Tutorial: Pengenalan terhadap POS tagging dan Probabilistic Parsing

Ruli Manurung

Fakultas Ilmu Komputer
Universitas Indonesia

maruli@cs.ui.ac.id

Workshop Nasional INACL
Kamis, 7 Januari 2016

Outline

Outline

Apa itu model bahasa?

- Model bahasa dapat memprediksi perilaku sebuah bahasa, mis. aturan tata bahasa, distribusi probabilitas.
- Dibutuhkan pada hampir setiap aplikasi NLP.
- Dengan bantuan model bahasa, kita dapat memahami dan memprediksi sifat, fungsi, dan makna sebuah teks.

Pendekatan Linguist ("top-down")

Implementasikan algoritma dan struktur data berdasarkan teori dan model linguistik.

Pendekatan Empiricist ("bottom-up")

Gunakan model "black-box" berdasarkan statistik atau *machine learning*.

Contoh sederhana

Bagaimana caranya menginterpretasikan sebuah sinyal lisan:

1. "*I scream* is delicious".
2. "*Ice cream* is delicious".

Model linguistik

Kalimat (1) tidak valid, sedangkan kalimat (2) adalah valid.

Model empiris

"Ice cream is" lebih sering dijumpai daripada *"I scream is"*.

Cara kita memodelkan bahasa pun beragam, mis. *n*-gram, bag of words, POS sequence, parse tree, dst.

Outline

Outline

POS

Kelas kata

Part-of-speech, POS, word class, morphological class, lexical tag: sebuah atribut dari kata. Secara umum, menentukan "tipe" dari kata tersebut.

Apa yang bisa diprediksi oleh POS?

- Sifat morphological: imbuhan apa saja yang bisa ditambahkan?
- Sifat syntactic: apa saja kata-kata yang bisa muncul di dekatnya?
- Sifat semantic: secara umum, apa "maksud" dari kata tersebut?

Apa gunanya?

- Speech recognition: *"I scream is delicious"*
- Information retrieval: stemming (tahu imbuhan yang mungkin)
- Masih banyak lagi... *shallow parsing*

Open vs closed class types

- **Open class words** adalah kelas kata yang keanggotaannya biasanya besar dan senantiasa bertambah (serapan bahasa lain, teknologi baru, etc.). Kata-kata ini, pada umumnya, menyatakan "isi dunia": obyek, kejadian, atribut.
- **Closed class words** adalah kelas kata yang keanggotaannya biasanya kecil dan stabil. Kata-kata ini, pada umumnya, digunakan untuk menyatakan hubungan antar open class words.

Open class words

Nomina

Kata-kata yang menyatakan orang, benda, tempat. Tunggal vs. jamak. Konkrit vs. abstrak. Proper vs. common noun.

Verbs

Kata-kata yang menyatakan tindakan, proses, kejadian. *English verbs* bisa berbeda bentuk (*eat, eats, eating, eaten*).

Adjectives

Kata-kata yang menyatakan sifat/atribut. Hampir semua bahasa memiliki *adjective* yang menyatakan warna (*hitam, putih*), usia (*tua, muda*), dst.

Adverbs

Merupakan *modifier*/keterangan terhadap *verb*. Seringkali menjadi category *"dan lain-lain"*. Contoh: "*Unfortunately*, John walked *home extremely slowly yesterday*"

Closed class words

- *Closed class words* sebuah bahasa jumlahnya biasanya terbatas.
- Penutur bahasa harusnya tahu (hampir) semua *closed class words*.
- Disebut juga **function words** – memainkan peran *grammatical*.

Contoh:

- **prepositions**: on, under, over, near, . . .
- **determiners**: a, an, the
- **pronouns**: she, who, I others
- **conjunctions**: and, but, or, as, if, when
- **auxiliary verbs**: can, may, should, are
- **particles**: up, down, on, off, in, out, at, by
- **numerals**: one, two, three, first, second, third

Bandingkan dengan *open class words*. Kira-kira apa ciri-ciri pembedanya?

Open vs. closed

Kata-kata ini *open* atau *closed*?

- Bakso
- pedoman
- menyakitkan
- tentang
- menyanyi
- melakukan
- melalui
- bawah

Analogi CS?

Anggap closed class word seperti *reserved keyword* dalam bahasa pemrograman: `for`, `if`, `while`, etc. Open class $\approx$ variable(?)

Tagset

Contoh tagset bahasa Inggris

- Brown corpus tagset: 87 tag (Francis and Kučera, 1982)
- Penn Treebank tagset: 45 tag (Marcus et al., 1993)
- C5 CLAWS BNC tagset: 61 tag (Garside et al., 1997)
- C7 tagset: 146 tag (Leech et al., 1994)

Perbedaannya?

- Penn Treebank menggabungkan beberapa tag menjadi satu, karena ada informasi *parse tree*.
- Tergantung kebutuhan!

Penn Treebank Tagset

Cuplikan Penn Treebank Tagset

Tag	Description	Example
DT	Determiner	*a, the*
IN	Preposition	*on, in, by*
JJ	Adjective	*big*
JJR	Adjective, comparative	*bigger*
NN	Noun, sing. or mass	*dog, snow, llama*
NNS	Noun, plural	*dogs, llamas*
NNP	Proper noun, singular	*IBM*
VB	Verb, base form	*eat*
VBD	Verb, past tense	*ate*
VBG	Verb, gerund	*eating*

Contoh kalimat yang di-tag: The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

Outline

Part-of-speech tagging

- Tagging adalah *proses* menentukan kelas kata / *part-of-speech tag* untuk setiap kata dalam sebuah teks.
- Input: rangkaian kata + tagset. Output: tag yang paling tepat untuk setiap kata.
- $\approx$ tokenization untuk bahasa pemrograman (di mana bedanya?) $\rightarrow$ POS tagging bisa *ambiguous*!
- Contoh:

Book	*that*	*flight*	.
VB	DT	NN	.

  Book bisa juga NN (malahan lebih sering?).

Mengamati masalah POS tagging

- Sebetulnya, seberapa sulitkah POS tagging ini?
- DeRose(1988): Hanya 11.5% kata Inggris unik (type) dari Brown corpus adalah rancu.

Unambiguous (1 tag)	35340
Ambiguous (2-7 tags)	4100
2 tags	3760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 (*"still"*)

- Namun demikian, 40% dari (token) di Brown corpus rancu!
- Type vs. token = class vs. instance.
  "the boy saw the man":

Mengamati masalah POS tagging

- Sebetulnya, seberapa sulitkah POS tagging ini?
- DeRose(1988): Hanya 11.5% kata Inggris unik (type) dari Brown corpus adalah rancu.

Unambiguous (1 tag)	35340
Ambiguous (2-7 tags)	4100
2 tags	3760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 (*"still"*)

- Namun demikian, 40% dari (token) di Brown corpus rancu!
- Type vs. token = class vs. instance.
  "the boy saw the man": 5 token, 4 type
  "pria itu berdiri di tengah pria-pria lainnya":

Mengamati masalah POS tagging

- Sebetulnya, seberapa sulitkah POS tagging ini?
- DeRose(1988): Hanya 11.5% kata Inggris unik (type) dari Brown corpus adalah rancu.

Unambiguous (1 tag)	35340
Ambiguous (2-7 tags)	4100
2 tags	3760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 (*"still"*)

- Namun demikian, 40% dari (token) di Brown corpus rancu!
- Type vs. token = class vs. instance.
  "the boy saw the man": 5 token, 4 type
  "pria itu berdiri di tengah pria-pria lainnya": 7 token, 7 type
- (Perhatikan ini adalah beda *tag*, bukan makna/*sense*.)

Metode POS tagging

Secara umum, ada 3 cara:

- **Rule-based tagging.** Cara *top-down* – konsultasi ahli linguistik; definisikan aturan-aturan yang biasa digunakan manusia.
- **Stochastic tagger.** Cara *bottom-up* – gunakan corpus sebagai *training data* untuk menentukan secara probabilistik tag yang terbaik untuk sebuah kata (dalam sebuah konteks).
- **Transformation-based tagger**. Semacam gabungan teknik di atas. Tetap belajar dari corpus, tapi *knowledge* yang dipelajari dinyatakan sebagai *rule*.

POS ditentukan oleh konteks

Ide dasar POS tagging

POS tag sebuah kata dapat ditentukan oleh konteks di mana ia muncul.

Bisa dalam corpus

1. Gue bisa menyelesaikan persoalan itu kok.
2. Penjinak ular menguras bisa hanya dengan cangkir plastik.
3. Masyarakat dan aparat bisa membersihkan sampah dengan baik.
4. Beliau menyatakan bisa menurunkan harga kedelai tahun ini.

Aturan apa yang dapat disimpulkan mengenai bisa?

- Dalam *rule-based tagger*, ide ini dinyatakan dalam rule yang dibuat secara manual (mis: *"jika kata sesudahnya ..., maka ..."*)
- Pada *stochastic* POS, konteks (corpus) diamati dan dipelajari secara otomatis.

Sebuah contoh kasus

Amati race pada dua kalimat berikut:

1. Secretariat/NNP is/VBZ expected/VBN to/TO race/VB tomorrow/NN
2. People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN

Bayangkan sudah diketahui POS tag yang benar **kecuali** untuk *race*. Konteks yang perlu diamati (secara *bigram*):

1. to/TO race/???
2. the/DT race/???

Perumusan statistik (untuk kasus pertama):

"Berapa kemungkinan *tag* **VB** (atau **NN**) jika *tag* sebelumnya **TO**?" (*tag sequence probability*) dikalikan dengan
"Jika diketahui sebuah kata adalah **VB** (atau **NN**), berapa kemungkinan ia adalah *race*?" (*lexical likelihood*)

Pemodelan statistik

- Pada intinya, kita ingin memilih tag yang memaksimalkan rumus berikut: $P(kata|tag) \times P(tag|n\ tag\ sebelumnya)$
- Sebagai aproksimasi, sebuah *bigram* tagger memilih tag untuk kata ke-i (t_i) berdasarkan tag sebelumnya (t_{i-1}) dan kata ke-i tersebut (w_i)

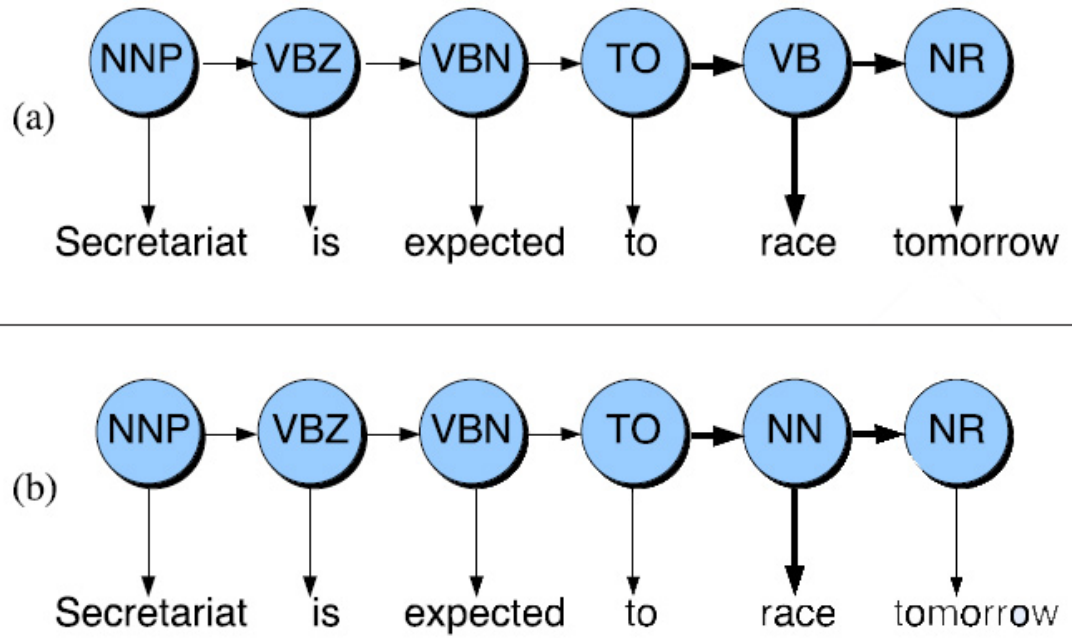
  $t_i = \text{argmax}_j\ P(t_j|t_{i-1}, w_i)$

  Melalui beberapa asumsi Markovian, diperoleh:

  $t_i = \text{argmax}_j\ P(t_j|t_{i-1}) \times P(w_i|t_j)$

  (Pada kenyataannya, kita ingin melakukan *tagging* pada seluruh kalimat sekaligus, bukan hanya satu kata!)

Probabilitas mengamati sekuens kata & tag



1. $P(VB|TO) \times P(race|VB) = 0.34 \times 0.00003 = 0.00001$
2. $P(NN|TO) \times P(race|NN) = 0.021 \times 0.00041 = 0.000007$

Menggunakan statistik

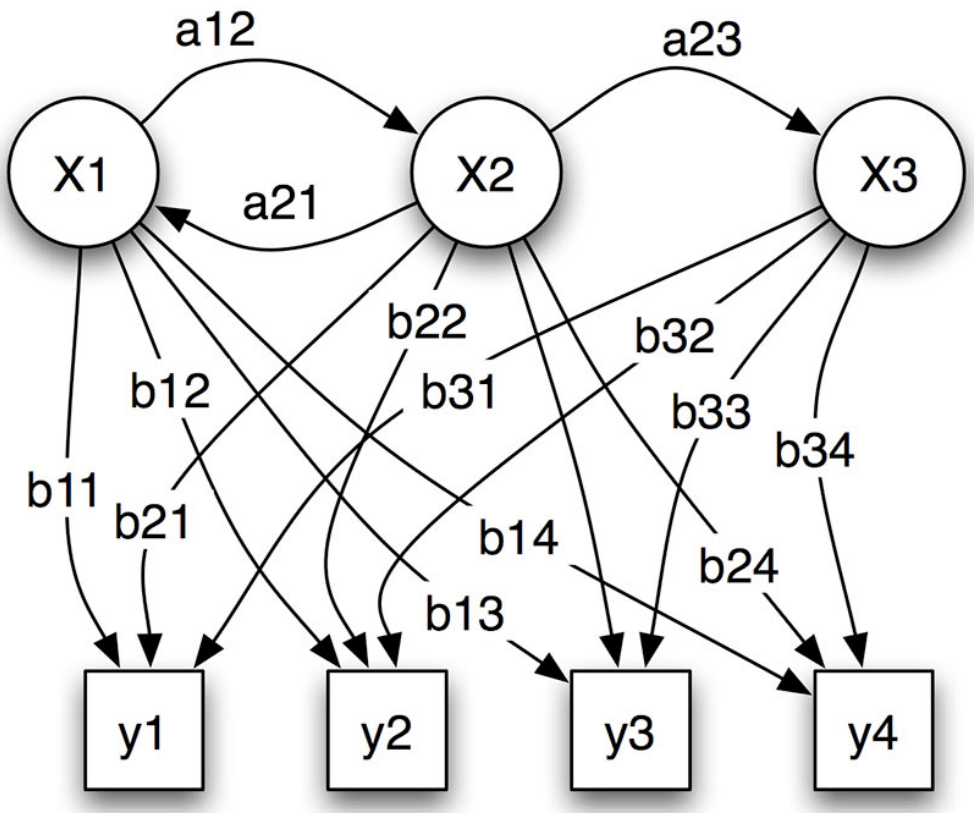
Definisi

Hidden Markov Model (HMM) adalah pemodelan statistik di mana sebuah sistem "menghasilkan" (*emit*) urutan simbol yang dapat diamati (*observation symbols*) berdasarkan sebuah proses probabilistik yang parameternya tidak diketahui (*hidden parameters*).

Proses probabilistik dinyatakan sebuah FSA:

- $x_1, x_2, \ldots$ adalah *state* yang menyatakan proses.
- a_{ij} adalah *state transition probabilities*: berapa kemungkinan proses berpindah dari state i ke j?
- $y_1, y_2, \ldots$ adalah *observation symbols*.
- b_{ij} adalah *output/emission probability*: berapa kemungkinan proses di state i menghasilkan symbol j?

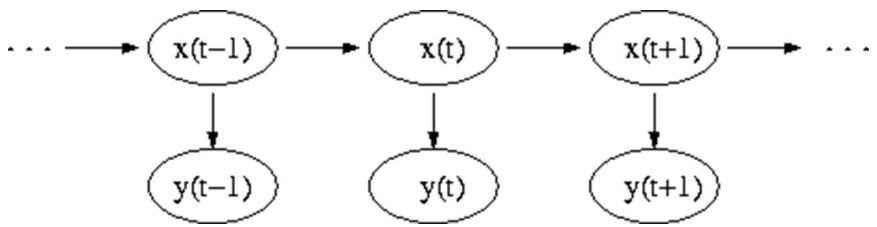
Contoh gambar HMM



Menggunakan statistik

HMM sering digunakan untuk memodelkan data *sequential/temporal*, di mana proses berjalan seiring waktu t, dengan asumsi berikut:

- Nilai *hidden state* $x(t)$ sepenuhnya ditentukan oleh *hidden state* sebelumnya, $x(t-1)$.
- Nilai *observed symbol* $y(t)$ sepenuhnya ditentukan oleh *hidden state* pada saat itu, $x(t)$.



Menghitung probabilitas sebuah *observation sequence*

Probabilitas $Y = y_0, y_1, y_2, \ldots, y_{L-1}$ dengan panjang L adalah $P(Y) = \sum_X P(Y|X)P(X)$. Jadi, kita menjumlahkan **semua kemungkinan** $X = x_0, x_1, x_2, \ldots, x_{L-1}$. Penghitungan *brute-force* dalam prakteknya bersifat *intractable*. Namun, ada algoritma *forward*.

HMM Tagger

- Contoh race: memilih *tag* terbaik untuk *kata* yang diamati.
- Sebuah HMM tagger memilih *tag sequence* terbaik untuk *word sequence* yang diamati.
  - Word sequence yang diamati: $W = w_1, w_2, \ldots, w_n$
  - Tag sequence yang terbaik/"benar": $T = t_1, t_2, \ldots, t_n$

  $\hat{T} = \text{argmax}_{T \in \tau} P(T|W)$

  Dengan Bayes Law:

  $\hat{T} = \text{argmax}_{T \in \tau} P(T)P(W|T)$

  Dengan chain rule:

  $\hat{T} = \text{argmax}_{T \in \tau} \prod_{i=1}^{n} P(w_i|w_1 t_1 \ldots w_{i-1} t_{i-1} t_i) P(t_i|t_1, \ldots t_{i-1})$
- Algoritma **Viterbi** dapat melakukan pencarian tag sequence terbaik ini secara cepat (*dynamic programming*).

Outline

Persiapan data

- **Data:** `http://bahasa.cs.ui.ac.id/postag/corpus`
- **Tool:** `http://nlp.stanford.edu/software/tagger.shtml`
- Siapkan dokumen training:

  Dokumen training berisi kalimat yang sudah diberi tagging secara manual dengan format:

```
Jokowi/NNP merayakan/VB tahun/NN baru/JJ di/IN Papua/NNP
```

Contoh file dokumen training dapat dilihat pada file: `postag_training_doc.txt`

Pembuatan model

- Buat file properties (contoh: `postag_training.props`) untuk menentukan setting untuk melakukan training model POS Tagger.
- Dua properti utama adalah: model: nama file model yang akan dihasilkan trainFile: dokumen training yang akan digunakan
- Jalankan perintah untuk melakukan training:
  ```
  java -classpath stanford-parser.jar
  edu.stanford.nlp.tagger.maxent.MaxentTagger -prop
  postag_training.props
  ```
- Akan dihasilkan sebuah model: `tagger.model`

Penggunaan model

- Berikan parameter path/nama_file_tagger_model (Pada contoh ini file model disimpan dalam folder model)
- Jalankan perintah: `java -classpath stanford-parser.jar edu.stanford.nlp.tagger.maxent.MaxentTagger -model model/tagger.model`

Outline

Outline

Apa itu grammar?

- *Grammar*: aturan-aturan yang menjelaskan struktur suatu fenomena.
- Dalam bahasa, *grammar* $\approx$ tata bahasa: aturan yang menjelaskan bagaimana merangkai kata-kata menjadi kalimat.
- Istilah lain: *syntax*. Secara umum, berarti "hal-hal yang berkaitan dengan tata-bahasa".
- Dengan grammar, kita bisa memahami struktur kalimat $\rightarrow$ model lebih kaya dan ekspresif.
- Kita akan melihat sebuah *grammar formalism*, yakni *Context-Free Grammar*.

Definisi formal CFG

Ingat, sebuah CFG adalah 4-tuple:

1. himpunan simbol non-terminal N
2. himpunan simbol terminal Σ (di mana $N \cap \Sigma = \emptyset$)
3. himpunan production rule P, masing-masing berbentuk $A \rightarrow \alpha$ di mana
   - $A \in N$, dan
   - $\alpha \in (\Sigma \cup N)*$ (dkl. α adalah string simbol terminal/nonterminal)
4. sebuah start symbol $S \in N$

Bahasa formal yang dinyatakan oleh sebuah CFG adalah himpunan string yang bisa di-*derive* dari symbol khusus: *start symbol (S)*. Dalam NLP, S sering diartikan sebagai *"sentence"*.

Context-Free Grammar

- Context-Free Grammar (CFG/Phrase-Structure Grammar/Backus-Naur Form) adalah suatu notasi matematis yang menyatakan aturan sebuah bahasa berdasarkan constituency.
- CFG terdiri dari dua bagian:
  1. sehimpunan (*rewrite*) *rule* atau *production*, yang menyatakan bagaimana symbol dalam bahasa dikelompokkan
  2. *lexicon*: daftar symbol/kata
- Symbol-symbol dalam CFG terbagi 2 kelas:
  1. **Terminal symbols**: symbol yang merepresentasikan kata dalam bahasa (muncul dama string/kalimat).
  2. **Non-terminal symbols**: symbol yang merepresentasikan kelompok/aggregate/generalisasi terminal.
- Lexicon adalah daftar terminal symbol.
- Non-terminal yang diasosiasikan dengan terminal kadang disebut **preterminal symbol**. Pada NLP, preterminal = part of speech!

Contoh CFG

Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

Contoh CFG

Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

$NP \rightarrow Det\ Nominal$

Contoh CFG

Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

$NP \rightarrow Det\ Nominal$

$NP \rightarrow ProperNoun$

Contoh CFG

Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

$NP \rightarrow Det\ Nominal$

$NP \rightarrow ProperNoun$

$Nominal \rightarrow Noun \mid Noun\ Nominal$

Contoh *lexicon*:

Lexicon bisa saja dinyatakan sebagai aturan context-free sebagai berikut:

Contoh CFG

Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

$NP \rightarrow Det\ Nominal$

$NP \rightarrow ProperNoun$

$Nominal \rightarrow Noun \mid Noun\ Nominal$

Contoh *lexicon*:

Lexicon bisa saja dinyatakan sebagai aturan context-free sebagai berikut:

$Det \rightarrow a$

Contoh CFG

Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

$NP \rightarrow Det\ Nominal$

$NP \rightarrow ProperNoun$

$Nominal \rightarrow Noun \mid Noun\ Nominal$

Contoh *lexicon*:

Lexicon bisa saja dinyatakan sebagai aturan context-free sebagai berikut:

$Det \rightarrow a$

$Det \rightarrow the$

Contoh CFG

Contoh *rule*:

Dalam bhs. Inggris, sebuah `NP` bisa terdiri dari `ProperNoun` atau `Determiner` diikuti `Nominal`. Sebuah `Nominal` bisa terdiri dari satu atau lebih `Noun`.

$NP \rightarrow Det\ Nominal$

$NP \rightarrow ProperNoun$

$Nominal \rightarrow Noun \mid Noun\ Nominal$

Contoh *lexicon*:

Lexicon bisa saja dinyatakan sebagai aturan context-free sebagai berikut:

$Det \rightarrow a$

$Det \rightarrow the$

$Noun \rightarrow flight$

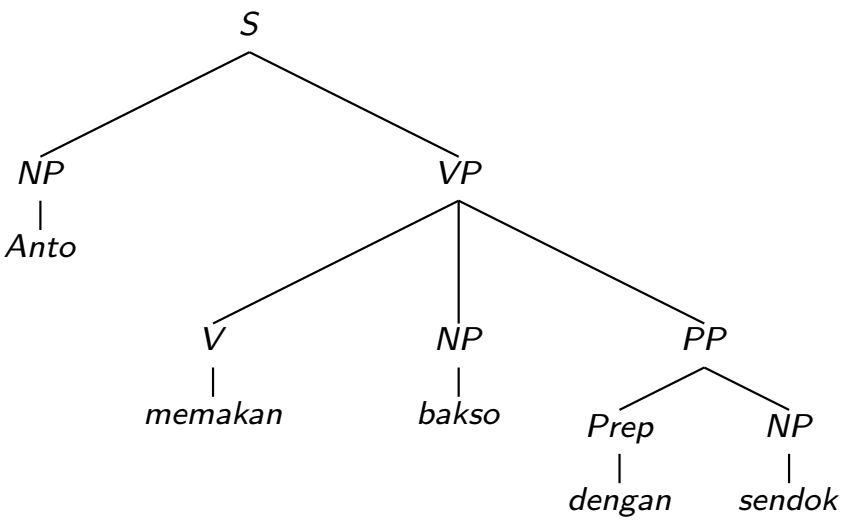
(dalam kenyataannya, kurang efisien)

CFG sebagai parser

Sebagai parser:

CFG sebagai mesin yang menghasilkan struktur untuk sebuah kalimat input.

Sebagai parser, bacalah tanda "$\rightarrow$" sebagai: *"if you see the symbols on the right, rewrite with the symbol on the left"*.



- Masalah bahasa manusia: satu kalimat, banyak struktur **(=ambiguity/kerancuan)**
- Tujuan akhir *parsing*: agar program dapat "memahami" **semantics/makna** dari kalimat.

Notasi alternatif, *bracketed notation*:
[$_S$ [$_{NP}$ Anto] [$_{VP}$ [$_V$ memakan] [$_{NP}$ bakso] [$_{PP}$ [$_{Prep}$ dengan] [$_{NP}$ sendok]]]]

Grammar singkat untuk bahasa Inggris (ATIS)

Lexicon untuk $\mathcal{L}_0$

Noun	$\rightarrow$	*flights — breeze — trip — morning — . . .*
Verb	$\rightarrow$	*is — prefer — like — need — . . .*
Adjective	$\rightarrow$	*cheapest — first — other — direct — . . .*
Pronoun	$\rightarrow$	*me — I — you — it — . . .*
ProperNoun	$\rightarrow$	*Alaska — Baltimore — Chicago — Garuda — . . .*
Determiner	$\rightarrow$	*the — a — an — this — . . .*
Preposition	$\rightarrow$	*from — to — on — near — . . .*

Grammar untuk $\mathcal{L}_0$

S	$\rightarrow$	*NP VP*	I + prefer a morning flight
NP	$\rightarrow$	*Pronoun*	I
	—	*ProperNoun*	Los Angeles
	—	*Det Nominal*	a + flight
Nominal	$\rightarrow$	*Noun Nominal*	morning + flight
	—	*Noun*	flights
VP	$\rightarrow$	*Verb*	do
	—	*Verb NP*	prefer a morning flight
	—	*Verb NP PP*	leave Boston in the morning
	—	*Verb PP*	leave in the morning
PP	$\rightarrow$	*Prep NP*	from Los Angeles

Grammaticality

- Kalimat-kalimat yang bisa di-*derive* dari S dikatakan *grammatical*.
- Kalimat-kalimat yang **TIDAK** bisa di-*derive* dari S dikatakan *ungrammatical*.
- Perbedaan yang sangat "tajam" untuk bahasa formal seperti ini terkadang kurang cocok untuk bahasa natural/manusia ...
- Dalam bidang *linguistics*, pemodelan ini disebut generative grammar (Chomsky), akhir '60-an.

Apakah kalimat-kalimat berikut grammatical (menurut $\mathcal{L}_0$)?

You need a trip to Baltimore.
I prefer a morning trip.
I prefer a trip morning.
I do to you.

Grammaticality

- Kalimat-kalimat yang bisa di-*derive* dari S dikatakan *grammatical*.
- Kalimat-kalimat yang **TIDAK** bisa di-*derive* dari S dikatakan *ungrammatical*.
- Perbedaan yang sangat "tajam" untuk bahasa formal seperti ini terkadang kurang cocok untuk bahasa natural/manusia ...
- Dalam bidang *linguistics*, pemodelan ini disebut generative grammar (Chomsky), akhir '60-an.

Apakah kalimat-kalimat berikut grammatical (menurut $\mathcal{L}_0$)?

You need a trip to Baltimore.
I prefer a morning trip.
I prefer a trip morning.
I do to you.

Pesan kepada sang *linguist*: Perhatikan masalah *overgeneration*!

Outline

Dunia tidak hitam dan putih

- CFG mempartisi string NLP menjadi 2: **sah** (di-*accept*) dan **tidak** (di-*reject*).
- Namun, bahasa tidak sediskrit ini. Ada kalimat yang "lebih sah" dari yang lain.
- Pendekatan formal tidak menangani disambiguation. Pokoknya, kembalikan semua kemungkinan!
- **Language modelling**: sebuah model yang lebih akurat. Aplikasi: *psycholinguistics*, NLU, NLG, *speech recognition*

Menambahkan probabilitas

Rule sebuah CFG secara non-deterministik menjabarkan semua kemungkinan *rewrite* sebuah non-terminal:

Contoh kemungkinan *expansion* VP:

$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ NP$

Menambahkan probabilitas

Rule sebuah CFG secara non-deterministik menjabarkan semua kemungkinan *rewrite* sebuah non-terminal:

Contoh kemungkinan *expansion* VP:

$VP \rightarrow Verb$
$VP \rightarrow Verb\ NP$
$VP \rightarrow Verb\ NP\ NP$

- Dengan PCFG, kita nyatakan **probabilitas** setiap kemungkinan *expansion* terjadi.

Menambahkan probabilitas

Rule sebuah CFG secara non-deterministik menjabarkan semua kemungkinan *rewrite* sebuah non-terminal:

Contoh kemungkinan *expansion* VP:

$VP \rightarrow Verb$	[0.55]
$VP \rightarrow Verb\ NP$	[0.40]
$VP \rightarrow Verb\ NP\ NP$	[0.05]

- Dengan PCFG, kita nyatakan **probabilitas** setiap kemungkinan *expansion* terjadi.
- Notasi: $A \rightarrow \beta[p]$, di mana $p = P(A \rightarrow \beta|A)$
- Jumlah probabilitas semua kemungkinan *rule* yang meng-*expand* A harus 1.

Definisi formal CFG

Sebuah CFG adalah 4-tuple $G = (N, \Sigma, P, S)$:

1. himpunan simbol non-terminal N
2. himpunan simbol terminal Σ (di mana $N \cap \Sigma = \emptyset$)
3. himpunan *production rule* P, masing-masing berbentuk $A \rightarrow \alpha$ di mana
   - $A \in N$, dan
   - $\alpha \in (\Sigma \cup N)*$ (dkl. α adalah string simbol terminal/nonterminal)
4. sebuah *start symbol* $S \in N$

Definisi formal PCFG

Sebuah PCFG adalah 5-tuple $G = (N, \Sigma, P, S, D)$:

1. himpunan simbol non-terminal N
2. himpunan simbol terminal Σ (di mana $N \cap \Sigma = \emptyset$)
3. himpunan *production rule* P, masing-masing berbentuk $A \rightarrow \alpha$ di mana
   - $A \in N$, dan
   - $\alpha \in (\Sigma \cup N)*$ (dkl. α adalah string simbol terminal/nonterminal)
4. sebuah *start symbol* $S \in N$
5. sebuah fungsi D yang menyatakan nilai probabilitas [0,1] untuk setiap rule $A \rightarrow \alpha \in P$

Contoh PCFG

$S \rightarrow NP\ VP$	[.80]	$Det \rightarrow that$ [.05] | the [.80] | a [.15]	
$S \rightarrow Aux\ NP\ VP$	[.15]	$Noun \rightarrow book$	[.10]
$S \rightarrow VP$	[.05]	$Noun \rightarrow flights$	[.50]
$NP \rightarrow Det\ Nom$	[.20]	$Noun \rightarrow meal$	[.40]
$NP \rightarrow Proper\text{-}Noun$	[.35]	$Verb \rightarrow book$	[.30]
$NP \rightarrow Nom$	[.05]	$Verb \rightarrow include$	[.30]
$NP \rightarrow Pronoun$	[.40]	$Verb \rightarrow want$	[.40]
$Nom \rightarrow Noun$	[.75]	$Aux \rightarrow can$	[.40]
$Nom \rightarrow Noun\ Nom$	[.20]	$Aux \rightarrow does$	[.30]
$Nom \rightarrow Proper\text{-}Noun\ Nom$	[.05]	$Aux \rightarrow do$	[.30]
$VP \rightarrow Verb$	[.55]	$Proper\text{-}Noun \rightarrow TWA$	[.40]
$VP \rightarrow Verb\ NP$	[.40]	$Proper\text{-}Noun \rightarrow Denver$	[.40]
$VP \rightarrow Verb\ NP\ NP$	[.05]	$Pronoun \rightarrow you$ [.40] | I [.60]	

Outline

Disambiguation dengan PCFG

Menghitung probabilitas sebuah *parse tree* T

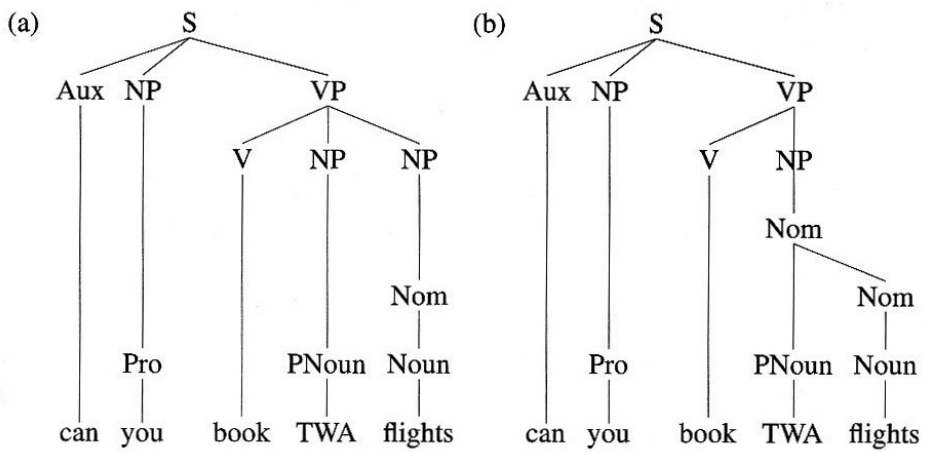
- Probabilitas T untuk sebuah kalimat S adalah **hasil perkalian probabilitas semua rule** r yang digunakan untuk meng-*expand* setiap node $n \in T$: $P(T,S) = \prod_{n \in T} p(r(n))$
- $P(T,S) = P(T)$. Mengapa? $P(T,S) = P(T) \times P(S|T)$. Namun $P(S|T)$ pasti 1, karena parse tree secara deterministik menentukan kalimat (postorder leaf traversal?)

Disambiguation dengan PCFG:

Definisi: $\tau(S)$ = himpunan semua kemungkinan *parse tree* untuk S.
Parse tree yang "tepat" untuk sebuah kalimat S adalah parse tree dengan probabilitas tertinggi:

$$\hat{T}(S) = \text{argmax}_{T \in \tau(S)} P(T|S) = \text{argmax}_{T \in \tau(S)} \frac{P(T,S)}{P(S)}$$
$$\hat{T}(S) = \text{argmax}_{T \in \tau(S)} P(T,S) = \text{argmax}_{T \in \tau(S)} P(T)$$

Contoh *disambiguation* dengan PCFG



Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP NP	.05
NP	→	Nom	.05
NP	→	PNoun	.35
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
PNoun	→	TWA	.40
Noun	→	flights	.50

Rules			P
S	→	Aux NP VP	.15
NP	→	Pro	.40
VP	→	V NP	.40
NP	→	Nom	.05
Nom	→	PNoun Nom	.05
Nom	→	Noun	.75
Aux	→	Can	.40
NP	→	Pro	.40
Pro	→	you	.40
Verb	→	book	.30
Pnoun	→	TWA	.40
Noun	→	flights	.50

- $T_a \approx$ "bisakah anda memesan penerbangan untuk TWA"?
- $T_b \approx$ "bisakah anda memesan penerbangan maskapai TWA"?
- $P(T_a) = .15 \times .40 \times .05 \times .05 \times .35 \times .75 \times .40 \times .40 \times .40 \times .30 \times .40 \times .50 = \mathbf{1.5 \times 10^{-6}}$
- $P(T_b) = .15 \times .40 \times .40 \times .05 \times .05 \times .75 \times .40 \times .40 \times .40 \times .30 \times .40 \times .50 = \mathbf{1.7 \times 10^{-6}}$

Language modelling

- PCFG juga menyatakan probabilitas sebuah **kalimat** S.
- Hal ini berguna untuk berbagai aplikasi, mis. *speech recognition*:
  - *I like ice cream*
  - *I like I scream*.
- Untuk kalimat yang *unambiguous*, $P(S) = P(T, S) = P(T)$.
- Probabilitas sebuah *kalimat* S yang *ambiguous* adalah jumlah probabilitas semua parse tree $T \in \tau(S)$:
  $P(S) = \sum_{T \in \tau(S)} P(T, S) = \sum_{T \in \tau(S)} P(T)$
- Penjabaran **semua** *parse tree* tidak efisien $\rightarrow$ dynamic programming.
  Inside algorithm pada PCFG $\approx$ *Forward algorithm* pada HMM.
  Menjumlahkan probabilitas semua {*parse tree* / *state sequence*} untuk sebuah {*input string* / *observation sequence*}.

Menghitung probabilitas dari *treebank*

- Dari mana asalnya nilai probabilitas PCFG?
- Dari sebuah corpus yang telah di-*parse* secara manual, atau **treebank**.
- Contohnya: Penn Treebank (Marcus et al. 1993)
- Probabilitas sebuah *expansion rule* $\alpha \rightarrow \beta$ bisa dihitung: berapa kali $\alpha \rightarrow \beta$ terjadi, dibagi dengan kemunculan α:

$$P(\alpha \rightarrow \beta|\alpha) = \frac{Count(\alpha \rightarrow \beta)}{\sum_{\gamma} Count(\alpha \rightarrow \gamma)} = \frac{Count(\alpha \rightarrow \beta)}{Count(\alpha)}$$

- *Treebank* harus dihasilkan secara manual, ongkosnya mahal.

Outline

Persiapan data

- **Data:** `http://bahasa.cs.ui.ac.id/treebank/corpus`
- **Tool:** `http://nlp.stanford.edu/software/lex-parser.shtml`
- Siapkan dokumen training:
  Dokumen training berisi kalimat yang sudah diberi bracketing secara manual dengan format:

```
(ROOT (S (NP-SBJ (NNP (Gates)) (CC (dan)) (NNP (Buffett))) (VP
(VB (mengatakan)) (SBAR (SC (O)) (S (NP-SBJ (PRP (mereka))) (VP
(VP (VB (berada)) (PP (IN (di)) (NP (NNP (Cina))))) (SBAR (SC
(untuk)) (S (NP-SBJ (*)) (VP (VB (mempelajari)) (NP (NP (NN
(kegiatan)) (NN (amal))) (PP (IN (di)) (NP (NN (negara)) (PR
(itu))))))))))) (Z (.))))
```

Contoh file dokumen training dapat dilihat pada file:
`parser_training.bracket`

Pembuatan model

- Jalankan perintah untuk melakukan training:
  ```
  java
  edu/stanford/nlp/parser/shiftreduce/ShiftReduceParser
  -trainTreebank parser_training.bracket
  -devTreebank parser_training.bracket
  -serializedPath Indonesian_Model.ser.gz
  ```
- Akan dihasilkan sebuah model: `Indonesian_Model.ser.gz`
- Jalankan perintah:
  ```
  java ParserShiftReduce "Jokowi merayakan tahun
  baru di Papua"
  ```

Ringkasan

- Model bahasa sangat berguna dalam berbagai aplikasi NLP
- POS tagging: memprediksi sekuens POS tag sebuah teks. Model linear.
- Parsing: memprediksi struktur konstituensi sebuah teks. Model hirarkis.
- Distribusi probabilitas dihitung dari *tagged corpus* / *treebank*, dimodelkan dengan HMM/PCFG.
- Lebih banyak training data, lebih baik :-)