Natural Language Processing and Machine Translation

Parsing, Sequence Labelling and Parts-of-Speech Tagging

Abhishek Koirala

M.Sc. in Informatics and Intelligent Systems Engineering

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Language Structure and Meaning

Our purpose is to know how meaning is mapped onto what language structures

- [***Thing*** The dog] is [***Place*** in the garden]
- [***Thing*** The dog] is [***Property*** fierce]
- [***Action*** [***Thing*** The dog] is chasing [***Thing*** the cat]]
- [***State*** [***Thing*** The dog] was sitting [***Place*** in the garden] [***Time*** yesterday]]

Word Categories: Traditional parts of speech

Noun	Names of things	boy, cat, truth
Verb	Action or state	become, hit
Pronoun	Used for noun	I, you, we
Adverb	Modifies V, Adj, Adv	sadly, very
Adjective	Modifies noun	happy, clever
Conjunction	Joins things	and, but, while
Preposition	Relation of N	to, from, into
Interjection	An outcry	ouch, oh, alas, psst

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Constituency

Group of words may behave as a single unit of phrase , called a **constituent**

Sentence have parts, some of which appear to have subparts. **Constituents** are those grouping of words that go together.

I hit the man with an axe.

I hit [the man with an axe].
I hit [the man] with an axe.

I talked to a man using a phone.

I talked to [a man using a phone].
I talked to [a man] using a phone.

Constituent Phrases

For constituents, we usually name them as phrases based on the word that heads the constituent:

the man from Amherst	is a Noun Phrase (NP) because the head man is a noun
extremely clever	is an Adjective Phrase (AP) because the head clever is an adjective
down the river	is a Prepositional Phrase (PP) because the head down is a preposition
killed the rabbit	is a Verb Phrase (VP) because the head killed is a verb

Note that a word is a constituent (a little one). Sometimes words also act as phrases. In:

Joe grew potatoes.
Joe and *potatoes* are both nouns and noun phrases.

Compare with:

The man from Amherst grew beautiful russet potatoes.

We say *Joe* counts as a noun phrase because it appears in a place that a larger noun phrase could have been.

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Constituent Phrases

1. They appear in similar environments (before a verb)
   Kermit the frog comes on stage
   They come to Massachusetts every summer
   December twenty-sixth comes after Christmas
   The reason he is running for president comes out only now.
   But not each individual word in the consituent
   **The comes our... *is comes out... *for comes out...*

2. The constituent can be placed in a number of different locations
   Consituent = Prepositional phrase: *On December twenty-sixth*
   On December twenty-sixth I'd like to fly to Florida.
   I'd like to fly on December twenty-sixth to Florida.
   I'd like to fly to Florida on December twenty-sixth.
   But not split apart
   **On December I'd like to fly twenty-sixth to Florida.*
   **On I'd like to fly December twenty-sixth to Florida.*

The most common way of modelling constituency is using Context Free Grammars (CFG)

Context Free Grammar (CFG)

$G = \langle T, N, S, R \rangle$

- T is set of terminals (lexicon)
- N is set of non-terminals For NLP, we usually distinguish out a set $P \subset N$ of *preterminals* which always rewrite as terminals.
- S is start symbol (one of the nonterminals)
- R is rules/productions of the form $X \to \gamma$, where X is a nonterminal and γ is a sequence of terminals and nonterminals (may be empty).
- A grammar G generates a language L.

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Context Free Grammar (CFG)

$G = \langle T, N, S, R \rangle$

$T = \{that,\ this,\ a,\ the,\ man,\ book,\ flight,\ meal,\ include,\ read,\ does\}$

$N = \{\text{S, NP, NOM, VP, Det, Noun, Verb, Aux}\}$

$S = S$

$R = \{$

S → NP VP
S → Aux NP VP
S → VP
NP → Det NOM
NOM → Noun
NOM → Noun NOM
VP → Verb
VP → Verb NP

Det → *that* | *this* | *a* | *the*
Noun → *book* | *flight* | *meal* | *man*
Verb → *book* | *include* | *read*
Aux → *does*

$\}$

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

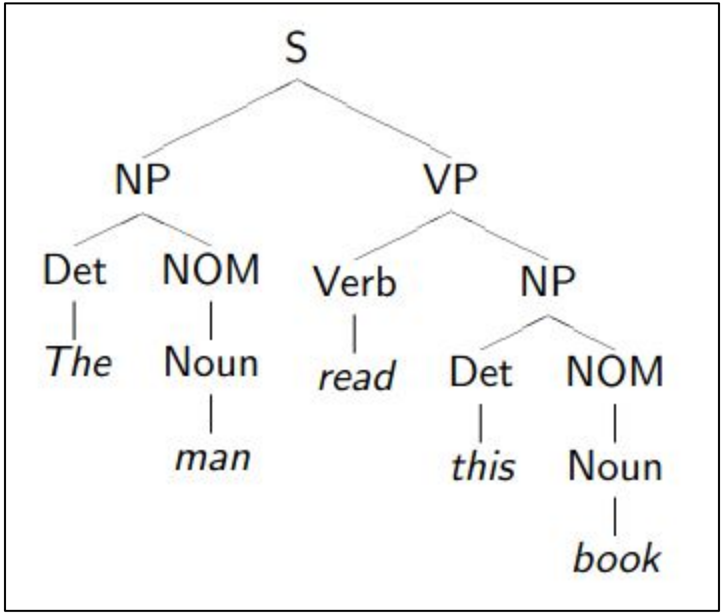
Parsing

- Our purpose is to run the grammar backwards to find the structure
- Parsing can be viewed as a search problem
- We search through the legal rewritings of the grammar. We want to find all structures matching an input string of words
- Two types:
  - Top - down
  - Bottom - up

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Context Free Grammar (CFG)

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	



Top-Down Parsing

Top down parsing is goal directed

- A top-down parser starts with a list of constituents to be built
- It rewrites the goals in the goal list by matching one against the LHS of the grammar rules
- and expanding it with the RHS
- attempting to match the sentence to be derived

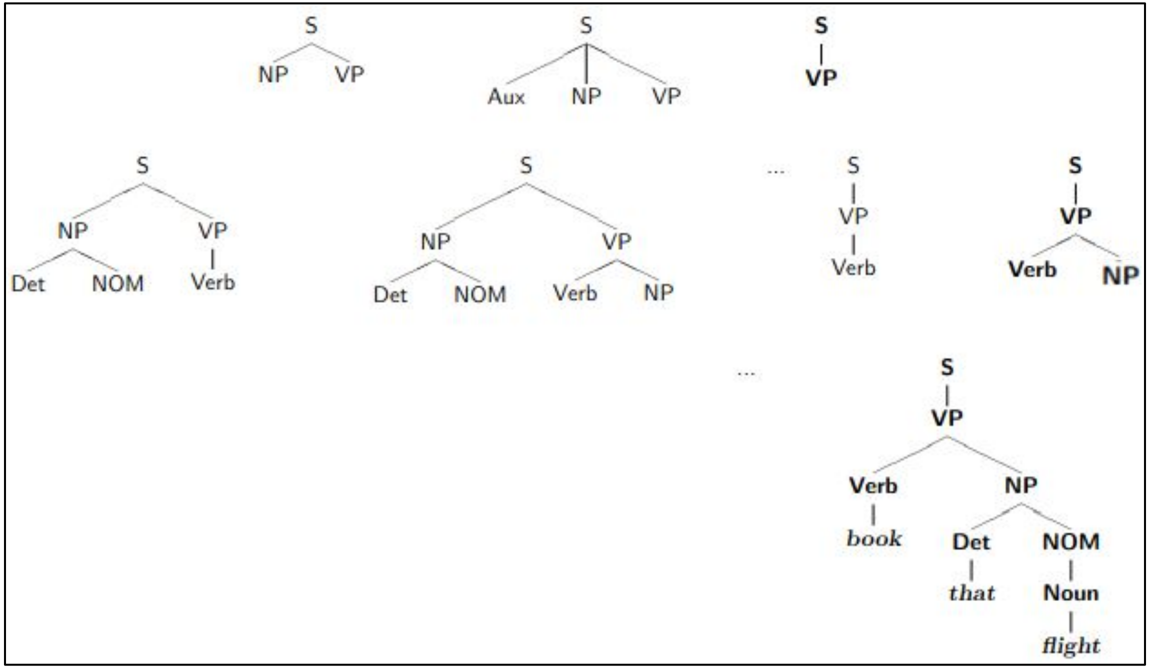
If a goal can be rewritten in several ways, then there is a choice of which rule to apply (search problem)

Top-Down Parsing

book that flight

S → NP VP
S → Aux NP VP
S → VP
NP → Det NOM
NOM → Noun
NOM → Noun NOM
VP → Verb
VP → Verb NP

Det → *that* | *this* | *a* | *the*
Noun → *book* | *flight* | *meal* | *man*
Verb → *book* | *include* | *read*
Aux → *does*



THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Button-up Parsing

Bottom up parsing is goal directed

- Initial goal list of a bottom-up parser is the string to be parsed
- If a sequence in the goal list matches the RHS of a rule, then this sequence may be replaced by the LHS of the rule
- Parsing is finished when the goal list contains just the start symbol

If the RHS of several rules match the goal list, then there is a choice of which rule to apply (search problem)

The standard presentation is as shift-reduce parsing.

Shift-Reduce Parser

Start with the sentence to be parsed in an input buffer

- a “shift” action corresponds to pushing hte next input symbol from the buffer onto the stack
- a “reduce” action occurs when we have a rule’s RHS on top of the stack. To perform the reduction, we pop the rule’s RHS off the stack and replace with the terminal on the LHS of the corresponding rul

If you end up with only the START symbol on the stac, then success !!

- If you don’t, and no “shift” or “reduce” actions are possible, backtrack

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Bottom-up parsing example

S → NP VP	Det → *that* | *this* | *a* | *the*
S → Aux NP VP	Noun → *book* | *flight* | *meal* | *man*
S → VP	Verb → *book* | *include* | *read*
NP → Det NOM	Aux → *does*
NOM → Noun	
NOM → Noun NOM	
VP → Verb	
VP → Verb NP	

Book that flight.

How do you parse this using shift reduce approach ?

CKY Parser

- (Cocke - Kasami - Younger) CKY Parser
- Bottom-up parser
- Dynamic programming approach
- Presumes a CFG in Chomsky Normal Form

Rules are all either **A → B C** or **A → a**
(with **A,B,C** nonterminals and **a** a terminal)

CKY Parser

function CKY (word w, grammar P) returns table

for i ← from 1 to LENGTH(w) do

  table[i-1, i] ← {A | A → w_i ∈ P }

for j ← from 2 to LENGTH(w) do

  for i ← from j-2 down to 0 do

    for k ← i + 1 to j – 1 do

      table[i,j] ← table[i,j] U {A | A → BC ∈ P,

        B ∈ table[i,k], C ∈ table[k,j] }

If the start symbol *S* ∈ table[0,n] then w ∈ L(G)

CKY Parser

Grammar rules

S → NP VP
NP → Det N
VP → V NP
V → includes
Det → the
Det → a
N → meal
N → flight

0 The 1 flight 2 includes 3 a 4 meal 5

	1	2	3	4	5
0					
1					
2					
3					
4					

CKY Parser

Grammar rules

- S → NP VP
- NP → Det N
- VP → V NP
- V → includes
- Det → the
- Det → a
- N → meal
- N → flight

0 The 1 flight 2 includes 3 a 4 meal 5

	1	2	3	4	5
0	DET	NP			S
1		N			
2			V		VP
3				DET	NP
4					N

CKY Parser (PCFG)

$_0$ The $_1$ flight $_2$ includes $_3$ a $_4$ meal $_5$

Grammar rules

S → NP VP [0.8]
NP → Det N [0.3]
VP → V NP [0.2]
V → includes [0.05]
Det → the [0.4]
Det → a [0.4]
N → meal [0.01]
N → flight [0.02]

	1	2	3	4	5
0	DET	NP			S
1		N			
2			V		VP
3				DET	NP
4					N

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

Earley Parser

General Principles:

- A clever hybrid *Bottom-Up* and *Top-Down* approach
- *Bottom-Up* parsing completely guided by *Top-Down* predictions
- Maintains sets of “dotted” grammar rules that:
  - Reflect what the parser has “seen” so far
  - Explicitly predict the rules and constituents that will combine into a complete parse
- Similar to Chart Parsing - partial analyses can be shared
- Time Complexity $O(n^3)$, but better on particular sub-classes
- Developed prior to Chart Parsing, first efficient parsing algorithm for general context-free grammars.

Earley Parser

Three Main Operations:

- **Predictor:** If state $[A \to X_1 ... \bullet C ... X_m, j] \in S_i$ then for every rule of the form $C \to Y_1 ... Y_k$, add to S_i the state $[C \to \bullet Y_1 ... Y_k, i]$
- **Completer:** If state $[A \to X_1 ... X_m \bullet, j] \in S_i$ then for every state in S_j of form $[B \to X_1 ... \bullet A ... X_k, l]$, add to S_i the state $[B \to X_1 ... A \bullet ... X_k, l]$
- **Scanner:** If state $[A \to X_1 ... \bullet a ... X_m, j] \in S_i$ and the next input word is $x_{i+1} = a$, then add to S_{i+1} the state $[A \to X_1 ... a \bullet ... X_m, j]$

THAPATHALI CAMPUS
INSTITUTES OF ENGINEERING

The Earley Recognition Algorithm

The Main Algorithm: parsing input $x = x_1 ... x_n$

1. $S_0 = \{[S' \rightarrow \bullet S\ \$, 0]\}$
2. For $0 \leq i \leq n$ do:
   Process each item $s \in S_i$ in order by applying to it the *single* applicable operation among:
   (a) Predictor (adds new items to S_i)
   (b) Completer (adds new items to S_i)
   (c) Scanner (adds new items to S_{i+1})
3. If $S_{i+1} = \phi$, *Reject* the input
4. If $i = n$ and $S_{n+1} = \{[S' \rightarrow S\ \$\bullet, 0]\}$ then *Accept* the input

The Earley Recognition Algorithm

The Main Algorithm: parsing input $x = x_1 \ldots x_n$

1. $S_0 = \{[S' \rightarrow \bullet S \ \$, 0]\}$
2. For $0 \leq i \leq n$ do:
   Process each item $s \in S_i$ in order by applying to it the *single* applicable operation among:
   (a) Predictor (adds new items to S_i)
   (b) Completer (adds new items to S_i)
   (c) Scanner (adds new items to S_{i+1})
3. If $S_{i+1} = \phi$, *Reject* the input
4. If $i = n$ and $S_{n+1} = \{[S' \rightarrow S \ \$\bullet, 0]\}$ then *Accept* the input


рахмат
danke
謝謝
ngiyabonga
شكراً جزيلاً
spas
Баярлалаа
mersi
kia ora
barka
welalin
tack
teşekkür ederim
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
kiitos
dankie
thank you
gracias
хвала
nanni
nandri
dhanyavad
asante
manana
mauruuru
köszönöm
aciū
obrigada
bedankt
bayarlalaa
gràcie
hvala
dankon
tenki
enkosi
akun
djiere dieuf
tau
mochchakkeram
murakoze
dziękuję
дякую
mamnun
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
chokrane
sobodi
dēkuji
taiku
sagolun
sukriya
kop khun krap
obrigado
mèsi
arigatô
takk
dakujem
trugarez
didi madloba
ありがとう
tanemirt
grazie
najis tuke
terima kasih
rahmet
diolch
dhanyavadagalu
shukriya
merce
мерси
kam sah hamnida
rahmat
ευχαριστώ
xièxie
merci
তোমাকে ধন্যবাদ
감사합니다