to allow for large-scale corpus parsing; supertagging, and statistical parsing.

Extensions

There are two modifications generally made to this process in order to allow for large-scale corpus parsing; supertagging, and statistical parsing.

Supertagging

The approaches presented above make the assumption that the words are tagged with functional categories to be used by the parser. For small-scale systems, it is sufficient to hand–code a lexicon; this is impossible for large-scale datasets.

Extensions

There are two modifications generally made to this process in order to allow for large-scale corpus parsing; supertagging, and statistical parsing.

Supertagging

The approaches presented above make the assumption that the words are tagged with functional categories to be used by the parser. For small-scale systems, it is sufficient to hand–code a lexicon; this is impossible for large-scale datasets. Supertaggers work effectively the same as POS-taggers for regular grammars do, except for functional categories, and many of the same techniques are used.

Extensions

There are two modifications generally made to this process in order to allow for large-scale corpus parsing; supertagging, and statistical parsing.

Supertagging

The approaches presented above make the assumption that the words are tagged with functional categories to be used by the parser. For small-scale systems, it is sufficient to hand–code a lexicon; this is impossible for large-scale datasets. Supertaggers work effectively the same as POS-taggers for regular grammars do, except for functional categories, and many of the same techniques are used.

Statistical parsing

CYK-based chart parsing algorithms have approximately $O(n^3)$ runtime. While this is fine for small systems, it is undesirable for large-scale parsing. As such, statistical parsing for CCGs has been developed, analogous to statistical parsing for CFGs.

Outline

Parser

The parser has been implemented using the simpler CYK algorithm, and allows parameterisation of both lexicon and rule set.
For ease of use, it has been closely based on the NLTK infrastructure, following the parsing interfaces already specified.

Demonstration

Demonstration of the parser.