BAB V
PEMROSESAN BAHASA ALAMI
(NATURAL LANGUAGE PROCESSING/NLP)

5.1. Pengenalan NLP

NLP adalah mencoba untuk membuat komputer dapat mengerti perintah-perintah yang ditulis dalam standar bahasa manusia.

NLP tidak memperdulikan bagaimana sebuah kalimat dimasukkan ke komputer tetapi mencopy informasi dari kalimat tersebut.

a. Pendekatan-pendekatan pada NLP: Inti dari NLP adalah *PARSER*

Dimana PARSER tersebut membaca setiap kalimat, kata demi kata, untuk menentukan apa yang dimaksud.

PARSER terdiri dari 3 jenis:

1. *PARSER STATE-MACHINE*
2. *PARSER CONTEXT-FREE RECURSIVE-DESCENT*
3. *PARSER NOISE-DISPOSAL*

Hal hal yang bertentangan dengan pendekatan NLP -:

1. Bahwa sesungguhnya NLP menggunakan semua informasi dalam sebuah kalimat, hanya manusia yang dapat melakukan hal tersebut.
2. Mencoba memperbolehkan komputer menerima perintah bahasa alami, tetapi hanya mengcopy inti informasi pada perintah (*command*).

b. Batasan Bahasa

Aspek yang paling sulit dalam pembentukan sistem pengendali NLP adalah: Pengakomodasian kekompleksan dan kefleksibelan bahasa manusia dalam sistem.

Contoh Bentuk standard : **Subyek-Verb-Obyek**

Diasumsikan:

- *Adjective* mengawali *Noun*
- *Adverb* mengawali *Verb*
- Semua kalimat diakhiri dengan titik

Contoh : *The child runs to the house*

The large child runs quickly to the window

PARSER akan menentukan:

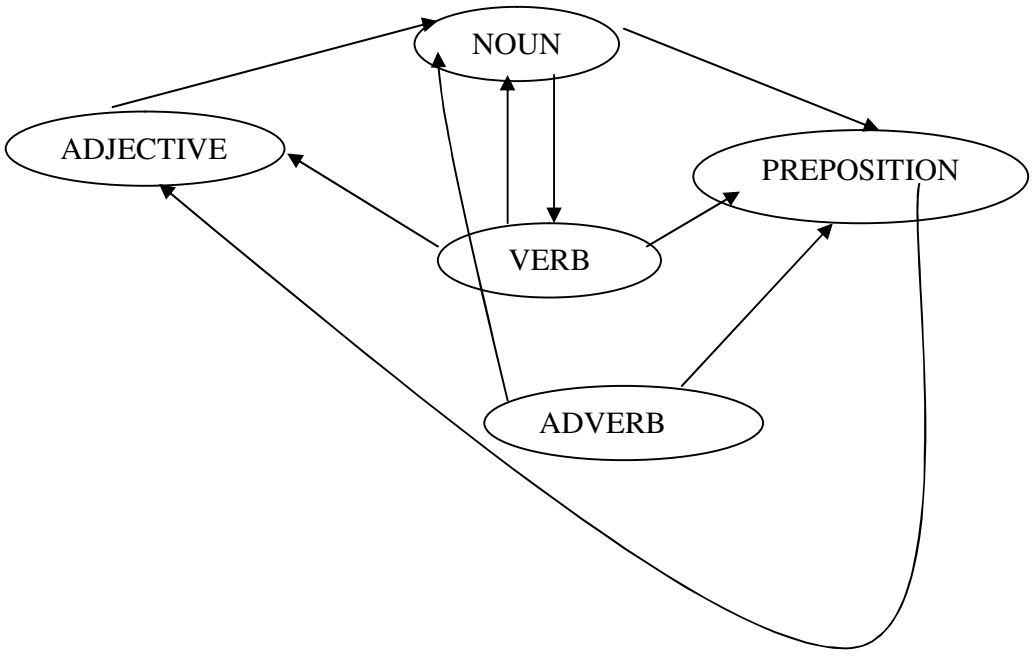
The child quickly runs to the house

5.2. Parser state-machine NLP

Parser state-machine menggunakan keadaan yang sesungguhnya dari kalimat untuk memprediksi tipe apa dari kata yang berlaku.

State-machine : *directed graph* yang menunjukkan transisi yang valid dari satu state ke yang lain.

Contoh: Grammar G1



Gambar 5.1. *State Machine Grammar* G1

Kegunaan *state-machine* :

a) dapat memotong atau memilah kalimat ke dalam komponen-komponennya.
b) menentukan apakah sebuah kalimat dibentuk dengan benar dalam batasan dari *grammar* G1

Database yang harus dibentuk sebelum implementasi:

i) Membentuk kosa kata(*vocabulary*) dari kata-kata yang dikenal ke sistem dengan mengikuti tipe yang ada.
ii) Menyimpan keadaan sesungguhnya dari kalimat.

Masalah yang paling buruk dengan *state-machine parser* adalah:

1. Kekompleksannya
   Contoh : Dalam grammar G1 dibutuhkan 14 clause yang terpisah untuk menunjukkan transisi keadaan.
2. Parser tidak mengetahui bagaimana mencapai suatu keadaan
   Contoh: Parser tidak dapat menghubungkan sebuah modifikasi phrase ke noun tertentu. Hal ini berarti tidak dapat memanggil parser state-machine untuk mendukung suatu informasi lain dari keadaan yang sesungguhnya.

Keuntungan *parser state-machine* : Ideal untuk aplikasi tertentu seperti: beberapa aplikasi database

5.3. Parser *Context-Free Recursive-Descent*

Contoh: sebuah kalimat adalah gabungan dari berbagai item dan item ini adalah gabungan dari item lain dan seterusnya sampai dipotong(dipilah) ke elemen-elemennya seperti Noun, Adjective, dan sebagainya.

Aturan-aturan yang ada pada setiap bagian yang telah dibentuk disebut: *Production rule* dari *grammar*.

Parser context-free menggunakan *production rule* untuk menganalisa sebuah kalimat.

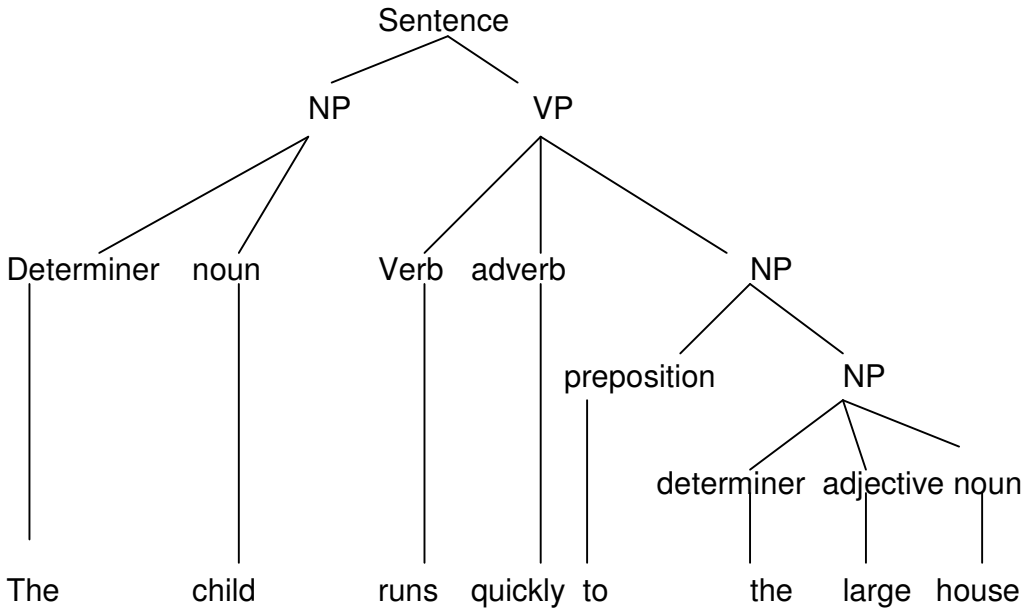
Production Rule untuk grammar G1:

Sentence	--> NP + VP
NP	--> determiner + noun
NP	--> determiner + adjective + noun
NP	--> preposition + NP
VP	--> Verb + NP
VP	--> Verb + adverb + NP
VP	--> Verb + adverb
VP	--> adverb

NP : Noun Phrase
VP : Verb Phrase

Noun Phrase: Definisi rekursif untuk proposisi clause

Verb Phrase: Rekursif tidak langsung karena *Verb Phrase* melibatkan *Noun Phrase* sebagai bagian dari definisinya.



Parser membentuk tipe *parse tree* yang disebut: '***context free***', sebab Tree bukan dasar dari konteks setiap elemen. Hal ini berarti bahwa aturan atau

rule akan bekerja untuk suatu *statement* yang menyerupai *grammar* G1 tanpa mengharapkan pada konteks setiap phrase.

Kegunaan *context free* selain untuk program-program AI NLP, juga untuk bahasa-bahasa komputer lainnya seperti:

- *PASCAL*
- *BASIC*
- *C*
- *MODULA-2*

Parser recursive-descent menggunakan:kumpulan rutin rekursif di mana menurunkannya melalui *production rule* sampai kalimat selesai ditelusuri seluruhnya.

Untuk membentuk *parser context-free recursive-descent* dibutuhkan beberapa *vocalbulary database* dan dukungan *predicate* untuk menyalin kata-kata dari sebuah kalimat sebagai *parser state-machine* yang digunakan.

Keuntungan *Parser Context-free recursive-descent*:

(1) Mudah diimplementasikan dalam turbo prolog.
(2) Dapat berkomunikasi dengan kalimat baik tingkatan kata dan phrase.
(3) Mengetahui di mana parser dalam kalimat pada setiap saat.

Kerugiannya: Tidak dapat menangani cara valid dalam jumlah besar di mana kalimat dalam suatu bahasa dibentuk.

5.4. Parser Noise-Disposal

Tipe parser ini sesungguhnya sangat umum dalam aplikasi tipe database, seperti : *Command processor*.

Contoh: sebuah database terdiri dari nama-nama perusahaan dan harga-harga stock.

Asumsi database menerima query seperti:

Lihatkan saya semua perusahaan dengan persediaan > 100

Lihatkan saya semua

Lihatkan saya xyz

Lihatkan saya satu dengan persediaan < 100

Tipe query: **Perintah<modifikasi><nama><operator><nilai>**

Perintah harus selalu ada, tetapi 4 elemen lainnya adalah optional. Namun demikian bila operator digunakan maka nilai harus digunakan.

Kerugian: tidak berguna untuk situasi tidak terbatas, karena didasari pada dua asumsi:

i. Kalimat mengikuti bentuk yang tegas
ii. Hanya beberapa *keyword* yang penting.

Keuntungan: sederhana untuk diimplementasikan dan mendapatkan informasi yang cepat.

5.5. *Natural Language Understanding*

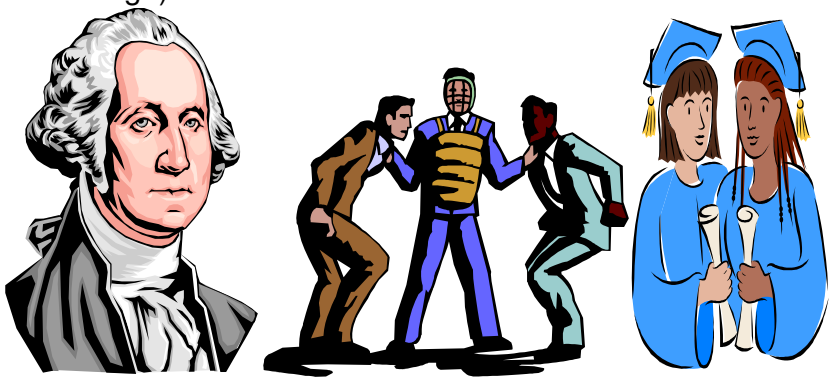
- Chomsky (*Psychologis*t) --> *Formal Language*
  S --> aSa

- *Machine Translation*
  English → Russian ≠ Russian → English

- Bagaimana struktur *grammar* daripada *sentence*?
  Bagaimana bahasa di organisis untuk mendapatkan artinya?

Conceptual Structure (Pengkodean Knowledge)

Conceptual Structure



string

1. object
2. relations
3. events

- *Object* : nama atau deskripsi atau *argument*

  :John :Black :Chair

- *Relation*: penghubung argument dan diatur

  (# loves :John :Mary)

  *argumen*t(object atau deskripsi)

  relation

  contoh lain: 'I want you to pick up the black'

  relation

  (# want :I :Rel1)
  (# pick up :you :Rel1)

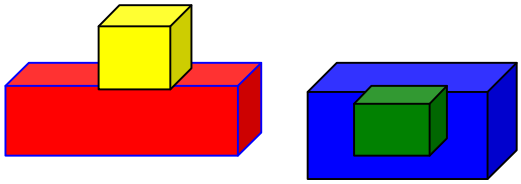
  nama relation

- *Events* : *Relation* berdasarkan waktu

  contoh: 'Kemarin John pergi ke Toko'

  (# pergi :John :Toko :Kemarin)

Contoh natural language understanding: “SHRDLU” ---> MIT

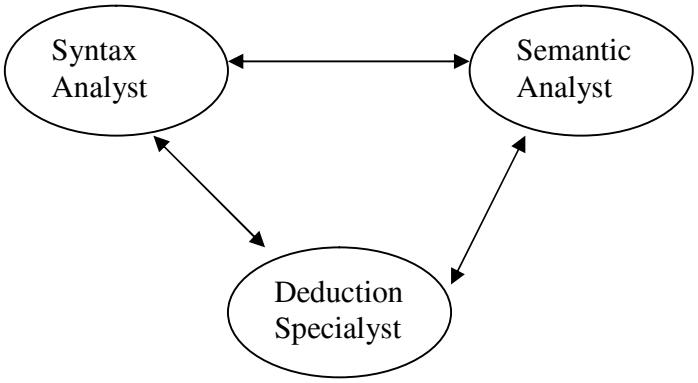


Word model :
1. (# support :B1 :B2)
2. (# yellow :B2)
3. (# Red :B1)
4. (# Contains :B3 :B4)
5. (# Blue :B3)
6. (# Green :B4)
7. (# Cube :B1)
8. (# Cube :B2)
9 (# Cube :B3)
10.(# Cube :B4)

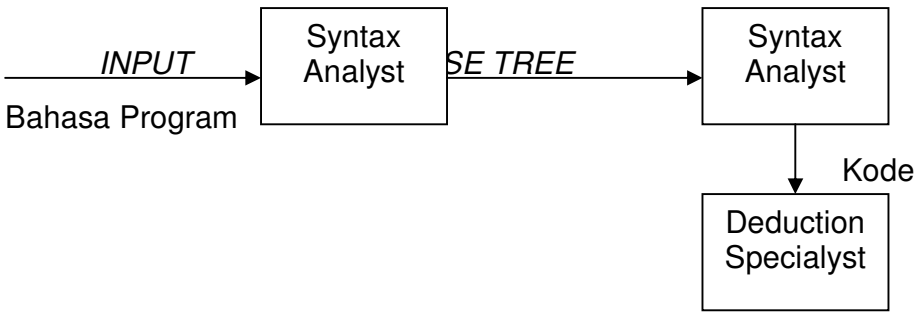
‘pick up the yellow block’

(# Grasp :B2) ;; hapus dan create model baru

Tiga aspek dasar dalam sistem *understanding*



Di *compiler* --> urutannya linear



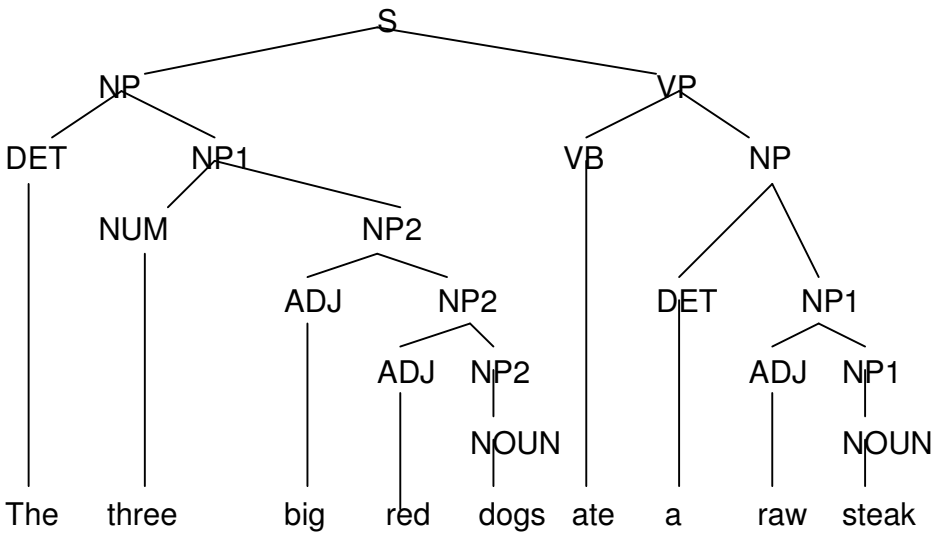
Bahasa program di input, lalu dicek atau *analyst word*nya ada/tidak di *dictionary*, sebelum dicek syntaknya, kemudian dicek *syntax*nya dengan melihat

*gramma*mya, hasilnya berupa *parse tree* atau *structure grammar*, setelah itu dianalysis arti atau makna katanya, kemudian direpresentasikan dalam kode yang sudah tahu maunya

Contoh:

Aturan(*Grammar)*		*Phrase(Group)*	
S	→ NG VERB NG	S	→ NP VP
NG	→ DET NOUN	NP	→ DET NP1
NG	→ DET ADJS NOUN	VP	→ VB NP
ADJS	→ADJ	NP1	→ NUM NP2
NG	→ PREP NG	NP2	→ ADJ NP2
NG	→ Q NOUN	NP2	→ NOUN
Q	→ NG	NP1	→ PREP NP2
NG	→ NOUN	NP	→ NP1
		NP1	→ ADJ NP1
		NP1	→ NOUN

Contoh: "*The three big red dogs ate a raw steak"*
"The graffis ate the apples and drank the vodka"



5.6. Proses Penterjemahan (*Transalation Process*)

Proses Penterjemahan dalam mengerti bahasa alami(*Natrural Language Understanding*), ada 4 tahap:

1. ***Lexical Analysis***: cek masing-masing word & *dictionary lock up*
2. ***Syntax Analysis***(Parsing): sesuai *grammar*
3. ***Semantic Analysis***: mengecek masing-masing arti kata
4. ***Discourse*** :
   - Melihat semua kalimat yang lain
   - Sifatnya kompleks
   - Mengecek arti kata secara keseluruhan
   - Hukum/sesuai daerahnya

Contoh: "**These Students had a good knowledge of LISP features**"

Lexical Analysis

Menghubungkan setiap kata dalam kalimat, dengan informasi tentang

- Kategori gramatik: Noun(NOUN), VERB(VB), Determiner(DET), Adjective(ADJ)
- Root(asal Kata) : have untuk "had", this untuk "these", student untuk "students"
- Tense atau form(bentuk) : present, past, ….
  Singular(SING), Plural (PLUR)

These (NOUN, this, Plur)	Students (NOUN, student, PLUR)	Had (VB, have, PAST)	A (DET, a, SING)	Good (ADJ, good)
Knowledge (NOUN, knowledge, sing)	Of (Prep)	LiSP (NOUN, LISP, Sing)	Features (Noun, feature, Plur)	

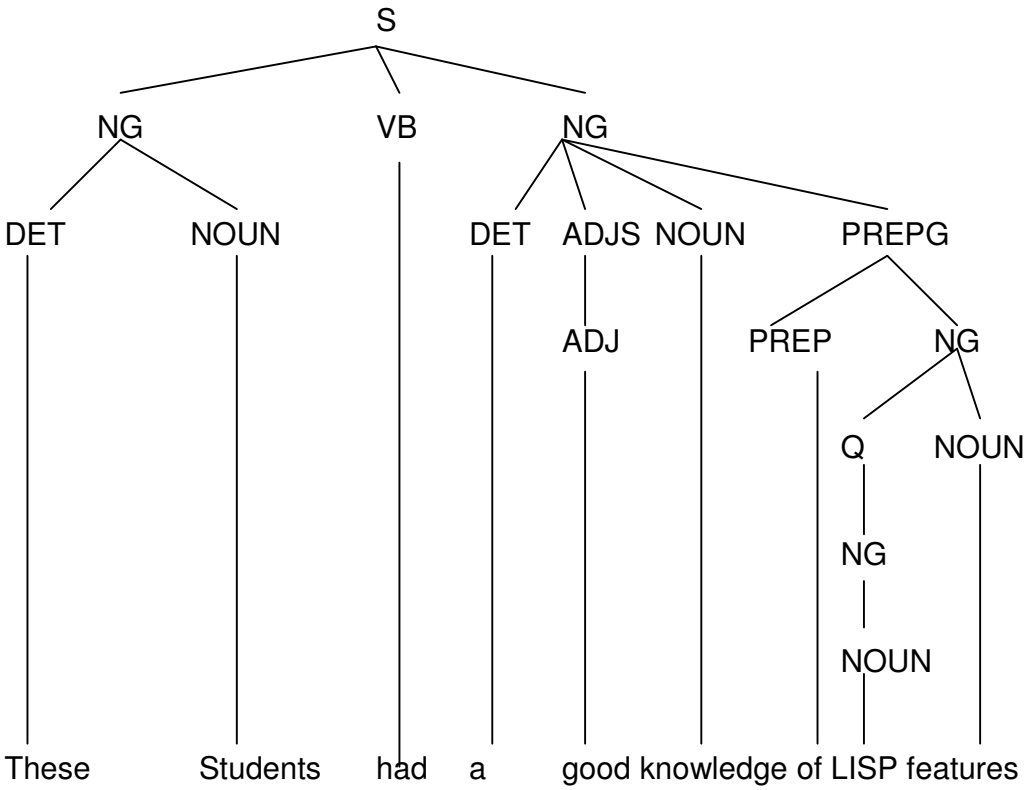
Syntax Analysis

Syntax Analysis

Menyusun *grammar* dari kalimat yang dapat direpresentasikan dengan "*Parse Tree*"

Setiap Node diberi Label:

S: Sentences	Adjs: Adjective Sequences	PREPG:Preposition Group
NG: Noun Group	Adj : Adjective	DET: Determiner
Q: Qualifier	VB: Verb	PREP: Preposition



Setelah parse tree di atas terbentuk, maka tersusunlah grammar sebagai berikut:

Grammar :

S	→ NG + VB + NG
NG	→ DET + NOUN
NG	→ DET + ADJS + NOUN + NOUN + PREPG
ADJS	→ ADJ
PREPG	→ PREP + NG
NG	→ Q + NOUN
Q	→ NG
NG	→ NOUN

Semantic Analysis

Semantic Analysis

Menentukan kegiatan utama dalam kalimat walaupun verbnya “have”, kegiatan utama : “***knowledge***” menentukan pelakunya, obyek dari kegiatan & karakteristik lainnya (waktu,lokasi)

Kegiatan : know

Qualifier : well

Pelaku : these students

Obyeks : lips

Waktu : past (lampau)

Lokasi :

DISCOURSE

Informasi yang dikumpulkan dari kalimat sebelumnya dari risalah dipergunakan untuk menyelesaikan hal-hal yang belum jelas :

-Two students = Peter, Jane

These

- Menghasilkan hubungan yang lebih spesifik.
- Menambah *knowledge*

Kegiatan : know

Qualifier : well

Pelaku : Peter & Jane

Obyek : lisp

Waktu : musim gugur 1988

Lokasi : San Fransisco