

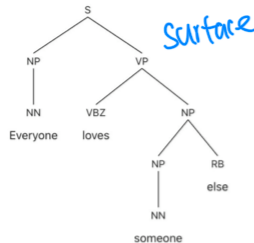
9. Semantic Parsing with CCG

POS Tagging, Constituency Parsing, and Dependency Parsing are for syntax. Semantic Parsing is for meaning.

"Everyone loves someone else."

Syntactic Representation

structural organization of text



Semantic Representation

underlying meaning of text

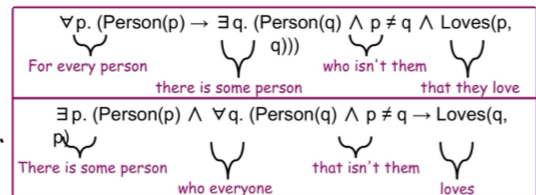
$\forall p. (\text{Person}(p) \rightarrow \exists q. (\text{Person}(q) \wedge p \neq q \wedge \text{Loves}(p, q)))$

Logical Forms

A representation of the meaning of a sentence.

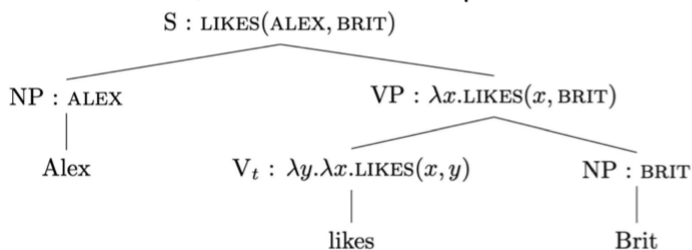
- Natural language: "Everyone loves someone else."

- Logical form: $\forall p. (\text{Person}(p) \rightarrow \exists q. (\text{Person}(q) \wedge p \neq q \wedge \text{Loves}(p, q)))$



Lambda Calculus

Principle of compositionality: The meaning of a complex expression is a function of the meanings of that expression's constituent parts.



Standard Lambda Calculus

Abstraction

- If M is a term and x is a variable, then $\lambda x. M$ is a term.
- $\lambda x. M$ can be understood as the function that takes x as input and produces M as output.
- If the expression N is the input, then the output expression is $M[x := N]$, where we replace every (free) occurrence of x in M by N .

Application

- If M and N are terms, then $(M N)$ is a term.
- $(M N)$ can be understood as applying M to N .
- Pay attention to the scope, example $\lambda x. \lambda y. (x(\lambda x. x x) y)$.
 - Scope of λx is $\lambda y. (x(\lambda x. x x) y)$
 - Scope of λy is $(x(\lambda x. x x) y)$

- Scope of inner λx is x

Free and bound variables

Example: $((\lambda x. \lambda y. (x ((\lambda x. x x) y)) \lambda x. x)(z)) \rightarrow z$ is free, all other are bounded.

α -Conversion

- Renaming a variable in a lambda term
- Rename a variable in an abstraction, together with all occurrences of the variable in the scope of the abstraction, which are bound to the same abstraction.
- Example:
 - $\lambda x. x \rightarrow \lambda y. y$
 - $\lambda x. (x x) y \rightarrow \lambda t. (t t) y$
 - $\lambda x. \lambda y. (x ((\lambda x. x x) y)) \rightarrow \lambda z. \lambda y. (z ((\lambda x. x x) y))$
- The renaming should not cause the free occurrence of a variable to become bound or cause a bound occurrence of a variable to become bound to a different abstraction.

β -reduction

- Applying one lambda term to another
- The β -reduction of a term $(\lambda x. M N)$ is obtained by applying the function $\lambda x. M$ to N , i.e., replacing every free occurrence of x in M by N . We denote the obtained expression as $M[x := N]$
- Example:
 - $(\lambda x. \lambda y. (x ((\lambda x. x x) y)) z) \rightarrow \lambda y. (z ((\lambda x. x x) y))$
 - $\lambda y. (z ((\lambda x. x x) y)) \rightarrow \lambda y. (z (z y))$
- The replacement of free occurrences of x in M should not cause the free occurrence of any variable in N to become bound in $M[x := N]$.
- Repeatedly applying β -reductions may or may not terminate
 - Example: $(FF) = (\lambda x. ((x x) x) F) \rightarrow ((FF) F) \rightarrow (((FF) F) F) \rightarrow \dots$

Enriched Lambda Calculus

Logical constants

- Representing objects and relations between objects in the real world
- Example: representing humans ALEX, BRIT, etc.
 representing cities NEW-YORK-CITY, ZURICH, etc.
 representing relations LIKES, TEACHER, etc.
- Every relation symbol has an arity that determines the number of objects that it relates
- Example: LIKES has arity 2, LIKES(x,y) means x likes y.

TEACHER has arity 1, and $\text{TEACHER}(x)$ means x is a teacher.

Variables

- Representing undetermined objects or undetermined relations.
- For objects, denote using lower-case letters.
- For relations, denote using upper-case letters.

Literals

- Formed by applying relations to objects.
 - Example: $\text{LIKES}(\text{ALEX}, \text{BRIT})$ means Alex likes Brit.
 - $\text{TEACHER}(\text{ALEX})$ means Alex is a teacher.
- We can use variables instead of logical constants in order to construct literals
 - Example: $\text{LIKES}(\text{ALEX}, y)$ means Alex likes an undetermined person y .
 - $P(\text{ALEX}, \text{BRIT})$ means an undetermined relation P holds between Alex and Brit.
 - $P(x, y)$ means an undetermined relation P holds between the undetermined objects x and y .

Basic logical forms

- Construct basic logical forms using logical connectives and quantifiers.
- Example: $\text{TEACHER}(x) \wedge \text{LIKES}(x, \text{BRIT})$ means the undetermined object x is a teacher that likes Brit.
 - $\exists y. \text{LIKES}(\text{ALEX}, y) \wedge \text{TEACHER}(y)$ means Alex likes some teacher y .

Lambda terms

- Starting from basic expressions, we build the lambda terms by recursively applying the process of abstraction.
- $\lambda x. \text{LIKES}(x, \text{BRIT})$, takes x as input and produces $\text{LIKES}(x, \text{BRIT})$ as output.
- $\lambda P. P(\text{ALEX}, \text{BRIT})$, takes (a relation) P as input and produces $P(\text{ALEX}, \text{BRIT})$ output.
- α -conversions and β -reductions can be applied exactly as in the standard lambda calculus.

Combinatory Logic

- The basic terms are
 - Variables: x, y, z
 - Primitive functions (or, combinators): These are higher order functions, $I, S, K, \dots$
- Terms are then recursively constructed using the rule of application
 - If M and N are combinatory terms, then (MN) is a term that means applying M to N .
- Parentheses are left associative, e.g. $(K x y)$ means $((K x) y)$ and $(S x y z)$ means $((S x) y z)$.

Some common combinators

Definition 8.4

The **I combinator** represents the identity function, and is defined as $(I x) = x$ for every x .

Definition 8.5

The **K combinator** is defined as $(K x y) = ((K x), y) = x, \forall x, y$. The **K combinator** produces constant functions, i.e., $(K x)$ is the constant function that always outputs x .

Definition 8.6

The **S combinator** is defined as $(S x y z) = ((x z (y z)) = ((x z)(y z)), \forall x, y, z$.

Definition 8.7

The **B combinator** is defined as $(B x y z) = ((x (y z)), \forall x, y, z$. The **B combinator** can be understood as being a composition operator, i.e., if f and g are functions, then $(B f g)$ is equivalent to $f \circ g$.

Definition 8.8

The **C combinator** is defined as $(C x y z) = ((x z), y) \forall x, y, z$.

Definition 8.9

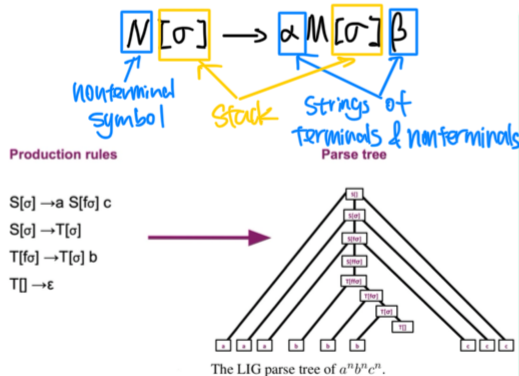
The **T combinator** is defined as $(T x y) = (y x) \forall x, y$. The **T combinator** is analogous to the canonical embedding of a vector space V into V^{**} , the dual of its dual. If f is a function, then $((T x) f) = (f x)$.

Combinatory Categorical Grammars (CCG)

We need to use CCG because

- To deal with "mildly context-sensitive grammars".
- To deal with cross-serial dependencies.

Linear Indexed Grammar (LIG)



Definition 8.10

A linear indexed grammar G is a quintuple:

$$G = \langle N, S, I, \Sigma, \mathcal{R} \rangle$$

(8.1)

with

1. A set of nonterminal symbols N ; written as upper-case letters, e.g., $N_1, N_2, N_3, \dots$
2. A distinguished start non-terminal S
3. A finite set of indices I ; written as usual function letters, e.g., $f, g, h, \dots$
4. An alphabet of terminal symbols Σ ; written as lower-case letters, e.g., $a_1, \dots$
5. A set of production rules $\mathcal{R}$ of one of the following forms:

- $N[\sigma] \rightarrow \alpha M[\sigma] \beta$
- $N[\sigma] \rightarrow \alpha M[f\sigma] \beta$
- $N[f\sigma] \rightarrow \alpha M[\sigma] \beta$

CCG (in general)

Definition 8.12

For a finite set V and two distinguished symbols $"/$ and $\backslash$, the set $C(V)$ of categories of V is inductively defined as follows:

- $V \subseteq C(V)$.
- If $v_1, v_2 \in V$, then $v_1 / v_2 \in C(V)$ and $v_1 \backslash v_2 \in C(V)$.

Definition 8.13

A combinatory categorical grammar (CCG) G is a quintuple

$$\langle V_T, V_N, S, f, R \rangle$$

(8.2)

that consists of the following:

- A finite set of terminals V_T (lexicon).
- A finite set of non-terminals V_N (atomic categories).
- A distinguished category $S \in V_N$.
- A function f mapping $V_T \cup \{\epsilon\}$ to finite subsets of $C(V_N)$, where $C(V_N)$ is the set of categories.
- A finite set of combinatory rules R , described below.

Definition 8.14

Let $|_i$, for $i \leq n$, denote either $\backslash$ or $/$. For categories x, y, z , the combinatory rules are the following:

1. Forward application:

$$(x/y) \quad y \mapsto x \quad (8.3)$$

2. Backward application:

$$y \quad (x \backslash y) \mapsto x \quad (8.4)$$

3. Generalized forward composition:

$$(x/y) \quad (\dots (y |_1 z_1) |_2 \dots |_n z_n) \mapsto (\dots (x |_1 z_1) |_2 \dots |_n z_n) \quad (8.5)$$

4. Generalized backward composition:

$$(\dots (y |_1 z_1) |_2 \dots |_n z_n) \quad (x \backslash y) \mapsto (\dots (x |_1 z_1) |_2 \dots |_n z_n) \quad (8.6)$$

Example: Deriving the meaning using CCG

What	states	border	Texas
$(S/(S \backslash NP))/N$	N	$(S \backslash NP)/NP$	NP
$\lambda f. \lambda g. \lambda x. f(x) \wedge g(x)$	$\lambda x. state(x)$	$\lambda x. \lambda y. borders(y, x)$	$texas$
$\rightarrow$			
$S/(S \backslash NP)$		$(S \backslash NP)$	
$\lambda g. \lambda x. state(x) \wedge g(x)$		$\lambda y. borders(y, texas)$	
$\rightarrow$			
S			
$\lambda x. state(x) \wedge borders(x, texas)$			

Relationship to CFGs

- Every CCG restricts itself to a finite set of rules. Considering higher-order composition rules, for every degree, there are forward and backward rules. However, each rule may give rise to infinitely many rule instances.
- CFGs have a finite set of non-terminals.

Parsing CCGs

- General rule of inference is $\frac{A_1 \dots A_k}{B}$, B is consequence of $A_1, \dots, A_k$.
- Derivation trees consist of unary and binary branchings.

- Unary corresponds to lexicon entries.

- Binary corresponds to composition rules.

- One inference rule for every forward rule:

$$\frac{[X/Y, i, j] [Y\beta, j, k]}{X\beta, i, k} \quad X/Y \quad Y\beta \Rightarrow X\beta$$

- Axioms have the form $[X, i, i+1]$, where $w_{i+1} = X$ is a lexicon entry

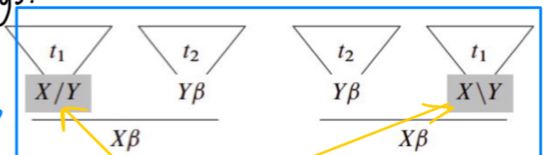
Example:

$$w_1 := A$$

$$w_5 := E/H/C$$

We have the following inference rule:

$$[X/Y, i, j] [Y\beta, j, k] \quad Y/V \quad V\beta \rightarrow V\beta$$



$w_2 := D$
 $w_3 := C \backslash A / F$
 $w_4 := S \backslash E$

$w_6 := F / G \backslash D$
 $w_7 := G$
 $w_8 := H$

$$\frac{}{X\beta, i, k} \Lambda / I \quad I \supset \implies \Lambda \supset$$

Through CCG parsing, we obtain the following result:

