BAB II
LANDASAN TEORI

2.1 Kecerdasan Buatan (Artificial Intelligent)

Kecerdasan buatan merupakan bagian dari ilmu pengetahuan komputer yang khusus ditujukan dalam perancangan otomatisasi tingkah laku cerdas dalam sistem kecerdasan komputer. Bagian utama dari kecerdasan buatan adalah basis pengetahuan (*knowledge base*), yaitu suatu pengertian atau pemahaman tentang wilayah subjek yang diperoleh melalui pembelajaran dan pengalaman (Kristanto, 2003).

Perangkat lunak dan perangkat keras merupakan slah satu bentuk untuk menirukan tindakan manusia pada kecerdasan buatan. Aktivitas manusia yang ditirukan seperti penalaran, penglihatan, pembelajaran, pemecahan masalah, pemahaman bahasa alami dan sebagainya. Teknologi kecerdasan buatan dapat dipelajari dalam berbagai bidang-bidang seperti robotika (*robotics*), penglihatan komputer (*komputer vision*), pengolahan bahasa alami (*natural language processing*), pengenalan pola (*pattern recognitiion*), sistem syaraf buatan (*artificial neural system*), pengenalan suara (*speech recognition*) dan sistem pakar (*expert system*). (Simarmata, 2006).

2.2 *Chatbot*

Salah satu program dalam kecerdasan buatan yang dirancang untuk dapat berkomunikasi langsung dengan manusia sebagai penggunanya adalah *Chatbot*. yang membedakan *chatbot* dengan sistem pemrosesan bahasa alami (*Natural language processing System*) adalah kesederhanaan *Algoritma* yang digunakan. Meskipun banyak *bots* yang dapat menginterpretasikan dan menanggapi *input* manusia, sebenarnya *bots* tersebut hanya mengartikan kata kunci dalam *input* dan membalasnya dengan kata kunci yang paling cocok, atau pola kata-kata yang paling mirip dari data yang telah ada dalam *database* yang telah dibuat sebelumnya. (Richard S, 2010)

Chat dapat diartikan sebagai pembicaraan. *Bot* merupakan sebuah program yang mengandung sejumlah data, jika diberikan masukan maka akan memberikan jawaban. *Chatbot* dapat menjawab pertanyaan dengan membaca tulisan yang diketikkan oleh pengguna melalui *keyboard*. (Adriyani, 2004).

Pada mulanya, program komputer (*bots*) ini diuji melalui *turing test*, yaitu dengan merahasiakan identitasnya sebagai mesin sehingga dapat membohongi orang yang berbicara dengannya. Jika pengguna tidak dapat mengidentifikasi *bots* sebagai suatu program komputer, maka *chatbot* tersebut dikategorikan sebagai kecerdasan buatan (*artificial intelligence*). (Richard S, 2010)

Salah satu *chatbot* yang terkenal adalah Eliza (Dr. Eliza) yang dikembangkan oleh Joseph Weizenbaum di MIT (*Massachusetts Institute of Technology*). Eliza mensimulasikan percakapan antara seorang psikiater dengan pasiennya dalam bahasa Inggris yang alami. *Chatbot* 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. (Rudiyanto N, 2005)

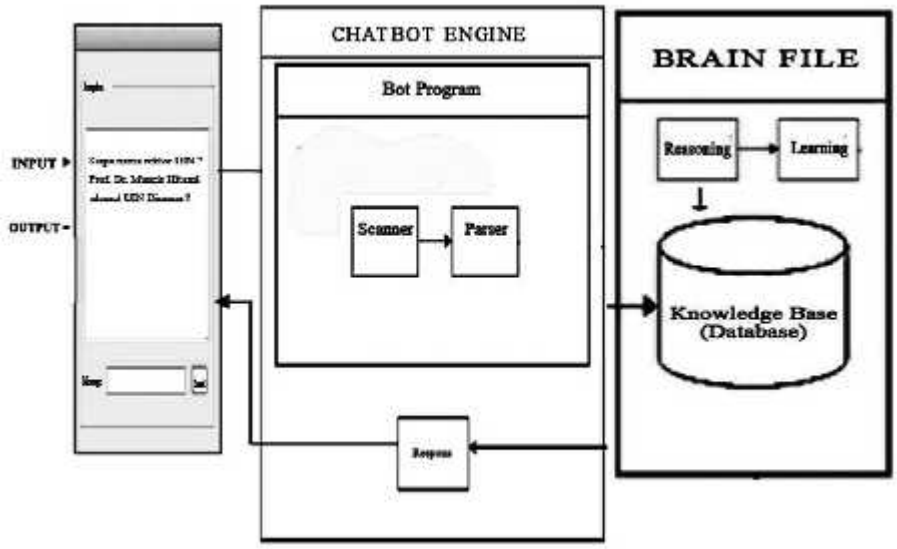
Eliza adalah pelopor *chatbot*, Eliza dikenal sebagai program *chat* yang memiliki profesi sebagai seorang psikiater. Eliza mensimulasikan percakapan antara seorang psikiater dengan pasiennya menggunakan metode biasa yang dapat mencerminkan perasaan pasien dengan mengajukan pertanyaan-pertanyaan seperti: "*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. (Rudiyanto N, 2005)

2.2.1 Perkembangan *Chatbot*

Chatbot merupakan teknologi yang sudah banyak perkembangan dalam penyempurnaan sistemnya. *Chatbot* pun sudah di implementasikan melalui jejaring sosial, seperti *twitter* dan *Windows Live Messenger*. Portal *online*

populer seperti *eBay* dan *PayPal* juga menggunakan agen *virtual multi* bahasa untuk memudahkan penggunanya. Misalnya, *PayPal* menggunakan *chatterbot* Louise untuk menangani *query* dalam bahasa Inggris dan *chatbot* Lea untuk *query* dalam bahasa Perancis. *Chatbot* tersebut Dikembangkan oleh VirtuOz.. Selain itu *Chatbot* juga di implementasikan untuk bidang komersial, pendidikan, entertainment dan sektor pelayanan publik. (Kerly, A. Hall, P. & Bull, S. 2006.)

2.2.2 Arsitektur *Chatbot*



Gambar 2.1 Arsitektur *Chatbot*

Chatbot terdiri dari dua komponen utama yakni *bot program* dan *brain file*.

2.2.2.1 Bot program

Bot Program merupakan program utama pada *chatbot* yang akan mengakses *input* dari pengguna, melakukan *parsing* dan kemudian membawanya ke *brain file (knowledge base)* untuk kemudian diberikan respon. Adapun *bot program* sendiri terdiri dari komponen *Scanner* dan *parser*. (Rudiyanto N, 2005)

1. ***Scanner***

Scanner merupakan salah satu bagian dari kompilator bahasa pada komputer yang bertugas melakukan analisis leksikal. *Scanner* menerima *input* berupa *stream* karakter kemudian memilah program sumber yang akan menjadi *input* bagi *parser*. Di dalam aplikasi *chatbot*, yang dimaksud dengan program sumber yang diolah oleh *Scanner* adalah berupa kalimat *input* dari pengguna. Contohnya: pada saat ada inputan dari user, maka otomatis dari *Scanner* akan memproses inputan, dan akan meneruskan ke tahap selanjutnya, namun pada tahap ini lebih khusus mendeteksi keberadaan inputan yang dimasukkan dan mengubah inputan menjadi bentuk normal kondisi yang diterima sistem. (Rudiyanto N, 2005)

2. ***Parser***

Parser atau *syntactic analyzer* pada kompilator bahasa pemrograman berfungsi untuk memeriksa kebenaran kemunculan inputan yang kemudian akan dijadikan *token*. Pada *Chatbot system*, fungsi dari *parser* ini agak berbeda karena *token* yang akan diolah, semuanya memiliki tipe yang sama yaitu berupa kata (*word*). Urutan *token* 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.* Contoh penerapan: inputan yang masuk pada *chatbot* akan dipecah menjadi token-token sehingga pada tahapan seanjutnya akan dilakukan pencocokan dengan kosakata kata kunci yang sudah ada pada *brain file.* (Rudiyanto N, 2005)

2.2.2.2 Brain file

Brain File merupakan otak dari *chat bot* itu sendiri yang menentukan bagaimana cara *chatbot* berpikir dan akan memberikan respon. *Brain file* berfungsi sebagaimana tabel informasi (*knowledge base*). Di dalam *brain file* inilah disimpan semua kosakata, kepribadian, dan pengetahuan (*knowledge*) dari

chatbot. Semakin banyak pengetahuan yang dimiliki *chat bot* maka akan semakin besar ukuran file dari *brain file* tersebut. Secara lebih rinci komponen *chatbot* itu berupa:

1. Reasoning

Reasoning adalah teknik penyelesaian masalah dengan cara mempresentasikan masalah ke dalam basis pengetahuan (*knowledge base*) menggunakan *logic* atau bahasa formal. Pada penerapan *chatbot* proses ini akan dikerahkan bila kata kunci terdapat dalam *knowledge base chatbot* dan dilakukan untuk mengembalikan respon ke pengguna. Proses ini menunjukkan bahwa masukan yang diberikan oleh pengguna tidak diproses sebagai satuan kata, tetapi sebagai kalimat utuh. Contoh penerapan dalam *chatbot* adalah pada pencocokan antara kata kunci pada *knowledge base* dengan inputan dari user. (Herdi, 2013)

Proses *reasoning* dapat digambarkan menggunakan algoritma sebagai berikut :

```
Function
Patternmatchingbruteforce (keyword As array)
Algoritma
     If keyword ← true then
          Return reasoning //cari jawaban yang sesuai
                           dengan kata kunci

               Keyword ← KeywordList //pencocokan kata
                                     Kunci dan jawaban
               return Respon

               return inputkalimat

     Else  keyword ← false
     Return learning
     End
```

Gambar 2.2 *Algoritma reasoning*

2. Learning

Learning merupakan pendefinisian aturan tertentu secara otomatis dalam menemukan aturan yang diharapkan bisa berlaku umum untuk data-data yang belum pernah kita ketahui. Pada penerapan *chatbot* proses *learning* dijalankan bila kata kunci pada masukan pengguna tidak terdapat dalam *knowledge base*. Kata kunci yang tidak ditemukan tersebut akan disimpan sebagai *dialog repository* untuk kemudian akan ditanyakan pada *bot program*. Contoh penerapannya pada *chatbot* adalah sewaktu *chatbot* tidak mendapati kesamaan antara kata kunci dalam knowledge base dengan inputan user, maka *chatbot* akan menyimpan kosakata pertanyaan tersebut untuk dipelajari dikemudian hari. (Kartika H, 2007).

Proses *learning* dapat digambarkan menggunakan algoritma sebagai berikut :

```
Function
Patternmatching Bruteforce (keyword As array)
Algoritma
     If keyword ← true then
          Return reasoning //cari jawaban yang sesuai
     Else keyword ← false then
                    Return learning
                    Keyword ← ResponList
// respon learning jika tidak sesuai dengan kata kunci
                    return Respon

                    return inputkalimat
     End
```

Gambar 2.3 *Algoritma Learning*

2.2.3 Prinsip Kerja *Chatbot*

Bot program atau bagian aplikasi menentukan kemampuan dan keterampilan *chat bot* untuk berbicara pada anda atau pada pengguna lainnya, atau dengan kata lain *bot program* berperan sebagai mulut. *Chatbot* tidak tahu apa-apa,

tidak punya nama dan tidak punya kepribadian. Untuk itu pengguna harus mengajarinya berbagai hal sehingga ia dapat berbicara dengan baik dan semua pelajaran tersebut dimasukkan ke *brain file* (Kartika H, Astari & Novita. 2007).

Sebagai contoh, pengguna mengeluh bahwa *chatbot* yang akan dirancang berbicara omong kosong dengan mengetik kalimat: "*You are talking nonsense, pinhead*!". Agar *chat bot* dapat memberikan respon terhadap kalimat tersebut, maka dapat ditambahkan baris berikut pada *brain file* dari *chat bot*:

You are talking nonsense ← trigger line

I am smarter than you ← respon bot

Dengan demikian jika pengguna sekarang *chatting* menggunakan kalimat yang disebutkan di atas maka *bot program* akan masuk ke dalam *brain file* dan kemudian memberikan respon dengan output yang sesuai dengan *trigger line* yaitu *you are talking nonsense* (Rudiyanto, 2005)..

Respon yang sama juga dapat muncul jika anda juga menuliskan baris berikut pada *brain file*:

Nonsense ← trigger line

I am smarter than you ← respon bot

Respon yang sama juga akan muncul jika anda menuliskan:

a. *Talking nonsense ← trigger line*

   I am smarter than you ← respon bot

b. *Pinhead ← trigger line*

   I am smarter than you ← respon bot

Namun akan lebih bagus jika *chatbot* diberikan kemampuan untuk merespon kalimat yang beragam misalnya:

c. *You are talking nonsense ← trigger line*

   I am smarter than you ← respon bot

d. *Talking Nonsense ← trigger line*

   Talking is funny, isnt it? ← respon bot

e. *Nonsense ← trigger line*

   Don't be stupid! ← respon bot

f. *Pinhead* ← *trigger line*

Please try not to be insulting,buddy ← *respon bot*

Dengan *brain file* seperti itu maka *chatbot* akan terasa lebih hidup karena dapat memberikan respon dengan berbagai kalimat yang berbeda.

2.3 Metode Case Base Reasoning

Case Base Reasoning (CBR) adalah metode yang biasanya diterapkan pada pembuatan sebuah penalaran system, yang pada umumnya digunakan pada system pakar. Namun tidak jarang ini juga dihjadikan sebagai metode alur berpikir dari sebuah proses sistem mesin inferensi *chatbot*. Secara garis besar, konsep dari *case based reasoning* adalah menyelesaikan suatu masalah berdasarkan pengalaman memecahkan masalah/kasus yang mirip di masa lalu (Sri Mulyana, 2009).

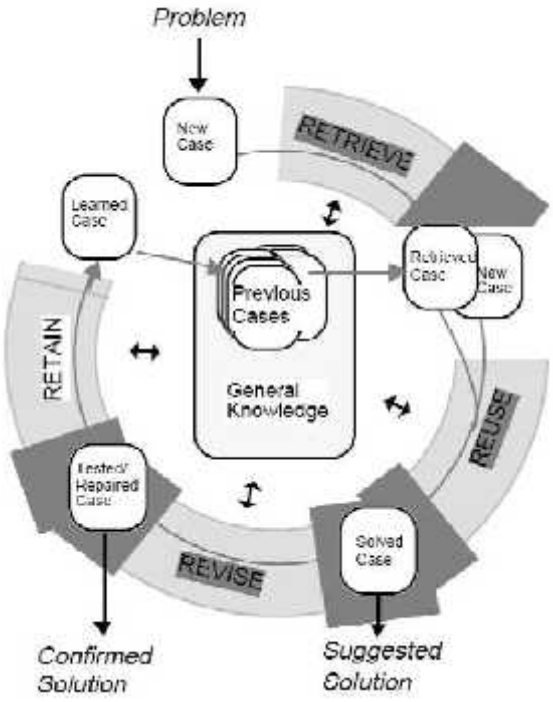
Case Base Reasoning dapat memiliki makna yang berbeda, tergantung tujuan dari penalaran berupa penyesuaian dan penggabungan solusi sebelumnya untuk menyelesaikan sebuah masalah baru, menjelaskan terhadap solusi berdasarkan kasus sebelumnya, menemukan alasan dari kondisi sebelumnya untuk memahami situasi baru atau membangun sebuah solusi yang disepakati berdasarkan kasus sebelumnya. Dari beberapa aspek yang berbeda tersebut dapat dikelompokkan ke dalam dua tipe utama : *case base reasoning* interpretatif dan *case base reasoning* untuk penyelesaian masalah

Pada *case base reasoning* interpretatif, aspek yang penting adalah argumentasi apakah suatu situasi baru seharusnya diperlakukan seperti sebelumnya berdasarkan persamaan dan perbedaannya, sedangkan *case base reasoning* untuk penyelesaian masalah, bertujuan mendapatkan penyelesaian masalah baru dengan melakukan adaptasi terhadap penyelesaian pada kasus-kasus sebelumnya. Pembagian ini tidak dimaksudkan untuk memisahkan antar keduanya, karena pada kenyataannya banyak masalah yang memiliki kedua tipe tersebut, bahkan kebanyakan aplikasi pembelajaran berbasis kasus menggunakan kedua metode tersebut.

Secara singkat, tahap-tahap penyelesaian masalah berbasis *case base reasoning* adalah sebagai berikut : (Mantaras dkk,2006)

1. Pengambilan kembali kasus-kasus yang sesuai dari memori (hal ini membutuhkan pemberian indeks terhadap kasus-kasus dengan menyesuaikan fitur-fiturnya).
2. Pemilihan sekelompok kasus-kasus yang terbaik.
3. Memilih atau menentukan penyelesaian.
4. Evaluasi terhadap penyelesaian (hal ini dimaksudkan untuk meyakinkan agar tidak mengulang penyelesaian yang salah)
5. Penyimpanan penyelesaian kasus terbaru dalam penyimpan kasus/memori.

Sesuai dengan tahap-tahap tersebut, Aamodt dan Plaza (Aamodt dan Plaza, 1994) menjelaskan sebuah *case base reasoning* sebagai sebuah siklus yang disingkat 4 R yaitu, *Retrieve, Reuse, Revise dan Retain* seperti pada gambar 2.2 berikut ini :



Gambar 2.4 Siklus *Case Base Reasoning* (Aamodt dan Plaza, 1994)

Penerapan *case based reasoning* pada *chatbot* di implementasikan pada alur *bot program* dan *brain file*, untuk penjelasannya sebagai berikut :

1. Siklus *retrieve* dilakukan pencarian kesamaan (*similarity*) antara kasus yang sedang ditangani (*new case*) berupa kosakata yang di *input*kan dalam bentuk pertanyaan dengan kosakata yang tersimpan didalam basis pengetahuan. Kosakata tersimpan yang sama akan dijadikan ditampilkan ke sistem patokan sebagai *retrieved case.*

Penerapan siklus *retrieve* ini pada *chatbot* adalah pada tahapan *reasoning*, dimana pada tahapan ini inputan yang dimasukkan ke dalam daftar knowledge base akan dilakukan pencocokan antara pertanyaan dan jawaban pada knowledge base akan dimunculkan ke tampilan bot program .

2. Siklus *reuse*, kosakata yang pernah diterapkan pada *retrieved case* dijadikan sebagai prediksi kosakata yang terpilih berupa solusi kosakata yang diusulkan (*suggested solution*) bagi kasus yang sedang ditangani. Kosakata yang terpilih tersebut dicobakan sebagai solusi untuk selanjutnya di konfirmasi untuk ketepatan calon jawaban dari pertanyaan yang di *input*kan untuk menjadi solusi (jawaban) yang tepat bagi *new case*. *Siklus reuse* pada aplikasi *chatbot* lebih mengarah pada proses *reasoning,* yaitu pengambilan jawaban dari daftar *knowledge base* yang ada dan sudah terkoreksi dengan benar untuk ditampilkan kembali pada tampilan bot program

3. Siklus *revise,* merupakan tahapan konfirmasi revisi tentang kosakata yang di usulkan akan dijadikan patokan untuk di pelajari berupa pencocokan pola dari kosakata yang terpilih dengan jawaban yang sesuai. Pada siklus ini akan dilakukan pembelajaran (*learning*) agar pada sesi berikutnya sistem *chatbot* dapat memberikan jawaban yang tepat bagi pertanyaan yang sama. Jika konfirmasi jawaban yang diterima adalah tepat, maka calon jawaban tersebut akan diresmikan menjadi jawaban yang tepat bagi pertanyaan yang sedang ditangani (*new case*) setelah dipastikan mendapatkan solusi yang tepat,

Siklus ini merupakan pencerminan dari tahapan *learning* pada aplikasi *chatbot*, karena pada tahap ini setiap inputan yang tidak diketemukan kecocokan dengan daftar kata kunci pada knowledge base maka sistem akan member usulan

kata kunci untuk diinputkan jawabannya ke dalam daftar jawaban pada knowledge base.

4. Siklus *retain* selanjutnya pertanyaan tersebut disimpan untuk referensi bagi pertanyaan-pertanyaan yang mirip yang ditemukan pada sesi selanjutnya.

2.4 Konsep *Algoritma*

2.4.1 Definisi *Algoritma*

Menurut, Munir (2001:4) *Algoritma* adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis, sedangkan menurut Kamus Besar Bahasa Indonesia (1976:30) *Algoritma* adalah urutan logis pengambilan putusan untuk pemecahan masalah. Adapun definisi *Algoritma* meliputi (Suarga, 2004 : 1) :

1. Teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata terbatas, tetapi tersusun secara logis dan sistematis.

2. Suatu prosedur yang jelas untuk menyelesaikan suatu persoalan dengan menggunakan langkah-langkah tertentu dan terbatas jumlahnya

Kata *Algoritma* sendiri diadaptasi dari nama ilmuwan muslim Abu Ja'far Muhammad ibn Musa Al-Khawarizmi (780-847 M) yang banyak menghasilkan karya dalam bidang matematika, disamping karya-karyanya dalam bidang lainnya seperti geografi dan musik (Wahid, 2004 : 1)

2.4.2 Ciri – ciri dan Sifat *Algoritma*

Donald E. Knuth (1973), menyatakan bahwa ada beberapa ciri-ciri *Algoritma*, yaitu :

a. *Algoritma* mempunyai awal dan akhir. Suatu *Algoritma* harus berhenti setelah mengerjakan serangkaian tugas atau dengan kata lain suatu *Algoritma* harus memiliki langkah yang terbatas.
b. Setiap langkah harus didefinisikan dengan tepat sehingga tidak memiliki arti ganda.

c. Memiliki masukan atau kondisi awal.

d. Memiliki keluaran atau kondisi akhir.

e. *Algoritma* harus efektif; bila dikuti dengan benar-benar akan menyelesaikan persoalan.

Berdasarkan ciri *Algoritma* dapat disimpulkan sifat utama suatu *Algoritma*, yaitu (Suarga, 2004 : 2): 11

a. *Input* : suatu *Algoritma* memiliki input atau kondisi awal sebelum *Algoritma* dilaksanakan dan bisa berupa nilai-nilai pengubah yang diambil dari himpunan khusus.

b. *Output* : suatu *Algoritma* akan menghasilkan output setelah dilaksanakan, atau *Algoritma* akan mengubah kondisi awal menjadi kondisi akhir, dimana nilai output diperoleh dari nilai input yang telah diproses melalui *Algoritma*.

c. *Definiteness* : langkah-langkah yang dituliskan dalam *Algoritma* terdefinisi dengan jelas sehingga mudah dilaksanakan oleh pengguna *Algoritma*.

d. *Finiteness* : suatu *Algoritma* harus memberi kondisi akhir atau output setelah melakukan sejumlah langkah yang terbatas jumlahnya untuk setiap kondisi awal atau input yang diberikan.

e. *Effectiveness* : setiap langkah dalam *Algoritma* bisa dilaksanakan dalam suatu selang waktu tertentu sehingga pada akhirnya member solusi sesuai yang diharapkan.

f. *Generality* : langkah-langkah *Algoritma* berlaku untuk setiap himpunan input yang sesuai dengan persoalan yang akan diberikan, tidak hanya untuk himpunan tertentu.

Algoritma sebagai langkah-langkah pemecahan masalah dapat dituliskan dengan berbagai cara , yaitu (Wahid, 2004 : 9) :

1. Uraian deskriptif yaitu Penulisan *Algoritma* dengan uraian deskriptif menggunakan bahasa yang biasa digunakan sehari-hari.

2. *Algoritma* pseudecode adalah *Algoritma* yang dituliskan dalam kode-kode yang disepakati dan mempunyai urutan-urutan tertentu. Kode-kode ini dapat dikembangkan sendiri asalkan arti dari setiap kode disepakati bersama.

3. Bagan alir (*Flowchart*)

Flowchart (bagan alir dokumen) adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program (Jogiyanto Hartono : 1989).

2.5 Algoritma Pencocokan String (*String Matching)*

2.5.1 Definisi *String Matching*

Menurut Black (dalam Syaroni dan Munir, 2004:1) string adalah susunan dari karakter-karakter (angka, alphabet, atau karakter yang lain) dan biasanya direpresentasikan sebagai struktu data array. String dapat berupa kata, frase, atau kalimat. Sedangkan string matching menurut Black (dalam syaroni dan munir, 2004:1) diartikan sebagai sebuah permasalahan untuk menemukan pola susunan karakter string didalam string lain atau bagian dari isi teks. String matching dalam bahasa Indonesia dikenal dengan istilah pencocokan string (Munir dalam Hadiati, 2007:1).

Pencarian string yang juga bisa disebut pencocokan string (String Matching) merupakan *Algoritma* untuk melakukan pencarian semua kemunculan string pendek *pattern* [0….n-1] yang disebut *pattern* di string yang lebih panjang teks [0…m-1] yang disebut teks (Charras, 1997 : 11). 16

2.5.2 Kerangka Kerja String Matching

Persoalan pencarian string dirumuskan sebagai berikut (Munir, 2004 : 1) Diberikan :

1. Sebuah teks (*text*), yaitu sebuah (*long*) string yang panjangnya n karakter.
2. *Pattern*, yaitu sebuah string dengan panjang m.

Dengan sebuah nilai karakter (m < n) yang akan dicari dalam teks. Dalam *Algoritma* pencocokan string, teks diasumsikan berada di dalam memori, sehingga bila kita mencari string di dalam sebuah arsip, maka semua isi arsip perlu dibaca terlebih dahulu kemudian disimpan di dalam memori. Jika *pattern* muncul lebih dari sekali di dalam teks, maka pencarian hanya akan memberikan keluaran berupa lokasi *pattern* ditemukan pertama kali.

2.5.3 Kerangka Pikir String *Matching*

Algoritma string matching dapat diklasifikasikan menjadi 3 bagian menurut arah pencariannya, yakni (Charras, 1997 : 12).

1. From left to right

Dari arah yang paling alami, dari kiri ke kanan, yang merupakan arah untuk membaca. *Algoritma* yang termasuk kategori ini adalah *Algoritma* brute force, *Algoritma* knuth moris Pratt.

2. From right to left

Dari arah kanan ke kiri, arah yang bisanya menghasilkan hasil terbaik secara partikal. *Algoritma* yang termasuk kategori ini adalah *Algoritma* boyer-moore.

3. In a specific order

Dari arah yang ditentukan secara spesifik oleh *Algoritma* tersebut, arah ini menghasilkan hasil terbaik secara teoritis. *Algoritma* yang termasuk kategori ini adalah *Algoritma colossi* dan *Algoritma crochemore-perrin*.

Beberapa konsep *string matching* antara lain:

1. *Approximate string matching*, yaitu sebuah pencarian terhadap pola-pola string (mengandung beberapa proses yaitu mengitung jumlah karakter yang berbeda, penyisipan dan penghapusan karakter) sehingga mendekati pola atau *pattern* dari string yang dicari. Dari wikipedia didefinisikan sebagai sebuah teknik untuk mencari sebuah pola yang mendekati string dari sebuah kumpulan teks.
2. *Algoritma* pencarian string adalah sebuah proses pencarian tempat dari suatu atau beberapa string yang ditemukan dalam sebuah kumpulan string atau teks. Jalan paling sederhana adalah dengan cara membaca karakter satu persatu dan melakukan perhitungan kesalahan posisi yang ada dari string yang dicari.

2.5.4 Macam *Algoritma String Matching*

Secara garis besar string matching dibedakan menjadi dua (Binstock & Rex dalam Syaroni & Munir, 2004 : 2), yaitu:

1. *Exact string matching*
2. *Inexact string matching* atau *Fuzzy string matching*.

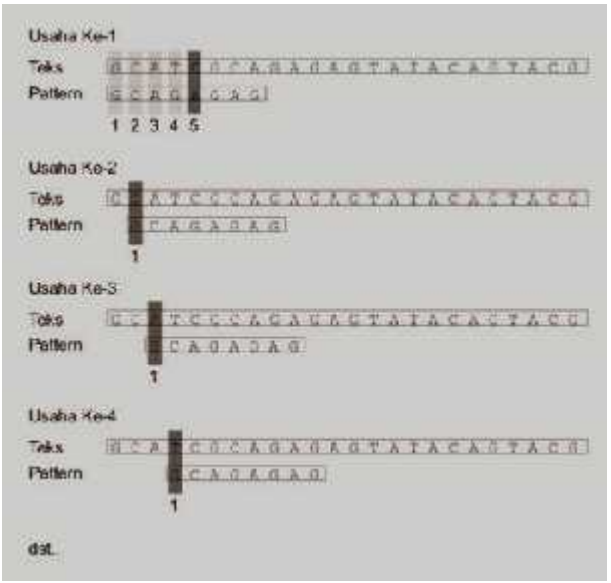
2.6 *Algoritma Brute Force*

Algoritma brute force merupakan *Algoritma* pencocokan string yang ditulis tanpa memikirkan peningkatan performa. *Algoritma* ini sangat jarang dipakai dalam praktik, namun berguna dalam studi pembanding dan studi-studi lainnya.

Secara sistematis, langkah-langkah yang dilakukan *Algoritma bruteforce* pada saat mencocokkan string adalah:

1. *Algoritma* brute force mulai mencocokkan *pattern* pada awal teks.
2. Dari kiri ke kanan, *Algoritma* ini akan mencocokkan karakter per karakter *pattern* dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
3. Karakter di *pattern* dan di teks yang dibandingkan tidak cocok (mismatch).
4. Semua karakter di *pattern* cocok. Kemudian *Algoritma* akan memberitahukan penemuan di posisi ini.
5. *Algoritma* kemudian terus menggeser *pattern* sebesar satu ke kanan
6. mengulangi langkah ke-2 sampai *pattern* berada di ujung teks.

Berikut adalah *Algoritma* brute force yang sedang bekerja mencari string:



Gambar 2.5 *Algoritma bruteforce*

Algoritma brute force string match adalah *Algoritma* yang paling sederhana untuk memecahkan masalah string match. Cara kerja *Algoritma* ini adalah dengan mencoba setiap posisi *pattern* (kata yang akan dicocokkan) terhadap teks, kemudian dilakukan proses pencocokan setiap karakter dan teks pada posisi tersebut.

Berikut adalah *psoudocode Algoritma bruth force string match*:

- Dari *psoudocode* tersebut terlihat bahwa *Algoritma* ini membutuhkan dua macam inputan, yakni input array karater teks dan input array *pattern*. Dalam perjalanan eksekusinya terdapat dua kali perulangan. Pada perulangan yang terdalam(while) terdapat suatu kondisi yang membandingkan suatu isi dari array *pattern* dengan array teks.

Kode program di atas adalah suatu program sederhana yang berusaha memeriksa apakah data array x terdapat pada array y sehingga kedua array ini memiliki kecocokan. Pengecekan kecocokan dengan menggunakan *Algoritma* brute force yang direpresentasikan oleh *method match*.

Program akan memberikan output berupa pesan “DATA COCOK” apabila memang terdapat kecocokan antara kedua data. Sebaliknya, program akan memberikan output pesan “DATA TIDAK COCOK” apabila setelah melalui prosedur pengecekan string tidak ditemukan kesamaan antara kedua data tersebut.

2.6.1 Kelebihan dan Kelemahan *Algoritma Brute Force.*

Menurut (Binstock & Rex dalam Syaroni & Munir, 2004 : 2), Kelemahan dan kelebihan dari *bruteforce* adalah :

a. Kekuatan:

1) Metode *brute force* dapat digunakan untuk memecahkan hampir sebagian besar masalah (*wide applicability*).

2) Metode *brute force* sederhana dan mudah dimengerti.

3) Metode *brute force* menghasilkan *Algoritma* yang layak untuk beberapa masalah penting seperti pencarian, pengurutan, pencocokan *string*, perkalian matriks.

4) Metode *brute force* menghasilkan *Algoritma* baku (standard) untuk tugas-tugas komputasi seperti penjumlahan/perkalian *n* buah bilangan, menentukan elemen minimum atau maksimum di dalam tabel (*list*).

b. Kelemahan:

1) Metode *brute force* jarang menghasilkan *Algoritma* yang mangkus.

2) Beberapa *Algoritma brute force* lambat sehingga tidak dapat diterima.

3) Tidak sekontruktif/sekreatif teknik pemecahan masalah lainnya.

Berikut ini adalah pseudecode dari *Algoritma* brute force :

```
procedure PencocokanString(input P:string, T: string,
n,m:integer,  output  idx  : integer)
{Masukan pattern P yang panjangnya m dan teks T yang
panjangnya  n.  Teks  T  diprepresentasikan  sebagai
string  (array  of  character)
Keluaran:  lokasi  awal  kecocokan  (idx)

}
Deklarasi
i: integer
ketemu:boolean
Algoritma:
i ←0
ketemu  ←false
while  (i<n-m)  and  (not  ketemu)  do
j ←1
while  (j<m)  and  (Pj  =  Ti+j)  do
j←j+1
endwhile
{j>m  or  Pj  <>  Ti+j}
if j  =  m  then  {kecocokan  string  ditemukan}
   ketemu<-true
else
   i<-  i+1 {geser pattern satu karakter ke
kanan teks}
endif
endfor
{i  >  n-m  or  ketemu}
if  ketemu  then
   idx<-i+1
else
   idx<-  -1
endif
```

Gambar 2.6 Pseudocode *Algoritma Bruteforce*

Algoritma Bruteforce penerapannya dalam aplikasi *chatbot* dengan metode case base reasoning adalah pada *siklus retrieve* pencocokan pola antara pertanyaan diajukan dengan kata kunci pada knowledge base untuk kemudian setelah diketemukan kecocokan maka akan dikeluarkan jawaban dari pertanyaan yang muncul.

2.7 Model Pengembangan Sistem

Siklus hidup pengembangan sistem (*System Development Life Cycle*) adalah proses evolusioner yang diikuti dalam menerapkan sistem atau sub sistem informasi berbasis komputer. Tahapan siklus hidup pengembangan sistem adalah kebijakan dan perencanaan, analisis sistem, perancangan, seleksi, implementasi, dan pemeliharaan.

Model ini bersifat linier karena prosesnya mengalir secara sekuensial mulai dari awal hingga akhir. Model ini mensyaratkan penyelesaian suatu tahap secara tuntas sebelum beranjak pada tahap selanjutnya. Hasil-hasilnya harus didokumentasikan dengan baik. Secara umum kerangka kerja model system ini berupa *Waterfall* (Pressman, 1997)

2.8 Teknik *Object Oriented*

Pengembangan sistem pada teknik pada teknik *object oriented* terjadi tiga aktifitas utama yaitu model spesifik dan model logika (analisa), model arsitektur (perancangan), dan implementasi (pengujian dan pengkodean). Dalam merancang sebuah sistem yang berorientasi objek, harus dianalisis terlebih dahulu hubungan sistem dengan lingkungan, mengidentifikasikan objek dan nama metode serta atribut, menetapkan hubungan antar objek, menetapkan antar muka dari beberapa objek, menerapkan dan menguji objek, dan memasang serta menguji sistem.

Salah satu bahasa pemrograman yang menjadi standar untuk merancang dan mendokumentasikan sistem perangkat lunak adalah UML (*Unified Modelling Language*). Menurut (Dharwiyanti dan Wahono, 2006) beberapa diagram yang dapat digunakan untuk memperjelas penggunaan UML dalam pemrograman berorientasi objek diantaranya :

1. *Use case diagram* menggambarkan sebuah fungsi yang dibutuhkan oleh sebuah sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, membuat sebuah daftar aktivitas, dan sebagainya.
2. *Sequence diagram* menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*.
3. *Class diagram* menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.

2.9 Program Berbasis *Web*

Pemrograman berbasis *web*, faktor yang menentukan kinerja aplikasi adalah kecepatan akses *database* dan kecepatan akses jaringan. untuk menjalankan sistem berbasi *web* dibutuhkan engine tertentu yakni *web server*.

2.9.1 *Apache*

Apache merupakan *web server* yang paling sering digunakan sebagai *server* internet dibandingkan *web server* lainnya. *Web server* merupakan *server* internet yang mampu melayani koneksi transfer di dalam protokol *HTTP* (*Hypertext Transfer Protocol*) saat ini *web server* merupakan inti dari *server-server* internet selain *e-mail server*, *FTP server*, dan *news server*.

2.9.2 *PHP*

PHP merupakan singkatan dari *PHP : Hypertext Preprocessor*, PHP adalah bahasa scripting server-side, artinya di jalankan di server, kemudian outputnya dikirim ke client (browser), PHP digunakan untuk membuat aplikasi *web*, PHP mendukung banyak database (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, dll.). Pada tugas akhir kali ini penulis akan menggunakan PHP 5 dalam pengerjaan systemnya.

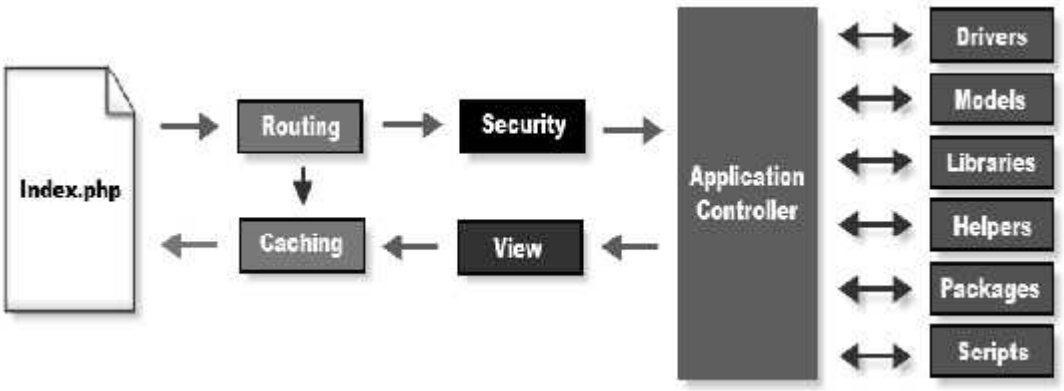
2.9.3 *Framework*

Framework adalah sekumpulan *class* dan *library* yang terpadu sehingga memudahkan menyelesaikan permasalahan seraca menyeluruh, efeknya waktu untuk membuat program menjadi lebih singkat, beberapa contoh *framework* di *PHP* adalah *CakePHP, Zend Framework, CodeIgniter* dan *Symfony.*

2.9.3.1 *Codeigniter* (CI)

Codeigniter adalah salah satu jenis dari *PHP framework* yang sedang berkembang sekarang ini. Secara umum, *framework CodeIgniter* menggunakan struktur *MVC* (*Model, View, Controller*). *MVC* merupakan suatu metode untuk memisahkan pengedali logika dan pengendali tampilan.

Gambar ilustrasi di bawah ini menunjukkan alur logika saat akan mengakses sebuah website yang dibangun menggunakan *codeigniter*.



Gambar 2.7 Alur kerja *codeigniter* (Awan Pribadi Basuki, 2010)

1. *Index.php*

   Pada saat akan mengakses sebuah alamat domain, browser akan mencari salah satu skrip: index.html, index.php, index.phtml. Index.php ini berperan sebagai *controller* awal, yang akan menginisialisasi sumber daya yang dibutuhkan untuk menjalankan *codeIgniter*.

2. *Router*

   Berperan layaknya penerjemah, *router* akan mengarahkan ke mana *skrip* selanjutnya di eksekusi. Hal yang pertama yang akan dilakukan route

adalah menganalisa *HTTP request*, setelah itu baru memutuskan apa yang selanjutnya akan dikerjakan.

3. *Cache*

   Cache adalah metode untuk menyimpan data-data yang sudah pernah di akses sebelumnya. Jadi apabila akan membuat panggilan baru, jika terdapat cache untuk permintaan yang sama, maka *cache* inilah yang akan digunakan terlebih dahulu. Hal ini akan mem *bypass* eksekusi normal yang biasanya dilakukan saat tidak ada *cache*.

4. *Security*

   Sebelum *controller* di panggil, semua data, baik permintaan HTTP sampai data yang dikirimkan oleh user akan di saring terlebih dahulu. Hal ini adalah security level pertama dalam sebuah website. Hal ini juga menjamin semua data yang akan disimpan ke dalam database sudah di escape terlebih dahulu.

5. *Controller*

   Layaknya seorang pengatur lalu lintas, *controller* akan memanggil model, *library* utama, *helper*, dan elemen lain yang dibutuhkan untuk sebuah *request* yang terjadi. *Request* yang dimaksud adalah URL sebuah *website*. Dalam *framework codeIgniter*, nama kelas dan fungsi di dalam *controller* inilah yang akan mengubahnya menjadi sebuah URL.

6. *View*

   Inilah akhir dari segala proses yang terjadi pada *framework CodeIgniter*. I akan selalu berinteraksi dengan *view*

Membandingkan *PHP Framework* saat ini menjadi kebutuhan dalam membangun sebuah aplikasi berbasis PHP. Kelebihan dari *codeigniter* adalah sebagai berikut :

1. Mudah digunakan dan tidak memerlukan konfigurasi yang rumit
2. Mendukung PHP4 dan PHP5
3. Merupakan *Framework* MVC paling populer dan paling banyak digunakan
4. Dokumentasinya lebih lengkap dan mudah dipahami.

5. Dokumentasi yang sangat bagus, *friendly* dan didukung oleh forum, wiki, dan komunitas yang besar (Awan Pribadi Basuki, 2010).

2.9.4 *MySQL*

Bahasa *PHP* mempunyai kelebihan yaitu kompabilitasnya dengan berbagai macam jenis *database*, dukungan dengan berbagai macam jenis sistem operasi. *PHP* lebih cocok dan umum digunakan jika di gabungkan dengan *database MySQL*.

Bahasa SQL pada umumnya informasi tersimpan dalam tabel-tabel yang secara logika merupakan struktur dua dimensi terdiri dari baris (*row* atau *record*) dan kolom (*column* atau *field*). Sedangkan dalam sebuah *database* dapat terdiri dari beberapa *tabel*.