Parse trees: from formal to natural languages

Matilde Marcolli

CS101: Mathematical and Computational Linguistics

Winter 2015

Context-free grammars $\mathcal{G} = (V_N, V_T, P, S)$

- V_N and V_T disjoint finite sets: *non-terminal* and *terminal* symbols
- $S \in V_N$ *start symbol*
- P finite rewriting system on $V_N \cup V_T$

$P =$ *production rules*: $A \to \alpha$ with $A \in V_N$ and $\alpha \in (V_N \cup V_T)^\star$

Language produced by a grammar $\mathcal{G}$:

$$\mathcal{L}_\mathcal{G} = \{w \in V_T^\star \mid S \stackrel{\bullet}{\to}_P w\}$$

language with alphabet V_T

Parse Trees of a context free language

• a finite, rooted, oriented (away from the root), planar tree (with a choice of a planar embedding)

• vertices decorated by elements of $V_N \cup V_T$ (terminal and non-terminal symbols)

• if an "internal vertex" (not a leaf) is decorated by A and if all the terminal vertices of oriented edges out of vertex A are labelled by $w_1, \ldots, w_n$ (with ordering specified by planar embedding) then

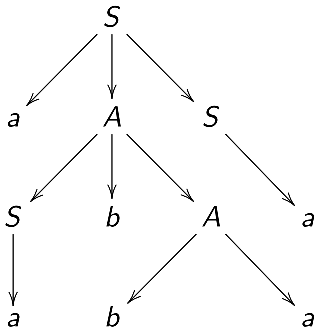
$$A \to w_1 \cdots w_n \quad \in P$$

Example

- Grammar: $\mathcal{G} = \{\{S, A\}, \{a, b\}, P, S\}$ with productions P

$$S \rightarrow aAS, \quad S \rightarrow a, \quad A \rightarrow SbA, \quad A \rightarrow SS, \quad A \rightarrow ba$$

- this is a possible parse tree for the string *aabbaa* in $\mathcal{L}_{\mathcal{G}}$



Fact: for context-free $\mathcal{G} = (V_N, V_T, P, S)$ have a chain of derivations in $\mathcal{G}$

$$A \xrightarrow{\bullet} w_1 \cdots w_n$$

if and only if there is a parse tree for $\mathcal{G}$ with root decorated by A and with n leaves decorated by $w_1, \ldots, w_n$

to see this: if have parse tree with input A and outputs $w_1, \ldots, w_n$, show by induction on number of internal vertices that $A \xrightarrow{\bullet} w_1 \cdots w_n$ in $\mathcal{G}$

• if only root and leaves (no other vertices) then $A \to w_1 \cdots w_n$ is a production rule in P

• otherwise, assume know for all trees with $\leq k$ vertices (induction hypothesis); if tree has $k+1$ vertices, look at immediate successor vertices from root: get a production in P (from A to the list of successors) then for each successor that not leaf get a tree with $\leq k$ vertices conversely if $A \stackrel{\bullet}{\to} w_1 \cdots w_n$ in $\mathcal{G}$

- then there is a chain of derivations in P,

$$A \to u_1, \quad \ldots \quad u_i \to u_{i+1}, \quad \ldots \quad u_k \to w_1 \cdots w_n$$

where the next derivation giving u_{i+1} is applied to some non-terminal element in the string u_i

- the first production rule $A \to u_1$, produces a string $u_1 = u_{11} \ldots u_{1k_1}$ and gives a root labelled A with valence k_1 and leaves labelled by u_{1j}

- the second $u_1 \to u_2$ consists of some production rules in P applied to some of the non-terminal symbols u_{1j} in the string u_1: append trees to the vertices labelled u_{1j} with leaves the resulting strings in u_2

- continue with successive derivations until obtain a tree with root A and with leaves (ordered by planar embedding) labelled by $w_1, \ldots, w_n$

Ambiguity of context-free languages (grammars)

- A context-free grammar $\mathcal{G}$ is *ambiguous* if there are words $w \in \mathcal{L}_\mathcal{G}$ that admit different (non-equivalent) parse trees

- Trivial example: $S \to A$, $S \to B$, $A \to a$, $B \to a$

- A language $\mathcal{L}$ is *inherently ambiguous* if every possible context-free grammar $\mathcal{G}$ with $\mathcal{L} = \mathcal{L}_\mathcal{G}$ is ambiguous

Example

$$\mathcal{L} = \{a^n b^n c^m d^m \,|\, n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n \,|\, n \geq 1, m \geq 1\}$$

is inherently ambiguous because there are infinitely many strings of the form $a^n b^n c^n d^n$ that have different parse trees

Sketch of argument for Example:

Suppose $\exists$ unambiguous context-free $\mathcal{G}$ for $\mathcal{L}$ above: then can always arrange that for all $A \in V_N \smallsetminus \{S\}$ have $A \xrightarrow{\bullet} x_1 A x_2$ with both x_1, x_2 not the empty word

• because of the form of words in $\mathcal{L}$ must have x_1 and x_2 consisting of only one type of symbol a, b, c, d (otherwise get a string not in $\mathcal{L}$)

• also symbol for x_1 different from symbol for x_2 (because of form of words cannot increase occurrences of only one type of symbol and still get words in $\mathcal{L}$) and also length of x_1 and x_2 has to be same (same reason)

• check only cases are x_1 made of a's and x_2 of b's or d's; x_1 made of b's and x_2 of c's; x_1 made of c's and x_2 of d's (divide variables other than S into C_{ab}, C_{ad}, C_{bc}, C_{cd})

• subdivide $\mathcal{G}$ into two grammars

$$\mathcal{G}_1 = \{\{S\} \cup C_{ab} \cup C_{cd}, V_T, P_1, S\} \quad \mathcal{G}_2 = \{\{S\} \cup C_{ad} \cup C_{bc}, V_T, P_2, S\}$$

$\mathcal{G}_1$ generates all $a^n b^n c^m d^m$ with $n \neq m$ and some $a^n b^n c^n d^n$; $\mathcal{G}_2$ generates all $a^n b^m c^m d^n$ with $n \neq m$ and some $a^n b^n c^n d^n$

• then show (set theoretic argument) that both $\mathcal{G}_1$ and $\mathcal{G}_2$ must generate all but finitely many of the $a^n b^n c^n d^n$: all these have two different parse trees

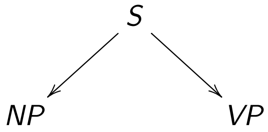
Parse trees and natural languages

Example How to *generate* the English sentence:
The book is believed to have been written by the Aztecs

- Two step process:

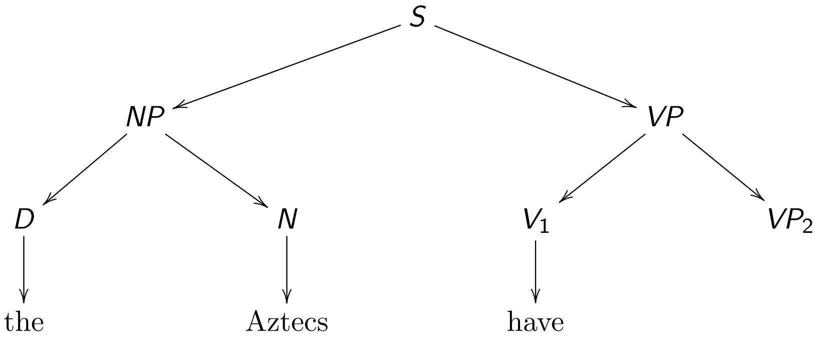
  1. *generate* two separate sentences:
     (1) *The Aztecs have written the book*;
     (2) *We believe it*
  2. combine them with appropriate *transformations*

- first sentence (S): noun phrase (NP) + verb phrase (VP)

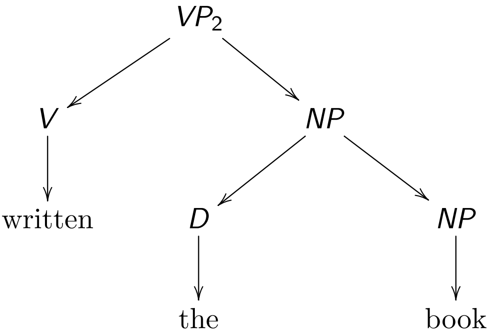


- The NP part is: determiner (D) + noun (N);

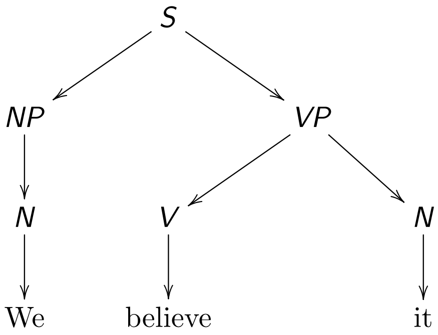
VP part has: auxiliary (V_1) + rest of phrase (VP_2)



- the VP_2 part consists of: verb (V) + noun phrase (NP)



- Similarly, the second sentence *We believe it* has a parse tree



- Operation 1: *Passive Transformation*

The Aztecs have written the book ⇒ *The book has been written by the Aztecs*

- Operation 2: *Insertion*

We believe IT ⇒ *We believe the book has been written by the Aztecs*

- Operation 3: *Passive Transformation*

We believe the book has been written by the Aztecs ⇒ *The book is believed by us to have been written by the Aztecs*

- Operation 4: *Agent Deletion*

The book is believed by us to have been written by the Aztecs ⇒ *The book is believed to have been written by the Aztecs*

Main idea:

Generative process with sentence (S) as start symbol; non-terminal symbols given by syntactic identifiers (NP, VP, N, V, D, etc.); terminals given by words; production rules encode syntactic structure, together with transformations on parse trees

Early formulation of Generative Grammar

1. Noam Chomsky, *The logical structure of linguistic theory* (1955), Plenum, 1975.
2. Noam Chomsky, *Syntactic structures*, Mouton, 1957.

Later developments focused more on transformations and less on production rules

A closer look at Transformational Grammar

• A set of trees (for example: parse trees of a context-free or context-sensitive grammar): *Base trees*

• finite, rooted, oriented, planar trees with decorated vertices: if one vertex v has only one outgoing edge e the label at $t(e)$ different from the label at $v = s(e)$

• Base trees $\mathcal{B} = \{B, V, V_T\}$ with B a collection of trees as above, $V_T \subset V$ a finite set of terminal symbols used to label leaves of trees in B and internal vertices labelled by non-terminal symbols $V_N = V \smallsetminus V_T$

- Additional data $\Delta = \{\Sigma, \Sigma_A, X, V''\}$ with
  - Σ finite set of abstract symbols with $V_T \subseteq \Sigma$ and $\Sigma \cap V_N = \emptyset$
  - X abstract symbol not in $V \cup \Sigma$ (dummy variable)
  - a subset $\Sigma_A \subseteq \Sigma$
  - V'' set containing $V \cup \Sigma \cup \{X\}$ and additional symbols $Y^{(k)}$ with $Y \in V \cup \Sigma \cup \{X\}$ and $k \in \mathbb{N}$

Σ_A represents the set of symbols over which the language generated by the grammar is defined

- $\mathcal{R} =$ finite set of transformation rules (T-rules) (D, C) with respect to V'', Σ and X

T-rules

- symbols $X, X^{(k)}$ in V'' mark parts of the tree that cannot be moved by the transformation T
- $D =$ domain statement: string $\alpha_1 \cdots \alpha_k$ of symbols in V''
- $C =$ structural change statement on D: string $\beta_1 \cdots \beta_k$ of symbols in $\{k\}_{k\in\mathbb{N}} \cup \Sigma$

$\beta_j = j$ if symbol $D_j = \alpha_j$ of D is some $X^{(r)}$ (unmoved by T) otherwise β_j is either some $i \neq j$ or some symbol in Σ

- Example: passivization in English

the cat ate the mouse $\mapsto$ *the mouse was eaten by the cat*

$N^{(1)}TVN^{(2)} \mapsto N^{(2)}\ T \text{ be } E_n\ VN^{(1)}$

$N^{(1)} = \text{cat},\ T = \text{tense, past};\ V = \text{eat},\ N^{(2)} = \text{mouse}$

$TV \mapsto T \text{ be } E_n V$ ate $\mapsto$ was eaten

- T_{pass} rule (D, C) where

$D = \alpha_1\alpha_2\cdots\alpha_8 = X^{(1)}\$N^{(1)}TVN^{(2)}\$X^{(2)}$

$C = \beta_1\beta_2\cdots\beta_8 = 1264(\text{ be } E_n\ 5 \text{ by })378$

$\$ = \text{boundary marker}$

States

• additional structure of transformational grammar:
$\Omega = \{K, \mathcal{N}, \delta, s_0\}$

- K = finite set of states, s_0 = start state
- $\mathcal{N} = \{N(s), s \in K\}$ with $N(s)$ partially ordered set over $\mathcal{R} \cup \{\#\}$ (with $\#$ stop symbol occurring as maximal element)
- $\delta : K \times \mathcal{R} \to K$ (next state function)

Keeps into account order of application of the T-rules (order matters)

Records the "past history" of the use of the rules (can reconstruct the path of rule applications)

Assume a rule T leaves a tree unchanged if it does not apply to it (continue to next rule in the ordered list)

Language generated by a transformational grammar

- the set of base trees = *deep structure*
- all the tree produced by applying compositions of transformations to base trees = *surface structure*

- (τ, s) with τ a tree and $s \in K$

$$(\tau, s) \vdash (\tau', s')$$

if $\exists\ T = (D, C)$ T-rule with $\tau' = T(\tau)$, $T \in N(s)$, $s' = \delta(s, T)$, there is no other τ'' and $T' \in N(s)$ with $T' < T$ and $\tau'' = T'(\tau)$

- string w generated by T-grammar if $w \in \Sigma_A^*$, there are τ, τ' and s' with $\tau \in \mathcal{B}$, $(\tau, s_0) \vdash^* (\tau', s') \vdash Stop$ and w is the terminal string of the tree τ'

Refences

- Noam Chomsky, *Selected Readings on Transformational Theory*, Dover 2012.
- Seymour Ginsburg, Barbara Partee, *A mathematical model of Transformational Grammars*, Information and Control 15 (1969) 297–334
- P.S.Peters, R.W.Ritchie, *On the generative power of transformational grammars*, Information Sci. 6 (1973), 49–83.
- Barbara Partee, Alice ter Meulen, Robert Wall, *Mathematical Methods in Linguistics*, Kluwer, 1990.

Tree Adjoining Grammars (Joshi, Levy, Takahashi)

Mathematical model for structural composition of parse trees: instead of production rules that rewrite strings as in the formal languages grammars, use a system of trees with tree rewriting rules

- a (finite) set of *Elementary Trees*
- *Substitution rule*: graft a terminal leaf of a tree T to the root of another tree (as in previous example: replace the *it* terminal vertex of the second tree with the root of the first tree, parsing the sentence *The book has been written by the Aztecs*)
- *Adjoining rule*: at an internal vertex of the tree labelled by X attach a tree with root labelled by X and with one of the leaves also labelled by X with anything outgoing from original tree at X then attached to the X-labelled leaf of the inserted tree.

Note: no additional transformations used (unlike example above with "passive transformation", "agent deletion" etc.) other than substitution and adjoining

Fundamental assumptions of TAG:

- all syntactic dependencies are encoded (locally) in the elementary trees

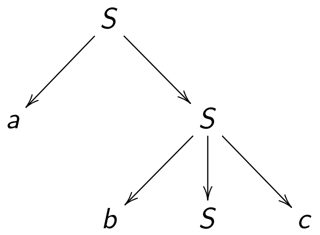
- non-local dependencies must be reducible to local ones (after contracting a certain number of adjoined trees)

TAG derivation: a combination of elementary trees via a sequence of substitutions and adjoining

Derivation structure: a tree whose vertices are labelled by elementary trees and daughter vertices of a given node T are the elementary trees that are substituted or adjoined into the tree T (requires "independence" of the operations performed)

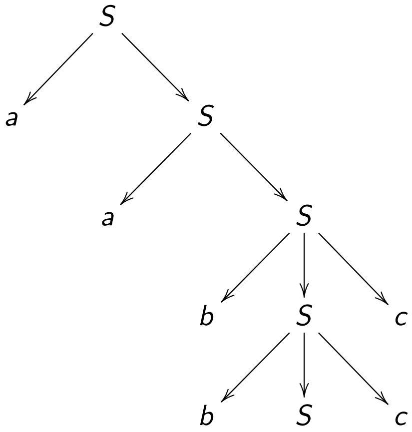
Generative power of TAG:

- All context-free languages can be generated by a TAG
- $\mathcal{L} = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ not generated by a context-free grammar, but can be generated by a TAG



repeatedly adjoin copies of this elementary tree into itself at the S vertex with the first b daughter

$a^2 b^2 c^2$ from first adjoining, etc.



But... simple examples of context-sensitive languages that cannot be generated by TAG's: (Vijay–Shanker)

$$\mathcal{L} = \{a^n b^n c^n d^n e^n \mid n \in \mathbb{N}\}$$

Representing natural languages?

• Question: How good are context-free grammars at representing natural languages?

- Not always good, but often good (better than earlier criticism indicated)
- Some explicit examples not context-free (cross-serial subordinate clause in Swiss-German)

1. G.K. Pullum, G. Gazdar *Natural languages and context-free languages*, Linguistics and Philosophy, Vol.4 (1982) N.4, 471–504
2. S. Shieber, *Evidence against the context-freeness of natural language*, Linguistics and Philosophy, Vol.8 (1985) N.3, 333–343

Are natural languages context-free?

- Try to show they are not by finding cross-serial dependencies of arbitrarily large size

- Example: the language $\mathcal{L} = \{xx^R \,|\, x \in \{a, b\}^*\}$ has cross serial dependencies of arbitrary length (the i-th and $(n+i)$-th term have to be the same (x^R = reversal of x)
- if cross serial dependencies of arbitrary length not context-free

- Example (Chomsky): English has arbitrarily long cross-serial dependencies because can combine dependencies such as *if ... then* and *either ... or* with subject-verb dependence and make arbitrarily long sequences

- Problem with this kind of argument: can have a non-context-free language embedded inside a context-free one

$$\mathcal{L} = \{xx^R \,|\, x \in \{a, b\}^*\} \subset \mathcal{L}' = \{a, b\}^*$$

context-free (regular)

- Better example from Swiss German cross-serial order in dependent clauses

$$wa^n b^m xc^n d^m y$$

Jan säit das mer (d'chind)^n (em Hans)^m es huus haend wele (laa)^n (häfte)^m aastrüche
non-context-free language (intersection of SG with a regular language, so SG also non-context-free)

Question: How good are TAG's at modeling natural languages?

• They give a class of languages that includes the context-free ones but is larger (seems to take care of the kind of $wa^n b^m xc^n d^m y$ type of problem)

• A lot of examples of explicit linguistic analysis using TAG in the book:

- A. Abeillé, O. Rambow (Eds.), *Tree Adjoining Grammars*, CSLI Publications, 2000.

Some References:

1. J.E. Hopcroft, J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison–Wesley, 1979
2. Robert Frank, *Phrase structure composition and syntactic dependences*, MIT Press, 2002
3. A.K. Joshi, L. Levy, M. Takahashi, *The tree adjunct grammars*, Journal of the Computer and System Sciences, 10 (1975) 136–163