Natural Language Processing
CSCI 4152/6509 — Lecture 19
DCG and PCFG Grammars

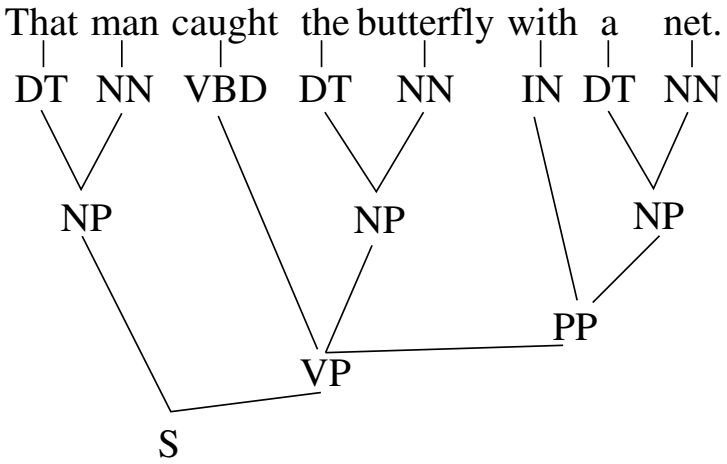
Instructors: Vlado Keselj
Time and date: 16:05 – 17:25, 20-Nov-2024
Location: Carleton Tupper Building Theatre C

Previous Lecture

- Stacked and bidirectional RNN
- LSTM, self-attention, transformers

  Part IV: Parsing (Syntactic Processing)
- Prolog introduction
  - unification and backtracking
  - variables, lists; examples: factorial, member
- Natural language syntax:
  - phrase structure, clauses, sentences
  - Parsing, parse tree examples
- Context-Free Grammars review:
  - formal definition
  - inducing a grammar from parse trees
  - derivations, and other notions

Bracket Representation of a Parse Tree



Bracket Representation of a Parse Tree

```
(S (NP (DT That)
       (NN man))
   (VP (VBD caught)
       (NP (DT the)
           (NN butterfly))
       (PP (IN with)
           (NP (DT a)
               (NN net)
) ) ) ) )
```

Some Notes on CFGs

- Left-hand side (lhs) and right-had side (rhs) of a production

$$\underbrace{S}_{lhs} \rightarrow \underbrace{NP\ VP}_{rhs}$$

- Empty rule (epsilon rule, epsilon production): $V \rightarrow \epsilon$
- Unit production: $A \rightarrow B$, where A and B are non-terminals
- Notational variations:
  - use of '|': $P \rightarrow N \mid A\ P$, instead of $P \rightarrow N,\ \ P \rightarrow A\ P$
  - BNF notation: $P ::= N \mid A\ P$
  - use of word 'opt': $NP\ ::=\ DT\ NN\ PP_{opt}$
  - or Kleene star: $NP\ ::=\ DT\ NN\ PP^*$

Using Prolog to Parse NL

Example: Consider a simple CFG to parse the following two sentences: "the dog runs" and "the dogs run"
The grammar is:

```
S  -> NP VP          N  -> dog
NP -> D N            N  -> dogs
D  -> the            VP -> run
                     VP -> runs
How to parse: the dog runs
```

Using Difference Lists: Idea

Consider rule: `S -> NP VP` and sentence [the,dog,runs]

Using Difference Lists to Parse CFG

The problem of parsing using this grammar can be expressed in the following way in Prolog:

```
s(S,R)  :- np(S,I), vp(I, R).
np(S,R) :- d(S,I), n(I,R).
d([the|R], R).
n([dog|R], R).
n([dogs|R], R).
vp([run|R], R).
vp([runs|R], R).
```

Parsing using Difference Lists

Save this in file `parse.prolog`. On Prolog prompt we type: `?- ['parse.prolog'].`

```
% parse.prolog compiled 0.00 sec, 1,888 bytes

Yes
?- s([the,dog,runs],[]).

Yes
?- s([runs,the,dog],[]).

No
```

Basic Definite Clause Grammar (DCG)

- DCG — Prolog built-in mechanism for parsing

Example

```
s  --> np, vp.
np --> d, n.
d  --> [the].
n  --> [dog].
n  --> [dogs].
vp --> [run].
vp --> [runs].
```

Building a Parse Tree

A parse tree can be built in the following way:

```
s(s(Tn,Tv))    --> np(Tn), vp(Tv).
np(np(Td,Tn)) --> d(Td), n(Tn).
d(d(the))      --> [the].
n(n(dog))      --> [dog].
n(n(dogs))     --> [dogs].
vp(vp(run))    --> [run].
vp(vp(runs))   --> [runs].
```

At Prolog prompt we type and obtain:

```
?- s(X, [the, dog, runs], []).
 X = s(np(d(the),n(dog)),vp(runs));
```

Handling Agreement

```
s(s(Tn,Tv))        --> np(Tn,A), vp(Tv,A).
np(np(Td,Tn),A)    --> d(Td), n(Tn,A).
d(d(the))          --> [the].
n(n(dog),sg)       --> [dog].
n(n(dogs),pl)      --> [dogs].
vp(vp(run),pl)     --> [run].
vp(vp(runs),sg)    --> [runs].
```

This grammar will accept sentences "the dog runs" and "the dogs run" but not "the dog run" and "the dogs runs". Other phenomena can be modeled in a similar fashion.

Embedded Code

We can embed additional Prolog code using braces, e.g.:

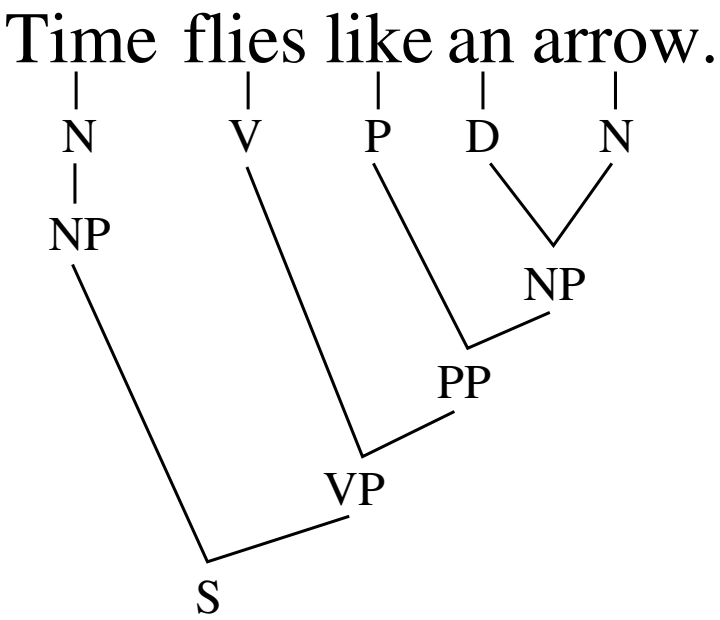
```
s(T)    --> np(Tn), vp(Tv), {T = s(Tn,Tv)}.
```

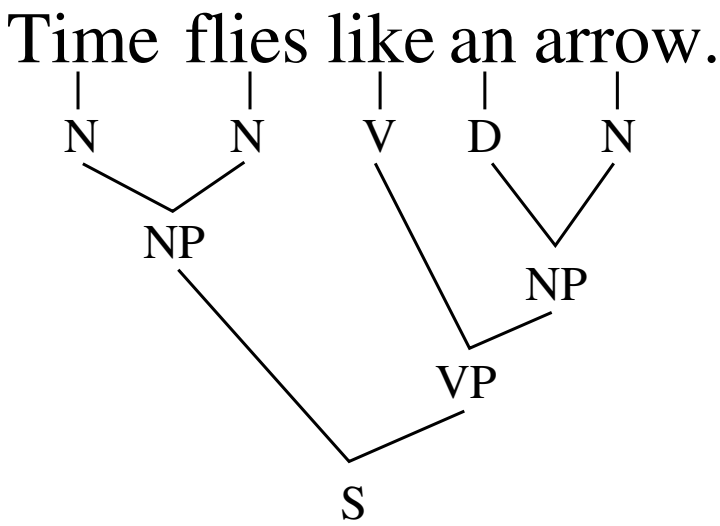
and so on, is another way of building the parse tree.

Probabilistic Context-Free Grammar (PCFG)

- Reading: Chapters 13 and 14
- also known as Stochastic Context-Free Grammar (SCFG)
- Handles ambiguous trees using a probabilistic model

Ambiguity Example





S	→	NP VP	VP	→	V NP	N	→	time	V	→	like
NP	→	N	VP	→	V PP	N	→	arrow	V	→	flies
NP	→	N N	PP	→	P NP	N	→	flies	P	→	like
NP	→	D N				D	→	an			

PCFG as a Probabilistic Model

- A generative model based on probabilistic derivation, for example:

$$\mathrm{S} \Rightarrow \mathrm{NP\ VP} \Rightarrow \mathrm{D\ N\ VP} \Rightarrow \ldots$$

- Each step is probabilistic use of one production

Probabilistic Context-Free Grammar Example

S	$\rightarrow$	NP VP	/1	VP	$\rightarrow$	V NP	/.5	N	$\rightarrow$	time	/.5
NP	$\rightarrow$	N	/.4	VP	$\rightarrow$	V PP	/.5	N	$\rightarrow$	arrow	/.3
NP	$\rightarrow$	N N	/.2	PP	$\rightarrow$	P NP	/1	N	$\rightarrow$	flies	/.2
NP	$\rightarrow$	D N	/.4					D	$\rightarrow$	an	/1
V	$\rightarrow$	like	/.3								
V	$\rightarrow$	flies	/.7								
P	$\rightarrow$	like	/1								

- The following condition must be satisfied for each nonterminal N:

$$\sum_{i=1}^{n} \mathrm{P}(N \rightarrow \alpha_i) = 1$$

Computational Tasks for PCFG Model

- Evaluation

$$P(\text{tree}) = ?$$

- Generation
- Learning
- Inference
  - Marginalization

$$\mathrm{P}(\text{sentence}) = ?$$

  - Conditioning

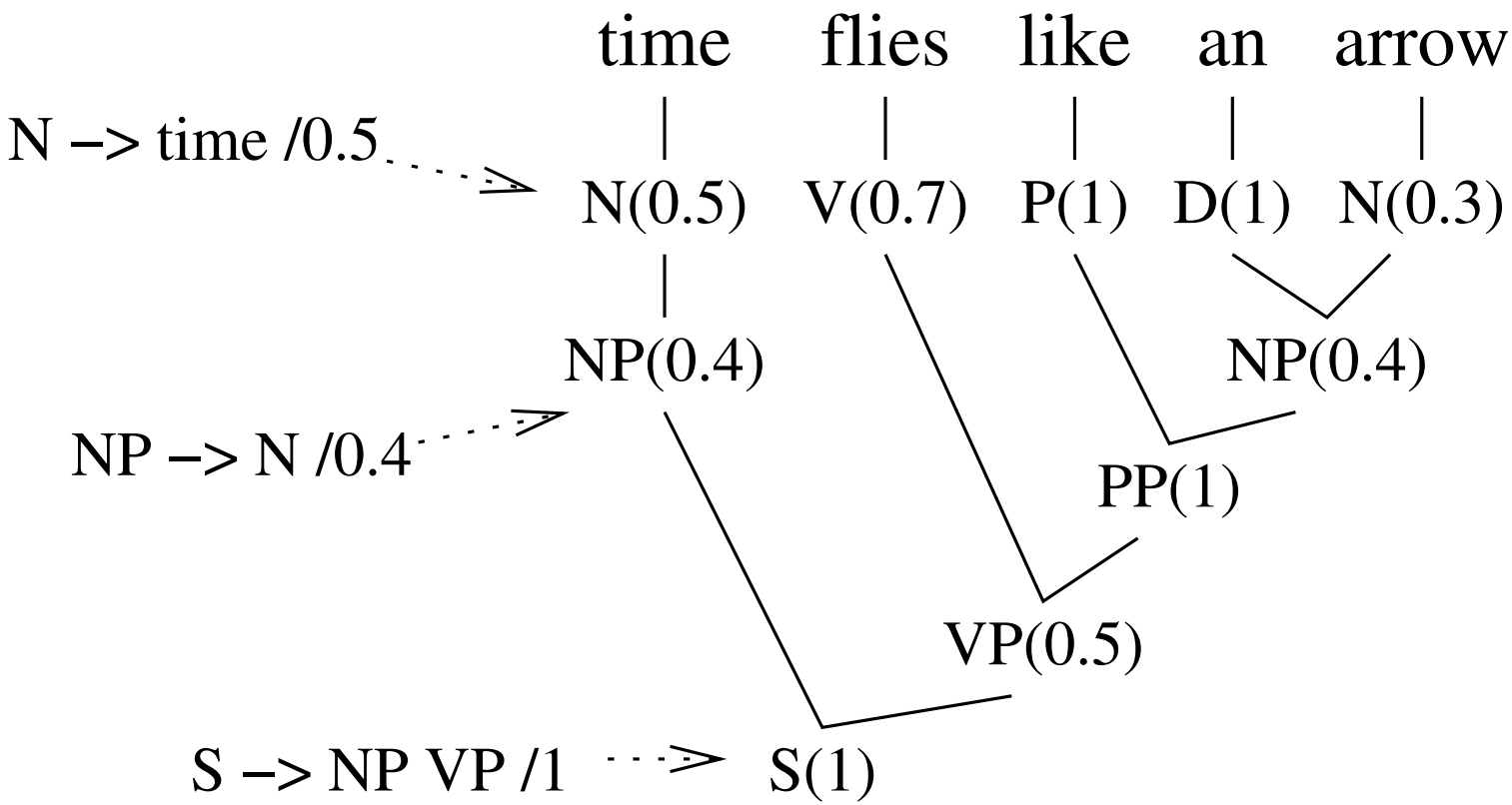
$$\mathrm{P}(\text{tree}|\text{sentence}) = ?$$

  - Completion

$$\arg\max_{\text{tree}} \mathrm{P}(\text{tree}|\text{sentence})$$

Evaluation example: time flies like an arrow (1st meaning)

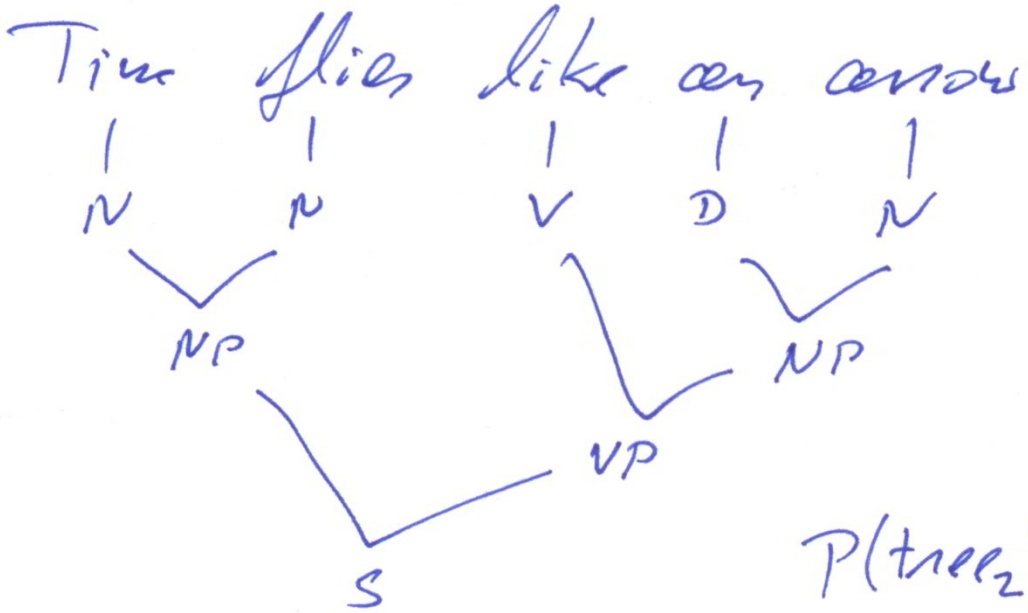
Evaluation



$$P(\text{tree}) = 0.5 \times 0.7 \times 1 \times 1 \times 0.3 \times 0.4 \times 0.4 \times 1 \times 0.5 \times 1 = 0.0084$$

Evaluation example: time flies like an arrow (2nd meaning)

Similarly



$P(tree_2) = 0.00036$

Generation (sampling)

S => NP VP => N VP => flies VP => ...

S → NP VP / 1

NP → N / 0.3
NP → N N / 0.2
NP → D N / 0.4

N → time / 0.5
N → arrow / 0.3
N → flies / 0.2

- choose rule randomly according to the given distribution

Question: Is the process going to stop?

A: Stops with probability 1 if the grammar is proper.

Good News: A grammar learned from a corpus is always proper.

Learning and Inference

Expressing PCFGs in DCGs

Let us consider the previous example of a PCFG:

S	$\rightarrow$	NP VP	/1	VP	$\rightarrow$	V NP	/.5	N	$\rightarrow$	time	/.5
NP	$\rightarrow$	N	/.4	VP	$\rightarrow$	V PP	/.5	N	$\rightarrow$	arrow	/.3
NP	$\rightarrow$	N N	/.2	PP	$\rightarrow$	P NP	/1	N	$\rightarrow$	flies	/.2
NP	$\rightarrow$	D N	/.4					D	$\rightarrow$	an	/1
V	$\rightarrow$	like	/.3								
V	$\rightarrow$	flies	/.7								
P	$\rightarrow$	like	/1								

The probabilities can be calculated as an additional argument:

```
s(T,P) --> np(T1,P1), vp(T2,P2),
           {T = s(T1,T2), P is P1 * P2 * 1}.
np(T,P) --> n(T1,P1), {T = n(T1), P is P1 * 0.4}.
```

and so on.

Full PCFG Expressed in DCG

```
s(s(Tn,Tv),P) --> np(Tn,P1), vp(Tv,P2), {P is P1 * P2}.
np(np(T),P) --> n(T,P1), {P is P1 * 0.4}.
np(np(T1,T2),P) --> n(T1,P1), n(T2,P2),
                                {P is P1 * P2 * 0.2}.
np(np(Td,Tn),P) --> d(Td,P1), n(Tn,P2),
                                {P is P1 * P2 * 0.4}.
v(v(like), 0.3) --> [like].
v(v(flies), 0.7) --> [flies].
p(p(like), 1.0) --> [like].
vp(vp(Tv,Tn), P) --> v(Tv, P1), np(Tn, P2),
                                {P is P1 * P2 * 0.5}.
vp(vp(Tv,Tp), P) --> v(Tv, P1), pp(Tp, P2),
                                {P is P1 * P2 * 0.5}.
pp(pp(Tp,Tn), P) --> p(Tp, P1), np(Tn, P2),
                                  {P is P1 * P2}.
n(n(time), 0.5) --> [time].
n(n(arrow), 0.3) --> [arrow].                    ...
```

Example Run in Prolog Interpreter

?- `s(T,P,[time,flies,like,an,arrow],[]).`
the interpreter would reply with: `T = s(np(n(time)),`
`vp(v(flies), pp(p(like), np(d(an), n(arrow)))))`
`P = 0.0084`
and after typing ; (semi-colon), we get: `T = s(np(n(time), n(flies)),`
`vp(v(like), np(d(an), n(arrow))))`
`P = 0.00036`
After typing second ';', the interpreter reports 'No' since there are no more parse trees.

Typical Phrase Structure Rules in English

- We will cover some typical phrase structure rules
- Specific to English but also generalizable to other languages
- **Not** *all rules are covered*, but the general principles should be adopted

Typical Sentence Rules (S)

```
S -> NP VP              Declarative sentences, e.g.:
                   I want a flight from Halifax to Chicago.

S -> VP                 Imperative sentences, e.g.:
                        Show the lowest fare.

S -> Aux NP VP          Yes-no questions, e.g.:
                      Do any of these flights have stops?
              Can you give me some information for United?

S -> Wh-NP VP           Wh-subject questions, e.g.:
                        What airlines fly from Halifax?

S -> Wh-NP Aux NP VP    Wh-non-subject questions, e.g.:
                      What flights do you have on Tuesday?
```