Stochastic Context-free Grammars and Parsing

Brigitte Krenn
Österreichisches Forschungsinstitut für Artificial Intelligence
ÖFAI
Freyung 6, 1010 Wien
brigitte@ai.univie.ac.at

Context-free Grammar

A **formal grammar** is a system for rewriting strings over some alphabet V.

A **context-free grammar** G is a quadruple $\langle V_N, V_T, S, R\rangle$ where:

V_N	is a finite set of nonterminal symbols.
V_T	is a finite set of terminal symbols. Let V denote $V_N \cup V_T$.
$S \in V_N$	is a distinguished start symbol, or axiom.
R	is a finite set of productions $X \to \beta$ where $X \in V_N$ and $\beta \in V^*$.

The string $\alpha X \gamma \in V^*$ can be rewritten in one step as $\alpha\beta\gamma \in V^*$ iff $X \to \beta$ is in R.

One occurrence of the nonterminal symbol X in a string is replaced by the string β:

$$\alpha X \gamma \Rightarrow \alpha\beta\gamma$$

A string $\phi \in V^*$ can be rewritten as the string ψ in a finite number of steps:

$$\phi \Rightarrow^* \psi$$

$L(G)$ denotes the set of strings over V_T generated by G and is defined as $\{\omega \in V_T^* : S \Rightarrow^* \omega\}$.

(Extended) Chomsky Normal Form

A context-free grammar is said to be in **Chomsky Normal Form (CNF)**,

if the productions are either of the form $X_i \rightarrow X_j X_k$,

a binary production rewriting a nonterminal as two nonterminals,

or of the form $X_i \rightarrow w$

where $w \in V_T$, a unary production rewriting a nonterminal as a single terminal symbol.

Extended Chomsky Normal Form (ECNF)

- $X_i \rightarrow X_j X_k$
- $X_i \rightarrow w$
- $X_i \rightarrow X_j$

Derivation

A **derivation** of a string of terminal symbols ω is a sequence of string rewrites over V.

$$S = \phi_0 \Rightarrow \ldots \Rightarrow \phi_M = \omega$$

A derivation is called **leftmost derivation** iff in each step, the leftmost nonterminal symbol of the sentential form is rewritten.

For a leftmost derivation we have

$$\begin{aligned}
\phi_m &= \alpha X \gamma \\
\alpha &\in V_T^* \\
X &\in V_N \\
\gamma &\in V^* \\
X \rightarrow \beta &\in R \\
\phi_{m+1} &= \alpha \beta \gamma
\end{aligned}$$

Trees

- A tree τ is a **directed acyclic graph.**
- There is a function ℓ that maps the nodes of the tree into some set of symbols, the so-called **labelling function.**
- The arcs in the tree depict the **immediate-dominance relation** $\mathcal{ID}$. The immediate-dominance relation is intransitive and irreflexive. Its transitive closure is called the **dominance relation,** denoted $\mathcal{D}$. This relation is transitive and antisymmetric, and thus reflexive.
- If n immediately dominates n', then n is called the **mother** of n' and n' is called a **daughter** of n.
- The **precedence relation** $\mathcal{P}$ is the partial order indicated by the horizontal position of each node. This relation is transitive and asymmetric (and thus irreflexive).
- Arcs in the tree are not allowed to cross. This is called **nontangling condition.**
- For any two distinct nodes n and n', they are related either by $\mathcal{D}$ or $\mathcal{P}$.
- A tree has a single **root**. The root is the unique minimal element w.r.t. $\mathcal{D}$.
- The maximal elements w.r.t. $\mathcal{D}$ are called the **leaves** of the tree.
- The leaves must be ordered by $\mathcal{P}$. The ordered sequence of the labels of the leaves is called the **yield** of the tree.
- The non-maximal elements w.r.t. $\mathcal{D}$ are called **internal nodes.**

Parse Trees

A parse tree (or derivation tree) of any string of terminal symbols ω generated by a context-free grammar G must obey the following:

- The root of the tree is labelled with the axiom S.
- All leaves of the tree are labelled with elements in V_T.
  More specificly, the yield of the tree must be ω.
- All internal nodes are labelled with elements in V_N.
- If there is a node labelled X in the parse tree that immediately dominates nodes $n_1, \ldots, n_K$ labelled $X_1, \ldots, X_K$ (where n_i precedes n_j for $i < j$, i.e., $\mathcal{P}(n_i, n_j)$), then there is a production in G of the form $X \rightarrow X_1, \ldots, X_K$.

Parse Trees

$T(G)$ denotes the set of parse trees generated by the grammar G and is defined as $\{\tau : \mathcal{Y}(\tau) \in L(G)\}$.

A grammar G is said to be *finitely ambiguous* iff there is only a finite number of parse trees for any finite string in $L(G)$, i.e., if

$$\forall \omega \in L(G) \quad |\omega| < \infty \rightarrow |\{\tau : \mathcal{Y}(\tau) = \omega\}| < \infty$$

This is equivalent to requiring that no (nonterminal) symbol can be rewritten as itself in one or more rewrite step, i.e., that $X \Rightarrow^+ X$ is impossible.

Usually it is assumed that the grammars are finitely ambiguous.

Even so, the number of parse trees in general grows exponetially in the string length.

Partial Parse Trees

A **partial parse tree** is a parse tree where

- the root does not need to be labelled with the axiom S and
- the leaves are not necessarily labelled with terminals.

Partial parse trees can be combined through tree substitution.

We will here define leftmost tree substitution.

Leftmost tree substitution:
Let τ and τ' be partial parse trees.
Let $\tau \circ \tau'$ denote τ extended by identifying the leftmost leaf of τ labelled with a non-terminal symbol with the root of τ'.
The selected leaf node of τ is called the $substitution\ site$.
The label of the substitution site must match the label of the root of τ'.

Stochastic Context-free Grammars SCFGs

A stochastic context-free grammar is thus a quintuple $\langle V_N, V_T, S, R, P\rangle$ where:

V_N	is a finite set of nonterminal symbols.
V_T	is a finite set of terminal symbols. Again V denotes $V_N \cup V_T$.
$S \in V_N$	is a distinguished start symbol, or axiom.
R	is a finite set of productions $X \rightarrow \beta$ where $X \in V_N$ and $\beta \in V^*$.
P	is a function from R to $[0,1]$ such that:
	$\forall X \in V_N \quad \sum_{\beta \in V^*} P(X \rightarrow \beta) = 1$

Example of a SCFG

A simple example of a SCFG:

$$
\begin{array}{lcll}
S & \rightarrow & NP\ VP & (1.0) \\
VP & \rightarrow & V & (0.5) \\
VP & \rightarrow & VP\ PP & (0.5) \\
NP & \rightarrow & John & (1.0) \\
V & \rightarrow & sleeps & (1.0) \\
PP & \rightarrow & outside & (1.0)
\end{array}
$$

Derivations as Stochastic Processes

Derivations: left-most derivation: at each step the leftmost nonterminal is expanded.

The probability of a tree can be formulated as

$$P(t) = \sum_{\{d: d \text{ is a derivation of } t\}} P(d)$$

The probability of a derivation can be formulated as

$$P(d) = P(S \rightarrow^{r_1} \alpha_1 \rightarrow^{r_2} \alpha_2 \rightarrow^{r_3} \cdots \rightarrow^{r_m} \alpha_m = s) = \prod_{i=1}^{m} P(r_i | r_1 \cdots r_{i-1})$$

The history of a parse:

$$h_i = (r_1 \cdots r_{i-1})$$

The use of derivation histories is the key characteristic of so-called *History-based Grammars*.

Derivation histories are usually modelled by decision trees.

String Probabilities

String probabilities can be defined in terms of tree probabilities
and
tree probabilities can be defined in terms of derivation probabilities

String probabilities can also be dirctly defined in terms of derivation probabilities, i.e.,

- The probability of a string is defined as the sum of the probabilities of the parse trees that have this string as a yield.
- The probability of a parse tree is defined as the probability of the corresponding leftmost derivation.

Weakening the Independence Assumption of SCFGs

- Recall: CFGs and also SCFGs do not take contexts (of utterances) into account.
- In NL, however, contextual info is relevant for disambiguation.
- SCFGs do not capture lexical dependencies.
- SCFGs do not capture structural context.

One way to take structural contexts into account are History-based Grammars.

A means to model lexical phenomena in NL are Stochastic Dependency Grammars.

Dependency Grammar

- Classical Dependency Grammar: Tesnière 1959
- Syntactic structure is determined on the basis of dependency (head-modifier) relations between words.
- Sentences and phrases are thus represented by the set of dependency relations that hold between their lexical elements.
- Dependency structures can be derived from context-free grammars by mapping trees into dependency structures, i.e.,
- A head-daughter is defined for each phrase and its lexical head is perculated up the tree.
- The other daughters of the phrase are considered as modifiers.
- A dependency relation for each head-modifier pair can be defined, for example, by a triple representing the modifier non-terminal, the mother non-terminal and the head non-terminal. $< NP, S, VP >$ for instance expresses a subject-verb dependency.

A Stochastic Dependency-based Model

cf. Collins 1997

The following information is used by the model

- the lexical head, and
- the argument structure defined by the head.

The probability of a parse tree T for a sentence S is defined by the product of the probabilities of its productions.

Productions are defined by the conditional probability of the right-hand sides RHS_i given the left-hand sides LHS_i.

Thus we have

$$P(T,S) = \prod_{i=1}^{n} P(RHS_i | LHS_i)$$

Collins 97

Right-hand sides RHSs are further decomposed into

- The probability P_H that a head category H is generated conditioned on a parent category P and a head word h, and
- the probabilities of the left and right contexts of $H(h)$ under P.

Collins 97

The contexts are defined by

- the probabilities of the left and right constituents (P_l, P_r), and
- the probability of the left and right complements (P_{LC}, P_{RC}).

Collins 97

Constituents are calculated by means of

- the probability of the head category (L_i or R_i) and
- the head word (l_i or r_i) conditioned on the parent category P,
- the parent's head category H and the head word h,
- a distance measure d between the head and the edge of the context constituent, and
- a set representing the left or the right arguments (LC, LR) of $H(h)$.
- The probability of left and right arguments is also conditioned by P, H and h.

Collins 97

Thus we have the following formula:

$$\prod_{i=1}^{n} P_H(H|P,h) * P_l(L_i, l_i|P,H,h,d_r,LC) * P_r(R_i, r_i|P,H,h,d_r,RC) * P_{LC}(LC|P,H,h) * P_{RC}(RC|P,H,h)$$