INTERNET RELAY CHAT BO T DENGAN MENGGUNAKAN ARTIFICIAL INTELLIGENT MARKUP LANGUANGE (AIML)

Wahyu Catur Pamungkas[1]

Abstrak :

Internet Relay Chat (IRC) merupakan media komunikasi yang banyak diminati oleh kalangan pengguna internet. *Chatterbot* pada *channel IRC* yang ada sekarang dapat dibedakan dengan pengguna asli, karena sering melakukan jawaban yang kaku dan berulang-ulang. Sifat *chatterbot* inilah yang memacu para *programmer* yang menyenangi dunia *IRC* untuk mengembangkan sebuah *chatterbot* yang lebih pandai dan efektif.

Pengembangan *chatterbot* mengimplementasikan *Artificial Intelligent Markup Language* (AIML), suatu bahasa XML yang dikhususkan untuk membangun sebuah *chatterbot* cerdas, yang bersifat *cross-platform* dengan algoritma pencarian khusus yaitu algoritma *graphmaster pattern matching*.

Chatterbot yang dikembangkan berkomunikasi lebih baik dan mampu memberikan informasi mengenai sinopsis film dan jadwal bioskop di setiap kota besar di Indonesia kepada pengguna yang menanyakannya. Karya tulis "*Internet Relay Chat Bot* dengan Menggunakan *Artificial Intelligent Markup Language*" dapat dijadikan sebagai acuan untuk membuat *chatterbot* yang berjalan pada jaringan komunikasi lainnya, seperti **YahooMessenger!**.

Kata kunci : *chatterbot*, *chatbot*, *irc bot*, *bot*, *AIML*, *graphmaster pattern matching*, XML.

1. Pendahuluan

1.1. Latar Belakang Masalah

Aktivitas kerja sekarang semakin membutuhkan media komunikasi yang cepat. Hal ini didukung dengan hadirnya teknologi internet yang semakin berkembang pesat dengan koneksi yang semakin cepat dan murah. Keadaan seperti inilah yang melahirkan fasilitas komunikasi secara *real-time* seperti *Internet Relay Chat (IRC)* dan *Instant messenger (IM)*.

Internet Relay Chat merupakan salah satu layanan internet yang paling banyak diminati oleh kalangan pengguna internet. Pada *IRC* komunitas terbagi berdasarkan forum diskusi yg disebut *channel*, kepopuleran *channel* sangat tergantung dari fasilitas yang ditawarankan. Umumnya, setiap *channel* memiliki satu atau beberapa *chatterbot* untuk meramaikan *channel*, dan untuk menjawab pertanyaan dari pengguna lain agar seolah-olah pengguna itu mendapat respon dari teman berkomunikasi.

Chatterbot yang ada sekarang dapat dibedakan antara *chatterbot* dengan pengguna asli, karena *chatterbot* sering melakukan jawaban yang kaku dan berulang-ulang. Sifat *chatterbot* inilah memacu para *programmer* yang menyenangi dunia *IRC* untuk menciptakan sebuah *chatterbot* yang lebih pandai dan efektif.

Dalam tugas akhir ini, penulis mengembangkan sebuah *chatterbot* yang mampu mengatasi masalah di atas. Penulis menggunakan teknologi yang sudah ada yaitu AIML, suatu bahasa XML yang dikhususkan untuk membangun sebuah *chatterbot* cerdas, yang bersifat *cross-platform* dengan algoritma yang juga dikhususkan untuk pencarian jawaban, yaitu algoritma *graphmaster pattern matching*.

1.2. Batasan Masalah

Pada penelitian yang dilakukan terdapat beberapa batasan masalah, yaitu:

Program hanya mampu melakukan percakapan di IRC

Basis pengetahuan program adalah seputar film, *chatterbot* sebagai pecinta film.

Pembahasan program hanya meliputi program *IRC bot*, implementasi dan analisis algoritma *Graphmaster Pattern Matching*.

---

[1] *mahasiswa Fakultas Teknik Program Studi Teknik Informatik a Universitas Kristen Duta Wacana, Yogyakarta. Email : czephie@yahoo.com*

Pembahasan mengenai database hanya berupa gambaran umum dan analisis. Fungsi-fungsi yang digunakan dalam program tidak dibahas secara mendetail kecuali dalam implementasi algoritma *Graphmaster Pattern Matching.*

1.3. Metodologi Penelitian

Penulisan tugas akhir ini menggunakan beberapa metode sebagai berikut:

Studi pustaka, berfungsi sebagai acuan pembelajaran untuk mendalami pengetahuan tentang AIML dan algoritma yang digunakannya yaitu *GraphMaster Pattern Matching.*

Mengimplementasikan dalam program serta melakukan uji coba program.

2. Landasan Teori

2.1 *Artificial Intelligent Markup Languange* (AIML)

AIML adalah bahasa *scripti*ng interpreter yang merupakan turunan dari Extensible Markup Language (XML) dengan fungsi yang lebih spesifik. Salah satu fungsinya adalah membuat sistem stimulus-response berbasis pengetahuan. Dokumen AIML terdiri dari objekobjek yang dipisahkan oleh *tag-tag* tertentu seperti layaknya dokumen XML atau HTML.

Element paling penting dari AIML diantaranya adalah sebagai berikut :

Category

Pada AIML, *category* merupakan unit dasar dari pengetahuan. *Category* minimal terdiri dari dua e*lement* AIML yaitu *pattern* dan *template*. Berikut adalah contoh *category* yang sederhana :

```
<category>
   <pattern>SIAPA NAMA KAMU</pattern>
   <template>Nama saya wahyu.</template>
</category>
```

Ketika *category* di atas dimuat di memory, maka sebuah *bot* AIML akan menjawab pertanyaan "Siapa nama kamu" dengan "Nama saya wahyu.".

Pattern

Sebuah *pattern* adalah sebuah rangkaian huruf yang diharapkan sesuai/cocok dengan satu atau bahkan lebih dengan masukan (input) pengguna. Suatu *pattern* dapat menggunakan *wildcard* yang akan cocok dengan satu atau lebih masukan pengguna. Suatu *pattern* seperti berikut :

```
SIAPA NAMA *
```

cocok dengan masukan "siapa nama kamu", "siapa nama mama kamu", "siapa nama dosen kamu", dan sebagainya. Sintaks *pattern* AIML merupakan pola yang jauh lebih sederhana dari regular expression.

"*The AIML pattern syntax is a very simple pattern language, far less complicated than regular expressions*"[9].

Template

Suatu *template* menentukan respon dari *pattern* yang sesuai. Sebuah *template* dapat berupa sebuah teks harafiah yang sederhana seperti berikut

```
Nama saya Wahyu.
```

Sebuah *template* juga dapat menggunakan variabel seperti :

```
Nama saya <bot name="name"/>.
```

Variabel bernilai sama dengan nama *bot* dan disisipkan ke dalam kalimat.

Template juga memungkinkan untuk meneruskan ke *pattern* lain dengan menggunakan *element* AIML bernama *srai*. Elemen *srai* dapat digunakan untuk mengimplementasikan persamaan arti seperti pada contoh berikut.

```
<category>
   <pattern>SIAPA NAMA KAMU</pattern>
   <template>Nama saya <bot name="name"/>.</template>
 </category>
 <category>
   <pattern>KAMU DIPANGGIL APA</pattern>
   <template>
     <srai>siapa nama kamu</srai>
   </template>
 </category>
```

Category pertama akan menjawab sebuah masukan "siapa nama kamu" dengan sebuah pernyataan mengenai nama *bot*. *Category* kedua akan menjawab masukan "kamu dipanggil apa" dengan meneruskan pertanyaan tersebut ke *category* pertama yang cocok dengan masukan "siapa nama kamu"—dengan kata lain bahwa dua frase tersebut adalah sebanding/sama.

That

That merupakan *element* AIML yang mengacu pada respon/keluaran sebelumnya. *That* sering digunakan pada pembuatan *category* agar respon yang dihasilkan masih berkaitan dengan pertanyaan/jawaban sebelumnya.

Masih terdapat juga 20 *element* (*tag*) dan *tag-tag* tambahan yang sering ditemukan dalam *file* AIML, bahkan memungkinkan untuk mendefinisikan *tag* sendiri yang disebut "*custompredicates*".

2.2 *Graphmaster Pattern Matching* (GPM)

Untuk mendapatkan waktu mencocokan pola (*pattern matching time*) yang efisien, dan penggunaan memori yang kecil, perangkat lunak AIML menyimpan semua kategori menjadi sebuah pohon (*Tree*) yang diatur menggunakan sebuah objek yang disebut *Graphmaster*.

Graphmaster pattern matching adalah algoritma *depth-first search(DFS)* yang lebih spesifik dengan berusaha menghilangkan *backtracking*, meskipun dalam beberapa kasus masih terdapat *backtracking*.

Graphmaster terdiri dari suatu koleksi dari titik-titik (*nodes*) yang disebut *Nodemappers*. *Nodemapper* memetakan cabang dari setiap *node* , dimana cabang dapat berupa kata tunggal ataupun *wildcard.*

Algoritma utama dari *Graphmaster* adalah sebagai berikut :

Apakah *node* mempunyai ' _ '? Jika ada, maka cari di *subgraph* yang akarnya adalah anak node yang mempunyai kaitan dengan ' _ '. Coba semua kemungkinan sufiks dari masukan untuk melihat apakah hal itu cocok. Jika tidak maka lanjutkan ke poin ke 2.

Apakah *node* mempunyai kata wh, dalam kalimat masukannya? Jika ada, maka cari di *subgraph* yang mempunyai kaitan dengan wh, menggunakan ekor dari masukan wh+1,...,wk. Jika tidak maka lanjutkan ke poin ke 3.

Apakah *node* mempunyai ' * '? Jika ada, cari di *subgraph* yang mempunyai root yang anak nodenya terkait dengan ' * '. Coba semua kemungkinan sufiks untuk menemukan 1 yang cocok. Jika tidak maka berikan kembalian false.

Secara sederhana, cara kerja *Graphmaster Pattern Maching* diilustrasikan seperti sebuah kamus ensiklopedia, yaitu jika ingin mencari suatu kata atau frase tidak perlu mencari dari awal hingga akhir buku sampai menemukan suatu kata atau frase yang cocok, tetapi dengan cara membuka bagian huruf pertama dari kata atau kata pada buku dicari, kemudian lanjutkan dengan mencari huruf berikutnya yang mengikuti huruf pertama.

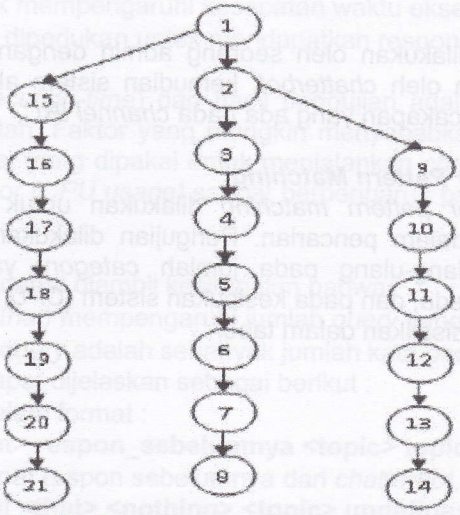
Pada *graphmaster*, karakter ' _ ' dan ' * ' (*wildcard*) berlaku seperti dua karakter special yang berada sebelum 0 setelah Z.secara berturut-turut. Karakter ' _ ' harus di cek pada awal operasi, setelah itu cek karakter alphabet, jika belum ditemukan, maka gunakan karakter ' * '.
Contoh :

```
<category>
<pattern>SIAPA *</pattern>
<template><star/> adalah hacker</template>
</category>
```

String AIML di atas akan diubah menjadi tabel *patterns* MySQL dengan ilustrasi AIML *Tree* seperti berikut "

Tabel 2.1
Tabel *patterns* pada MySQL

id	word	parent	isend
1	<input>	-1	0
2	SIAPA	1	0
3	NULL	2	1
4	<that>	3	0
5	NULL	4	1
6	<topic>	5	0
7	NULL	6	1
8		7	0



Gambar 2.1 AIML *Tree* dengan 3 *category*

Dan tabel *templates* seperti berikut :

Tabel 2.2
Tabel *templates* pada MySQL

id	*template*	*Pattern*
8	<template><star/> adalah hacker</template>	SIAPA _

Pada Gambar 2.1, *category* yang telah dikonversi menjadi tabel MySQL diilustrasikan sebagai rangkaian *node* 1-2-3-4-5-6-7-8, sedangkan rangkaian *node* lainnya adalah *categorycategory* lain yang juga telah dikonversi ke dalam tabal MySQL.

3. Pembahasan

3.1 Perancangan Sistem *Chatterbot*

Chatterbot merupakan gabungan dari dua macam *bot* yaitu *IRC bot* menangani masalah koneksi dengan jaringan *IRC* dan *parsing* untuk mendapatkan *input,* Sedangkan tugas AIML *bot* adalah mencari respon untuk suatu *input* yang diberikan oleh *IRC bot.* Setiap ketikan *user* pada *channel IRC*, merupakan *input* yang akan diproses oleh *chatterbot.* Alur program *chatterbot* terlihat seperti gambar berikut :

Gambar 3.1 Alur Program *Chatterbot*

Aktivasi *chatterbot* dilakukan oleh seorang admin dengan menentukan *nick, channel, server* yang akan digunakan oleh *chatterbot,* kemudian sistem akan bekerja secara otomatis untuk menanggapi setiap percakapan yang ada pada *channel IRC.*

3.1 Analisis *Graphmaster Pattern Matching*

Analisis *graphmaster pattern matching* dilakukan untuk mengetahui kelebihan dan kekurangan dari algoritma dalam pencarian. Pengujian dilakukan dengan memberikan *input* "siapa kamu" secara berulang-ulang pada jumlah *category* yang berbeda, pada respon sebelumnya (*that)* yang berbeda, dan pada kesibukan sistem (*CPU usage)* yang berbeda. Berikut adalah hasil pengujian yang disajikan dalam tabel :

Tabel 3.1
Tabel Hasil Pengujian
Algoritma Graphmaster Matching
Berdasarkan Jumlah *Category*

Category	Execution time (detik)	Jumlah Query
10	0.062047	10
20	0.049077	10
30	0.062694	10
40	0.107908	10
50	0.054682	10
60	0.05507	10
70	0.05131	10
80	0.057071	10
90	0.055716	10
100	0.05254	10

Tabel 3.2
Tabel Hasil Pengujian
Algoritma Graphmaster Matching
Berdasarkan respon sebelumnya *(that)*

Respon sebelumnya	Execution time (detik)	Jumlah Query
	0.062047	8
hai	0.049077	9
halo	0.062694	9
kamu siapa	0.107908	10
nama kamu siapa	0.054682	11
nama saya wahyu	0.046877	11
saya wahyu	0.059271	10
ya	0.056836	9
tidak	0.064790	9
senang berkenalan dengan anda	0.053420	12

Tabel 3.3
Tabel Hasil Pengujian *Algoritma Graphmaster Matching*
Berdasarkan Jumlah *Category*

Category	Execution time (detik)	Jumlah Query
10	6.024044	10
20	2.083075	10
30	0.062824	10
40	5.10763	10
50	1.057633	10
60	0.055716	10
70	0.051313	10
80	2.052347	10
90	16.05278	10
100	7.052624	10

Berdasarkan tabel 3.1 dapat diambil kesimpulan bahwa :

Jumlah *category* mempengaruhi banyaknya *query* yang dilakukan dalam pencarian.

Jumlah *category* tidak mempengaruhi kecepatan waktu eksekusi (*execution-time*).

Waktu rata-rata yang diperlukan untuk mendapatkan respon dari *input* "Selamat pagi" adalah 0.05 detik.

Waktu eksekusi (*execution-time*) dari hasil pengujian adalah acak tetapi masih dalam kisaran angka yang berdekatan. Faktor yang mungkin menyebabkan nilai acak pada pengujian adalah keadaan dari komputer yang dipakai untuk menjalankan *chatterbot*. Kecepatan komputer dan tingkat kesibukan prosesor *(CPU usage)* sangat berpengaruh pada lamanya waktu eksekusi, seperti terlihat pada tabel 3.3.

Berdasarkan tabel 3.2, maka dapat diambil kesimpulan bahwa :

Respon sebelumnya(*that)* mempengaruhi jumlah *query* yang dilakukan dalam pencarian.

Penambahan jumlah *query* adalah sebanyak jumlah kata pada respon sebelumnya(*that)*.

Penambahan jumlah query dapat dijelaskan sebagai berikut :

Input akan disusun dalam format :

<input> kalimat <that> respon_sebelumnya <topic> topic_yg_aktif

Pada saat tidak terdapat respon sebelumnya dari *chatterbot*, maka *input* adalah :

<input> selamat pagi <that> <nothing> <topic> undefined
1 2 3 4 5 6 7

Jumlah kata dari *input* di atas yaitu 7 kata, dan untuk mengecek apakah *graphmaster* sudah memproses semua kata pada kalimat, ditambahkan *input* berupa *string* kosong. Jumlah kata dari input menjadi 8 kata. Banyaknya kata pada *input* inilah yang mempengaruhi jumlah *query*.

4. Kesimpulan

Berdasarkan pembahasan sebelumnya, penulis menarik kesimpulan sebagai berikut :

Graphmaster pattern matching memahami kalimat dengan cara mencari urutan kata penyusun kalimat yang paling sesuai.

Jumlah *category* pada *chatterbot* tidak mempengaruhi banyaknya *query* yang dilakukan oleh *graphwalker()* untuk mendapatkan *pattern* yang sesuai.

Respon sebelumnya oleh *chatterbot* mempengaruhi jumlah *query* yang dilakukan dalam pencarian.

Jumlah *category* tidak mempengaruhi kecepatan waktu eksekusi (*execution-time*).

Kecepatan pencarian (*execution-time)* sangat dipengaruhi oleh keadaan komputer (*cpu usage)* pada saat melakukan pencarian.

Ketepatan dan kecepatan respon mengenai jadwal dan sinopsis film dipengaruhi oleh kecepatan akses google.com dan 21cineplex.com.

Algoritma *graphmaster pattern matching* merupakan algoritma yang optimal untuk *chatterbot*, diukur dari kecepatan respon yang diberikan.

5. Daftar Pustaka

[1]. Bush, Noel, (2001). "Artificial Intelligence Markup Language (AIML) Version 1.0.1: A.L.I.C.E. AI Foundation Working Draft", dalam http://Alicebot.org/TR/2001/WD-aiml/; Juni 2008.

[2]. Friedl, Jeffrey E. F., (2002). "Mastering Regular Expressions 2nd Edition" : O'Reilly. [3]. Huynh, Michael, (2004) "Brain, the Framework for a Chatterbot" dalam http://www.mikexstudios.com/research/brain - chatterbot-framework .pdf ; 18 Juni 2008.

[4]. Ringate, Thomas,(2001)"AIML Reference Manual", dalam http://alicebot.org/documentation/aiml-reference.html; 2 Juni 2008.

[5]. Syafi'i, M., (2006). "Membangun Aplikasi Web dengan PHP dan MySQL", Edisi III, Yogyakarta: Andi.

[6]. Stubblebine, Tony, (2007). "Regular Expression Pocket Reference, Second Edition", 2nd Edition: O'Reilly.

[7]. Wallace, Richard,(2000). "A.L.I.C.E and AIML Specification, Don't Read me", dalam http://alicebot.org/articles/wallace/dont.html; 29 Mei 2008.

[8]. Wallace, Richard, (2001) "AIML Pattern Matching Simplified", dalam http://alicebot.org/documentation/matching.html; 29 Mei 2008.

6. Biodata Penulis

Wahyu Catur Pamungkas, lahir di Solo pada tanggal 18 September 1984 merupakan anak terakhir dari empat bersaudara, sekarang bertempat tinggal di Cirebon. Saat ini sedang menempuh pendidikan S1 Teknik Informatika di Universitas Kristen Duta Wacana Yogyakarta.

Pekerjaan sekarang adalah mahasiswa dan aktivis dari sebuah forum komunitas internet (http://forum.cyberphreaking.com) sebagai member VIP dikenal dengan *nick* czephie.