

**Membangun sebuah Combinatory Categorical  
Grammar (CCG) Supertagger Berbasis  
Maximum Entropy untuk Bahasa Indonesia**

**Proposal Tugas Akhir**

**Kelas TA NLP**

**Wisnu Adi Nurcahyo**

**NIM: 1301160479**



**Program Studi Sarjana Informatika**

**Fakultas Informatika**

**Universitas Telkom**

**Bandung**

**2019**

## Lembar Persetujuan

Membangun sebuah Combinatory Categorical Grammar (CCG)  
Supertagger Berbasis Maximum Entropy untuk Bahasa Indonesia

*Building a Combinatory Categorical Grammar (CCG)  
Supertagger Based on the Maximum Entropy for Bahasa  
Indonesia*

Wisnu Adi Nurcahyo  
NIM: 1301160479

Proposal ini diajukan sebagai usulan pembuatan tugas akhir pada  
Program Studi Sarjana Informatika  
Fakultas Informatika Universitas Telkom

Bandung, 13 November 2019  
Menyetujui

Calon Pembimbing 1

Dr. Ade Romadhony, S.T., M.T.  
NIP: 06840042

## Abstrak

Dalam pemrosesan bahasa alami, combinatory categorial grammar (CCG) merupakan salah satu formalisme tata bahasa yang dapat digunakan untuk membangun sebuah *parser* yang umumnya dikenal sebagai CCG *parser*. CCG *parser* dapat digunakan untuk berbagai macam keperluan dalam pemrosesan bahasa alami. Sebagai contoh, CCG *parser* dapat digunakan untuk memperoleh informasi (*information extraction*) dari suatu kalimat yang kemudian membentuk sebuah *query*. Agar dapat bekerja, CCG *parser* membutuhkan CCG *lexicon*. CCG *lexicon* diperoleh dari proses yang bernama *supertagging*. *Supertagging* adalah proses pelabelan suatu token kata terhadap *supertag*-nya. Perangkat lunak yang melakukan *supertagging* disebut sebagai *supertagger*. Demikian itu, *supertagging* merupakan langkah pertama yang perlu dilakukan sebelum membangun sebuah CCG *parser*. *Supertagger* yang dibangun dalam tugas akhir ini dimaksudkan sebagai produsen CCG *lexicon* bahasa Indonesia untuk riset-riset yang berkenaan dengan CCG di masa yang akan datang.

**Kata Kunci:** natural language processing, combinatory categorial grammar, supertagger, maximum entropy model, bahasa indonesia, haskell

# Daftar Isi

<b>Abstrak</b>	<b>i</b>
<b>Daftar Isi</b>	<b>ii</b>
<b>I Pendahuluan</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Perumusan Masalah . . . . .	1
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Rencana Kegiatan . . . . .	2
1.6 Jadwal Kegiatan . . . . .	2
<b>II Kajian Pustaka</b>	<b>4</b>
2.1 Categorical Grammar . . . . .	4
2.2 Combinatory Categorical Grammar . . . . .	4
2.3 Category Theory . . . . .	6
2.4 Lambda Calculus . . . . .	6
2.5 Supertagging . . . . .	7
2.6 Maximum Entropy Model . . . . .	7
<b>III Metodologi dan Desain Sistem</b>	<b>9</b>
3.1 Pembuatan Dataset . . . . .	9
3.1.1 Input Teks Bahasa Indonesia . . . . .	9
3.1.2 POS Tagging . . . . .	9
3.1.3 Transformasi POS Tag . . . . .	10
3.1.4 Output CCG Supertag . . . . .	11
3.2 Melatih Model Supertagger . . . . .	12
<b>Daftar Pustaka</b>	<b>14</b>
<b>Lampiran</b>	<b>15</b>

# Bab I

## Pendahuluan

### 1.1 Latar Belakang

Riset pemrosesan bahasa natural untuk bahasa Indonesia saat ini terbilang sedikit. Bahkan, masih banyak area riset yang belum tersentuh seperti contohnya *combinatory categorial grammar* (CCG). CCG merupakan formalisme tata bahasa yang salah satu manfaatnya adalah untuk memperoleh informasi (*information extraction*) dari suatu kalimat. Informasi tersebut diperoleh setelah melakukan *parsing* berdasarkan formalisme CCG dengan menggunakan perangkat lunak bernama CCG *parser*. Untuk dapat melakukan *parsing*, CCG *parser* membutuhkan CCG *lexicon* yang mengandung bentuk formal dari suatu token kata. Bentuk formal yang dimaksud adalah *category* dalam *category theory*. CCG *lexicon* diperoleh dari proses pelabelan suatu token kata terhadap bentuk formalnya yang mana dikenal sebagai *supertagging*. Proses *supertagging* akan menghasilkan *supertag* yang kemudian disebut sebagai CCG *supertag* karena formalisme yang digunakan adalah formalisme CCG. Dalam hal ini, CCG *supertag* adalah CCG *lexicon* itu sendiri.

Tugas akhir dengan judul “Membangun sebuah Combinatory Categorical Grammar (CCG) Supertagger Berbasis Maximum Entropy untuk Bahasa Indonesia” berusaha untuk membangun versi awal dari CCG *supertagger* untuk bahasa Indonesia yang mana harapannya dapat menjadi inisiator riset pemrosesan bahasa natural dengan tema CCG sehingga ke depannya akan ada lebih banyak riset mengenai CCG yang tersedia. *Supertagger* yang dimaksud dalam tugas akhir ini akan dibangun dengan menggunakan model Maximum Entropy (MaxEnt) dan implementasinya akan ditulis dalam bahasa pemrograman Haskell. Model MaxEnt digunakan karena keterbatasan *dataset* untuk melakukan *learning*. Adapun bahasa pemrograman Haskell digunakan karena abstraksi bahasanya yang sangat mendekati *category theory* serta kemampuannya yang sangat baik dalam pemrosesan data.

### 1.2 Perumusan Masalah

Rumusan masalah yang akan diangkat yaitu:

1. Mengapa CCG *supertagger* diperlukan?

2. Apa saja yang harus dipersiapkan untuk membangun CCG *supertagger*?
3. Bagaimana proses pembangunan CCG *supertagger*?

### 1.3 Tujuan

Tujuan yang diharapkan dapat tercapai oleh tugas akhir ini yaitu:

1. Mengenalkan alternatif metode yang dapat digunakan dalam pemrosesan bahasa alami untuk bahasa Indonesia.
2. Merilis CCG *supertagger* pertama untuk bahasa Indonesia.
3. Membuka peluang riset untuk CCG *parser* bahasa Indonesia.

### 1.4 Batasan Masalah

Hipotesis dari tugas akhir ini yaitu:

1. Memberikan label CCG untuk proses *learning* merupakan permasalahan utama dari tugas akhir ini.
2. *Supertagger* yang akan dibangun kemungkinan besar memiliki akurasi yang cenderung rendah.
3. CCG *lexicon* sudah dapat digunakan oleh CCG *parser* bahasa Indonesia (apabila ada).

### 1.5 Rencana Kegiatan

Rencana kegiatan yang akan dilakukan adalah sebagai berikut:

- Studi literatur
- Studi *tools* yang tersedia
- Studi bahasa pemrograman yang akan digunakan
- Perancangan sistem *supertagger*
- Membangun *supertagger*
- Memeriksa hasil

### 1.6 Jadwal Kegiatan

Laporan proposal ini akan dijadwalkan sesuai dengan tabel 1.1.

Tabel 1.1: Jadwal kegiatan proposal tugas akhir.

No	Kegiatan	Bulan ke-																							
		1				2				3				4				5				6			
1	Studi Literatur																								
2	Studi <i>Tools</i> yang Tersedia																								
3	Studi Bahasa Pemrograman																								
4	Pengumpulan Data																								
5	Analisis dan Perancangan Sistem																								
6	Implementasi Sistem																								
7	Analisa Hasil Implementasi																								
8	Penulisan Laporan																								

## Bab II

### Kajian Pustaka

#### 2.1 Categorical Grammar

Categorical Grammar (CG) merupakan sebuah istilah yang mencakup beberapa formalisme terkait yang diajukan untuk sintaks dan semantik dari bahasa alami serta untuk bahasa logis dan matematis [2]. Karakteristik yang paling terlihat dari CG adalah bentuk esktrim dari leksikalismenya di mana beban utama (atau bahkan seluruh beban) sintaksisnya ditanggung oleh leksikon. Konstituen tata bahasa dalam *categorical grammar* dan khususnya semua leksikal diasosiasikan dengan suatu *type* atau “*category*” (dalam *category theory*) yang mendefinisikan potensi mereka untuk dikombinasikan dengan konstituen lain untuk menghasilkan konstituen majemuk. *Category* tersebut adalah salah satu dari sejumlah kecil *category* dasar (seperti NP) atau *functor* (dalam *category theory*).

Ada beberapa notasi berbeda untuk *category* dalam merepresentasikan *directional*-nya. Notasi yang paling umum digunakan adalah “*slash notation*” yang dipelopori oleh Bar-Hilel, Lambek, dan kemudian dimodifikasi dalam kelompok teori yang dibedakan sebagai tata bahasa “*combinatory*” *categorical grammar* (CCG). Sebagai contoh, *category*  $(S \backslash NP) / NP$  merupakan suatu *functor* yang memiliki dua buah notasi *slash* yaitu  $\backslash$  dan  $/$ . Masing-masing notasi *slash* tersebut merepresentasikan *directionality* yang berbeda. Notasi *forward slash*,  $/$ , mengindikasikan bahwa argumen dari suatu *functor*  $X/Y$  ada di bagian kanan atau dengan kata lain  $Y$ . Adapun *backward slash*,  $\backslash$ , mengindikasikan bahwa argumen dari suatu *functor*  $X \backslash Y$  ada di bagian kiri atau dengan kata lain  $X$ . Demikian itu, penggunaan notasi *slash* yang tepat sangat penting dikarenakan hal ini dapat mempengaruhi konstituen dari hasil “kombinasi” *category*-nya.

#### 2.2 Combinatory Categorical Grammar

Combinatory Categorical Grammar (CCG) merupakan salah satu formalisme tata bahasa yang gaya aturannya diturunkan dari *categorical grammar* dengan beberapa penambahan aturan dan istilah baru. Di CCG, *category* dapat dipasangkan dengan *semantic representation*. Dalam hal ini, *semantic repre-*



Pamungkas  $\vdash$  NP :  $pamungkas'$   
 Setyo  $\vdash$  NP :  $setyo'$   
 dan  $\vdash$  CONJ :  $\lambda x.\lambda y.\lambda f. (f\ x) \wedge (f\ y)$   
 menyukai  $\vdash$  (S\NP)/NP :  $\lambda x.\lambda y. suka(y, x)$   
 rendang  $\vdash$  NP :  $rendang'$

Gambar 2.1: Kamus yang memetakan token kata ke bentuk CCG *lexicon*-nya.

*sensation* yang dimaksud adalah abstraksi fungsi lambda (dalam *lambda calculus*, *lambda function*). Sebagai contoh, *category* (S\NP)/NP dapat dipasangkan dengan fungsi lambda  $\lambda x.fx$  sehingga dapat ditulis menjadi (S\NP)/NP :  $\lambda x.fx$ . Adapun pemetaan dari suatu token kata ke *category*-nya menggunakan notasi  $\vdash$ . Sebagai contoh, anggap saja kita memiliki kamus pemetaan seperti pada Gambar 2.1. Apabila kita memiliki kalimat “Pamungkas dan Setyo menyukai rendang”, maka kita dapatkan:

Pamungkas	dan	Setyo	menyukai	rendang
NP	CONJ	NP	(S\NP)/NP	NP
: $pamungkas'$	: $\lambda x.\lambda y.\lambda f. (f\ x) \wedge (f\ y)$	: $setyo'$	: $\lambda x.\lambda y. suka(y, x)$	: $rendang'$

Ada beberapa operasi yang dapat dilakukan dalam CCG. *Operand* dari operasi yang dimaksud adalah *category*. Berdasarkan contoh di atas, akan ada tiga operasi yang dijalankan yaitu *coordination*, *forward application*, dan *type rising*. Untuk mendapatkan hasil yang diinginkan, kita lakukan *type rising* sebelum *forward application* di akhir. Sehingga, kita dapatkan:

Pamungkas	dan	Setyo	menyukai	rendang
NP	CONJ	NP	(S\NP)/NP	NP
: <i>pamungkas'</i>	: $\lambda x.\lambda y.\lambda f. (f\ x) \wedge (f\ y)$	: <i>setyo'</i>	: $\lambda x.\lambda y. suka(y, x)$	: <i>rendang'</i>
			< & >	>
NP		S\NP		
: $\lambda f. (f\ pamungkas') \wedge (f\ setyo')$		: $\lambda y. suka(y, rendang')$		
			> T	
S/(S\NP)				
: $\lambda f. (f\ pamungkas') \wedge (f\ setyo')$				
S				
: $suka(pamungkas', rendang') \wedge suka(setyo', rendang')$				

Berdasarkan hasil evaluasi tersebut, kita dapatkan *query* 2.1 yang diperoleh dari kalimat “Pamungkas dan Setyo menyukai rendang”. Demikian itu, komputer dapat melakukan komputasi berdasarkan *query* yang telah diperoleh.

Kegiatan tersebut merupakan apa yang disebut dengan CCG *parsing*. Untuk dapat melakukan parsing, CCG *lexicon* diperlukan. Untuk mendapatkan CCG *lexicon* kita dapat menggunakan CCG *supertagger* yang akan melakukan pelabelan suatu token kata ke CCG *lexicon* berdasarkan pemetaannya.

$$\textit{suka}(\textit{pamungkas}', \textit{rendang}') \wedge \textit{suka}(\textit{setyo}', \textit{rendang}') \quad (2.1)$$

## 2.3 Category Theory

*Category Theory* (CT) merupakan formalisme yang dapat digunakan untuk memformalkan struktur matematis. CT mempelajari *category* yang merupakan sebuah representasi dari suatu abstraksi konsep matematis. Suatu *category* memiliki kumpulan *object* dan *morphism*. Untuk mempermudah pemahaman mengenai CT, kita akan gunakan *category of set* (kategori dari himpunan) sebagai contoh. Dalam *category of set*, *object*-nya adalah himpunan dan *morphism*-nya (terkadang disebut dengan *arrow*) adalah fungsi (*function*, sebuah pemetaan). Kemudian, pemetaan dari suatu *category C* ke *category D* yang dipetakan oleh  $F$  ( $F : C \rightarrow D$ ) disebut sebagai *functor*.

## 2.4 Lambda Calculus

*Lambda calculus* ( $\lambda$ -*calculus*) merupakan sebuah formalisme yang dikembangkan oleh Alonzo Church sebagai alat yang digunakan untuk memahami konsep komputasi yang efektif [1]. Formalisme  $\lambda$ -*calculus* cukup populer dan bahkan dijadikan sebagai pondasi teori bagi paradigma pemrograman *functional programming*. Konsep utama dari  $\lambda$ -*calculus* adalah apa yang disebut dengan *expression*. Suatu *expression* dalam  $\lambda$ -*calculus* terdiri dari tiga bagian yaitu *lambda notation* ( $\lambda$ ), *argument* (seperti  $a$ ,  $b$ ,  $c$ ,  $x$ , dan lain-lain), dan *body* yang dipisahkan dengan tanda titik. Sebagai contoh, fungsi lambda  $\lambda x.x$  merupakan sebuah fungsi identitas yang mengambil argumen  $x$  kemudian mengembalikan nilai  $x$  itu sendiri. Dalam hal ini, terlihat bahwa notasi  $\lambda$  merupakan sebuah penanda bagi suatu fungsi lambda. Kemudian, pengubah  $x$  setelah notasi  $\lambda$  merupakan argumen dari fungsi tersebut. Selanjutnya, tanda titik merupakan pemisah antara *head* dan *body* fungsi lambda. Terakhir, setelah tanda titik adalah *body* dari suatu fungsi lambda yang mana berupa *expression*.

Untuk mempermudah pemahaman,  $\lambda$ -*calculus* dapat diperlakukan seperti fungsi tanpa nama. Sebagai contoh, fungsi lambda  $(\lambda x.x+5)$  apabila diberikan nilai 2 sehingga menjadi  $(\lambda x.x+5)2$  akan dievaluasi menjadi  $\lambda(2).(2)+5$ . Demikian itu, nilai yang dikembalikan oleh fungsi tersebut adalah 7. Sama seperti fungsi pada umumnya, konsep ini bernama *substitution* (substitusi). Memahami  $\lambda$ -*calculus* dirasa perlu berhubung dalam tugas akhir ini  $\lambda$ -*calculus* digunakan sebagai bentuk formal di *category* dalam konteks CCG *lexicon*. Meskipun  $\lambda$ -*calculus* tidak sesederhana yang dijelaskan sebelumnya, setidaknya memahami

$\lambda$ -calculus seperti ini sudah cukup untuk dapat membangun *supertagger* yang ada di tugas akhir ini.

## 2.5 Supertagging

*Supertagging* merupakan proses yang memetakan suatu token kata ke bentuk *supertag*-nya. CCG *supertagging* artinya proses pemetaan dari suatu token kata ke dalam bentuk CCG *supertag* atau dapat juga disebut sebagai CCG *lexicon*. *Supertag* mirip dengan POS *tag*. Perbedaannya, *supertag* memiliki bentuk formalisme yang lebih kompleks dari POS *tag*. Hal ini dikarenakan *supertag* menyimpan informasi lain selain tanda gramatikalnya (NP, NN, VB, dan sebagainya). Sebagai contoh, CCG *supertag* memiliki bentuk dengan format:

$$< \text{categorial grammar tag} > : < \text{semantic representation} >$$

Kita dapat menggunakan notasi  $\vdash$  untuk memetakan token kata ke bentuk CCG *supertag*-nya. Anggap kita memiliki kata kerja “menyukai” yang akan dipetakan ke  $(S \backslash NP) / NP : \lambda x. \lambda y. \text{suka}(y, x)$ , kita dapatkan:

$$\text{menyukai} \vdash (S \backslash NP) / NP : \lambda x. \lambda y. \text{suka}(y, x)$$

Adapun sebuah *tool* yang melakukan proses *supertagging* ini dinamakan *supertagger*. *Supertagger* sederhananya mengambil daftar token kata yang kemudian untuk setiap token kata tersebut “dilabelkan” dengan *supertag*-nya.

## 2.6 Maximum Entropy Model

*Maximum Entropy* (MaxEnt) merupakan model statistik yang dapat digunakan untuk melakukan *train* korpus teranotasi dengan Part-Of-Speech (POS) *tag*. Salah satu aplikasi MaxEnt adalah POS *tagger* yang mana memiliki hasil akurasi yang lebih baik ketimbang *state-of-the-art*. Adapun model probabilitasnya didefinisikan dalam  $\mathcal{H} \times \mathcal{T}$ , dimana  $\mathcal{H}$  adalah himpunan dari kemungkinan kata dan konteks *tag*, atau “*history*” (riwayat), dan  $\mathcal{T}$  adalah himpunan dari *tag* yang diizinkan. Model probabilitas dari suatu *history*  $h$  bersama dengan *tag*  $t$  didefinisikan dalam persamaan 2.2.

$$p(h, t) = \pi \mu \prod_{j=1}^k a_j^{f_j(h, t)} \quad (2.2)$$

dimana  $\pi$  merupakan konstan normalisasi,  $\{\mu, a_1, \dots, a_k\}$  merupakan parameter model positif, dan  $\{f_1, \dots, f_k\}$  merupakan apa yang kita sebut sebagai “*feature*”, dimana  $f_j(h, t) \in \{0, 1\}$ .

Meskipun MaxEnt dipernalkan untuk POS *tagging*, MaxEnt dapat pula digunakan untuk *supertagging*. Stephen Clark 2002 mempublikasikan literatur

*supertagging* untuk CCG yang mana MaxEnt merupakan model yang digunakan. Adapun persamaan modelnya dapat dilihat di persamaan 2.3.

$$p(c|h) = \frac{1}{Z(h)} e^{\sum_i \lambda_i f_i(c,h)} \quad (2.3)$$

dimana  $c$  merupakan *category*,  $h$  merupakan *context*, fungsi  $f_i(c, h)$  merupakan “*feature*” dari suatu *category* dan *context*, dan  $Z(h)$  merupakan konstan normalisasinya. Adapun contoh “*feature*” yang dimaksud dapat dilihat di persamaan 2.4.

$$f_j(c, h) = \begin{cases} 1 & , \text{if } \textit{merupakan\_kata\_yang}(h) = \textit{true} \ \& \ c = \text{NP/N} \\ 0 & , \text{selainnya} \end{cases} \quad (2.4)$$

## Bab III

# Metodologi dan Desain Sistem

### 3.1 Pembuatan Dataset

*Dataset* untuk melakukan *train* CCG sayangnya belum tersedia. Karena-nya, kita akan membuat *dataset* secara semi otomatis dengan memanfaatkan POS *tagger* untuk bahasa Indonesia. Secara formal, *flowchart* untuk proses pembuatan *dataset* dapat dilihat pada Gambar 3.1.

#### 3.1.1 Input Teks Bahasa Indonesia

Pada bagian *input* dalam pembuatan *dataset*, kita dapat mengambil teks bahasa Indonesia dari Indonesian Treebank<sup>1</sup> dan/atau dari beberapa contoh artikel yang terdapat di *website* Wikipedia Indonesia<sup>2</sup>. Indonesian Treebank terdapat setidaknya 1000 kalimat yang dirasa cukup untuk melakukan *training* menggunakan model MaxEnt. Apa yang perlu dilakukan setelah mengambil *input* adalah membersihkan bentuk *tree*-nya untuk kemudian diambilkan kalimat yang sebenarnya. Caranya cukup sederhana yaitu dengan mengambil *leaf* dari *tree* tersebut kemudian disatukan di suatu pengubah dengan tipe data *string*.

Sebagai contoh, salah satu *tree* yang terdapat di Indonesian Treebank dapat dilihat pada Gambar 3.2. Apa yang dimaksud dengan *leaf* pada *tree* tersebut adalah (*Kera*), (*untuk*), (*\**), (*amankan*), (*pesta olahraga*). Terkhusus untuk *leaf* dengan bentuk spesial, seperti (*\**), kita hilangkan sehingga teks yang diperoleh dari *tree* tersebut adalah “Kera untuk amankan pesta olahraga”. Selanjutnya, contoh teks yang akan digunakan agar konsisten yaitu “Pamungkas dan Setyo menyukai rendang”.

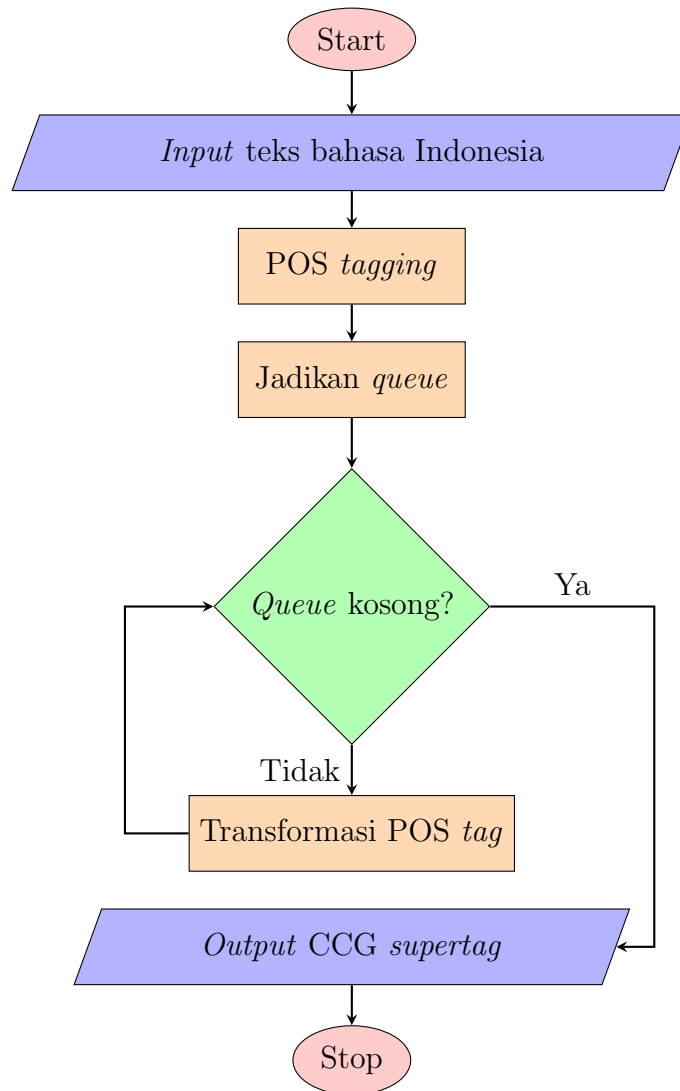
#### 3.1.2 POS Tagging

Berdasarkan teks bahasa Indonesia yang telah diambil, kita memanfaatkan *tool* POS *tagger* bahasa Indonesia untuk mendapatkan *lexical category* atomik untuk masing-masing token. Dengan memanfaatkan POS *tag* kita dapat membuat *dataset* untuk CCG *supertag* lebih mudah dibandingkan dengan

---

<sup>1</sup>[github.com/famrashel/idn-treebank/blob/master/Indonesian\\_Treebank.bracket](https://github.com/famrashel/idn-treebank/blob/master/Indonesian_Treebank.bracket)

<sup>2</sup>[id.wikipedia.org](https://id.wikipedia.org)



Gambar 3.1: Alur kerja pembuatan *dataset* untuk CCG *supertag*.

memberikan *tag* CCG secara manual. Ide dasarnya yaitu kita akan mentransformasikan POS *tag* yang didapatkan menjadi CCG *supertag* berdasarkan aturan-aturan khusus yang telah ditentukan. Sebagai contoh, kita dapat membuat aturan seperti mentransformasikan VB menjadi (S\NP)/NP. Dengan menggunakan contoh kalimat yang sama seperti di bagian sebelumnya, yaitu “Pamungkas dan Setyo menyukai rendang”, setelah menggunakan POS *tagger* bahasa Indonesia kita dapatkan hasil sesuai dengan Gambar 3.3.

### 3.1.3 Transformasi POS Tag

Pada bagian proses transformasi POS *tag* ke bentuk CCG *supertag*-nya, kita buat aturan-aturan transformasinya. Aturan transformasi tersebut

$(NP$   
 $(NN (Kera))$   
 $(SBAR$   
 $(SC (untuk))$   
 $(S (NP - SBJ (*))$   
 $(VP$   
 $(VB (amankan))$   
 $(NP (NN (pesta olahraga))))))$

Gambar 3.2: Salah satu contoh *tree* dalam Indonesian Treebank.

Pamungkas	dan	Setyo	menyukai	rendang
NNP	CC	NNP	VB	X

Gambar 3.3: Kalimat contoh dengan POS *tag*-nya.

merupakan pemetaan berbasis aturan. Sebagai contoh, kata “menyukai” memiliki POS *tag* VB. Anggap saja dalam aturan transformasi terdapat pemetaan  $VB \vdash (S \backslash NP) / NP$ . Sehingga, kita dapatkan CCG *supertag* untuk “menyukai” yaitu  $(S \backslash NP) / NP$ . Kendati demikian, masih ada bagian yang belum kita dapatkan yaitu *semantic representation*-nya. Kita dapat menggunakan *stemmer* bahasa Indonesia agar mendapatkan *root words* dari kata “menyukai” yaitu “suka”. Langkah terakhirnya adalah membuat *semantic representation*-nya berdasarkan *root words* yang telah diperoleh sehingga kita dapatkan fungsi lambdanya yaitu  $\lambda x. \lambda y. \text{suka}(y, x)$ .

Selain menggunakan *stemmer*, kita dapat menggunakan *morphological analyzer*. Untuk bahasa Indonesia, kita dapat menggunakan *tool* bernama MorphInd<sup>3</sup>. *Morphological analyzer* salah satu kegunaannya yaitu dapat menghasilkan *root words* sehingga dapat kita manfaatkan untuk membuat fungsi lambda. Namun, kita dapat menggunakan MorphInd sebagai pelengkap POS *tagger* untuk bahasa Indonesia. Hal ini agar *dataset* yang dibuatkan secara semi-otomatis ini dapat memiliki kualitas yang baik.

### 3.1.4 Output CCG Supertag

Keluaran dari bagian pembuatan *dataset* ini adalah sebuah berkas JSON (JavaScript Object Notation) berisi CCG *supertag* lengkap dan beberapa berkas JSON dari CCG *supertag* yang lemanya dikelompokkan berdasarkan alfabet. Adapun format JSON dari *dataset* yang disimpan dalam berkas tersebut dapat dilihat pada Gambar 3.4. Untuk setiap kalimat dalam berkas tersebut direp-

<sup>3</sup>[septinalarasati.com/morphind](http://septinalarasati.com/morphind)

```
[
  {
    "tokens": ["token1,1", "token1,2", ..., "token1,m"],
    "supertags": ["supertag1,1", "supertag1,2", ..., "supertag1,m"]
  },
  {
    "tokens": ["token2,1", "token2,2", ..., "token2,m"],
    "supertags": ["supertag2,1", "supertag2,2", ..., "supertag2,m"]
  },
  ⋮
  {
    "tokens": ["tokenn,1", "tokenn,2", ..., "tokenn,m"],
    "supertags": ["supertagn,1", "supertagn,2", ..., "supertagn,m"]
  }
]
```

Gambar 3.4: Format JSON *dataset* yang disimpan di dalam berkas.

representasikan oleh dua objek yaitu (1) “tokens” berupa daftar token, dan (2) “supertags” berupa daftar *supertag* untuk token ke- $(i, j)$  dimana  $1 \leq i \leq n$  dan  $1 \leq j \leq m$  dalam sistem *1-indexed array*. Sebagai contoh, dengan kalimat yang sama dengan sebelum-sebelumnya, dapat dilihat di Gambar 3.5. Demikian itu, kita dapat mengambil *dataset* secara lengkap dengan CCG *supertag*-nya.

### 3.2 Melatih Model Supertagger

Model yang akan digunakan oleh *supertagger* ini adalah Maximum Entropy (MaxEnt). Model ini dipilih karena ketersediaan *dataset* bahasa Indonesia yang masih sangat kurang. Selain itu, model MaxEnt digunakan di sebuah riset yang dilakukan oleh Stephen Clark dalam pembuatan *supertagger*-nya. Bahkan, performansi *supertagger* yang dikembangkan sangat baik. Riset tersebut pada intinya membuktikan bahwasannya MaxEnt dapat digunakan di *supertagger* juga meskipun pada peruntukannya MaxEnt dibuat untuk POS *tagger*. Tentunya terdapat beberapa penyesuaian yang harus dilakukan. Salah satunya adalah formula probabilitas yang digunakan. Formula yang telah disesuaikan tersebut dapat dilihat di persamaan 2.3.



```
[
  {
    "tokens": [
      "Pamungkas",
      "dan",
      "Setyo",
      "menyukai",
      "rendang",
      "."
    ],
    "supertags": [
      "NP: pamungkas'",
      "CONJ:  $\lambda x.\lambda y.\lambda f. (f\ x) \wedge (f\ y)$ ",
      "NP: setyo'",
      "(S\NP)/NP:  $\lambda x.\lambda y. suka(y, x)$ ",
      "NP: rendang'",
      "Z"
    ]
  }
]
```

Gambar 3.5: Contoh isi dari berkas *dataset* dalam format JSON.

## Daftar Pustaka

- [1] Raul Rojas. A tutorial introduction to the lambda calculus. *CoRR*, abs/1503.09060, 2015.
- [2] Mark Steedman. Categorical grammar. Technical report, 1992.

## Lampiran