BAB 2
LANDASAN TEORI

2.1 Bahasa Inggris

Bahasa Inggris adalah bahasa Jermanik Barat, yang berasal dari Inggris. Bahasa ini merupakan kombinasi antara beberapa bahasa lokal yang dipakai oleh orang-orang Norwegia, Denmark, dan Anglo-Saxon dari abad ke-6 sampai 10. Lalu pada tahun 1066 dengan ditaklukkan Inggris oleh William the Conqueror sang penakluk dari Normandia, Perancis Utara, maka bahasa Inggris dengan sangat intensif mulai dipengaruhi bahasa Latin dan bahasa Perancis. Dari seluruh kosakata bahasa Inggris modern, diperkirakan 50% berasal dari bahasa Perancis dan Latin.

Bahasa Inggris adalah media komunikasi utama bagi masyarakat di negara Inggris, Amerika Serikat, Kanada, Australia, New Zealand, Afrika Selatan, dan di banyak negara lainnya. Bahasa Inggris merupakan bahasa resmi dari banyak negara-negara persemakmuran dan dipahami serta dipergunakan secara meluas. Bahasa Inggris dipergunakan di lebih banyak negara di dunia dibanding bahasa yang lain kecuali bahasa Cina, bahasa ini juga lebih banyak dipergunakan orang.

Bahasa Inggris termasuk rumpun bahasa-bahasa Anglo-Frisia pada cabang barat bahasa-bahasa Jerman, dan merupakan sebuah bahasa subfamili dari bahasa-bahsa Indo-Eropa. Bahasa Inggris hampir mendekati bahasa Frisia, sedikit lebih luas dari bahasa Netherlandic (Belanda–Flemish) dan dialek Jerman tingkat rendah (Plattdeutsch), serta jauh dari bahasa Jerman Modern tingkat tinggi [3].

2.1.1 *Grammar*

Grammar adalah himpunan dari aturan-aturan yang terstruktur yang mengatur susunan kalimat, frase, dan kata dalam bahasa apapun. Di dalam tata bahasa Inggris ada tiga dasar tata bahasa, yaitu etimologi, orthographi, dan sintaks.

1. Etimologi

   Etimologi yaitu ilmu yang mempelajari tentang cara penyusunan huruf-huruf bahasa inggris yang tepat ke dalam sebuah kata bahasa Inggris. Dalam Etimologi, kata-kata bahasa inggris dikelompokkan ke dalam 8 bagian atau yang disebut dengan *the Eight Parts of Speech*. Ke delapan bagian tersebut adalah *noun, adjective, verb, pronoun,adverb,preposition, conjuction, dan interjection*

2. Orthographi

   Orthographi merupakan sistem ejaan suatu bahasa atau gambaran bunyi bahasa yang berupa tulisan atau lambang. Ortografi antara lain meliputi masalah ejaan, kapitalisasi, pemenggalan kata, serta tanda baca. Materi yang dibahas dalam orthography, yaitu:

   a. *Letter* (huruf), merupakan lambang dari bunyi atau dalam arti yang lain disebutkan bahwa letter adalah tanda goresan yang diucapkan atau dibunyikan.
   b. *Word*, merupakan rangkaian dari huruf-huruf yang membentuk sebuah arti, misalnya : *take* (mendapatkan), *year* (tahun).
   c. *Syllable* (suku kata), merupakan bagian-bagian dari sebuah kata yang dapat langsung diucapkan, misalnya *glass*, *book, clever*.

3. Sintaks

   Sintaks, adalah ilmu yang mempelajari tentang cara penyusunan kata kata bahasa Inggris menjadi sebuah kalimat bahasa Inggris yang tepat dan sesuai atau dapat juga dikatakan bahwa syntax merupakan bagian dari tata bahasa yang mempelajari tentang proses pembentukan sebuah kalimat [4].

2.1.2 Frase *(Phrase)*

Frase (*Phrase*) adalah rangkaian kata yang bermakna tapi tidak berpola S + V. Frase (Phrase) ini sangat berkaitan dengan *Word Order* yang di dalamnya terdapat Head (H) dan Modifier (M) (dalam bahasa Indonesia dikenal dengan

istilah Dijelaskan dan Menjelaskan). Ada sembilan frase dalam bahasa Inggris yaitu sebagai berikut:

1. *Noun Phrase* adalah frase yang *head* (inti)-nya berupa noun dan *modifier*-nya bisa berupa *adjective* atau *noun*.

   Contoh:

   - Old car.
     Adj. N
     M H
     
   - Book store.
     N N
     M H
     
   - Someone special.
     Pron. Adj.
     H M
     

2. *Verb Phrase* adalah frase yang head (inti)-nya berupa *verb* (*ordinary verb*) dan modifier-nya bisa berupa *auxilary verb*, *adverb* atau *object of transtive verb*.

   Contoh:

   - Am studying.
     Aux. Ord.
     M H
     
   - Study hard.
     Ord. Adv.
     H M
     

3. *Adjective Phrase* adalah frase ynag head (inti)-nya berupa *adjective* dan *modifier*-nya berupa *adverb*.

Contoh:

- So good.
  Adv. Adj.
  M H



- Good enough
  Adj. Adv.
  H M



4. *Adverbial Phrase* adalah frase yang *hea*d (inti)-nya berupa *adverb* dan *modifier*-nya berupa *adverb*.
   Contoh:
   - So beautifully.
     Adv. Adv.
     M H
   - Seriously enough.
     Adv. Adv.
     H M





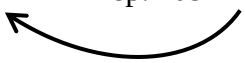
5. *Prepositional Phrase* adalah frase yang *head* (inti)-nya berupa preposisi dan *modifier*-nya berupa *object of preposition* (*noun, pronoun, noun phrase* dan lain-lain).
   Contoh:
   - With you.
     Prep. Pron.
     H M



   - Because of you.
     Prep. Pron.
     H M



6. *Gerund Phrase* adalah frase yang *head* (inti)-nya berupa *gerund* dan *modifier*-nya berupa *object of gerund* atau *adverb*.
   Contoh:
   - Studying English.
     G N
     H M
     
   - Working hard.
     G Adv.
     H M
     

7. *Participle Phrase* adalah frase yang *head* (inti)-nya berupa *participle* (baik berupa present participle [Ving] maupun past participle [V3]) dan *modifier*-nya berupa *object of participle* atau *adverb* (*prepositional phrase*).
   Contoh
   - Known as a writer.
     Ps.Part. H NP as M
     H Prep.P as M
     
   - Climbing the tree.
     Pr.Part. M H
     H NP as M
     

8. *Infinitive Phrase* adalah frase yang *head* (inti)-nya berupa ‘to’ dan *modifier*-nya berupa *bare infinitiv*e (V1).
   Contoh:
   - To help.
     To inf. V1
     H M
     

9. *Exclamatory Phrase* adalah frase yang terbentuk dari pola "*What + Noun Phrase*" atau " *How + Adjectiv*e!". Fungsi *exclamatory phrase* adalah untuk menunjukkan perasaan yang kuat seperti kagum [5].

   Contoh:

   - What a beautiful girl!

     M M H

     What NP

2.1.3 Klausa (*Clause*)

Definisi menurut Ruggiero (1981): klausa adalah sekelompok kata yang mengandung subyek dan predikat. Klausa mungkin independen dari bagian kalimat yang lain (karenanya mempu berdiri sendiri sebagai kalimat yang terpisah) atau dependen atas klausa yang lain. Contohnya, dalam kalimat "*If it snows this weekend, I'm going skiing*," lima kata pertama membentuk satu klausa, tiga kata terakhir membentuk yang lain. Tetapi hanya klausa yang kedua adalah independen; yang pertama dibuat dependen dengan kata *if*. ("If it snows this weekend" adalah pikiran yang tidak lengkap.) Klausa dependen diklasifikasikan oleh fungsi, yakni oleh kelas kata. Maka dari itu, ada klausa nomina, klausa adjektiva, dan klausa adverbia.

Klausa nomina diperkenalkan oleh kata seperti *that, what, why, whether, who, which,* dan *how*. Contohnya, "*Whether I go to the party or stay at home* makes no difference to me." (Klausa yang dicetak miring adalah subyek dari verba *makes*.) Dan "All I ask is *that you listen to my ideas on the subject*." (Klausa yang dicetak miring adalah nomina predikat.)

Klausa adjektiva diperkenalkan oleh kata seperti *who, whom, whose, which,* dan *that*. Contoh, "The boy *who sits next to me in class* is my roommate's cousin." (Klausa yang dicetak miring menerangkan nomina *boy*.)

Klausa adverbia diperkenalkan oleh kata yang menunjukkan waktu (*when*), tempat (*where*), sebab (*because*), maksud (*so that*), keadaan (*unless*),

konsensi (*although*), cara (*as if*), dan perbandingan (*than*). Contoh, "You may wash the car *whenever you wish*." Dan "*Because inflation had made it impossible for me to save any money*, I'll have to give up my Bermuda vacation this year." Dan "He sacrificed *so that his children could have a better life than he had* [5]."

2.1.4 Kalimat (Sentence)

Kalimat, secara gramatika sekelompok kata yang bebas—karenanya sekelompok kata yang secara struktural lengkap. Kalimat mungkin mengungkapkan pernyataan atau seruan, bertanya suatu pertanyaan, atau mengungkapkan perintah. Ada empat tipe dasar struktur kalimat dalam Bahasa Inggris.

2.1.4.1 Kalimat Sederhana (Simple)

Kalimat simpel mengandung klausa independen tunggal.

Contoh:

The cat is asleep.

What a beautiful girl! Walaupun seruan tersebut tak memiliki predikat yang terungkap, maknanya jelas menyiratkan itu dan ini adalah cara dramatik berujar. "*That girl is beautiful*."

Are your roommates going to the movies with us tonight?

Please clean your closet! Walaupun perintah tersebut tidak memiliki subyek yang terungkap, maknanya jelas menyiratkan itu—"*You please clean your closet*."

Perhatikan bahwa sebuah kalimat tetap kalimat sederhana meski subyek, verba, atau keduanya dimajemukkan.

Contoh:

The cat and the dog fought.

The dog barked and growled.

The cat and the dog snarled and fought.

2.1.4.2 Kalimat Majemuk

Kalimat majemuk mengandung dua atau lebih klausa independen.

Contoh:

I may decide to forgive Mary for her behavior at the party, but I certainly won't forget the embarrassment she caused me. Bahwa dua klausa itu independen terbukti karena koma dapat diubah ke titik, menciptakan dua kalimat lengkap.

I may decide to forgive Mary for her behavior at the party. I certainly won't forget the embarrassment she caused me.

John went to Bermuda for a week, and Dave went to Florida for ten days, but I stayed home for the entire semester break. Kalimat ini mengandung tiga klausa independen.

2.1.4.3 Kompleks

Kalimat kompleks mengandung satu klausa independen dan satu atau lebih klausa dependen.

When the movie is over, let's go for a pizza.

If my sister doesn't visit me this weekend, I'm going to begin my term paper.

That story about Martha isn't true, although I must admit it's plausible.

2.1.4.4 Kalimat Majemuk-Kompleks

Kalimat majemuk-kompleks mengandung dua atau lebih klausa independen dan sedikitnya satu klausa dependen [5].

Contoh:

If you can't relive the past, it makes no sense to brood over it; learn from it and then forget it.

Sally can't seem to live without him, and yet every time she's with him they quarrel incessantly; so unless she wants a lifetime of happiness, she'd better think twice before marrying him.

2.1.5 *Tenses*

Tenses merupakan suatu bentuk kata kerja dalam bahasa Inggris untuk menunjukkan waktu (sekarang, masa depan, atau masa lalu) terjadinya suatu

perbuatan atau peristiwa. *Tenses* terdiri dari empat macam waktu yaitu *Present Tense* (Masa Kini), *Past Tense* (Masa Lalu), *Future Tense* (Masa yang Akan Datang), dan *Past Future Tense* (Masa Kini Untuk Dikerjakan Dimasa depan). Dari empat macam tenses yang telah dipaparkan sebelumnya, terbagi kembali menjadi empat jenis pekerjaan yang akan dilakukan yaitu *simple* (Sederhana), *Continuous* (yang sedang berlangsung), *Perfect* (telah terjadi), dan *Perfect Continuous* (telah dan sedang terjadi) [5].

2.1.5.1 *Simple Present Tense*

Digunakan untuk pekerjaan yang berulang ulang misalnya dilakukan setiap hari setiap bulan dan sebagainya. Adapun rumusnya adalah :

Positif : *Subjec*t + *Verb I* (s, es)

Negatif : *Subject* + *do/does* + not + *Verb I*

Interrogatif : *do/does* + *Subject* + *Verb I* ?

Contoh :

She goes to office every day.

2.1.5.2 *Simple Past Tense*

Digunakan untuk pekerjaan yang telah dilakukan pada masa lampau. Adapun rumusnya adalah :

Positif : Subject + Verb II

Negatif : Subject + did + not + Verb I

Interrogatif : Did + Subject + Verb I ?

Contoh :

I visited Singapore one year ago.

2.1.5.3 *Simple Future Tense*

Digunakan untuk pekerjaan yang akan dilakukan pada masa mendatang. Adapun rumusnya adalah :

Positif : *Subject + am/is/are+going to +Verb I*

Negatif : *Subject + am/is/are+ not + going to +Verb I*

Interrogatif : *Am/is/are+Subject+going to +Verb I* ?

Contoh :

I am going to teach tonight

To be bisa juga diganti dengan Will. (I will teach tonight)

2.2 *Part Of Speech Tagging*

Part-of-speech tagging adalah sebuah sistem yang memberikan label kata secara otomatis pada suatu kalimat. Pengelompokkan kata dengan Penn Treebank *POS Tag* dapat dilihat pada tabel 2.1.[11]

Tabel 2.1 Pengelompokan Kata dengan *POS Tag*

No	Tag	Deskripsi	Contoh
1	CC	*Coordinating Conjuction*	*and, but, or*
2	CD	*Cardinal Number*	*one, two, three*
3	DT	*Determiner*	*the*
4	EX	*Existential 'there'*	*there*
5	ARTICLE	*Existential*	*a, an*
6	IN	*Preposition/sub-conj*	*of, in, by*
7	JJ	*Adjective*	*yellow, old, happy*
8	MD	*Modal*	*can, should*
9	NN	*Noun, Singular Or Mass*	*monkey, tomatoe*
10	NNS	*Noun, Plural*	*monkeys, tomatoes*
11	NNP	*Proper Noun, Singular*	*The Himalaya*
12	NNPS	*Proper Noun, Plural*	*Americans*
13	PP1	*Personal Pronoun 1*	*i, you, they, we*
14	PP2	*Personal Pronoun 2*	*she, he , it*
15	RB	*Adverb*	*quickly, never*
16	TO	*"to'*	*to*
17	VB	*Verb, Base Form*	*eat*

18	VBB	*Verb Be*	*am, are, is*
19	VBD	*Verb, Past Tense*	*ate*
20	VDB	*Verb Be Past Tense*	*were ,was*
21	VBG	*Verb, Gerund*	*eating*
22	VBN	*Verb, Past Participle*	*eaten*
23	VBZ	*Verb, 3rd person singular*	*eats*
24	WH	*Ask 5W+1H*	*what,where,why*
25	X	Kata yang tidak diketahui kategorinya	schol, eet
26	Z	*Sentence-final punc*	., !, ?

2.3 Teori Bahasa dan Automata

Tata bahasa (*grammar*) bisa didefinisikan secara formal sebagai kumpulan dari himpunan-himpunan variabel, simbol,-simbol terminal, simbol awal, yang dibatasi oleh aturan-aturan produksi. Pada tahun 1959 seorang ahli bernama Noam Chomsky melakukan penggolongan tingkatan bahasa menjadi empat, yang disebut dengan *Hirarki Chomsky*.

Tabel 2.2 Tabel penggolongan tingkatan bahasa menurut *Chomsky*

Bahasa	Mesin Otomata	Batasan Aturan Produksi
Natural Language	Mesin Turing	Tidak ada batasan
Context Sensitive	*Linear Bounded Automata*	$\lvert \alpha \rvert \leq \lvert \beta \rvert$
Bebas Konteks	Push Down Automata	α berupa sebuah simbol variabel.
Regular	*Finite State Automata* (FSA) meliputi *Deterministic Finite Automata* (DFA) & *Nondeterministic Finite Automata* (NFA)	α adalah sebuah simbol variabel. β maksimal memiliki sebuah simbol variabel yang bila ada terletak di posisi paling kanan.

Semua penggolongan diatas berdasarkan pembatasan yang dilakukan pada aturan produksinya. Aturan produksi merupakan pusat dari tata bahasa, yang menspesifikasikan bagaimana suatu tata bahasa melakukan transformasi suatu *string* ke bentuk lainnya. Melalui aturan produksi tersebut didefinisikan suatu bahasa yang berhubungan dengan tata bahasa tersebut. Semua aturan produksi dinyatakan dalam bentuk "α → β" (dibaca: α menghasilkan β atau α menurunkan β). Dimana α menyatakan simbol-simbol pada ruas kiri aturan produksi dan β menyatakan menyatakan simbol-simbol pada ruas kanan aturan produksi (disebut juga sebagai hasil produksi). Simbol-simbol tersebut dapat berupa simbol terminal atau variabel. Simbol variabel masih bisa diturunkan sedangkan simbol terminal sudah tidak dapat diturunkan lagi. Simbol variabel dinyatakan dalam huruf besar sedangkan simbol terminal dinyatakan dengan huruf kecil [6].

Berdasarkan tabel 2.2 pembahasan lebih diarahkan ke bahasa bebas konteks karena algoritma CYK hanya mengenali aturan produksi dari *Chomsky Normal Form* (CNF), sedangkan CNF berawal dari *Context Free Grammar* (CFG) atau tata bahasa bebas konteks.

2.3.1 Tata Bahasa Bebas Konteks (TBBK)

TBBK atau CFG adalah suatu cara untuk menunjukkan bagaimana menghasilkan untai-untai dalam sebuah bahasa. Suatu CFG terdiri dari himpunan simbol terminal dan simbol variabel. Pada CFG aturan produksinya dapat dilihat pada aturan produksi berikut.

$$\alpha \rightarrow \beta$$

Dimana α merupakan sebuah simbol variabel dan β merupakan hasil produksi dari α. Contoh aturan produksi yang dapat dibuat dari CFG dapat dilihat pada aturan produksi berikut .

$$S \rightarrow aBCde$$

$$B \rightarrow fGe$$

Dimana S merupakan variabel awal dan aBCde merupakan hasil produksi dari variabel S begitu juga dengan variabel B yang merupakan turunan dari variabel S. Saat menurunkan suatu *string*, simbol-simbol variabel akan mewakili bagian-bagian yang belum terturunkan dari *string* tersebut. Pada CFG letak bagian yang belum

terturunkan bisa dimana saja. Saat penurunan itu telah lengkap, semua bagian yang belum terturunkan telah diganti oleh *string-string* (yang mungkin saja kosong) dari himpunan simbol terminal. Tata bahasa bebas konteks menjadi dasar dalam pembentukan suatu proses analisis sintaksis.

2.3.2 Penyederhanaan Tata Bahasa Bebas Konteks

Penyederhanaan ini bertujuan untuk melakukan pembatasan sehingga tidak menghasilkan aturan produksi yang rumit dan tidak berguna. Suatu tata bahasa bebas konteks dapat disederhanakan dengan cara:

1. Menghilangkan Produksi *Useless* (tidak berguna)

   Produksi *useless* merupakan produksi yang memuat simbol variabel yang tidak memiliki penurunan yang akan menghasilkan terminal-terminal. Produksi ini tidak berguna karena bila diturunkan tidak akan pernah selesai. Contoh dapat dilihat pada aturan produksi berikut.

   $$S \rightarrow Aa \mid B$$
   $$A \rightarrow ab \mid D \mid \varepsilon$$
   $$B \rightarrow b \mid E \mid F$$
   $$C \rightarrow bb$$
   $$F \rightarrow G$$
   $$G \rightarrow bc$$

   Berdasarkan contoh diatas dapat dilihat bahwa variabel D dan E tidak memiliki penurunan yang menuju terminal, sehingga variabel ini dapat dihilangkan. Konsekuensinya semua variabel D dan E dihapus dari aturan produksi. Variabel C juga dapat dihapus karena tidak ada variabel yang mengarah ke variabel C. Hasil penyederhanaannya dapat dilihat pada aturan produksi berikut.

   $$S \rightarrow Aa \mid B$$
   $$A \rightarrow ab \mid \varepsilon$$
   $$B \rightarrow b \mid F$$
   $$F \rightarrow G$$
   $$G \rightarrow bc$$

2. Menghilangkan Produksi Unit

   Produksi unit adalah produksi dimana ruas kiri dan kanan aturan produksi hanya berupa satu simbol variabel. Keberadaan produksi unit membuat tata bahasa memiliki kerumitan yang tak perlu atau menambah panjang penurunan. Penyederhanaan ini dilakukan dengan cara mengganti aturan produksi unit. Hasil penyederhanaan dapat dilihat pada aturan produksi berikut.

   $$S \rightarrow Aa \mid B$$
   $$A \rightarrow ab \mid \varepsilon$$
   $$B \rightarrow b \mid F$$
   $$F \rightarrow G$$
   $$G \rightarrow bc$$

   Berdasarkan contoh aturan produksi sebelumnya, variabel B memiliki turunan F, variabel F memiliki turunan G dan variabel G memiliki turunan bc, sehingga dapat digambarkan bahwa B → F → G → bc. Aturan ini dapat disederhanakan dengan cara mengganti aturan produksi dari variabel B menjadi bc. Hasil penyederhanaannya dapat dilihat pada aturan produksi berikut.

   $$S \rightarrow Aa \mid B$$
   $$A \rightarrow ab \mid \varepsilon$$
   $$B \rightarrow b \mid bc$$

3. Menghilangkan Produksi Kosong (ε)

   Produksi ε atau produksi kosong dapat dihilangkan, karena produksi ini tidak menghasilkan apa-apa. Contoh aturan produksi dapat dilihat pada aturan produksi berikut.

   $$S \rightarrow Aa \mid B$$
   $$A \rightarrow ab \mid \varepsilon$$
   $$B \rightarrow b \mid bc$$

Pada aturan produksi sebelumnya variabel A mengandung produksi kosong sehingga dapat ihilangkan. Hasil penyederhanaannya dapat dilihat pada aturan produksi berikut.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow ab$$
$$B \rightarrow b \mid bc$$

Apabila terdapat kasus seperti aturan produksi berikut.

$$S \rightarrow Aa \mid B$$
$$A \rightarrow \varepsilon$$
$$B \rightarrow b \mid bc$$

Semua variabel A yang ada didalam aturan produksi dihapus karena variabel A mengandung produksi kosong. Hasil penyederhanannya dapat dilihat pada aturan produksi berikut.

$$S \rightarrow a \mid B$$
$$B \rightarrow b \mid bc$$

2.3.3 *Chomsky Normal Form* (CNF)

CNF merupakan salah satu bentuk normal yang sangat berguna untuk CFG. Bentuk CNF dapat dibuat dari sebuah CFG yang telah mengalami penyederhanaan dengan menghilangkan produksi *useless*, unit, dan *empty* (himpunan kosong yang dilambangkan dengan ε).

Aturan produksi dalam bentuk normal *Chomsky* ruas kanannya tepat berupa sebuah terminal atau dua variabel. Contoh aturan produksi dapat dilihat pada 2.11.

$$A \rightarrow BC$$
$$A \rightarrow b$$
$$B \rightarrow a$$
$$C \rightarrow BA \mid d$$

Langkah-langkah dalam pembentukan CNF yaitu:

1. Biarkan aturan produksi yang sudah dalam bentuk CNF.

2. Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan > 1.
3. Lakukan penggantian aturan produksi yang ruas kanannya memuat > 2 simbol variabel.
4. Penggantian dapat dilakukan berkali-kali sampai semua aturan produksi berada dalam bentuk CNF.
5. Selama dilakukan penggantian, kemungkinan akan memperoleh aturan-aturan produksi baru dan memunculkan simbol-simbol variabel baru.

Contoh CFG yang sudah disederhanakan dapat dilihat pada aturan produksi 2.12.

$$S \rightarrow bC \mid aB \mid AB$$
$$A \rightarrow bAC \mid a$$
$$B \rightarrow aCB \mid b$$
$$C \rightarrow cBA \mid c$$

Pada bentuk CFG diatas, aturan produksi yang sudah dalam bentuk CNF adalah $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$, dan $C \rightarrow c$. Selebihnya harus dilakukan penggantian aturan produksi. Hasil penggantian dapat dilihat pada aturan produksi 2.13.

$S \rightarrow bC$	menjadi	$S \rightarrow V_1C$
$S \rightarrow aB$	menjadi	$S \rightarrow V_2B$
$A \rightarrow bAC$	menjadi	$A \rightarrow V_1AC$
$A \rightarrow V_1AC$	menjadi	$A \rightarrow V_1V_3$
$B \rightarrow aCB$	menjadi	$B \rightarrow V_2CB$
$B \rightarrow V_2CB$	menjadi	$B \rightarrow V_2V_4$
$C \rightarrow cBA$	menjadi	$C \rightarrow V_5BA$
$C \rightarrow V_5BA$	menjadi	$C \rightarrow V_5V_6$

Aturan produksi baru dapat dilihat pada aturan 2.14.

$$V_1 \rightarrow b$$

$V_2 \rightarrow a$

$V_3 \rightarrow AC$

$V_4 \rightarrow CB$

$V_5 \rightarrow c$

$V_6 \rightarrow BA$

Hasilnya dapat dilihat pada aturan produksi 2.15.

$S \rightarrow V_1C$

$S \rightarrow V_2B$

$S \rightarrow AB$

$A \rightarrow V_1V_3$

$A \rightarrow V_2$

$B \rightarrow V_2V_4$

$B \rightarrow V_3$

$C \rightarrow V_5V_6$

$C \rightarrow V_3$

$V_1 \rightarrow b$

$V_2 \rightarrow a$

$V_3 \rightarrow AC$

$V_4 \rightarrow CB$

$V_5 \rightarrow c$

$V_6 \rightarrow BA)$

2.4 Algoritma *Cocke-Younger-Kasami* (CYK)

Algoritma CYK merupakan algoritma *parsing* dan keanggotaan (*membership*) untuk tatabahasa bebas konteks. Syarat untuk menggunakan algoritma ini adalah tata bahasa harus berada dalam bentuk CNF. Tujuan dari algoritma ini adalah untuk menunjukkan apakah suatu *string* dapat diterima dari suatu tata bahasa.

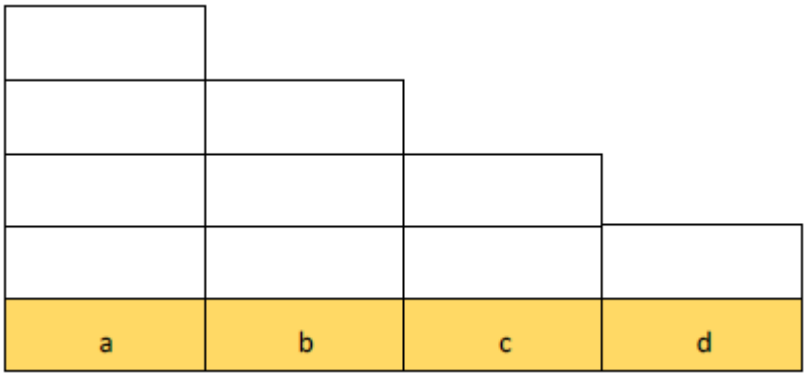
Proses *parsing* untai dengan algoritma CYK menggunakan struktur data sebuah *array* dua dimensi dengan jumlah baris dan kolom n x n. Nilai n ditentukan dari jumlah kata dalam suatu kalimat. Proses *parsing* dari algoritma ini

memanfaatkan hasil *parsing* sebelumnya untuk menentukan apakah proses yang sedang berlangsung dapat diterima atau tidak.

Sebagai contoh terdapat suatu aturan produksi 2.16 dengan variabel awal S.

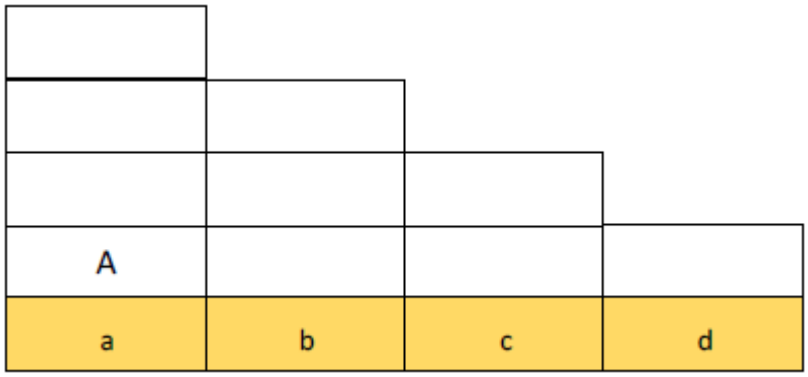
S → EF | EC
E → AB
F → CD
A → a
B → b
C → c
D → d

String yang akan diperiksa misalnya "abcd". Langkah pertama yang dilakukan adalah dengan membuat tabel dua dimensi seperti gambar 2.1.



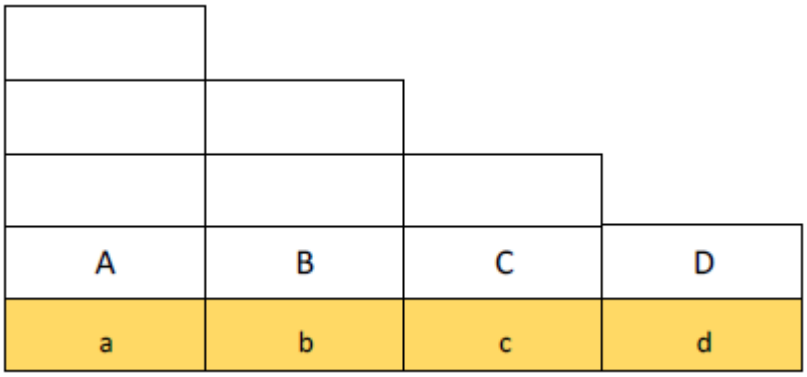
Gambar 2.1. Pembentukan tabel *parsing* algoritma CYK

Setiap kata yang ada pada suatu *string* ditempatkan pada masing-masing blok seperti gambar 2.1. Setiap blok yang kosong diisi berdasarkan variabel yang sesuai dengan aturan produksi dari CNF yang sudah dibentuk sebelumnya. Sebagai contoh pada baris pertama dan kolom pertama diisi dengan variabel yang dapat menghasilkan huruf a, yaitu variabel A. Hasilnya dapat dilihat pada gambar 2.2.



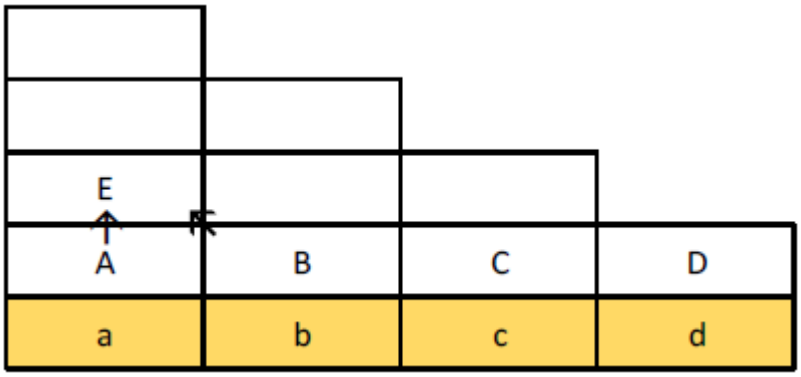
Gambar 2.2. Pengisian tabel dengan variabel A

Blok selanjutnya diisi dengan langkah yang sama, sehingga dapat dilihat baris pertama isiannya seperti gambar 2.3.



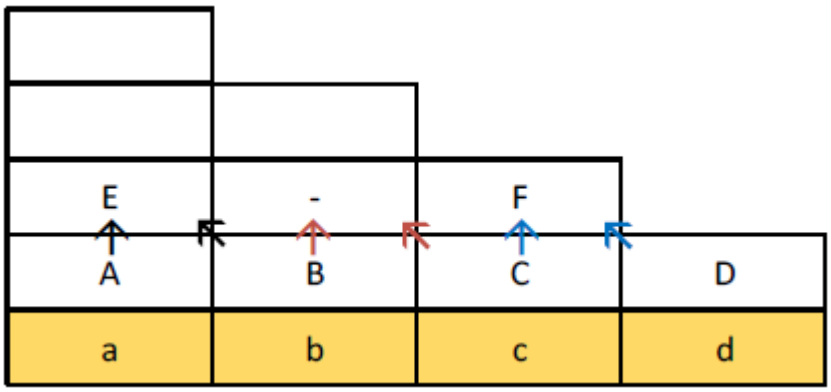
Gambar 2.3. Pengisian tabel baris pertama

Pada baris selanjutnya cara pengisian dilakukan dengan melihat variabel pada dua blok, kemudian dilihat variabel mana yang menghasilkan kedua variabel tersebut. Sebagai contoh baris kedua kolom pertama diisi dengan melihat variabel pada blok pertama dan blok dua, yaitu variabel A dan B yang termasuk ke dalam variabel E, sehingga cara pengisiannya terlihat seperti pada gambar 2.4.



Gambar 2.4. Pengisian tabel dengan variabel B

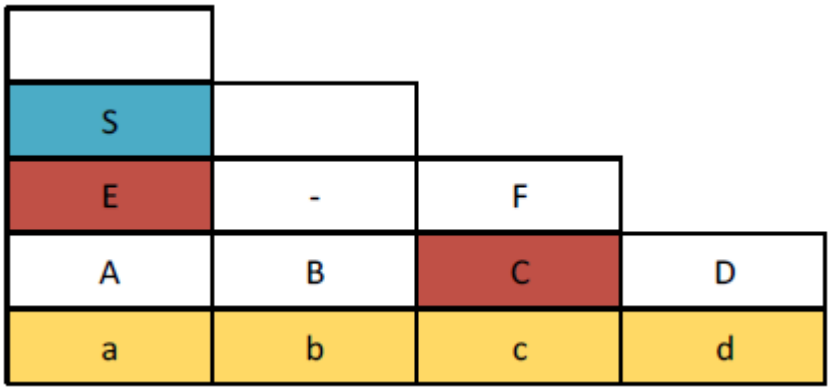
Blok selanjutnya diisi dengan langkah yang sama sesuai dengan blok yang akan diisi. Apabila pengisian dilakukan pada blok kedua, variabel diambil dari kolom kedua dan ketiga. Untuk lebih jelasnya dapat dilihat pada gambar 2.5.



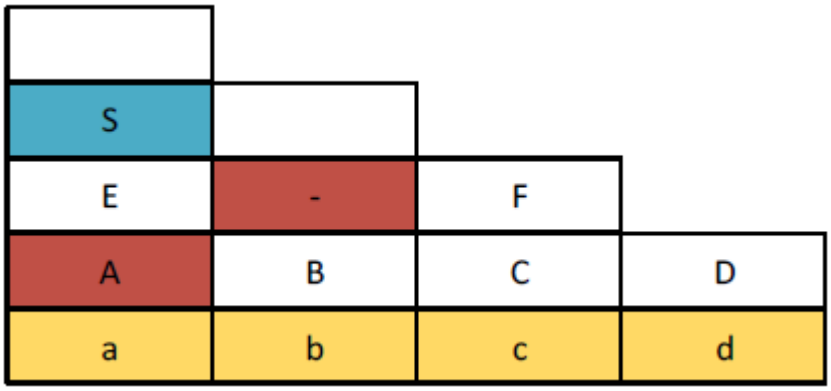
Gambar 2.5. Pengisian tabel baris kedua

Pengisian pada baris ketiga dan selanjutnya hampir sama seperti baris kedua, namun terdapat perbedaan dari cara pengecekkan yang dilakukan.

Pengisian baris ketiga dilakukan dari baris pertama hingga sebelum baris ketiga (n-1). Cara pengecekkan dapat dilihat seperti gambar 2.6 dan 2.7.

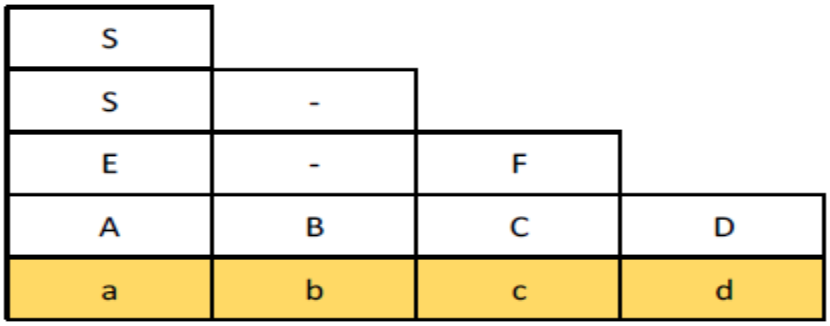


Gambar 2.6. Cara pengecekkan langkah pertama



Gambar 2.7. Cara pengecekkan langkah kedua

Warna biru menunjukkan blok yang akan diisi, sedangkan warna merah merupakan blok yang dicek variabelnya. Pada kedua gambar diatas dapat dilihat untuk baris ketiga cara pengecekkan dilakukan secara diagonal dan vertikal. Gambar pertama pada saat pengecekkan ditemukan variabel yang menghasilkan variabel E dan C. Cara pengisian untuk blok selanjutnya sama seperti gambar 2.7. Hasil akhir dari pengecekan *string* diatas dapat dilihat pada gambar 2.8.



Gambar 2.8. Hasil *parsing string* dengan algoritma CYK

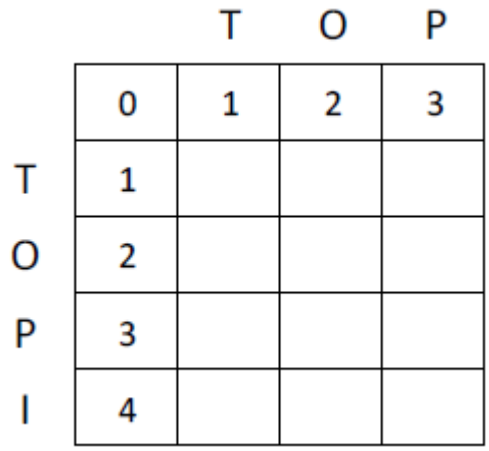
Dapat dilihat dari gambar diatas *string* "a b c d" termasuk ke dalam aturan produksi yang sudah dibuat, artinya *string* tersebut dapat dikatakan sebagai kalimat dalam contoh ini. Jika baris dan kolom terakhir tidak menghasilkan tanda "-", *string* tersebut belum dapat dikenali oleh algoritma CYK.[1]

2.5 Algoritma *Levenshtein*

Algoritma *Levenshtein* merupakan algoritma pencarian jumlah perbedaan *string* yang ditemukan oleh Vladimir Levenshtein, seorang ilmuwan Rusia, pada tahun 1965. Algoritma ini digunakan secara luas dalam berbagai bidang, misalnya mesin pencari, pengecek ejaan (*spell checking*), pengenalan pembicaraan (*speech recognition*), pengucapan dialek, analisis DNA, dan lain-lain.

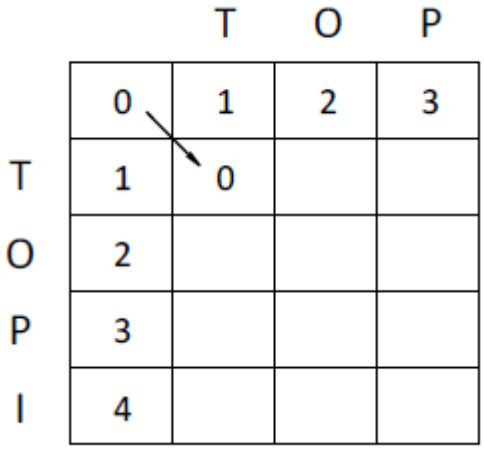
Dua buah *string* dibandingkan untuk mengetahui nilai *edit distance*-nya. *Edit distance* adalah suatu pengukuran (metrik) yang dihasilkan melalui perhitungan jumlah perbedaan yang terdapat pada dua *string*. *Edit distance* antara dua *string* didefinisikan sebagai jumlah minimum perubahan yang diperlukan untuk mengganti suatu *string* dengan *string* lain, dengan operasi penambahan (*insert*), operasi penggantian (*substitute*), operasi penghapusan (*delete*).

Algoritma ini menggunakan matriks dua dimensi dalam membandingkan kedua *string*. Algoritma ini berjalan mulai dari pojok kiri atas hingga ke pojok kanan bawah. Seluruh baris pertama diisi dengan nilai berurut mulai dari 0 hingga n begitu juga dengan kolom pertama. Cara pengisiannya dapat dilihat pada gambar 2.9.



Gambar 2.9. Inisialisasi awal tabel *levenshtein*

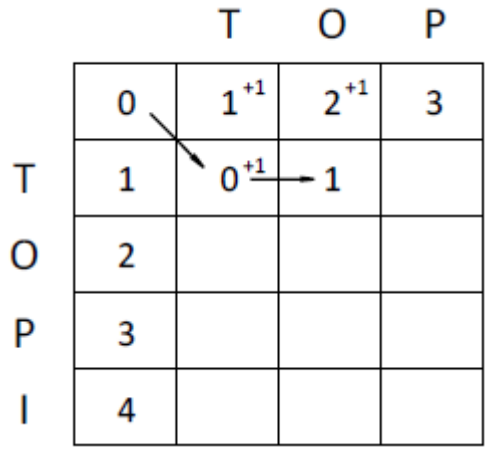
. Pada gambar diatas kata “top” akan dilakukan perbandingan dengan kata “topi” untuk melihat nilai *edit distance*-nya. Pengisian dilakukan mulai dari blok [1,1] dengan membandingkan karakter yang ada pada blok [0,1] dan blok [1,0]. Apabila karakter yang dibandingkan sama, turunkan nilai yang ada pada blok [0,0] yaitu angka 0. Nilai *edit distance*-nya dapat dilihat pada gambar 2.10.



Gambar 2.10. Penurunan angka 0 ke baris 1 kolom 1

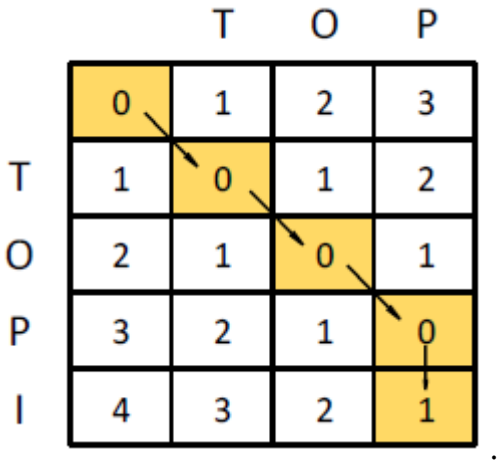
Pengisian blok berikutnya dilakukan dengan membandingkan karakter dengan baris yang sama. Apabila karakter yang dibandingkan tidak sama, maka nilai blok [0,2] ditambahkan dengan angka 1. Sama halnya dengan blok[1,1] dan blok [0,1]. Setelah itu lakukan perbandingan nilai dari blok [0,2], [1,1], dan [0,1]

untuk mencari nilai minimumnya. Selanjutnya simpan pada blok [1,2]. Hasilnya dapat dilihat pada gambar 2.11.



Gambar 2.11. Pengisian blok [1,2]

Langkah selanjutnya isi semua blok sampai akhir dengan langkah yang sama. Hasil akhir dari pengisian tabel Levenshtein diatas dapat dilihat pada gambar 2.12



Gambar 2.12. Hasil akhir pengisian tabel Levenshtein

Nilai *edit distance* dapat dilihat dari baris dan kolom terakhir, yaitu angka 1. Nilai inilah yang akan mengetahui seberapa banyak perbedaan karakter antara dua *string* [3].

2.6 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan Buatan adalah salah satu cabang ilmu pengetahuan berhubungan dengan pemanfaatan mesin untuk memecahkan persoalan yang rumit. Hal ini biasanya dilakukan dengan mengikuti atau mencontoh karakteristik dan analogi berpikir dari kecerdasan/Inteligensia manusia, dan menerapkannya sebagai algoritma yang dikenal oleh komputer. Pengertian lain dari kecerdasan buatan adalah bagian ilmu komputer yang membuat agar mesin komputer dapat melakukan pekerjaan seperti dan sebaik yang dilakukan manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan jaman, maka peran komputer semakin mendominasi kehidupan manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu,komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia. Agar komputer bisa bertindak seperti dan sebaik manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu AI akan mencoba untuk memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa menjadi mesin pintar. Dari beberapa perspektif, AI dapat dipandang sebagai:

1. Dari perspektif kecerdasan, AI adalah bagaimana membuat mesin yang cerdas dan dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan manusia.
2. Dari perspektif bisnis, AI adalah sekelompok alat bantu *(tools)*yang berdayaguna dan metodologi yang menggunakan alat-alat bantu tersebut untuk menyelesaikan masalah-masalah bisnis.
3. Dari perspektif pemrograman, AI meliputi studi tentang pemrograman simbolik, pemecahan masalah, dan proses pencarian *(search)*.
4. Dari perspektif penelitian
    a. Riset tentang AI dimulai pada awal tahun 1960-an, percobaan pertama adalah membuat program permainan catur, membuktikan teori, dan *general problem solving*.
    b. *Artificial intelligence* adalah nama pada akar dari studi area.

Ada dua hal yang sangat mendasar mengenai penelitian-penelitian AI, yaitu *knowledge representation* (representasi pengetahuan) dan *search* (pelacakan). Para peneliti AI terus mengembangkan berbagai jenis teknik baru dalam menangani sejumlah permasalahan yang tergolong ke dalam AI seperti *vision* dan percakapan, pemrosesan bahasa alami, dan permasalahan khusus seperti diagnosa medis [7].

2.6.1 Konsep Kecerdasan Buatan

Ada beberapa konsep yang harus dipahami dalam kecerdasan buatan, diantaranya (kusrini, 2006) :

1. *Turing Test* – Metode Pengujian Kecerdasan

   Turing Test merupakan sebuah metode pengujian kecerdasan yang dibuat oleh Alan Turing. Proses uji ini melibatkan seorang penanya (manusia) dan dua obyek yang ditanyai. Yang satu adalah seorang manusia dan yang satunya adalah sebuah mesin yang akan diuji. Penanya tidak dapat melihat langsung kepada obyek yang ditanyai.Penanya diminta untuk membedakan mana jawaban komputer danmana jawaban manusia berdasarkan jawaban kedua obyek tersebut. Jika penanya tidak dapat membedakan mana jawaban mesin dan mana jawaban manusia maka Turing berpendapat bahwa mesin yang diuji tersebut dapat diasumsikan CERDAS.

2. Pemrosesan Simbolik

   Komputer semula didesain untuk memproses bilangan atau angka-angka (pemrosesan numerik). Sementara manusia dalam berfikir dan menyelesaikan masalah lebih bersifat simbolik, tidak didasarkan pada sejumlah rumus atau melakukan komputasi matematika. Sifat penting dari AI adalah bahwa AI merupakan bagian dari ilmu komputer yang melakukan proses secara simbolik dan nonalgoritmik dalam penyelesaian masalah.

3. *Heuristic*

Istilah heuristic diambil dari bahasa Yunani yang berarti menemukan. Heuristic merupakan suatu strategi untuk melakukan proses pencarian (*search*) ruang *problem* secara selektif, yang memandu proses pencarian yang kita lakukan disepanjang jalur yang memiliki kemungkinan sukses paling besar.

4. Penarikan Kesimpulan (*Inferencing*)

AI mencoba membuat mesin memiliki kemampuan berfikir atau mempertimbangkan (*reasoning*). Kemampuan berfikir (*reasoning*) termasuk didalamnya proses penarikan kesimpulan (*inferencing*) berdasarkan fakta-fakta dan aturan dengan menggunakan metode heuristic atau pencarian lainnya.

5. Pencocokan Pola (*Pattern Matching*)

AI bekerja dengan metode pencocokan pola (*pattern matching*) yang berusaha untuk menjelaskan objek, kejadian (*event*) atau proses, dalam hubungan logika atau komputasional.

2.6.2 Lingkup Utama Kecerdasan Buatan

Lingkup utama kecerdasan buatan adalah :

1. Sistem pakar (*Expert System*). Komputer digunakan sebagai saranuntuk menyimpan pengetahuan para pakar. Dengan demikian komputer akan memiliki keahlian untuk menyelesaikan masalah dengan meniru keahlian yang dimiliki para pakar
2. Pengolahan bahasa alami (*Natural Language Processing*).Dengan pengolahan bahasa alami ini diharapkan user mampu berkomunikasi dengan komputer dengan menggunakan bahasa sehari-hari.
3. Pengenalan ucapan (*Pattern Recognition*). Melalui pengenalan ucapan diharapkan manusia mampu berkomunikasi dengan komputer dengan menggunakan suara.
4. Robotika dan Sistem sensor.
5. *Computer vision*, mencoba untuk dapat mengintrepetasikan gambar atau objek-objek tampak melalui komputer.

6. *Intelligent Computer aid Instruction*. Komputer dapat digunakan sebagai tutor yang dapat melatih dan mengajar.

2.7 Bahasa Alami (*Natural Language*)

Pada prinsipnya bahasa alami adalah suatu bentuk representasi dari suatu pesan yang ingin dikomunikasikan antar manusia. Bentuk utama representasinya adalah berupa suara/ucapan (*spoken language*), tetapi sering pula dinyatakan dalam bentuk tulisan.

Bahasa dapat dibedakan menjadi (1) Bahasa Alami, dan (2) Bahasa Buatan. Bahasa alami adalah bahasa yang biasa digunakan untuk berkomunikasi antar manusia, misalnya bahasa Indonesia, Sunda, Jawa, Inggris, Jepang, dan sebagainya. Bahasa buatan adalah bahasa yang dibuat secara khusus untuk memenuhi kebutuhan tertentu, misalnya bahasa pemodelan atau bahasa pemrograman komputer.

Chomsky adalah orang yang pertama kali merepresentasikan bahasa sebagai rangkaian simbol. Chomsky berhasil memperlihatkan bahwa bahasa apapun dapat direpresentasikan dengan suatu cara yang universal. Pemikiran Chomsky yang merepresentasikan bahasa sebagai kumpulan simbol-simbol dan aturan yang mengatur susunan simbol-simbol tersebut telah membuka peluang untuk melakukan pemrosesan bahasa secara simbolik dengan teknologi komputer, sehingga melahirkan bidang ilmu *Natural Language Processing* (NLP).

Linguistik adalah bidang ilmu yang khusus mengkaji tentang bagaimana bahasa distrukturkan dan digunakan. Bidang ilmu ini salah satunya membahas tentang perbendaharaan kata dan leksikon.

Perbendaharaan kata adalah sekumpulan kata-kata dan frase-frase yang digunakan dalam bahasa tertentu. Sebagai bagian dari pengkajian bahasa, linguis mendefinisikan semua kata-kata dan frase-frase yang digunakan secara umum kemudian mengorganisasikannya ke dalam sebuah leksikon. Leksikon adalah kamus yang mendaftar kata-kata bahasa itu secara

Alfabet [7].

2.7.1 Pengertian *Natural Language Processing*

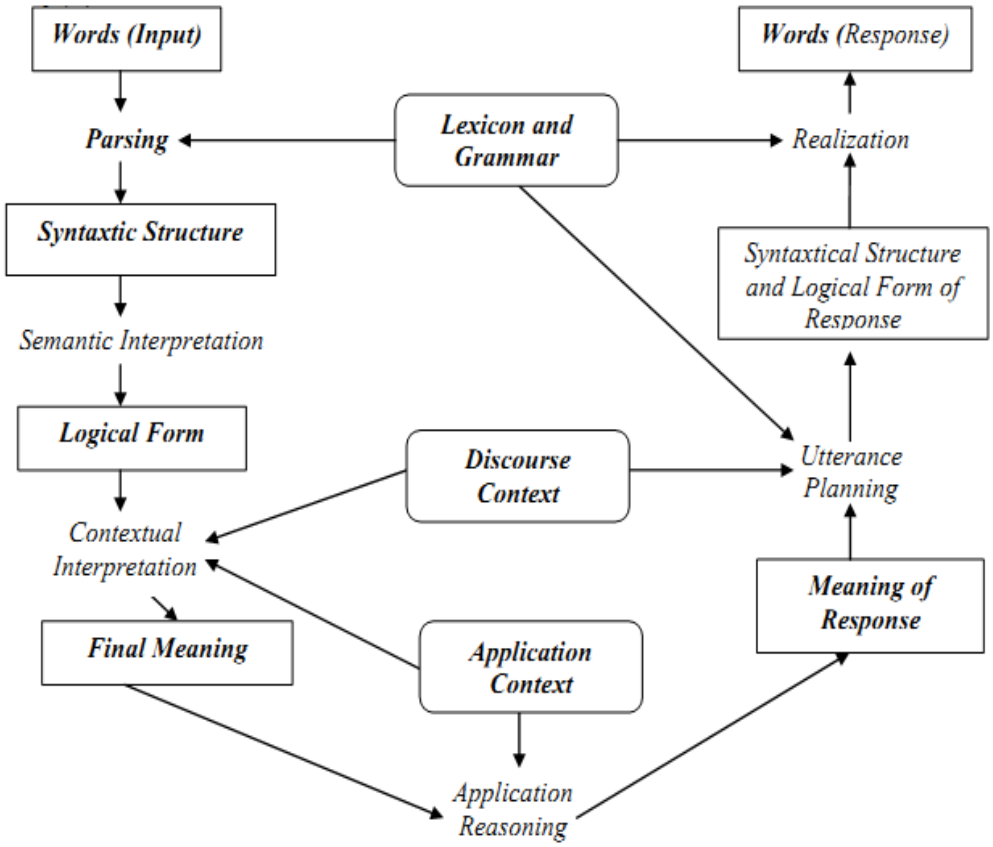
Pengolahan bahasa alami (NLP) adalah bidang ilmu komputer dan linguistik berkaitan dengan interaksi antara komputer dan manusia. Seluler generasi bahasa sistem komputer yang mengubah informasi dari database ke dalam bahasa manusia yang dapat dibaca.

Natural language processing, biasanya disingkat dengan NLP, mencoba membuat komputer mampu memahami suatu perintah yang dituliskan dalam bentuk bahasa sehari-hari dan diharapkan komputer juga merespon dalam bahasa yang mirip dengan bahasa natural. Setelah komputer bisa memahami perintah dalam bahasa natural, maka diharapakan sistem komputer juga dapat memberikan respon dalam bahasa natural pula.

Sebuah sistem natural language harus memperhatikan pengetahuan terhadap bahasa itu sendiri baik dari segi kata yang digunakan bagaimana kata-kata tersebut digabung untuk menghasilkan suatu kalimat, apa arti suatu kata, apa fungsi sebuah kata dalam sebuah kalimat dan sebagainya. Natural Language processor tidak memperdulikan bagaimana suatu kalimat diinputkan ke komputer. Tugasnya adalah mengekstrak informasi dari kalimat. Inti dari sistem NLP adalah parser. Parser adalah bagian dari program atau system yang membaca setiap kalimat, kata demi kata, untuk menentukan "*what is what*".

NLP tidak bertujuan untuk mentransformasikan bahasa yang diterima dalam bentuk teks atau suara menjadi data digital dan/atau sebaliknya pula; melainkan bertujuan untuk memahami arti dari kalimat yang diberikan dalam bahasa alami dan memberikan respon yang sesuai, misalnya dengan melakukan suatu aksi tertentu atau menampilkan data tertentu. Untuk mencapai tujuan ini dibutuhkan tiga tahap proses. Proses yang pertama ialah parsing atau analisa sintaksis yang memeriksa kebenaran struktur kalimat berdasarkan suatu *grammar* (tata bahasa) dan *lexicon* (kosa kata) tertentu. Proses kedua ialah *semantic* interpretation atauinterpretasi semantik yang bertujuan untuk merepresentasikan arti dari kalimat secara context-independent untuk keperluan lebih lanjut. Sedangkan proses ketiga ialah contextual interpretation atau interpretasi kontekstual yang bertujuan untuk merepresentasikan arti secara *context dependent*

dan menentukan maksud dari penggunaan kalimat. Gambaran organisasi dari sebuah sistem NLP yang lengkap ditunjukkan pada gambar 2.13.



Gambar 2.13. Organisasi Sebuah Sistem NLP

2.7.2 *Question Answering System* (*QA System*)

Question answering system (QA system) adalah sistem yang mengijinkan user menyatakan kebutuhan informasinya dalam bentuk *natural language question* (pertanyaan dalam bahasa alami), dan mengembalikan kutipan teks singkat atau bahkan frase sebagai jawaban. Ketersediaan sumber informasi yang luas dan bervariasi, serta adanya perkembangan yang pesat dari teknik *Natural Language Processing* (NLP), *Information Extraction* (IE), dan *Information Retrieval* (IR) sangat mempengaruhi perkembangan dari QA *system*, yang mana dulunya hanya dapat menjawab pertanyaan-pertanyaan yang terbatas pada suatu

bidang (*domain*) tertentu dengan berdasarkan pada sumber informasi yang terstruktur seperti database, hingga kini dapat menjawab berbagai jenis pertanyaan dengan bersumber pada informasi dari sebuah koleksi teks yang tidak terstruktur. Umumnya arsitektur *question answering system* yang berbasis teks dibangun atas enam tahapan proses, yaitu analisis pertanyaan, preprocessing koleksi dokumen, pemilihan kandidat dokumen, analisis kandidat dokumen, ekstraksi jawaban, dan pemberian respon.

Aplikasi-aplikasi *QA system* (baik yang dapat diakses melalui internet maupun tidak) yang dikembangkan dengan ciri khasnya masing-masing memiliki urutan proses yang tidak jauh berbeda satu dengan lainnya. Jawaban yang dikembalikan oleh sebuah *QA system* sebagai respon terhadap pertanyaan perlu dievaluasi untuk menilai performansi sistem.

2.7.3 *Scanner* (Analisis Leksikal)

Scanner merupakan salah satu bagian dari kompilator bahasa pada komputer yang bertugas melakukan analisis leksikal. Analisis leksikal adalah proses pengidentifikasian semua besaran yang membangun suatu bahasa pada program sumber. *Scanner* menerima input berupa *stream* karakter kemudian memilah program sumber menjadi satuan leksik yang disebut dengan token. Token ini akan menjadi input bagi parser. Tugas dari *scanner* adalah sebagai berikut:

1. Melakukan pembacaan kode sumber dengan merunut karakter demi karakter.
2. Mengenali besaran leksik.
3. Mentransformasikan menjadi sebuah *token* dan menentukan jenis *token*-nya.
4. Mengirimkan token .
5. Membuang/mengabaikan *blank* dan komentar dalam program .
6. Menangani kesalahan .
7. Menangani *table symbol* .

Di dalam aplikasi NLP sistem cerdas yang akan dibuat, yang dimaksud dengan program sumber yang diolah oleh *scanner* adalah berupa kalimat input dari pengguna dalam bentuk sms.

Ketika scanner menerima input berupa *stream* karakter kemudian memilah menjadi satuan leksik, satuan leksik tersebut terdiri atas simbol-simbol satuan yang jika dikombinasikan akan mempunyai arti yang berbeda-beda. Simbol-simbol yang bisa dipergunakan dalam sebuah bahasa tentunya terbatas jumlahnya, yang membentuk sebuah himpunan dan disebut sebagai abjad (*alphabet*).

Tata bahasa (grammatika) adalah sekumpulan dari himpunan variabel-variabel, simbol-simbol terminal, simbol non-terminal, simbol awal yang dibatasi oleh aturan-aturan produksi. Aturan produksi adalah pusat dari tata bahasa yang menspesifikasikan bagaimana suatu tata bahasa melakukan transformasi suatu string ke bentuk lainnya. Dalam pembicaraan grammar, anggota alfabet dinamakan simbol terminal atau token. Kalimat adalah string yang tersusun atas simbol-simbol terminal. Bahasa adalah himpunan kalimat-kalimat. Anggota bahasa bisa berupa tak berhingga hingga kalimat.

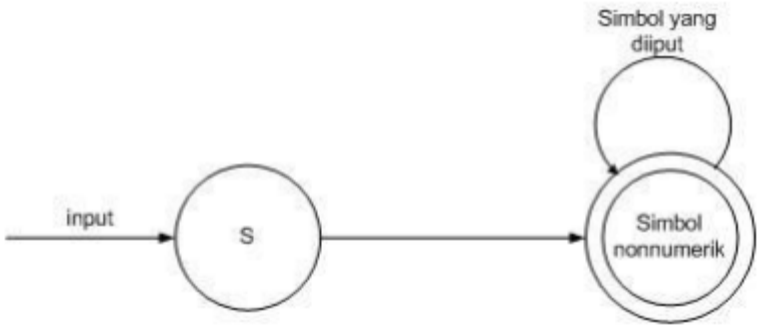
Simbol-simbol berikut adalah symbol terminal :

1. huruf kecil alphabet, misalnya :a, b, c
2. symbol operator, misalnya : +, -, dan „
3. symbol tanda baca, misalnya : (,), dan ;

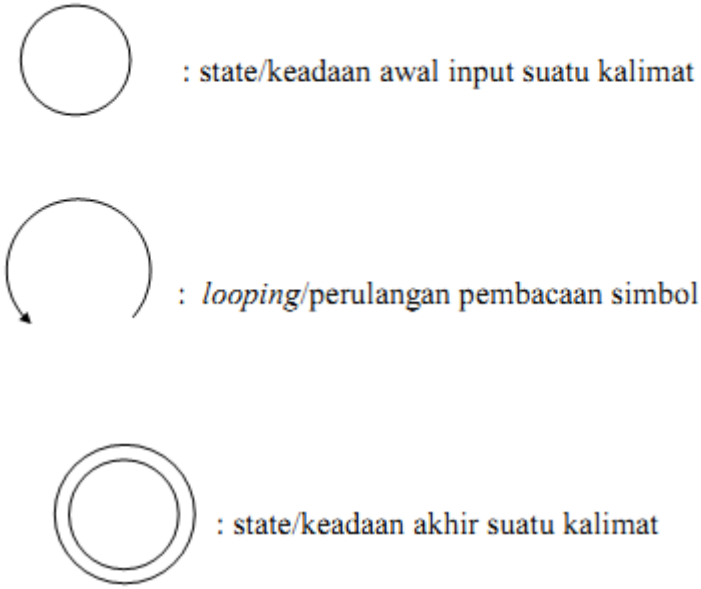
Sedangkan simbol-simbol berikut adalah simbol non terminal :

1. huruf besar awal alphabet, misalnya : A, B, C
2. huruf S sebagai simbol awal

Pada saat *scanner* membaca *input*, *tools* yang digunakan untuk menggambarkan perpindahan dari posisi satu ke posisi lainnya adalah diagram transisi.



Gambar 2.14. Diagram Transisi



2.7.4 *Parser* (Analisis Sintaksis)

Parser atau *syntactic analyzer* pada kompilator bahasa pemrograman berfungsi untuk memeriksa kebenaran kemunculan setiap *token*.

Pada *QA system*, fungsi dari *parser* ini agak berbeda karena *token* yang akan diolah semua memiliki tipe yang sama yaitu berupa kata (word). Urutan kemunculan *token* yang berupa kata-kata tersebut akan diolah dengan mengacu pada *brain file* agar didapatkan makna kalimat yang sesungguhnya. Dengan kata lain, tahap analisa semantik terjadi di bagian *brain file*. Kemampuan dari *parser* untuk mengolah *token* dan bekerja sama dengan *brain file* inilah yang paling menentukan tingkat kecerdasan dari sebuah *chatbot*.

2.7.5 *Parsing*

Proses parsing tidak hanya dapat dilakukan dalam proses information retrieval, melainkan juga pada bidang lain seperti pada pembuatan sebuah compiler dan bahasa alami. Sebelumnya perlu diketahui arti dari istilah *parser* yaitu program yang melakukan proses parsing.

Untuk pemrosesan, dokumen dipisahkan menjadi unit-unit yang lebih kecil misalnya berupa kata, frasa atau kalimat. Unit pemrosesan tersebut disebut sebagai token. Parsing merujuk pada proses pengenalan token yang terdapat dalam rangkaian teks.

Proses *parsing* (penguraian kalimat) juga merupakan proses yang dilakukan untuk menterjemahkan masukan dari pengguna agar dapat dimengerti oleh sistem. Secara *default*, seluruh kalimat masukan pengguna yang dimasukan akan dianggap sebagai kata-kata yang harus ada pada data yang akan dicari. Misal pengguna memasukan kalimat sebagai berikut :

Penggunaan bahasa alami

Maka sistem akan melakukan pencarian data pada dokumen yang memiliki kata *bahasa* atau *alami*.Artinya jika terdapat dokumen yang mengandung kata *bahasa inggris* maka dokumen tersebut dianggap sesuai dengan yang dicari pengguna, karena mengandung kata *bahasa*. Hal ini membuat pengguna yang ingin mencari informasi mengenai *bahasa alami* akan mendapatkan hasil yang tidak sesuai dengan yang diinginkan.

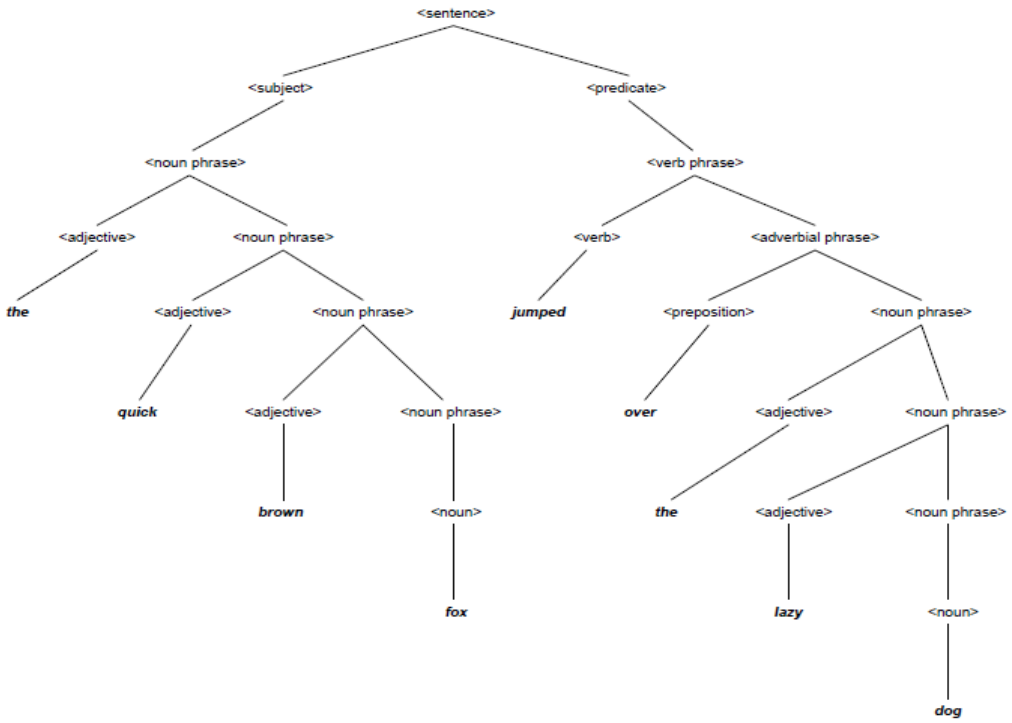
2.7.6 Pohon Sintaks

Pohon (*tree*) adalah suatu *graph* terhubung tidak sirkuler, yang memiliki satu simpul (*node*)/*vertex* disebut akar (*root*) dan dari situ memiliki lintasan ke setiap simpul.

Pohon sintaks/pohon penurunan (*syntax tree/derivation tree/ parse tree*) berguna untuk menggambarkan bagaimana memperoleh suatu *string* dengan cara menurunkan simbol-simbol variabel menjadi simbol-simbol terminal. Setiap simbol variabel akan diturunkan menjadi terminal, sampai tidak ada yang belum

tergantikan. Gambar 2.15 memberikan contoh sebuah *tree* yang menguraikan kalimat dalam bahasa Inggris.

The quick brown fox jumped over the lazy dog



Gambar 2.15. Pohon Sintaks

Proses penurunan atau *parsing* bisa dilakukan dengan cara:

1. Penurunan terkiri (*leftmost derivation*): simbol variabel terkiri yang diperluas terlebih dulu
2. Penurunan terkanan (*rightmost derivation*): simbol variabel terkanan yang diperluas terlebih dulu.

2.7.7 Analisis Semantik

Analisis semantik merupakan kelanjutan dari proses *scanning* dan *parsing*. Fungsi dari analisis semantik adalah untuk menentukan makna dari serangkaian instruksi yang terdapat dalam program sumber atau masukan dari penguna.

Penganalisa semantik harus mampu menentukan aksi atau respon apa yang yang akan dilakukan terhadap instruksi yang diberikan.

2.7.8 *Turing Test*

Turing Test adalah usulan untuk mengetes kemampuan sebuah mesin untuk melakukan percakapan seperti manusia. Digambarkan oleh Alan Turing di Koran pada 1950 ‘’Mesin komputasi dan kecerdasan‟. Hasilnya sebagai berikut : seorang manusia penilai melakukan percakapan bahasa alami dengan dua pihak lain, satu manusia dan satunya lagi mesin, jika penilai tidak dapat mengatakan yang mana mesin dan yang mana manusia, maka mesin tersebut telah berhasil melalui *test* tersebut. Diasumsikan bahwa baik manusia dan mesin mencoba untuk terlihat seperti manusia. Untuk menjaga *test* diatur lebih sederhana dan *universal* (secara eksplisit menguji kemampuan bahasa mesin), percakapan biasanya dibatasi dengan teks saja.

2.8 *Chat Bot*

Chat Bot adalah sebuah program komputer yang dirancang untuk mensimulasikan sebuah percakapan atau komunikasi yang interaktif kepada user (manusia) melalui bentuk teks, suara, dan atau visual. Percakapan yang terjadi antara komputer dengan manusia merupakan bentuk respon dari program yang telah dideklarasikan pada database program pada komputer. Respon yang dihasilkan merupakan hasil pemindaian kata kunci pada inputan user dan menghasilkan respon balasan yang dianggap paling cocok, atau pola kata-kata yang dianggap paling mendekati, dari database tentunya. Dalam bahasa sehari-harinya Chat Bot merupakan Aplikasi atau Program komputer yang dirancang untuk meniru manusia itu sendiri, batasan yang diambil dari Chat Bot adalah mampu meniru komunikasi manusia. Jadi jika manusia sedang bercakap-cakap dengan program ini, seakan-akan ada 2 pribadi manusia yang saling berkomunikasi. Nyatanya tidak, manusia berkomunikasi dengan Bot. Bot sudah dirancang untuk merespon segala jenis pertanyaan dan pernyataan yang diinputkan oleh manusia (user). Hal ini terjadi karena sebelumnya sudah

dideklarasikan pada database, berupa entitas-entitas kata, pola kalimat, dan berbagai jenis pernyataan dan pertanyaan.

Bot biasa digunakan dalam IRC (*Internet Chat Groups*) di mana *bot* tersebut melakukan berbagai fungsi seperti menyapa partisipan baru dan memonitor penggunaan kata-kata yang tidak pantas. Sekarang ini, *bot* dapat mengumpulkan informasi, mengolah informasi tersebut, dan memberikan respon terhadap pengguna dan *event* yang terjadi di internet. Beberapa jenis *bot* bekerja secara *silent* di internet tanpa sepengetahuan kita namun ada pula *bot* yang bekerja secara interaktif yang dapat kita ajak berbicara yaitu *chat bot*.

Chat bot adalah sebuah simulator percakapan yang berupa program komputer yang dapat berdialog dengan penggunanya dalam bahasa alami. Karena *chat bot* hanya sebuah program, dan bukan robot (*chat bot* tidak memiliki tubuh dan tidak memiliki mulut sehingga tidak dapat berbicara seperti manusia), maka yang dimaksud dengan dialog antara manusia sebagai pengguna dengan *chat bot* dilakukan dengan cara mengetik apa yang akan dibicarakan dan *chat bot* akan memberikan respon. Orang yang membuat dan mengembangkan program *chat bot* disebut *bot master*.

Chat bot yang pertama adalah ELIZA yang dibuat pada tahun 1964 sampai 1966 oleh Professor Joseph Weizenbaum di MIT (*Massachusetts Institute of Technology*), dengan tujuan untuk mempelajari komunikasi *natural language* antara manusia dengan mesin. Eliza bertindak seolah-olah dia adalah seorang psikolog yang dapat menjawab pertanyaan-pertanyaan dari pasien dengan jawaban yang cukup masuk akal atau menjawabnya dengan pertanyaan balik.

ELIZA adalah pelopor atau dapat disebut sebagai nenek moyangnya *chat bot.* ELIZA (juga dikenal sebagai Dr. Eliza) terkenal sebagai program *chat* yang “memiliki profesi” sebagai seorang psikiater. ELIZA mensimulasikan percakapan antara seorang psikiater dengan pasiennya menggunakan metode biasa yang dapat mencerminkan perasaan pasien dengan mengajukan pertanyaan-pertanyaan seperti: "*How do you ...*", "*Why do you feel like ...*", "*What do you think about ...*".

Program ini akan mencari pola kata-kata tertentu pada input yang diberikan oleh pengguna, dan kemudian memberikan output yang sesuai.

ELIZA dapat memberikan respon yang tepat hanya jika pernyaataan pengguna ditulis dalam ejaan yang benar. Meskipun begitu, pengguna dapat berbicara secara santai dengan ELIZA.

ALICE (*Artificial Linguistic Internet Computer Entity*) adalah sebuah program *chat bot* yang diciptakan oleh Dr. Richard Wallace. ALICE mampu menciptakan suasana seperti komunikasi antara dua manusia seperti biasanya, ALICE bahkan dapat berbohong, menyebarkan gosip, dan mengetahui banyak kejadian yang terjadi di dunia nyata.

Ada banyak lagi *chat bot* lain yang menarik seperti Erin the Bartender, Madame Celeste, Mega HAL, dan lain-lain [7].

2.9 Basis Data

Basis data terdiri dari dua kata yaitu basis dan data. Basis dapat diartikan sebagai gudang tempat berkumpul. Sedangkan data adalah represebtasi fakta dunia nyata yang mewakili objek seperti manusia, barang, hewan, peristiwa konsep, yang nyatakan dalam bentuk angka, huruf, simbol teks, gambar, bunyi atau kombinasinya.

Basis data dapat didefinisikan dari beberapa sudut pandang diantaranya sebagai berikut:

1. Sekumpulan data *persistence* (data disimpan *defile sekunder* atau data yang tahan lama) yang saling terkait, menggambarkan suatu organisasi (*enterprise*).
2. Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.
3. Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikina rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
4. Kumpulan file atau arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronis.

2.10 Kamus Data (*Data Dictionary*)

Kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi.

Dengan adanya kamus data, analisis sistem dapat mdendefinisikan data yang mengalir di dalam sistem dengan lengkap. Pada tahap perancangan sistem, kamus data dapat digunakan untuk merancang input, output, dan merancang database program. Kamus data dibuat berdasarkan arus data yang ada.

2.11 ERD (*Entity Relationship Diagram*)

Model E-R didasarkan pada persepsi bahwa dunia nyata merupakan sekumpulan dari sejumlah objek dasar (entitas) dan relasi antar objek-objek data tersebut. Diagram yang menggambarkan struktur lojik keseluruhan basis data, simbol yang digunakan adalah Persegi empat, merepresentasikan himpunan entitas (untuk entitas lemah diberi garis ganda), Elips, merepresentasikan atribut, Wajik, merepresentasikan himpunan keterhubungan, Garis, menghubungkan simbol-simbol pada diagram. Label dari persegi empat, elips, dan wajik menunjukkan nama, Kardinalitas pemetaan dinyatakan dengan 2 cara : [Korth] garis berarah (1) dan garis tidak berarah (Banyak), [*Date*] menuliskan kardinalitasnya pada garis dan Peran dapat dituliskan sebagai label dari garis.

Pemakaian elemen-elemen dalam ERD ada tiga diantaranya sebagai berikut:

1. *Entity* (Entitas) adalah sebuah objek yang dapat dibedakan dari objek-objek lainnya, yang memiliki sejumlah *property* atau atribut, dimana setiap atribut memiliki sekumpulan nilai yang diizinkan yang disebut domain, himpunnan entitas yaitu kumpulan jumlah entitas yang memiliki tipe yang sama dan sebuah basis data mengandung sekumpulan himpunan entitas yang masing-masingnya memiliki sejumlah entitas dari tipe yang sama.
2. *Relationship* (relasi) merupakan hubungan antar entitas yaitu sebuah relasi menggambarkan suatu asosiasi antar sejumlah entitas, himpunan relasi (*Relationsip set*) adalah kumpulan sejumlah relasi yang memiliki tipe yang sama yang merupakan relasi matematis terhadap dua atau lebih himpunan entitas : {(e1, e2,…, en) (e1 E1, e2 E2 ,…, en En)}, Jumlah

entitas terlihat dalam 2 buah relasi disebut derajat. Kebanyakan relasi yang muncul adalah relasi binary, ada beberapa yang ternary, lebih dari itu sangat jarang, Fungsi sebuah entitas di dalam relasi disebut peran (role) dan Sebuah relasi dapat memiliki atribut.

2.12 *Web Application*

Web application dikenal sebagai aplikasi yang diakses melalui *web browser* dan melalui jaringan seperti internet atau intranet. Kemampuan untuk memperbarui dan memelihara aplikasi web tanpa harus mendistribusikan dan menginstal perangkat lunak pada kemungkinan ribuan komputer klien merupakan keunggulan teknologi ini, selain juga untuk *cross-platform compatibility*. Termasuk aplikasi *web common webmail*, penjualan ritel online, online pelelangan, wiki dan banyak fungsi lainnya [9].

Banyak situs internet yang memiliki halaman dengan sifat statis seperti profil perusahaan, artikel, dan keterangan-keterangan lain. Situs ini mempunyai dokumen dengan teks yang sederhana, image dan *hyperlinks* ke dokumen yang dimilikinya. Untuk mengembangkan situs yang bersifat statis, kita menggunakan teknologi client side. HTML dan *Cascading Style Sheet* (CSS) dapat digunakan untuk mengatur struktur dan menampilkan halaman isi. Seandainya ingin diperindah dapat ditambahkan *script* yang sifatnya *client side*, seperti JavaScript, Jscript ataupun VBScript.

Dengan berkembangnya internet, situs yang ada di internet tidak hanya berfungsi untuk mempresentasikan content tetapi cenderung berupa aplikasi yang kebanyakan terhubung ke suatu basis data. Pada tahapan ini situs akan bersifat dinamis, karena content yang dipresentasikan akan bervariasi dan berubah-ubah sesuai dengan data yang diminta dan action dari user. Untuk mengembangkan situs yang dinamis diperlukan teknologi *server side* seperti PHP, ASP, Perl dan CGI yang lain. Dengan teknologi *server side* kita dapat mengembangkan suatu aplikasi berbasis internet yang dapat mengahsilkan dan menampilkan content secara dinamis [9]

2.13 PHP

PHP adalah bahasa pemrograman *script* yang paling banyak dipakai saat ini. PHP banyak dipakai untuk memrogram situs web dinamis, walaupun tidak

tertutup kemungkinam untuk digunakan untuk pemakaian lain. PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu PHP bernama FI (*Form Interpreted*). Pada saat tersebut PHP adalah sekumpulan *script* yang digunakan untuk mengolah data *form* dari web. Perkembangan selanjutnya adalah Rasmus melepaskan kode sumber tersebut dan menamakannya PHP/ FI, pada saat tersebut kepanjangan dari PHP/ FI adalah *Personal Home Page*/ *Form Interpreter*. Dengan pelepasan kode sumber ini menjadi *open source,* maka banyak programmer yang tertarik untuk ikut mengembangkan PHP. Pada November 1997, dirilis PHP/ FI 2.0. Pada rilis ini interpreter sudah diimplementasikan dalam C. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan PHP/ FI secara signifikan.

Pada tahun 1997, sebuah perusahaan bernama Zend, menulis ulang interpreter PHP menjadi lebih bersih, lebih baik dan lebih cepat. Kemudian pada Juni 1998 perusahaan tersebut merilis interpreter baru untuk PHP dan meresmikan nama rilis tersebut menjadi PHP 3.0.Pada pertengahan tahun 1999, Zend merilis interpreter PHP baru dan rilis tersebut dikenal dengan PHP 4.0. PHP 4.0 adalah versi PHP yang paling banyak dipakai. Versi ini banyak dipakai sebab versi ini mampu dipakai untuk membangun aplikasi web kompleks tetapi tetap memiliki kecepatan proses dan stabilitas yang tinggi.Pata Juni 2004, Zend merilis PHP 5.0. Versi ini adalah versi mutakhir dari PHP. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Dalam versi ini juga dikenalkan model pemrograman berorientasi objek baru untuk menjawab perkembangan bahasa pemrograman kearah pemrograman berorientasi objek.

PHP merupakan bahasa pemrograman web, penulisan *script* PHP dapat dijadikan satu dengan kode html dan disimpan dalam satu *file* yang sama ataupun berdiri sendiri dengan nama *file* berekstensi *.php. Ciri khas PHP diawali dengan tanda “<?php“ dan diakhiri dengan “?>”. PHP memiliki fasilitas untuk melakukan koneksi ke *database*. Script program PHP disimpan dan dijalankan di sisi *server* sehingga keamanan data lebih terjamin [9].

2.14 MySQL

MySQL merupakan aplikasi *database server*. Perkembangannya disebut SQL yang merupakan kepanjangan dari *Structured Query Language*. SQL merupakan bahasa terstruktur yang digunakan untuk mengolah *database*. MySQL dapat digunakan untuk membuat dan mengelola database beserta isinya. Kita dapat memanfaatkan MySQL untuk menambahkan, mengubah, dan menghapus data yang berada dalam database.

MySQL adalah sebuah sistem manajemen *database* relasi (*relation database management system*) yang bersifat *open source.* MySql merupakan buah pikiran dari Michael "Monty" Widenius, Daveid axmark dan Allan Larson yang dimulai tahun 1995.Mereka bertiga kemudian mendirikan perusahaan bernama MySQL AB di Swedia.

MySQL versi 1.0 rilis pada Mei 1996 dan penggunaannya hanya terbatas di kalangan perusahaan saja. Barulah pada bulan Oktober 1996, MySQL versi 3.11.0 di rilis ke masyarakat luas. MySQL menggunakan bahasa standar SQL (*Structure Query Languange*) sebagai bahasa interaktif dalam mengelola data. MySQL memiliki kinerja, kecepatan proses dan ketangguhan yang tidak kalah dibanding *database-database* besar lainnya yang komersil seperti ORACLE, Sysbase, Unify dan sebagainya. MySQL dapat berjalan di atas banyak sistem operasi Linux, Windows, Solaris, FreeBSD, Mac OS X, dan lain sebagainya [9].

2.15 *Sublime Text*

Sublime Text itu adalah salah satu *text editor* yang biasa digunakan oleh para *programmer*, khususnya *Web Developer*. Bisa diibaratkan sebagai senjata-nya *Web Developer*. Setiap *web developer* wajib untuk menggunakan *Sublime Text* sebagai senjata hariannya. Dengan banyaknya fitur-fitur yang bagus menjadikan *Sublime Text* pilihan para Web Developer. Beberapa fiturnya sebagai berikut :

1. *Multiple Selection,* fitur untuk merubah banyak text dengan satu kali ketik.

2. *Find in Project,* fitur untuk mencari file dalam project

3. *Go to line,* fitur untuk berpindah baris dengan cepat

4. *Package Control dan Themes,* fitur untuk menambahkan fitur dan mengganti tema

2.16 Perancangan Sistem

Dalam membangun aplikasi ini perancangan yang digunakan adalah perancangan berorientasi objek dengan menggunakan pemodelan *Unified Modelling Language* (UML), serta pemrograman berorientasi objek. Berikut ini penjelasan dari perancangan sistem yang digunakan:

2.16.1 Pengertian *Unified Modelling Language* (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari tiga notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). [10]

UML ini menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu : [10]

1. *Use Case Diagram* untuk memodelkan proses bisnis.
2. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. *Sequence Diagram* untuk memodelkan pengiriman pesan (*message*) antar objek.
4. *Collaboration Diagram* untuk memodelkan interaksi antar objek.
5. *State Diagram* untuk memodelkan perilaku objek di dalam sistem.
6. *Activity Diagram* untuk memodelkan perilaku userdan objek di dalam sistem.
7. *Class Diagram* untuk memodelkan struktur kelas.
8. *Objek Diagram* untuk memodelkan struktur objek.
9. *Component Diagram* untuk memodelkan komponen objek.
10. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

Berikut akan dijelaskan 6 macam diagram yang digunakan dalam perancangan aplikasi ini, yaitu *use case diagram, activity diagram, class diagram, sequence diagram, deployment diagram,* dan *component diagram.*

1. *Use case diagram*

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas *diagram* untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.

Use case merepresentasikan operasi-operasi yang dilakukan oleh *actor*.*Use case* digambarkan berbentuk elips dengan nama operasi dituliskan didalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case.*

2. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa object swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

3. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi*). Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. [13]

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private* , tidak dapat dipanggil dari luar class yang bersangkutan
2. *Protected* , hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya

3. *Public* , dapat dipanggil oleh siapa saja

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Sesuai dengan perkembangan class model, class dapat dikelompokkan menjadi package.

4. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai *respons* dari sebuah *event* untuk menghasilkan *output* tertentu.

Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada *fase* desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

5. *Deployment Diagram*

Deployment diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah node adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Hubungan antar node (misalnya TCP/IP) dan *requirement* dapat juga didefinisikan dalam diagram ini.

6. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

2.16.2 Pemrograman Berorientasi Objek

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Dasar pembuatan adalah objek, yang merupakan kombinasi antara struktur data dab perilaku dalam suatu entitas. Model berorientasi objek bermanfaat untuk memahami masalah, komunikasi dengan ahli aplikasi, pemodelan suatu organisasi, meyiapkan dokumentasi serta perancangan program dan basis data. Pertama-tama suatu model analisis dibuat untuk menggambarkan aspek dasar dari domain aplikasi, dimana model tersebut berisi objek yang terdapat dalam domain aplikasi termasuk deskripsi dari keterangan objek dan perilakunya. [10]

Secara spesifik, pengertian berorientai objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karateristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan).

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karateristik utama, yaitu : [10]

a. *Encapsulation*

Encapsulation (pengkapsulan) merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas dalam bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya. Data terlindung dari prosedur atau objek lain kecuali prosedur yang berada dalam objek itu sendiri.

b. *Inheritance*

Inheritance (pewarisan) adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi atribut dan metoda dari induknya langsung. Atribut dan metoda dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pendefinisian objek dipergunakan untuk membangun suatu hirarki dari objek turunannya, sehingga tidak perlu membuat atribut dan metoda lagi pada anaknya, karena telah mewarisi sifat induknya.

c. *Polymorphism*

Polymorphism (polimorfisme) yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.