BAB II
TINJAUAN PUSTAKA

II.1 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi chatbot dengan pendekatan Natural Language Processing.

II.1.1 Kecerdasan Buatan (*Artificial Intelligence*)

Kecerdasan Buatan adalah salah satu cabang ilmu pengetahuan berhubungan dengan pemanfaatan mesin untuk memecahkan persoalan yang rumit [1]. 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.

II.1.2 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 dan mana 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 non-algoritmik 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.

II.1.3 Lingkup Utama Kecerdasan Buatan

Lingkup utama kecerdasan buatan adalah :

1. Sistem pakar (*Expert System*). Komputer digunakan sebagai saran untuk 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.

II.2 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.

II.2.1 Pengolahan Bahasa Alami (*Natural Language Processing*)

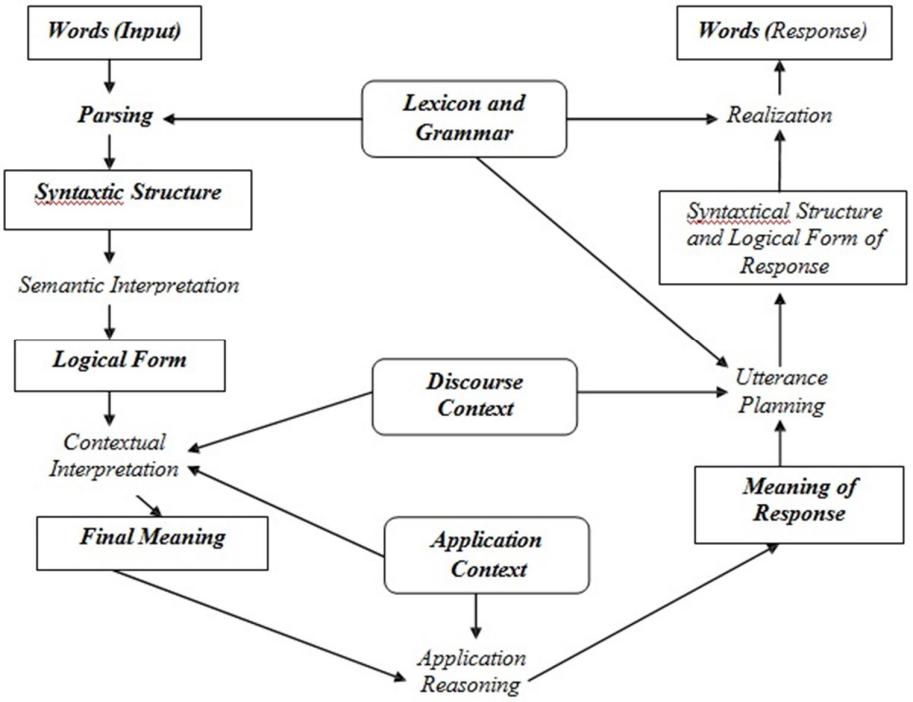
"*Natural Language Processing* (NLP) atau Pengolahan Bahasa Alami (PBA) merupakan salah satu bidang ilmu kecerdasan buatan (*Artificial Intelligence*) yang mempelajari komunikasi antara manusia dengan komputer". Pengolahan Bahasa Alami tidak bertujuan untuk mentransformasikan bahasa yang diterima dalam bentuk suara menjadi data digital dan/atau sebaliknya pula, melainkan bertujuan untuk memahami arti dari teks/tulisan yang diberikan dalam format bahasa alami dan memberikan respon yang sesuai, misalnya dengan melakukan suatu aksi tertentu atau menampilkan data tertentu.

Pengolahan Bahasa Alami merupakan metode komunikasi AI dengan komputer yang menggunakan bahasa Inggris atau bahasa lainnya yang biasa digunakan, contohnya bahasa Indonesia.

Agar komputer dapat memahami pertanyaan dalam bahasa alami, komputer harus mempunyai pengetahuan analisis dan interpretasi input dalam data *knowledgenya*. Komputer harus mengerti gramatika dan definisi kata-kata. Dalam hubungan ini teknik AI digunakan untuk menampilkan pengetahuan internal dan mengolah input. Pelacakan klasik dan teknik penyocokan pola (pattern matching) digunakan bersama dengan basis pengetahuan agar komputer bisa mengerti apa yang pengguna masukkan dalam bahasa alami. Bila komputer sudah mengerti ucapan yang pengguna berikan, maka komputer bisa melakukan hal-hal yang pengguna harapkan/respon kembali yang dinyatakan atau diekspresikan dalam bahasa alami juga.

Untuk mencapai tujuan tersebut dibutuhkan tiga tahap proses. Proses yang pertama adalah parsing atau analisa sintaksis yang memeriksa kebenaran struktur kalimat berdasarkan suatu grammar (tata bahasa) dan lexion (kosa kata) tertentu.

Proses kedua adalah semantic interpretation atau interpretasi semantik yang bertujuan untuk merepresentasikan arti dari kalimat secara context-independent untuk keperluan lebih lanjut. Sedangkan proses ketiga adalah contextual interpretation atau interpretasi kontekstual yang bertujuan untuk merepresentasikan arti secara context-dependent dan menentukan maksud dari penggunaan kalimat. Gambaran sebuah organisasi sistem NLP dapat dilihat pada gambar II.1



Gambar II.1 Organisasi sebuah sistem NLP

Jenis aplikasi yang bisa dibuat pada bidang bidang *natural language* adalah *text - based application* dan *dialogue - based applications*.

1. *Text - based application*

   Mencakup segala macam aplikasi yang melakukan proses terhadap text tertulis seperti misalnya buku, berita di surat kabar, e-mail dan lain sebagainya. Contoh penggunaan dari text - based application ini adalah :

   a. mencari topik tertentu dari buku yang ada pada perpustakaan
   b. memberikan respon atas input yang diberikan
   c. mencari isi dari surat atau e-mail
   d. menterjemahkan dokumen dari satu bahasa ke bahasa yang lain

2. *Dialogue - based application*

   Idealnya pedekatan ini melibatkan bahasa lisan atau pengenalan suara, akan tetapi bidang ini juga memasukkan interaksi

dengan cara memasukkan teks pertanyaan melalui keyboard. Aplikasi yang sering ditemui untuk bidang ini adalah :

a. sistem tanya jawab, dimana *natural language* digunakan dalam mendapatkan informasi dari suatu database.
b. sistem otomatis pelayanan melalui telepon
c. kontrol suara pada perlatan sistem
d. sistem *problem solving* yang membantu untuk melakukan penyelesaian masalah yang umum dihadapi dalam suatu pekerjaan.

II.2.2 *Question Answering 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*.

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.

II.2.3 *Scanner* (Analisis Leksikal)

Analisis Leksikal (Scanner) merupakan antarmuka antara kode program sumber dan analisa sintaktik (parser). Atau dalam pengertiannya adalah sebuah proses yang mendahului parsing sebuah rangkaian karakter. Scanner melakukan pemeriksaan karakter per karakter pada teksmasukan, memecah sumber program menjadi bagian-bagian disebut Token. Proses parsing akan lebih mudah dilakukan bila inputnya sudah berupa token.

Analisis leksikal membuat pekerjaan membuat sebuah parser jadi lebih mudah daripada membangun nama setiap fungsi dan variabel dari karakter-karakter yang menyusunnya, dengan analisis leksikal parser cukup hanya berurusan dengan sekumpulan token dan nilai sintaksis masing-masing.

Terlepas dari efisiensi pemrograman yang dapat dicapai dengan penggunaannya, proses kerja analisis leksikal yang membaca lebih dari sekali setiap karakter dari input yang diberikan menjadikan penganalisa leksikal sebagai sub-sistem yang paling intensif melakukan komputasi, terutama bila digunakan dalam sebuah kompilator.

Kompilator adalah sebuah program yang membaca suatu program yang ditulis dalam suatu bahasa sumber (source language) dan menterjemah-kannya ke dalam suatu bahasa sasaran (target language). Dalam penguraian struktur kalimat, penganalisa leksikal menganalisa setiap kata dalam kalimat, kemudian menentukan jenis kelas katanya.

Hasil dari penganalisa leksikal ini digunakan oleh penganalisa sintaks yang akan memeriksa urutan simbol-simbol kelas kata tersebut dalam kalimat. Analisa kata dalam kalimat ini dilakukan oleh penganalisa leksikal berdasarkan kecocokan kata dengan aturan-aturan leksikal berupa ekspresi regular yang sudah didefinisikan.

Tugas dari *scanner* adalah sebagai berikut :

1. Melakukan pembacaan kode sumber dengan merunut karakter demi karakter
2. Mengenali besaran leksik

3. Mentransformasi menjadi sebuah *token* dan menentukan jenis *token*-nya.
4. Mengirimkan *token*
5. Membuang/mengabaikan *blank* dan komentar dalam program
6. Menangani kesalahan
7. Menangani tabel simbol

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 :

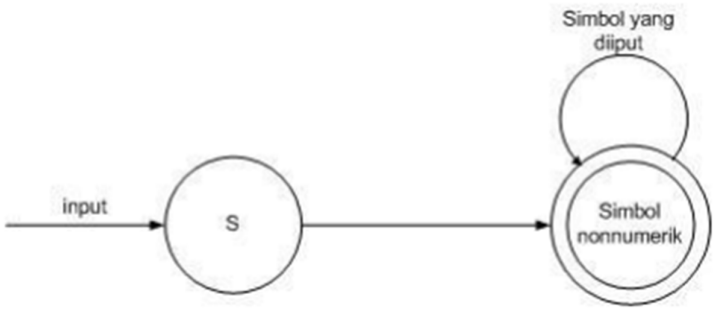
1. huruf kecil alphabet, misalnya :a, b, c
2. symbol operator, misalnya : +, -, dan ‘
3. symbol tanda baca, misalnya : (,), dan ;

Sedangkan simbol-simbol berikut adalah simbol non terminal :

1. huruf besar awal alphabet, misalnya : A, B, C

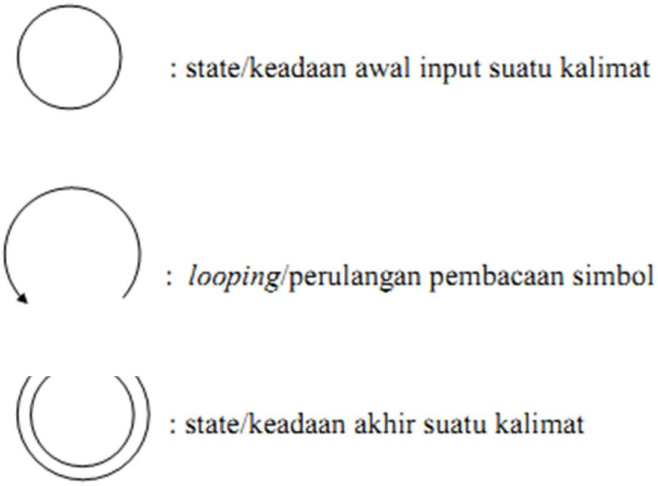
2. huruf S sebagai simbol awal

Pada saat scanner membaca input, tools yang digunakan untuk menggambarkan perpindahan dari posisi satu ke posisi lainnya adalah diagram transisi. Diagram transisi dapat dilihat pada gambar II.2



Gambar II.2 Diagram transisi

Keterangan :



II.2.4 *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 *chat bot*.

II.2.4.1 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.

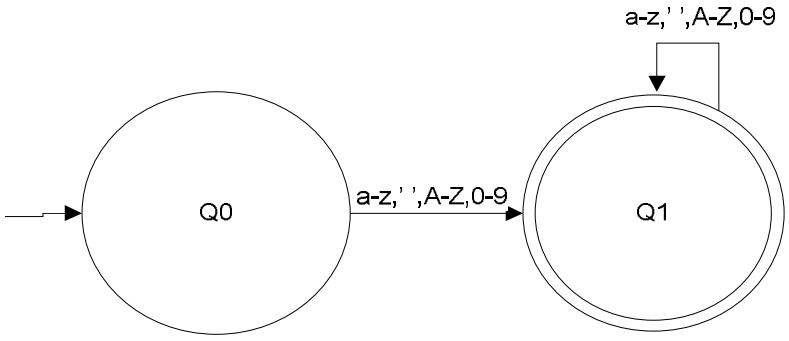
Untuk pemrosesan, dokumen dipisahkan menjadi unit-unit yang lebih kecil misalnya berupa kata, frasa atau kalimat. Unit pemrosesan tersebut disebut sebagai token. Parsing merujuk pada proses pengenalan token yang terdapat dalam rangkaian teks.

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.

Jika pengguna ingin melakukan proses pencarian yang lebih spesifik, agar informasi yang didapat lebih tepat sasaran, maka pengguna haru mengikuti aturan proses penguraian kalimat yang dimiliki sistem dapat dilihat pada gambar II.3.



Gambar II.3 *Finite State Automata* Proses *Parsing*

Konfigurasi FSA diatas secara formal dinyatakan sebagai berikut :

Q = {Q0, Q1}

∑ = {a-z,' ',A-Z,0-9}

S = Q0

F = {Q1}

Fungsi transisi yang ada sebagai berikut :

δ(Q0, a-z|' '|A-Z|0-9) = Q1

δ(Q1, a-z|' '|A-Z|0-9) = Q1

Fungsi tersebut bisa disajikan dan dilihat dalam tabel II.1 berikut :

Tabel II.1 Tabel Transisi

δ	a-z|' '|A-Z|0-9
Q0	Q1
Q1	Q1

Dari diagram state gambar II.3 , didapat aturan produksi sebagai berikut:

S ->aA..zA| AA..ZA | 0A-9A|<space>A|a..Z|A..Z|0..9|<space>

A ->aA..zA| AA..ZA |0A-9A|<space>A|a..Z|A..Z|0..9|<space>

Dimana secara formal dapat ditulis :

V = {S, A}

T = {a-z,' ' A-Z,0-9}

P = {S->aA..zA|AA..ZA|0A..9A|<space>A|a..Z|A..Z|0..9|<space>,

A->aA..zA| AA..ZA |0A..9A|<space>A|a..Z|A..Z|0..9|<space>}

S = S

Aturan produksi diatas menggunakan *left to right* sehingga jika pengguna memasukan pertanyaan “hai juga”, maka proses akan dilakukan dari depan (*left*) ke belakang (*right*), seperti berikut: :

S ->hA

S ->haA

S ->haiA

S ->haiA

S ->hai jA

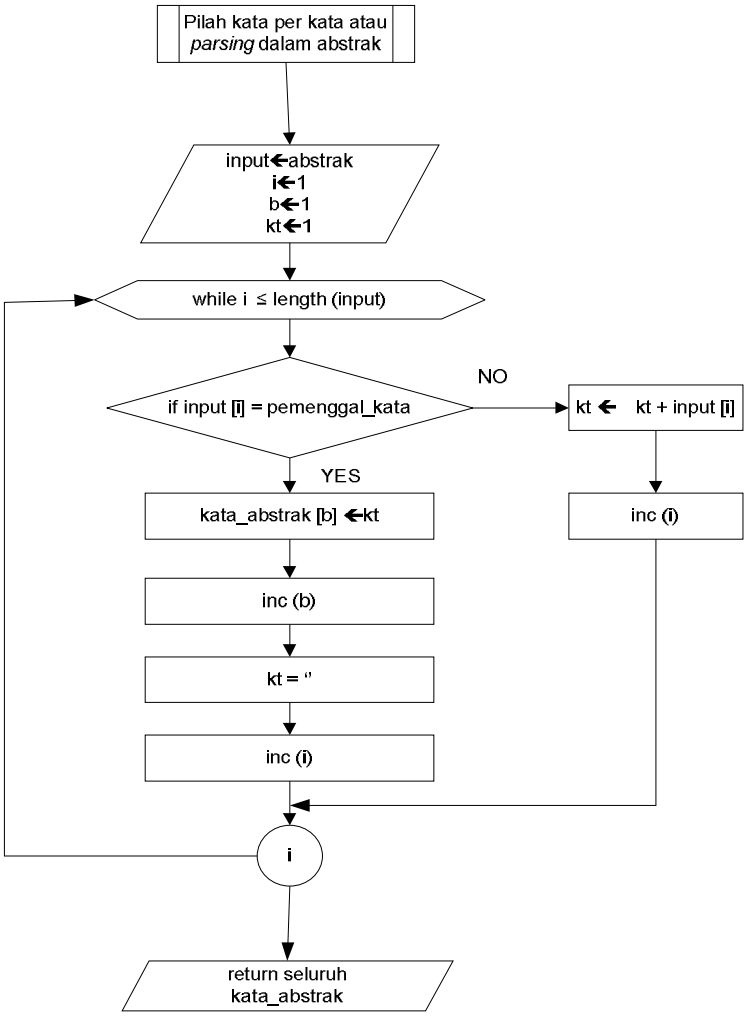
S ->hai juA

S ->hai jugA

S ->hai jugaA

S ->hai juga

Pada proses di atas setiap kalimat atau kata akan dipecah menjadi perbagian karakter dalam bentuk array. Flowchart proses parsing abstak dapat dilihat pada gambar II.4 , proses ini digunakan untuk memotong - motong dokumen kata per kata dan menyimpannya pada array kata abstrak.



Gambar II.4 ***Flowchart Parsing***

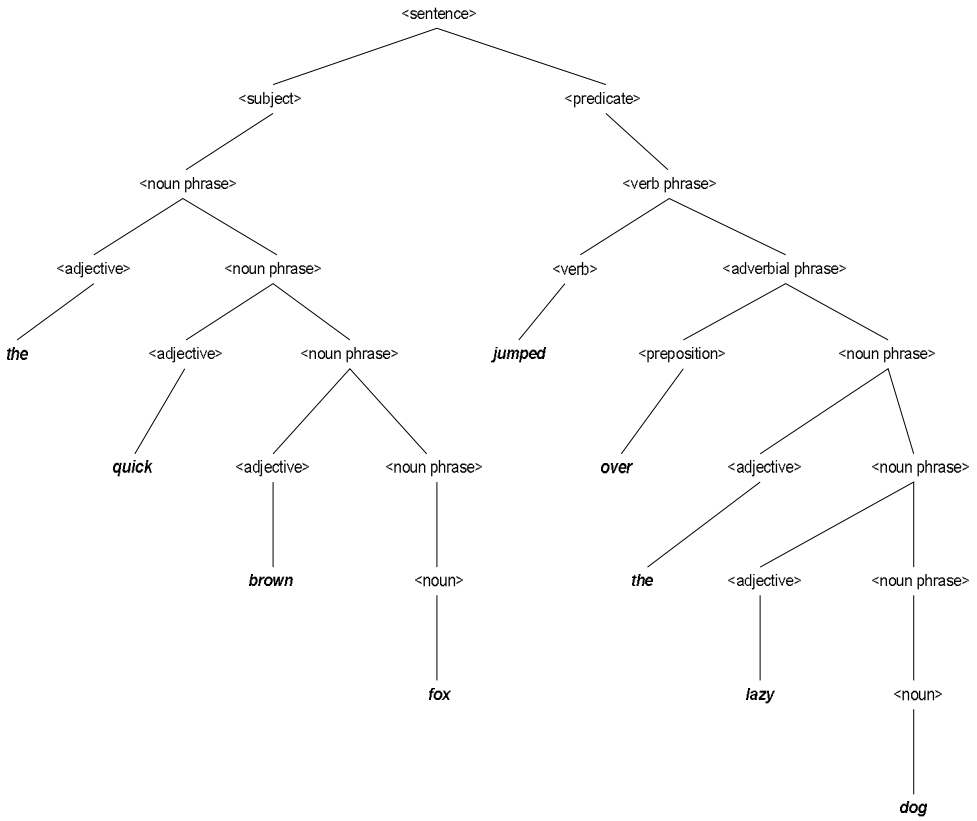
II.2.5 Pohon Sintaks

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. Contoh dapat dilihat pada gambar II.5 , dimana contoh sebuah *tree* yang menguraikan kalimat dalam bahasa Inggris.

The quick brown fox jumped over the lazy dog



Gambar II.5 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.

II.3. *Text Mining*

Text mining merupakan salah satu aplikasi dari *data mining*. *Text mining* juga sering disebut sebagai *Text Data Mining* (TDM) dan *knowladge Discovery in Textual Databases* (KDT) [8]. *Text mining* merupakan proses mengesktrak *petterns* dan *knowladge* yang bersifat menarik dan nontrivial

(penting) dari dokumen-dokumen teks. Pada intinya proses kerja *text mining* sama dengan proses kerja *data mining* pada umumnya hanya saja data yang di *mining* merupakan *text databases*.

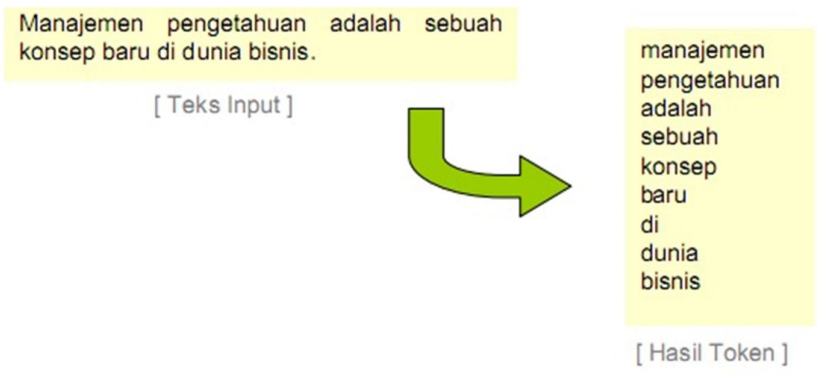
Data teks akan diproses menjadi data numerik agar dapat dilakukan proses lebih lanjut. Sehingga dalam *text mining* ada istilah *preprocessing data*, yaitu proses pendahulu yang diterapkan terhadap data teks yang bertujuan untuk menghasilkan data numerik.

Pada proses *preprosesing* merupakan tahap dimana deskripsi di tangani untuk dapat siap diproses memasuki tahap *text mining*. Tahap-tahap tersebut adalah:

1. *Parsing/ Tokenizing*
2. *Stopwords Removal/ Filtering*
3. *Stemming*
4. *Tagging*
5. *Anayizing*

II.3.1 Parsing / Tokenizing

Parsing adalah sebuah proses yang dilakukan seseorang untuk menjadikan sebuah kalimat menjadi lebih bermakna atau berada dengan cara memecah kalimat tersebut menjadi kata-kata atau *frase-frase ("Parsing")*. *Parsing* di dalam pembuatan aplikasi *text mining* ini merupakan proses penguraian deskripsi yang semula berupa kalimat-kalimat berisi kata-kata dan tanda pemisah antara kata seperti titik(.), koma(,), spasi dan tanda pemisah lain menjadi kata-kata saja baik itu berupa kata-kata penting maupun kata-kata tak penting. Secara sederhana proses *parsing* ini terlihat sebagai proses pengambilan kata jika ketemu tanda spasi namun pada kenyataannya tidak sesederhana itu. Contoh tahap ini dapat dilihat pada gambar II.6.

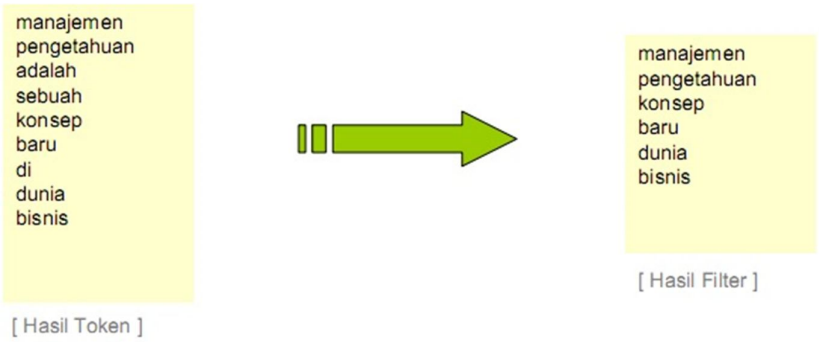


Gambar II.6 Tahap *Parsing / Tokenizing*

II.3.2 Stopword Removal / Filtering

Kebanyakan bahasa resmi di berbagai negara memiliki kata fungsi dan kata sambung seperti artikel dan preposisi yang hampir selalu muncul pada dokumen-dokumen teks. Biasanya kata-kata ini memiliki arti yang lebih di dalam memenuhi kebutuhan seorang *searcher* di dalam mencari informasi. Kata-kata tersebut (misalnya *a, an, the on* pada bahasa inggris) disebut sebagai *stopwords*. Di dalam bahasa Indonesia *stopwords* dapat disebut sebagai kata tidak penting misalnya “di”, “oleh”, “pada”, “sebuah”, “karena”. Sebelum proses *stopwords removal* dilakukan, terlebih dulu dibuat daftar stopwords (stoplist). Preposisi, kata hubung dan partikel biasanya merupakan kandidat *stoplist*.

Stopwords removal merupakan proses penghilangan kata tidak penting pada deskripsi melalui pengecekan kata-kata hasil *parsing* deskripsi apakah termasuk di dalam daftar kata tidak penting *(stoplist)* atau tidak. Jika termasuk di dalam *stoplist* maka kata-kata tersebut akan di-*remove* dari deskripsi sehingga kata-kata yang tersisa di dalam deskripsi di anggap sebagai kata-kata penting atau *keywords*. Tahap *filtering* dapat dilihat pada gambar II.7.

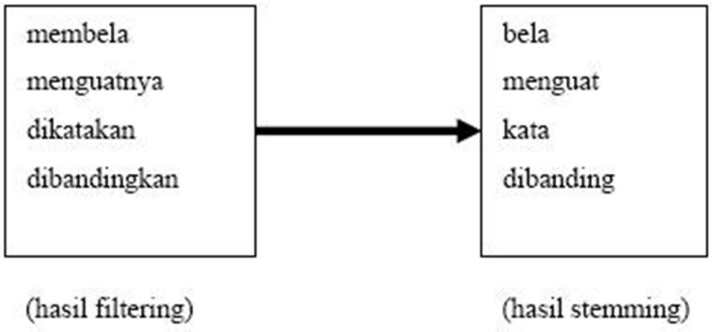


Gambar II.7 Tahap *Filtering*

II.3.3 Stemming

Stemming adalah proses pemetaan dan penguraian berbagai bentuk *(variants)* dari suatu kata menjadi bentuk kata dasarnya *(stem)*. Proses ini juga disebut sebagai *conflation*. Proses *stemming* secara luas sudah digunakan di dalam *Information retrieval* (pencarian informasi) untuk meningkatkan kualitas informasi yang didapatkan. Kualitas informasi yang dimaksud misalnya untuk mendapatkan hubungan antara variant kata yang satu dengan yang lainnya. Sebagai contoh kata “diculik”, “menculik” (melakukan tindakan menculik) dan “penculik” (orang yang menculik) yang semula mengandung arti yang berbeda dapat di-*stem* menjadi sebuah kata “culik” yang memiliki arti yang sama sehingga kata-kata diatas saling berhubungan.

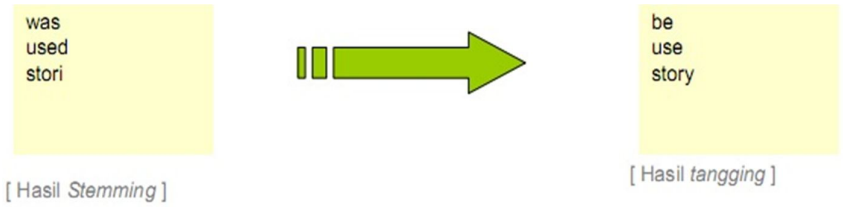
Selain itu *stemming* juga dapat digunakan untuk mengurangi ukuran dari suatu ukuran *index file*. Misalnya dalam suatu deskripsi terdapat *variant* kata “memberikan”, “diberikan”, “memberi” dan “diberi” hanya memiliki akar kata (*stem*) yaitu “beri”. Ukuran file daftar *index* yang semula berjumlah lima *record* akan di-*reduce* sehingga menjadi satu *record* saja. Gambaran tahap *stemming* dapat dilihat pada gambar II.8.



Gambar II.8 Tahap *Stemming*

II.3.4 Tagging

Tahap *tagging* adalah tahap mencari bentuk awal/ root dari tiap kata lampau atau kata hasil *stemming* yang bukan Bahasa Indonesia. Gambaran tahap *tagging* dalam Bahasa Inggris dapat dilihat pada gambar II.9.



Gambar II.9 Tahap *Tagging*

II.3.5 Analyzing

Tahap *analyzing* merupakan tahap penentuan seberapa jauh keterhubungan antara kata-kata dengan dokumen yang ada.

II.3.6 *Stemming* untuk Bahasa Indonesia Menggunakan *Confix Stripping Stemmer*

Confix Stripping (CS) stemmer adalah metode stemming untuk Bahasa Indonesia yang diperkenalkan oleh Jelita Asian. Stemmer ini merupakan pengembangan dari metode stemming untuk Bahasa Indonesia yang diperkenalkan oleh Nazief dan Adriani (1996). Algoritma stemming Nazief dan Adriani ini dikembangkan berdasarkan pada aturan morfologi Bahasa Indonesia yang mengelompokkan dan mengenkapsulasi imbuhan-imbuhan, termasuk di dalamnya adalah awalan (prefix), sisipan 17 (infix), akhiran (suffix) dan gabungan awalan-akhiran (confixes).

Algoritma ini menggunakan kamus kata dasar dan mendukung recoding, yakni penyusunan kembali kata-kata yang mengalami proses stemming berlebih. Kombinasi awalan dan akhiran yang dilarang dapat dilihat pada tabel II.2.

Tabel II.2 Kombinasi Awalan-Akhiran yang dilarang

Awalan (*prefix)*	Akhiran (*suffix)* yang tidak diperbolehkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

Algoritma stemmer Nazief dan Adriani mengelompokkan imbuhan ke dalam beberapa kategori sebagai berikut:

1. Inflection suffixes yakni kelompok-kelompok akhiran yang tidak merubah bentuk kata dasar. Sebagai contoh, kata “duduk” yang diberikan akhiran “-lah” akan menjadi “duduklah”. Kelompok ini dapat dibagi menjadi dua :
   a. Particle (P) atau partikel, yakni termasuk di dalamnya “-lah”, “-kah”, “-tah”, dan “-pun”.
   b. Possessive Pronoun (PP) atau kata ganti kepunyaan, termasuk di dalamnya adalah “-ku” , “-mu”, dan “-nya”.
2. Derivation Suffixes (DS) yakni kumpulan akhiran asli Bahasa Indonesia yang secara langsung ditambahkan pada kata dasar. Termasuk di dalamnya adalah akhiran “-i”, “-kan”, dan “-an”.
3. Derivation Prefixes (DP) yakni kumpulan awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah mendapatkan penambahan sampai dengan 2 awalan. Termasuk di dalamnya adalah awalan yang dapat bermorfologi (“me-”, “be-”, “pe-”, dan “te-”) serta awalan yang tidak bermorfologi (“di-”, “ke-” dan “se-”).

Berdasarkan pengklasifikasian imbuhan-imbuhan di atas, maka bentuk kata berimbuhan dalam Bahasa Indonesia dapat dimodelkan sebagai berikut:

[DP + [DP + [DP +]]] Kata Dasar [[+ DS][+ PP][+ P]]

Dengan batasan-batasan sebagai berikut :

a. Tidak semua kombinasi diperbolehkan, sebagai contoh pada kata yang diberi awalan “di-”, maka penambahan akhiran “-an” tidak diperkenankan. Kombinasi-kombinasi imbuhan yang tidak diperbolehkan dapat dilihat pada Tabel 2.1.
b. Penggunaan imbuhan yang sama secara berulang tidak diperkenankan.
c. Jika suatu kata hanya terdiri dari satu atau dua huruf, maka proses stemming tidak dilakukan.
d. Penambahan suatu awalan tertentu dapat mengubah bentuk asli kata dasar, ataupun awalan yang telah diberikan sebelumnya pada kata dasar bersangkutan (bermorfologi). Sebagai contoh, awalan “me-” dapat berubah menjadi “meng-”, “men-”, “meny-”, dan “mem-”. Oleh karena itu, diperlukan suatu aturan yang mampu mengatasi masalah morfologi ini.

Pada dasarnya, algoritma confix stripping stemmer dikembangkan dari algoritma stemming yang dibuat oleh Nazief dan Adriani, dengan beberapa penambahan aturan tertentu yang telah terbukti mampu meningkatkan kinerja stemmer tersebut. Algoritma stemmer yang diperkenalkan Nazief dan Adriani didefinisikan sebagai berikut:

1. Kata yang hendak di-stemming dicari terlebih dahulu pada kamus. Jika kata ditemukan dalam kamus, berarti kata tersebut sudah berbentuk kata dasar, jika tidak maka tahap selanjutnya dilakukan.
2. Hilangkan inflectional particle P (“-lah”, “-kah”, “-tah”, “-pun”) dan kata ganti kepunyaan atau possessive pronoun PP (“-ku”, “-mu”, “-nya”).
3. Hilangkan derivation suffixes DS (“-i”, “-kan”, atau “-an”).
4. Hilangkan derivation prefixes DP {“di-”,“ke-”,“se-”,“me-”,“be-”,“pe”, “te-”} dengan iterasi maksimum adalah 3 kali:
   a. Langkah 4 berhenti jika:

Terjadi kombinasi awalan dan akhiran yang terlarang seperti pada Tabel 2.1.

Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.

Tiga awalan telah dihilangkan.

b. Identifikasikan tipe awalan dan hilangkan. Awalan ada dua tipe:
   Standar: “di-”, “ke-”, “se-” yang dapat langsung dihilangkan dari kata.
   Kompleks: “me-”, “be-”, “pe”, “te-” adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 2.2 untuk mendapatkan pemenggalan yang tepat.
c. Cari kata yang telah dihilangkan awalannya ini di dalam kamus. Apabila tidak ditemukan, maka langkah 4 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.

5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses recoding dilakukan dengan mengacu pada aturan pada Tabel 2.2. Recoding dilakukan dengan menambahkan karakter recoding di awal kata yang dipenggal. Pada Tabel 2.2, karakter recoding adalah huruf kecil setelah tanda hubung (‘-’) dan terkadang berada sebelum tanda kurung.
6. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

Dilihat segi performa, stemmer ini mampu memberikan hasil yang lebih baik jika dibandingkan dengan algoritma stemmer untuk Bahasa Indonesia seperti yang dikembangkan oleh Arifin dan Setiono (2002) , Vega (2001), Ahmad (1996), dan Idris (2001). Hal ini wajar, mengingat algoritma ini merupakan algoritma yang paling kompleks. Jelita Asian kemudian mengembangkan algoritma CS stemmer, dengan menambahkan beberapa perbaikan yang bertujuan untuk meningkatkan hasil stemming yang diperoleh. Perbaikan yang ditambahkannya adalah sebagai berikut:

1. Menggunakan kamus kata dasar yang lebih lengkap.
2. Memodifikasi dan menambahkan aturan pemenggalan untuk tipe awalan yang kompleks (memodifikasi aturan pada Tabel 2.2 sesuai

modifikasi pada Tabel 2.3, dan menambahkan aturan pada Tabel 2.4 ke Tabel 2.2).

3. Menambahkan aturan stemming untuk kata ulang dan bentuk jamak, semisal pada “buku-buku” yang seharusnya di-stemming menjadi “buku”. Caranya, adalah dengan melakukan pemisahan menjadi dua sub-kata, yang masing-masing di-stemming. Apabila stemming memberikan kata dasar yang sama, maka output kata dasarnya adalah hasil stemming tersebut. Namun apabila hasil stemming 2 sub-kata ini berbeda, maka dapat disimpulkan bahwa input adalah kata ulang semu, dan tidak memiliki bentuk kata dasar lagi.
4. Mengubah urutan stemming untuk beberapa kasus tertentu. Algoritma stemmer Nazief dan Adriani akan menghilangkan akhiran terlebih dahulu, baru diikuti penghilangan awalan. Menurut Jelita Asian, cara ini tidak selalu berhasil pada beberapa kata. Oleh karena itu, diberikan beberapa aturan yang akan mengubah urutan stemming, dimana penghilangan awalan dilakukan terlebih dahulu, lalu diikuti penghilangan akhiran. Aturan ini disebut rule precedence, dan berlaku jika suatu kata memiliki pasangan awalan-akhiran “be-lah”, “be-an”, “me-i”, “di-i”, “pe-i”, atau “te-i”.

Pada Tabel II.3, II.4, dan II.5, simbol C merupakan konsonan, simbol V menandakan vokal, simbol A merupakan vokal atau konsonan, dan simbol P merepresentasikan partikel atau fragmen dari suatu kata, misalnya “er”.

Tabel II.3 Aturan Pemenggalan Awalan Stemmer Nazief dan Andriani

Aturan	Format Kata	Pemenggalan
1	berV…	ber-V…| ber-rV…
2	berCAP…	ber-CAP…dimana C!='r' & P!='er'
3	berCAerV…	ber-CaerV…dimana C!='r'
4	belajar	bel-ajar
5	beC_1erC_2 …	be-C_1erC_2… dimana C_1!={'r'|'l'}
6	terV…	ter-V…| te-rV…
7	terCerV…	ter-CerV… dimana C!='r'
8	terCP…	ter-CP… dimana C!='r' dan P!='er'
9	teC_1erC_2…	te-C_1erC_2… dimana C_1!='r'
10	me{l|r|w|y}V…	me-{l|r|w|y}V…
11	mem{b|f|v}…	mem-{b|f|v}…
12	mempe{r|l}…	mem-pe…
13	mem{rV|V}…	me-m{rV|V}… | me-p{rV|V}…
14	men{c|d|j|z}…	men-{c|d|j|z}…
15	menV…	me-nV… | me-tV
16	meng{g|h|q}…	meng-{g|h|q}…
17	mengV…	meng-V… | meng-kV…
18	menyV…	meny-sV…
19	mempV…	mem-pV… dimana V!='e'
20	pe{w|y}V…	pe-{w|y}V…
21	perV…	per-V…| pe-rV…
23	perCAP…	per-CAP… dimana C!='r' dan P!='er'
24	perCAerV…	per-CAerV… dimana C!='r'
25	pem{b|f|V}…	Pem-{b|f|V}…
26	pem{rV|V}…	Pe-m{rV|V}… | pe-p{rV|V}…
27	pen{c|d|j|z}…	pen-{c|d|j|z}…
28	penV…	pe-nV… | pe-tV…
29	peng{g|h|q}…	peng-{g|h|q}…
30	pengV…	peng-V… | peng-kV…
31	penyV…	peny-sV…
32	pelV…	pe-lV… kecuali "pelajar" yang menghasilkan "ajar"
33	peCerV…	per-erV… dimana C!={r|w|y|l|m|n}
34	peCP…	pe-CP… dimana C!={r|w|y|l|m|n} dan P!='er'

Tabel II.4 Modifikasi Aturan pada Tabel II.3

Aturan	Format Kata	Pemenggalan
12	mempe…	mem-pe…
16	meng{g|h|q|k}…	meng-{g|h|q|k}…

Tabel II.5 Tambahan Aturan untuk Tabel II.3

Aturan	Format Kata	Pemenggalan
35	terC_1erC_2 …	ter-C_1erC_2… dimana C_1!='r'
36	peC_1erC_2 …	pe-C_1erC_2… dimana C_1!={r|w|y|l|m|n}

Algoritma confix stripping (CS) stemmer yang dikembangkan Jelita Asian bekerja sebagai berikut :

1. Cek rule precedence, apabila bernilai benar maka lakukan penghilangan awalan terlebih dahulu. Apabila bernilai salah, maka penghilangan akhiran dilakukan terlebih dahulu.
2. Lakukan recoding apabila diperlukan.
3. Cek apakah terdapat tanda hubung ('-') yang menandakan bahwa input kata tersebut adalah kata ulang. Jika benar, maka lakukan proses stemming pada potongan kata di sebelah kiri dan kanan tanda hubung tersebut. Apabila stemming dua kata ini memberikan hasil yang sama, maka kata dasar kata ulang tersebut adalah hasil stemming yang didapatkan.
4. Jika ketiga proses di atas gagal, maka input kata yang di-stemming dianggap sebagai kata dasar. Pada setiap perpindahan langkah, dilakukan proses pencarian output stemming ke kamus. Apabila ditemukan di kamus, maka algoritma ini berhenti. Berikut adalah contoh proses stemming pada kata "menangkapnya" dengan menggunakan confix stripping stemmer:
    1) Rule precedence bernilai false karena tidak ditemukan kombinasi awalan dan akhiran yang tidak diperbolehkan.Pemenggalan akhiran dilakukan terlebih dahulu, dan menyisakan kata "menangkap".

2) Pemenggalan awalan menyisakan kata “nangkap”.
3) Karena “nangkap” tidak terdapat di kamus, maka lakukan recoding dengan mengganti ‘n’ dengan ‘t’ (aturan 15 pada Tabel II.3).
4) Kata “tangkap” ada di kamus, oleh karena itu proses berhenti. Kata dasar “menangkapnya” adalah “tangkap”.

Sayangnya, setelah melakukan beberapa analisa dan percobaan kecil, ditemukan beberapa contoh kata yang tidak dapat di-stemming. Analisa atas beberapa kata yang gagal di-stemming tersebut adalah sebagai berikut:

1) Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format “mem+p...”. Hal ini terjadi pada kata “mempromosikan”, “memproteksi”, dan “memprediksi”.
2) Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format “men+s...”. Hal ini terjadi pada kata “mensyaratkan”, dan “mensyukuri”.
3) Kurang relevannya aturan 17 untuk pemenggalan awalan pada kata-kata dengan format “menge+kata dasar”, seperti pada kata “mengerem”.
4) Kurang relevannya aturan 30 untuk pemenggalan awalan pada kata-kata dengan format “penge+kata dasar”, seperti pada kata “pengeboman”.
5) Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format “peng+k...” seperti pada kata “pengkajian”.
6) Adanya elemen pada beberapa kata dasar yang menyerupai suatu imbuhan. Kata-kata seperti “pelanggan”, “perpolitikan”, dan “pelaku” gagal di-stemming karena akhiran “-an”, “-kan” dan “-ku” seharusnya tidak dihilangkan.

Berdasarkan kegagalan-kegagalan ini, maka untuk memperbaiki kesalahan algoritma CS stemmer dengan menambah beberapa perbaikan, diantaranya adalah sebagai berikut:

1) Merevisi aturan 19 pada Tabel II.3 agar stemming berhasil pada kata-kata dengan format “mem+p...”. Revisi aturan ini dapat dilihat pada Tabel II.6.

2) Merevisi aturan 14 pada Tabel II.3 agar stemming berhasil pada kata-kata dengan format "men+s...". Revisi aturan ini dapat dilihat pada Tabel II.6.
3) Merivisi aturan 17 pada Tabel II.3 agar stemming berhasil pada kata-kata dengan format "menge+...". Revisi aturan ini dapat dilihat pada Tabel II.6.
4) Merivisi aturan 30 pada Tabel II.3 agar stemming berhasil pada kata-kata dengan format "penge+...". Revisi aturan ini dapat dilihat pada Tabel II.6.
5) Merevisi aturan 29 pada Tabel II.3 agar stemming berhasil pada kata-kata dengan format "peng+k...". Revisi aturan ini dapat dilihat pada Tabel II.6.
6) Menambahkan suatu algoritma tambahan untuk mengatasi kesalahan pemenggalan akhiran yang seharusnya tidak dilakukan. Algoritma ini disebut loopPengembalianAkhiran, dan dilakukan apabila proses recoding gagal.

Algoritma loopPengembalianAkhiran dideskripsikan sebagai berikut:

1. Kembalikan seluruh awalan yang telah dihilangkan sebelumnya, sehingga menghasilkan model kata seperti berikut: [DP+[DP+[DP]]] + Kata Dasar. Pemenggalan awalan dilanjutkan dengan proses pencarian di kamus kemudian dilakukan pada kata yang telah dikembalikan menjadi model tersebut.
2. Kembalikan akhiran sesuai dengan urutan model. Ini berarti bahwa pengembalian dimulai dari DS ("-i", "-kan", "-an"), lalu PP("-ku", "-mu", "-nya"), dan terakhir adalah P ("-lah", "-kah", "-tah", "-pun"). Untuk setiap pengembalian, lakukan langkah 3) hingga 5) berikut. Khusus untuk akhiran "-kan", pengembalian pertama dimulai dengan "k", baru kemudian dilanjutkan dengan "an".
3. Lakukan pengecekan di kamus. Apabila ditemukan, proses dihentikan. Apabila gagal, maka lakukan proses pemenggalan awalan berdasarkan aturan pada Tabel II.3 (dengan revisi Tabel II.6).
4. Lakukan recoding apabila diperlukan.
5. Apabila pengecekan di kamus tetap gagal setelah recoding, maka awalan-awalan yang telah dihilangkan dikembalikan lagi.

Tabel II.6 Revisi untuk Tabel II.3

Aturan	Format Kata	Pemenggalan
14	men{c|d|j|s|z}…	men-{c|d|j|s|z}…
17	mengV…	meng-V… |meng-kV…| (mengV-…jika V=’e’)
19	mempA…	mem-pA… dengan A!=’e’
29	pengC…	peng-C…
30	pengV…	peng-V… |peng-kV… | (pengV-…jika V=’e’)

II.4. *Chatbot*

Penerapan AI salah satunya adalah dalam bentuk chat bot. Chat bot merupakan sebuah program komputer yang diprogram untuk dapat berkomunikasi dengan manusia menggunakan bahasa manusia itu sendiri. Salah satunya contoh kongkritnya adalah Help Bot pada Yahoo!Messenger dan ALICE (Artificial Linguistic Internet Computer Entity) yang dikembangkan oleh Dr. Richard S. Wallace.

Atau dapat dikatakan juga chatbot (atau *chatterbot*, atau *bots*) adalah sebuah program komputer yang dirancang untuk menstimulasikan percakapan intelektual dengan satu atau lebih manusia baik secara audio maupun teks.

Pada mulanya, program komputer (*bots*) ini diuji melalui *Turing Test*, yaitu dengan merahasiakan identitasnya sebagai mesin sehingga dapat mengelabui orang yang bercakap-cakap dengannya. Jika pengguna tidak dapat mengidentifikasi *bots* sebagai suatu program komputer, maka *chatterbot* tersebut dikategorikan sebagai kecerdasan buatan (atau *artificial intelligence*). Zaman sekarang ini, *chatterbot* telah dimanfaatkan untuk tujuan praktis seperti bantuan online, layanan personal, atau akuisisi informasi, dalam hal ini dapat dilihat fungsi program sebagai suatu jenis agen percakapan (atau *conversational agent*). Yang membedakan *chatterbot* dengan sistem pemrosesan bahasa alami (atau *Natural Language Processing System*) adalah kesederhanaan algoritma yang digunakan.

Meskipun banyak *bots* yang tampaknya dapat menginterpretasikan dan menanggapi masukan dari manusia, sebenarnya *bots* tersebut hanya

menganalisis kata kunci dalam masukan dan membalasnya dengan kata kunci yang paling cocok, atau pola kata-kata yang paling mirip dari basis data tekstual.

Chatbot juga merupakan *QA system* atau *question-answering system*, yaitu memberikan kemampuan pada sebuah mesin (komputer) untuk menginterpretasikan bahasa alami untuk melakukan dialog dengan pengguna hampir seperti dialog antara dua orang manusia dalam bahasa sehari-hari.

Baru-baru ini telah muncul ide-ide baru di dalam pengembangan ALICE di antaranya adalah kemampuan untuk menjaga alur pembicaraan untuk setiap user yang melakukan pembicaraan dengannya. Selain itu juga mulai telah dipikirkan suatu cara agar nantinya program tersebut dapat memprediksi atau mengidentifikasi umur, jenis kelamin,lokasi geografis dan pekerjaan dari client (user) yang melakukan perbincangan dengannya.

II.4.1 Penggunaan *Brainfile* Pada *Chatbot*

Brain file merupakan otak dari *chat bot* itu sendiri yang menentukan bagaimana cara *chat bot* berpikir dan akan memberikan respon. *Brain file* biasanya berupa file *plain text. Brain file* berfungsi sebagaimana tabel informasi pada kompilator bahasa pemrograman tingkat tinggi. Di dalam *brain file* inilah disimpan semua kosakata, kepribadian, dan pengetahuan (*knowledge*) dari *chat bot*. Semakin banyak pengetahuan yang dimiliki *chat bot* maka akan semakin besar ukuran file dari *brain file* tersebut.

II.5. Perangkat Lunak

Perangkat Lunak (*software*) merupakan suatu program yang dibuat oleh pembuat program untuk menjalankan perangkat keras komputer. Perangkat Lunak adalah program yang berisi kumpulan instruksi untuk melakukan proses pengolahan data. *Software* sebagai penghubung antara manusia sebagai pengguna dengan perangkat keras komputer, berfungsi menerjemahkan bahasa manusia ke dalam bahasa mesin sehingga perangkat keras komputer memahami keinginan pengguna dan menjalankan instruksi

yang diberikan dan selanjutnya memberikan hasil yang diinginkan oleh manusia tersebut.

Perangkat lunak terdiri dari kode-kode yang dibuat atau ditulis dalam bahasa pemrograman tertentu. Banyak bahasa pemrograman yang dapat dijadikan alat bantu untuk merancang sebuah perangkat lunak antara lain bahasa PASCAL, C, C++, BASIC, Java, HTML, PHP, JavaScript, AIML, dan bahkan tersedia *tool* yang berbasis grafis seperti Visual Basic, Visual C++, Borland Delphi, dan masih banyak lagi *tool* dan bahasa pemrograman lainnya.

II.6. *Object Oriented Programming* (OOP)

Object-Oriented Programming (OOP) adalah sebuah pendekatan untuk pengembangan / *development* suatu *software* dimana dalam struktur *software* tersebut didasarkan kepada interaksi *object* dalam penyelesaian suatu proses/tugas. Interaksi tersebut mengambil form dari pesan-pesan dan mengirimkannya kembali antar *object* tersebut. *Object* akan merespon pesan tersebut menjadi sebuah tindakan / *action* atau metode.

Object-oriented programs terdiri dari *objects* yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Seperti dunia nyata, *users* dari *software* programs dilibatkan dari logika proses untuk menyelesaikan tugas. Adapun karakteristik yang dimiliki OOP yaitu :

a. Objects

Object adalah sebuah *structure* yang menggabungkan data dan prosedur untuk bekerja bersama-sama.

b. Abstraction

Ketika manusia berinteraksi dengan *object-object* di dunia ini, manusia sering hanya konsentrasi dengan sebuah bagian dari propertiesnya.

Ketika membangun *objects* dalam aplikasi OOP, adalah penting untuk menggabungkan konsep *abstraction* ini. Jika membangun aplikasi *shipping*, maka harus membangun *object* produk dengan atribut seperti ukuran dan berat. Warna adalah contoh informasi yang tidak ada hubungannya dan harus dibuang. Tetapi ketika akan membangun *order-*

entry application, warna menjadi penting dan harus termasuk atribut *object* produk.

c. Encapsulation

Ciri penting lainnya dari OOP adalah *encapsulation*. *Encapsulation* adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan, bahkan *hidden*. Jika ingin mendapat data, harus berinteraksi dengan *object* yang bertanggung jawab atas data tersebut.

d.Polymorphism

Polymorphisms adalah kemampuan 2 buah *object* yang berbeda untuk merespon pesan permintaan yang sama dalam suatu cara yang unik. Dalam OOP, diterapkan tipe *polymorphism* melalui proses yang disebut *overloading*. Dapat dilakukan dalam implementasi metode yang berbeda pada sebuah *object* yang mempunyai nama yang sama.

e.Inheritance

Penggunaan *inheritance* dalam OOP untuk mengklasifikasikan *objects* dalam program sesuai karakteristik umum dan fungsinya.

II.6.1 *Unified Model Language*

UML *(Unified Modeling Language)* adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. Artifact dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

Bagian-bagian utama dari UML adalah view, diagram, model element, dan *general mechanism*.

1. *View*

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *view* dalam UML antara lain: *use case view, logical view, component view, concurrency view, dan deployment view.*

a) Use case view

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *View* ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

b) Logical view

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,dan relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam class diagrams untuk struktur statis dan dalam *state, sequence, collaboration, dan activity diagram* untuk model dinamisnya. View ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

c) Component view

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari code module diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. View ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).

d) Concurrency view

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis *(state, sequence, collaboration, dan activity diagrams)* dan diagram implementasi *(component dan*

deployment diagrams) serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

e) Deployment view

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

2. Diagram

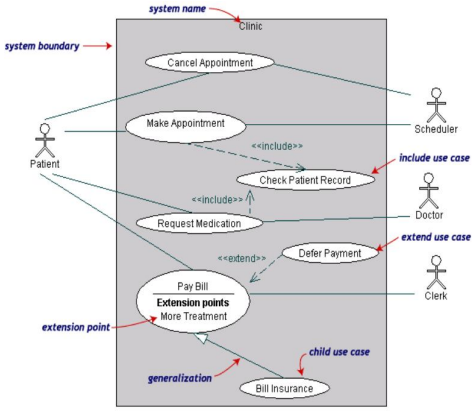
Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

a) *Use Case Diagram*

Use case adalah abstraksi dari interaksi antara sistem dan actor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Contoh *use case diagram* dapat dilihat pada gambar II.10

.



Gambar II.10 *Use Case Diagram*

b) *Class Diagram*

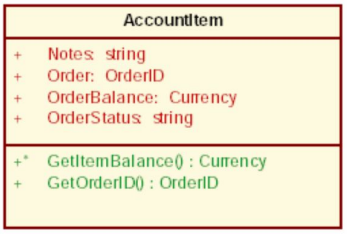
Class adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class* diagram dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class- class yang ada. Sebuah sistem biasanya mempunyai beberapa *class* diagram. *Class* diagram sangat membantu dalam visualisasi struktur kelas dari suatu sistem. *Class* memiliki tiga area pokok :

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

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
c. *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.* Contoh *class diagram* dapat dilihat pada gambar II.11.



Gambar II.11 *Class Diagram*

c) *Component Diagram*

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi *software* dari sebuah atau lebih *class*. Komponen dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga component diagram merepresentasikan dunia nyata yaitu *component software* yang mengandung *component, interface dan relationship.*

d) *Deployment Diagram*

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

e) *State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

f) *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

g) *Collaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequence* diagrams. Dalam menunjukkan pertukaran pesan, *collaboration* diagrams menggambarkan *object* dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence* diagrams, tapi jika penekanannya pada konteks gunakan *collaboration* diagram.

h) *Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

II.7. Basis Data

Basis data merupakan kumpulan dari kelompok data (arsip) yang saling berhubungan dimana masing-masing kelompok data tersebut memiliki jenis dan fungsi tertentu. Biasanya suatu basis data diorganisir sedemikian rupa sehingga dapat dimanipulasi dengan cepat dan mudah.

Basis data terdiri dari 2 kata, yaitu basis dan data. Basis dapat diartikan sebagai tempat berkumpul, sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia, barang, hewan, peristiwa, dan sebagainya yang direkam dalam bentuk angka, huruf, symbol, teks, gambar, bunyi atau kombinasinya.

Ada beberapa operasi dasar yang dapat dilakukan berkenaan dengan basis data, antara lain:

1. Pembuatan basis data baru (create database)
2. Penghapusan basis data (drop database)

3. Pembuatan file/tabel baru ke suatu basis data (create table)
4. Penghapusan file/tabel dari suatu basis data (drop table)
5. Penambahan/pengisian data baru ke sebuah file/tabel (insert)
6. Pengambilan/pencarian data dari sebuah file/tabel (retrieve/search)
7. Pengubahan data dari sebuah file/tabel (update)
8. Penghapusan data dari sebuah file/tabel (delete)

Istilah-istilah dalam database :

a. *Entity*

   Entity adalah orang, tempat, kejadian atau konsep yang informasinya direkam. Misalnya pada sekolah terdapat entity mahasiswa, matakuliah, dosen, nilai test dan lain-lain.

b. *Atribute*

   Setiap entity mempunyai atribute atau sebutan untuk mewakili suatu entity. Seorang mahasiswa dapat dilihat dari atributenya, misalnya nim, nama, alamat, jenis kelamin dan lain-lain. Atribute juga disebut sebagai data elemen, data field, data item.

c. *Data value* (nilai atau isi data)

   Adalah data actual atau informasi yang disimpan pada tiap data elemen atau atribute. Atribute nama mahasiswa menunjukan tempat dimana informasi nama mahasiswa disimpan, sedang data value adalah Pita, Respati, merupakan isi data nama mahasiswa tersebut.

d. *Record* (tupel)

   Yaitu kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entity secara lengkap. Satu record mewakili satu data atau informasi tentang seseorang misalnya, nomor induk mahasiswa, nama, alamat, jenis kelamin dan seterusnya.

e. *File*

   Kumpulan record-record sejenis yang mempunyai panjang elemen yang sama, atribute yang sama, namun berbeda-beda data valuenya.

f. *Database*

Kumpulan file-file yang mempunyai kaitan antara satu file dengan file yang lain sehingga membentuk satu bangunan data untuk menginformasikan satu perusahaan, instansi dalam batasan tertentu.

g. *Database Management System* (DBMS)

Kumpulan file yang saling berkaitan bersama dengan program untuk pengelolaannya disebut DBMS. Database adalah kumpulan datanya, sedangkan program pengelolanya berdiri sendiri dalam satu paket program yang berfungsi untuk membaca data, mengisi data, menghapus data serta melaporkan data dalam database.

II.8. Teknologi Web

Web adalah sebutan bagi sekelompok halaman web (*web page*), yang umumnya merupakan bagian dari suatu nama domain (domain name) atau subdomain di *World Wide Web* (WWW) di Internet. WWW terdiri dari seluruh situs web yang tersedia kepada publik.

Halaman-halaman sebuah situs web diakses dari sebuah URL yang menjadi "akar"(*root*), yang disebut homepage dan biasanya disimpan dalam server yang sama. Web memiliki kemampuan link yang sangat bagus. Keistimewaan inilah yang telah menjadikan Web sebagai *service* yang paling cepat pertumbuhannya. Web mengizinkan pemberian highlight pada kata atau gambar dalam sebuah dokumen untuk dihubungkan ke media lain seperti dokumen, frase, movie clip, atau file suara.

Tipe yang paling populer adalah dokumen hypertext yang disusun menurut bahasa khusus, seperti *Hypertext Markup Language* (HTML) sedangkan untuk interaktif web atau dynamic page terdapat banyak pilihan, salah satunya adalah *PHP Hypertext Preprocessor* (PHP).

Pada prinsipnya web bekerja dengan cara menampilkan file PHP yang berasal dari server web pada program client, yaitu browser web. Program browser pada client mengirim permintaan kepada server web, yang kemudian dikirimkan oleh serverdalam bentuk Format HTML. File PHP berisi instruksi-instruksi yang diperlukan untuk membentuk dynamic page.

Perintah-perintah PHP ini kemudian diterjemahkan oleh browser sehingga isi informasinya dapat ditampilkan secara visual kepada pengguna di layar komputer. Pada penggunaannya, mekanisme web juga dapat diakses melalui Local Area Network (LAN) sehingga tidak perlu terhubung dengan internet.

II.9. Perangkat Lunak Pendukung

Perangkat lunak pendukung adalah program yang digunakan sebagai perantara antara program aplikasi dan perangkat keras komputer.

II.9.1 PHP

PHP singkatan dari *Personal Home Page Tools*, adalah sebuah bahasa *scripting* yang dibundel dengan HTML, yang dijalankan disisi server. Sebagian besar perintahnya berasal dari C, Java dan dengan beberapa tambahan fungsi khusus PHP. Bahasa ini memungkinkan para pembuat aplikasi web menyajikan halaman HTML dinamis dan interaktif dengan cepat dan mudah, yang dihasilkan server.

Sistem manajemen database yang sering digunakan bersama PHP adalah MySQL. Namun PHP juga mendukung system manajemen Database Oracle, Microsoft Acces, Interbase, d-Base, PostgreSQL dan sebagainya.

PHP membuat proses pengembangan aplikasi menjadi mudah karena kelebihan-kelebihannya yaitu:

1. *Script developer* (kode program) terintegrasi dengan file HTML, sehingga dapat berkonsentrasi langsung pada penampilan dokumen *web*nya.
2. Tidak ada proses *compiling* dan *linking*.
3. Berorientasi objek (*Object Oriented*).
4. Sintaksis pemogramannya mudah dipelajari, sangat menyerupai C dan Perl.
5. Integrasi yang sangat luas ke berbagai server database.

Script PHP berkedudukan sebagai tag dalam bahsa HTML. Suatu skrip akan dikenali sebagai script PHP bila diapit oleh tanda :

a. <?php……?>
b. <?.....?>
c. <script language=”PHP”>……</script>

Script yang dibuat dengan PHP disimpan dengan nama file dan diikuti dengan ekstensi *.php, misalnya : contoh.php. Bila script PHP diakses melalui komputer local maka file PHP disimpan di folder htdocs di local web server.

PHP mempunyai fungsi khusus untuk mengakses MySQL. Ada sekitar 48 fungsi yang didukung PHP dalam mengakses MySQL dalam membuat aplikasi. Adapun yang bisa digunakan diantaranya adalah :

a. *mysql_connect*()

   Fungsi mysql_connect adalah untuk menghubungkan PHP dengan database MySQL. Format fungsinya adalah :

   mysql_connect(string hostname, string username, string password);

b. mysql_select_db

   Setelah terhubung ke database MySQL dengan menggunakan mysql_connect, langkah selanjutnya adalah memilih database yang akan digunakan. Fungsi mysql_select_db digunakan untuk memilih database. Format fungsinya adalah :

   mysql_select_db(string database, koneksi)

   koneksi ialah variable yang terhubung MySQL. Jika tidak mengisi variable koneksi maka koneksi yang terbuka saat itulah yang digunakan. Manfaat berbagai macam koneksi adalah bahwa dengan pilihan seperti itu maka dalam satu file dimungkinkan mengambil query dari 2 database sekaligus.

c. mysql_query

   Dalam database MySQL, perintah untuk melakukan transaksi ialah perintah SQL. Sebutan untuk mengirim perintah SQL dinamakan query. Query memberi perintah kepada database untuk melakukan apa yang diinginkan. Format fungsinya :

Int mysql_query(string query, int[link, identifier];

d. *mysql_num_rows*

Kegunaan dari fungsi ini adalah untuk menghitung jumlah baris yang dikenai oleh proses SQL. Format fungsinya adalah :

Int mysql_num_rows(int result);

e. *mysql_fetch_array*

fungsi ini berkaitan dengan menampilkan data. Untuk menampilkan data, digunakan fungsi mysql_fetch_array. Dengan fungsi ini, hasil query ditampung dalam bentuk array. Format fungsinya adalah :

Array mysql_fetch_array(int result, int[result type]);

II.9.2 *My SQL*

Pada awal perkembangannya disebut SQL yang merupakan kepanjangan dari Structured Query Language. SQL merupakan bahasa terstruktur yang khusus digunakan untuk mengolah database. SQL pertama kali didefinisikan oleh American National Standards Institute (ANSI) pada tahun 1986. MySQL adalah sebuah sistem manajemen database yang bersifat open source. MySQL adalah pasangan serasi dari PHP. MySQL dibuat dan dikembangkan oleh MySQL AB yang berada di Swedia.

MySQL dapat digunakan untuk membuat dan mengola database beserta isinya. Kita dapat memanfaatkan MySQL untuk menambahkan, mengubah dan menghapus data yang berada dalam database. MySQL merupakan sisitem manajemen database yang bersifat ***at relational.*** Artinya data-data yang dikelola dalam database akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan menjadi jauh lebih cepat.

MySQL (*My Strukture Query Language*) atau yang biasa dibaca " mai-se- kuel" adalah salah satu jenis *database* server yang sangat terkenal. MySQL sangat popular karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *database*. MySQL bersifat *free* pada berbagai *platform* (kecuali pada Windows, yang bersifat *shareware*

atau anda perlu membayar setelah melakukan evaluasi yang memutuskan untuk digunakan untuk keperluan produksi).

MySQL dalam operasi klien-server melibatkan server daemon MySQL di sisi server dan berbagai macam program serta pustaka yang berjalan di sisi klient. MySQL mampu menangani data yang cukup besar. MySQL juga dapat menjalankan perintah- perintah *Structured Query Language (SQL)* untuk mengelola database-database yang ada di dalamnya.

II.9.3 Macromedia Dreamweaver

Dreamweaver dalah suatu bentuk program editor web yang dibuat oleh makromedia. Dengan program ini seorang programmer web dapat dengan mudah membuat dan mendesain webnya.

Dreamweaver merupakan editor yang komplit, yang dapat digunakan untuk membuat animasi sederhana yang berbentuk layer. Dengan program ini pembuatnya tidak akan susah-susah mengetik script-script format HTML, PHP, ASP maupun bentuk program yang lainnya.

Dengan kelebihan ini, seorang programmer dapat langsung melihat hasil buatannya tanbpa harus di browser. Dreamweaver MX juga memiliki dua bentuk layer, yaitu bentuk halaman *Design* dan halaman *Code.* Dua bentuk layer ini akan mempermudah dalam menambahkan *script* yang berbasis PHP maupun javascript. Selain mendukung pembuatan web yang berbasis HTML, juga dapat mendukung program–program web yang lain diantaranya PHP, ASP, Perl, Javascript, dan lain-lain.

II.9.4 Wamp Server

Wampserver adalah paket web server yang bekerja secara pada localhost yang dibuat secara independen dan di instal pada sistem operasi Windows.

WAMP adalah singkatan dari Windows and the principal components of the package : Apache, MySQL and PHP (or Perl or Python).

Apache adalah Web server, MySQL adalah database, PHP adalah bahasa scripting yang dapat memanipulasi informasi yang dibuat di database dan menghasilkan halaman web dinamis konten setiap waktu diminta oleh browser. Program lain juga dapat dimasukkan dalam paket, seperti phpMyAdmin yang menyediakan antarmuka pengguna grafis untuk manajer database MySQL, atau bahasa scripting Python alternatif atau Perl. Gambar wamp server dapat dilihat pada gambar II.12.



Gambar II.12 Wamp Server