L95: Natural Language Syntax and Parsing
4) Categorial Grammars

Paula Buttery

Dept of Computer Science & Technology, University of Cambridge

Reminder:

For statistical parsing generally we need...

- a grammar
- a parsing algorithm
- a scoring model for parses
- an algorithm for finding best parse

- Parsing **efficiency** is dependent on the parsing and best-parse algorithms
- Parsing **accuracy** is dependent on the grammar and scoring model
- There are reasons that we might use a more sophisticated (and perhaps less robust) grammar formalism even if at the expense of accuracy

Some grammars provide a mapping between syntax and semantic structure

- **Combinatory Categorial Grammars** provide a mapping between syntactic structure and predicate-argument structure
- CCG parsers exist that are robust and efficient (Clark & Currans 2007) `https://www.cl.cam.ac.uk/~sc609/candc-1.00.html`
- The **C&C parser** uses a CCG treebank (CCGBank) derived from the Penn Treebank to build a grammar and training the scoring model
- A **supertagging** phase is needed before parsing commences
- Uses a discriminative model over complete parses

First, what is a CCG?

Categorial grammars are **lexicalized grammars**

In a **classic categorial grammar** all symbols in the alphabet are associated with a finite number of **types**.

- Types are formed from primitive types using two operators, $\backslash$ and $/$.
- If P_r is the set of **primitive types** then the set of all types, T_p, satisfies:
  - $P_r \subset T_p$
  - if $A \in T_p$ and $B \in T_p$ then $A \backslash B \in T_p$
  - if $A \in T_p$ and $B \in T_p$ then $A/B \in T_p$
- Note that it is possible to arrange types in a hierarchy: a type A is a *subtype* of B if A occurs in B (that is, A is a subtype of B iff $A = B$; or ($B = B_1 \backslash B_2$ or $B = B_1/B_2$) and A is a subtype of B_1 or B_2).

Categorial grammars are **lexicalized grammars**

- A relation, $\mathcal{R}$, maps symbols in the alphabet Σ to members of T_p.
- A grammar that associates at most one type to each symbol in Σ is called a **rigid grammar**
- A grammar that assigns at most k types to any symbol is a **k-valued grammar**.
- We can define a classic categorial grammar as $G_{cg} = (\Sigma, P_r, S, \mathcal{R})$ where:
  - Σ is the alphabet/set of terminals
  - P_r is the set of primitive types
  - S is a distinguished member of the primitive types $S \in P_r$ that will be the root of complete derivations
  - $\mathcal{R}$ is a relation $\Sigma \times T_p$ where T_p is the set of all types as generated from P_r as described above

Categorial grammars are **lexicalized grammars**

A string has a valid parse if the types assigned to its symbols can be combined to produce a derivation tree with root S.

Types may be combined using the two rules of **function application**:

- FORWARD APPLICATION is indicated by the symbol $>$:

$$\frac{A/B \qquad B}{A} >$$

- BACKWARD APPLICATION is indicated by the symbol $<$:

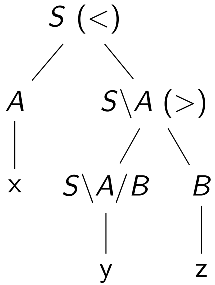
$$\frac{B \qquad A\backslash B}{A} <$$

Categorial grammars are **lexicalized grammars**

Derivation tree for the string *xyz* using the grammar $G_{cg} = (\Sigma, P_r, S, \mathcal{R})$ where:

$$
\begin{aligned}
Pr &= \{S, A, B\} \\
\Sigma &= \{x, y, z\} \\
S &= S \\
\mathcal{R} &= \{(x, A), (y, S\backslash A/B), (z, B)\}
\end{aligned}
$$

$$\dfrac{\dfrac{x}{A}\,\mathcal{R} \qquad \dfrac{\dfrac{y}{S\backslash A/B}\,\mathcal{R} \qquad \dfrac{z}{B}\,\mathcal{R}}{S\backslash A}\,>}{S}\,<$$

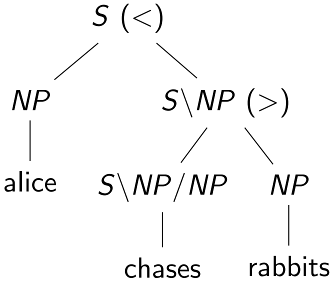


Categorial grammars are **lexicalized grammars**

Derivation tree for the string *Alice chases rabbits* using the grammar $G_{cg} = (\Sigma, P_r, S, \mathcal{R})$ where:

$$
\begin{array}{rcl}
Pr & = & \{S, NP\} \\
\Sigma & = & \{alice, chases, rabbits\} \\
S & = & S \\
\mathcal{R} & = & \{(alice, NP), (chases, S\backslash NP/NP), \\
 & & (rabbits, NP)\}
\end{array}
$$

$$
\dfrac{\dfrac{alice}{NP}\,\mathcal{R} \qquad \dfrac{\dfrac{chases}{S\backslash NP/NP}\,\mathcal{R} \qquad \dfrac{rabbits}{NP}\,\mathcal{R}}{S\backslash NP}\,>}{S}\,<
$$



We can construct a **strongly equivalent** CFG

To create a context-free grammar $G_{cfg} = (\mathcal{N}, \Sigma, S, \mathcal{P})$ with strong equivalence to $G_{cg} = (\Sigma, P_r, S, \mathcal{R})$ we can define G_{cfg} as:

$$
\begin{aligned}
\mathcal{N} &= P_r \cup range(\mathcal{R}) \\
\Sigma &= \Sigma \\
S &= S \\
P &= \{A \rightarrow B \;\; A\backslash B \mid A\backslash B \in range(\mathcal{R})\} \\
&\quad \cup \{A \rightarrow A/B \;\; B \mid A/B \in range(\mathcal{R})\} \\
&\quad \cup \{A \rightarrow a \mid \mathcal{R} : a \rightarrow A\}
\end{aligned}
$$

Combinatory categorial grammars **extend** classic CG

Combinatory categorial grammars use **function composition** rules in addition to function application:

- FORWARD COMPOSITION is indicated by the symbol $> B$:

$$\frac{X/Y \qquad Y/Z}{X/Z} > B$$

- BACKWARD COMPOSITION is indicated by the symbol $< B$:

$$\frac{Y\backslash Z \qquad X\backslash Y}{X\backslash Z} < B$$

They also use **type-raising** rules (only applies to NP, PP, $S[adj]\backslash NP$):

$$\frac{X}{T/(T\backslash X)}\ T$$

$$\frac{X}{T\backslash(T/X)}\ T$$

- Also backward crossed composition and co-ordination (see Steedman)

CCG examples in class

The C&C parser uses a **log-linear** model

- Recall that discriminative models define $P(T|W)$ directly (rather than from subparts of the derivation)
- C&C is a discriminative parser that uses a log-linear model to score parses based on their features:

$$P(T|W) = \frac{1}{Z_W} \exp^{\lambda.F(T)}$$

  where $\lambda.F(T) = \sum_i \lambda_i f_i(T)$ and λ_i is the weight of the ith feature, f_i (and Z_W is a normalising factor)
- Train by maximising log-likelihood over the training data (minus a prior term to prevent overfitting)
- Requires building a packed chart of all the trees using CKY (instance of a **feature forest**)
- Packing requires the features in the model are **local**—confined to a single rule application

The C&C parser uses a **log-linear** parsing model

The features used in the C&C parser are:

- features encoding local trees (that is two combining categories and the result category)
- features encoding word-lexical category pairs at the leaves of the derivation
- features encoding the category at the root of the derivation
- features encoding word-word dependencies, including the distance between them

- Each feature type has variants with and without head information (lexical items and pos tags)

Lexicalised grammar parsers have two steps

Parsing with lexicalised grammar formalisms is a two-stage process:

1. Lexical categories are assigned to each word in the sentence
2. Parser combines the categories together to form legal structures

For C&C:

1. Uses a **supertagger** (log-linear model using words and PoS tags in a 5-word window)
2. Uses the CKY chart parsing algorithm and Viterbi to find the best parse

Ambiguous CCG parse example in class