PEMANFAATAN *SENTENCE-SIMILARITY MEASUREMENT* UNTUK PROSES PENCARIAN POLA PADA *CHATBOT* BERBASIS *PATTERN-MATCHING*

Ayu Mutiara Oktavia Dewi [1)], **Bayu Setiaji** [2)]

Jurusan Teknik Informatika STMIK AMIKOM Yogyakarta
Jl Ring road Utara, Condongcatur, Sleman, Yogyakarta 55281
Email : ayu.d@students.amikom.ac.id [1)], bayusetiaji@amikom.ac.id [2)]

Abstrak

Chatbot adalah salah satu aplikasi dalam bidang Pemrosesan Bahasa Alami atau Natural Language Processing (NLP) yang dapat melakukan percakapan dengan manusia. Chatbot memungkinkan interaksi antara manusia dan komputer dapat dilakukan dengan bahasa alami melalui media tulisan atau suara. Salah satu cara yang digunakan dalam pemrosesan tulisan dalam chatbot adalah dengan pencocokan pola (pattern-matching). Pola – pola yang mungkin ditemukan selama proses percakapan tersimpan sebagai pengetahuan dalam bentuk plain text atau basis data.

Kemampuan interaksi chatbot berbasis pattern-matching adalah pada jumlah pola yang tersimpan dalam pengetahuan. Semakin banyak pola yang tersimpan akan membuat chatbot terlihat semakin pintar. Namun, jumlah pola yang banyak tersebut dapat mengakibatkan proses pencocokan menjadi semakin lama. Sentence similarity measurement (SSM) adalah salah metode untuk menghitung tingkat kemiripan suatu kalimat. Pemanfaatan SSM dalam proses pencarian memungkinkan pengurangan penyimpanan jumlah pola karen satu pola dapat mewakili beberapa input dengan arti yang hampir sama.

Hasil penelitian berupa stored program dalam basis data yang mengimplementasikan sentence similarity measurement. Proses pengujian menggunakan perintah Structured Query Language (SQL) untuk menjalankan stored program tersebut.

Kata kunci: *chatbot, pattern-matching, sentence-similarity, SQL.*

1. Pendahuluan

Chatbot merupakan salah satu aplikasi dalam bidang *Natural Language Processing* (NLP). Aplikasi *chatbot* sendiri dibuat untuk banyak tujuan, seperti *customer service* perusahaan atau hanya sekedar untuk percakapan biasa. Pada umumnya *chatbot* menggunakan media teks sebagai sarana berkomunikasi dengan lawan bicaranya (*user*), yang bisa berupa manusia atau *chatbot* yang lain. Ketika ada input dari *user chatbot* kemudian memberikan respon yang sesuai. Respon yang diberikan tergantung dari pengetahuan yang dimiliki *chatbot* itu sendiri. Pengetahuan tersimpan dalam berbagai macam bentuk seperti file teks sederhana atau dalam sistem basis data.

Pencocokan pola (*pattern-matching*) adalah salah satu metode yang digunakan untuk proses pemilihan respon. Dalam metode ini pengetahuan berupa kumpulan pola yang saling berpasangan satu – satu dengan respon. *Chatbot* akan mencari pola dalam pengetahuan yang cocok dengan input dari *user* kemudian mengambil respon yang bersesuaian. Respon itu yang nantinya dikirimkan kembali kepada *user*. Semakin banyak dan beragam *pattern* yang tersimpan akan membuat percakapan terlihat lebih "natural", karena dapat dicocokkan dengan berbagai kemungkinan input yang masuk dari *user*. Jumlah *pattern* yang tersimpan berbanding terbalik dengan waktu pencocokan pola. Gambar 1 berikut ini adalah ilustrasi sederhana pencocokan antara input dari *user* dengan *pattern* yang ada.

Gambar 1. Pencocokan Pola

Berdasar Gambar 1 di atas terdapat beberapa pasangan *pattern* dan *template* (respon) di dalam pengetahuan *chatbot*. *User* memberikan input berupa string "siapa namamu" dan *chatbot* akan mencari *pattern* yang cocok dengan input tersebut. Dari gambar di atas *chatbot* berhasil menemukan pola yang sesuai dan akan memberikan *template* "Namaku TIARA" sebagai respon.

Namun, ketika *user* memberikan input "namamu siapa" atau "nama kamu siapa" *chatbot* tidak dapat menemukan *pattern* yang sesuai padahal ketiga input tadi memiliki arti yang sama dengan "siapa namamu". Oleh karena itu

untuk satu kalimat dengan arti yang sama *chatbot* harus memiliki koleksi beberapa *pattern* yang kemungkinan dijadikan input oleh *user*. Selain itu *chatbot* juga harus bisa mengantisipasi adanya penggunaan tanda baca sehingga sebelum melakukan proses *pattern-matching* input *user* harus sudah dinormalkan terlebih dahulu sesuai *rule* yang ada pada *chatbot* seperti penghilangan tanda baca, perubahan *case*, dan lain – lain.

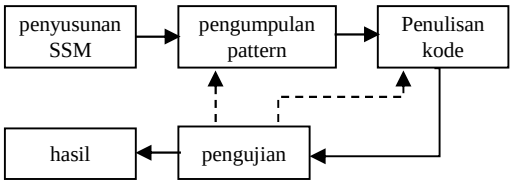
Salah satu contoh adalah *Artificial Intelligence Markup Language* (AIML), sebuah bahasa berdialek XML yang digunakan sebagai representasi pengetahuan *chatbot*. Di dalam dokumen AIML terdapat puluhan *tag* untuk mendefinisikan apa yang harus dilakukan oleh *interpreter* untuk melakukan proses pencocokan pola sehingga alur pencocokan pola ditentukan sekaligus di dalam *pattern* yang tersimpan. Salah satu *tag* dalam AIML adalah `<srai/>` yang digunakan untuk memerintahkan *interpreter* agar melakukan pencocokan secara rekursif. [1]

Berdasar uraian latar belakang di atas penelitian ini akan membahas tentang penggunaan salah satu metode pencocokan yaitu *sentence-similarity measurement* yang mendasarkan pada prosentase kemiripan antara input *user* dengan *pattern* sehingga dapat mengurangi penyimpanan jumlah koleksi dan keragaman *pattern* untuk dicocokkan dengan berbagai kemungkinan input dengan arti yang sama. Metode ini nantinya akan diimplentasikan pada *Relational Database Management System* (RDBMS) sebagai penyimpan pengetahuannya. Implementasinya dalam bentuk kode *Structured Query Language* (SQL).

Tujuan penelitian ini adalah membuat representasi pengetahuan *chatbot* sekaligus metode pencarian polanya menggunakan *sentence-similarity measurement* yang terpaket dalam RDBMS sehingga peran bahasa pemrograman hanya sebatas pada perantara antara *user* dengan sistem *chatbot*. Agar lebih terfokus penelitian ini dibatasi hanya pada hal – hal berikut ini.

1. *Sentence-similarity measurement* (SSM) menggunakan *bigram*
2. Menggunakan RDBMS MySQL sebagai penyimpan pengetahuan
3. SSM disimpan dalam *stored program*.
4. Tidak membahas proses normalisasi input *user.*
5. Pengujian hanya berfokus pada proses *pattern-matching*.
6. Pengujian menggunakan *console* MySQL lewat *query* untuk mengakses *stored program* yang sudah dibuat.
7. Pengetahuan menggunakan Bahasa Indonesia.

Secara umum langkah penelitian digambarkan dalam Gambar 2 berikut ini.



Gambar 2. *Urutan Penelitian*

Gambar 2 di atas menunjukkan bahwa penelitian dimulai dengan penyusunan *flowchart* untuk SSM. Selanjutnya adalah memasukkan beberapa contoh *pattern* dan pasangan respon (*template*) untuk keperluan pengujian. Kemudian rumus tersebut ditulis dalam *stored program* di RDBMS. Berikutnya adalah mengujikan *stored program*. Bila hasil pengujian belum sesuai dengan perhitungan manual maka akan dilakukan penulisan ulang terhadap kode atau penyusunan ulang *pattern.*

2. Tinjauan Pustaka

Berikut ini adalah eberapa penelitian berkaitan dengan *chatbot* dan NLP pada umumnya yang sudah pernah dilakukan sebelumnya:

1. Aplikasi BotQA, sebuah aplikasi *chatbot offline* dan *stand alone* yang dilengkapi dengan animasi ekspresi wajah. *Chatbot* tersebut dapat menyusun ulang input dari *user* menyesuaikan pola yang sudah ditentukan sehingga dapat mengurangi koleksi *pattern*. Sebagai contoh di dalam pengetahuan tersimpan *pattern* "siapa namamu" maka ketika ada input "namamu siapa" *chatbot* akan menyusun ulang input tersebut menjadi "siapa namamu". [2]
2. *Chatbot service* berbasis AIML dengan arsitektur pengetahuan modular. Chatbot yang berupa *service* yang dapat digunakan untuk percakapan melalui program *client*. *Service chatbot* terhubung dengan modul – modul pengetahuan tertentu yang bersifat independen. *Interpreter* dan semua fungsi pendukungnya ditulis dalam bahasa PHP. Penggunaan RDBMS MySQL pada *chatbot* ini hanya untuk penyimpanan *pattern* saja sehingga untuk membuat *service* dalam bahasa yang berbeda harus menulis ulang kode untuk *interpreter*-nya. [3]

Berikut adalah beberapa teori yang dapat dijadikan landasan atau sumber referensi penelitian ini:

1. Chatbot

Secara harfiah *chatbot* berasal dari dua kata yaitu *chat* dan *bot*. Dalam dunia komputer *chat* dapat diartikan sebagai kegiatan komunikasi yang menggunakan sarana tulisan. Sedangkan *bot* merupakan program yang memiliki sejumlah data yang bila diberi input akan menghasilkan output sebagai jawaban. Dari dua

istilah di atas dapat diartikan bahwa *chatbot* adalah program komputer yang dapat melakukan percakapan melalui media tulisan. Percakapan dapat terjadi dengan manusia atau *chatbot* yang lain. [2]

2. *Pattern-matching*

   Dalam ilmu komputer, pencocokan pola (*pattern-matching* atau *string-matching*) adalah kegiatan pemeriksaan serangkaian token yang diberikan untuk menemukan beberapa pola yang konstituen dalam string. Berbeda dengan pengenalan pola, kecocokan dalam *pattern matching* harus tepat. [4]

   Sebagai contoh adalah fasilitas *search* pada editor teks. Misalnya akan dicari pola "hai" maka akan ditemukan pada kata – kata "aduhai", "Thailand", 'wahai" dan lain – lain.

3. *Bigram*

   Dalam kamus Oxford *bigram* adalah kata benda sebagai istilah di bidang linguistik yang memiliki arti sepasang unit yang tertulis berurutan, berupa huruf, suku kata, atau kata. Berikut ini adalah contoh *bigram* yang merupakan pasangan 2 huruf dari sebuah string.

   string: "minnion"
   bigram: {"mi", "in", "nn", "ni", "io", "on"}

4. *Sentence-similarity measurement* (SSM)

   SSM adalah perhitungan tingkat kemiripan antara dua kalimat atau string. Skor yang dihasilkan dari perhitungan *similarity* adalah berupa bilangan real 0 sampai 1. [5]

   Dalam kaitannya dengan penggunaan *bigram* dapat dibuat sebuah rumus sederhana untuk menghitungnya. Berikut ini adalah rumus yang digunakan. [6]

$$\frac{count(b1 \in b2)+count(b2 \in b1)}{count(b1)+count(b2)} \quad .....(1)$$

   Penjelasan dari rumus (1) di atas adalah sebagai berikut.

   b1 adalah himpunan *bigram* yang dihasilkan dari string pertama dan *b2* adalah himpunan *bigram* yang dihasilkan dari string kedua.

   $count(b1 \in b2)$ adalah operasi himpunan untuk jumlah "setiap *bigram* di *b1* yang merupakan anggota *bigram* di *b2*". Demikian sebaliknya untuk $count(b2 \in b1)$.

   $count(b1)$ adalah jumlah *bigram* yang ada di *b1*. Demikian sebaliknya untuk $count(b2)$.

   Berikut ini adalah contoh penggunaan *rumus* (1) di atas.

   string1: "tiara"

   string2: "tiarra"

   b1 = {"ti", "ia", "ar", "ra"}

   b2 = {"ti", "ia", "ar", "rr", "ra"}

   $b1 \in b2$ = {"ti", "ia", "ar", "ra"}

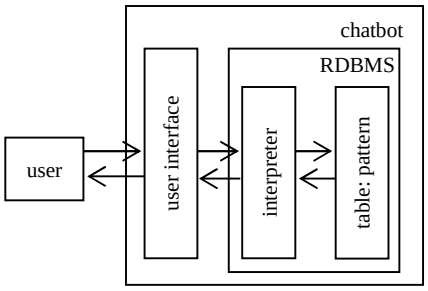
   $b2 \in b1$ = {"ti", "ia", "ar", "ra"}

   Maka diperoleh nilai *similarity* untuk string1 dan string2:

$$\frac{4+4}{4+5} = \frac{8}{9} = 0{,}88$$

3. Pembahasan

Sebelum melakukan penelitian sesuai langkah yang sudah dipaparkan pada Gambar 2 maka harus dipersiapkan terlebih dahulu rancangan arsitektur *chatbot* untuk keperluan pengujian. Gambar 3 berikut ini adalah gambaran umum arsitekur *chatbot* yang akan digunakan.



Gambar 3. Gambaran Umum Arsitektur *Chatbot*

Gambar 3 di atas menjelaskan bahwa *chatbot* terdiri dari 3 bagian yaitu *user interface*, *interpreter*, dan *table* sebagai penyimpan *pattern*. *Interpreter* dan *table* terletak pada sisi RDBMS sedangkan *user interface* berada di luarnya, yang ditulis menggunakan bahasa pemrograman tertentu. Penelitian ini hanya membahas *interpreter* sebagai bagian yang berisi langkah *pattern-matching* untuk pencocokan antara input dengan *pattern* yang ada di *table*.

Penyusunan SSM sudah dipaparkan pada rumus (1) di bagian tinjauan pustaka sebelumnya, sehingga langkah selanjutnya adalah mengumpulkan contoh *pattern-template* yang tersimpan dalam tabel. Tabel 1 berikut ini adalah struktur tabel *pattern* yang akan digunakan sebagai penyimpan *pattern-template.*

Tabel 1. Struktur Tabel *pattern*

Field	Tipe	Keterangan
idpatt	UNSIGNED INT	Id pattern
pattern	VARCHAR(50)	Pattern
template	VARCHAR(50)	Respon

Berdasar pada Tabel 1 di atas tabel *pattern* terdiri dari 3 *field*, yaitu *idpatt* digunakan sebagai identitas untuk pasangan *pattern-template*. *Field pattern* digunakan untuk menyimpan *pattern* yang akan dicocokkan dengan

input *user*, dan *field template* digunakan untuk menyimpan *template* sebagai respon kepada *user*. Tabel 2 berikut ini adalah sampel *pattern-template* yang digunakan sebagai bahan untuk pengujian.

Tabel 2. Sampel *pattern-template*

id	pattern	template
01	assalamualaikum	Wa'alaikum salam
02	hai	Hai juga
03	apa kabar	Kabar baik
04	siapa namamu	Namaku TIARA
05	aku mau tanya	OK,silakan
06	siapa presiden pertama indonesia	Soekarno
07	siapa soekarno	Presden pertama Indonesia
08	presiden pertama indonesia adalah soekarno	Betul, presiden pertama Indonesia adalah Soekarno
09	di mana ibu kota indonesia	Ibu kota Indonesia adalah Jakarta
10	jakarta adalah ibu kota indonesia	Betul sekali

Tabel *pattern* berserta sample *pattern-template* di atas kemudian diimplementasikan dalam database ***SSMTes***.

Langkah selanjutnya adalah membuat fungsi untuk perhitungan SSM sesuai dengan rumus (1) yang sudah dipaparkan pada pembahasan sebelumnya. Berikut ini adalah *pseudocode* untuk fungsi **f_similar()**.

```
 1 FUNC f_similar(str1, str2):
 2 BEGIN
 3  b1 ← BIGRAM(str1)
 4  b2 ← BIGRAM(str2)
 5  b12 ← INTERSECT(b1, b2)
 6  b21 ← INTERSECT(b2, b1)
 7  c1 ← COUNT(b1)
 8  c2 ← COUNT(b2)
 9  c12 ← COUNT(b12)
10  c21 ← COUNT(b21)
11  f_similar ← (c12+c21)/(c1+c2)
12 END
```

Berdasar *pseudocode* di atas fungsi *f_similar()* memiliki argumen *str1* dan *str2* dengan tipe *string* dengan kembalian bertipe *real*. Baris 3 dan 4 menunjukkan pembuatan *bigram* dari *str1* dan *str2* yang masing – masing di-*assign* ke variabel *b1* dan *b2*. Baris 5 dan 6 menunjukkan operasi $b1 \in b2$ dan $b2 \in b1$ yang hasilnya masing – masing di-*assign* ke variabel *b12* dan *b21*. Baris 7 dan 8 menghitung banyaknya *bigram* yang dihasilkan dalam *b1* dan *b2* yang hasilnya masing – masing di-*assign* ke variabel *c1* dan *c2*. Baris 9 dan 10 menghitung banyaknya *bigram* di *b12* dan *b21*. Terakhir di baris ke-11 fungsi mengembalikan nilai (c12+21)/(c1+c2) sebagai skor *similarity*. *Pseudocode* di atas diimplementasikan dalam *stored function* di database *SSMTes* dengan nama yang sama yaitu **f_similar().**

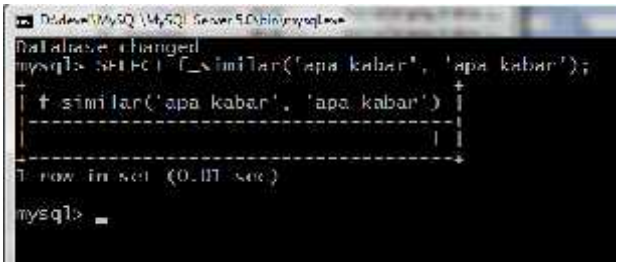
Berikutnya adalah pengujian terhadap fungsi *f_similar()*. Tabel 3 berikut menunjukkan input dan *pattern* yang akan diujikan.

Tabel 3. Nilai Input dan *Pattern*

Pattern	apa kabar	
Input	1	apa kabar
	2	apa kabarmu
	3	apa kabar kamu
	4	bagaimana kabarmu
	5	aku mau tanya

Dari Tabel 3.3 di atas *pattern* yang akan dijadikan *sample* adalah "apa kabar", sedangkan input berupa 4 frase yang memiliki arti sama dan 1 frase yang memiliki arti berbeda dengan nilai *pattern* yang diujikan. Pengujian dilakukan melalalui *console* MySQL dengan menggunakan perintah berikut:

```
SELECT f_similar(input, pattern);
```



Gambar 4. *Console* MySQL

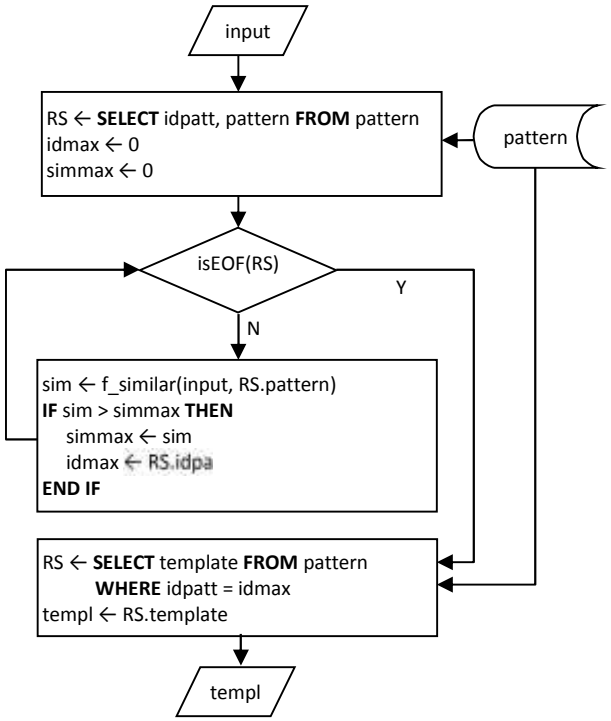
Hasil pengujian disajikan dalam Tabel 4 berikut ini.

Tabel 4. Hasil Pengujian Fungsi *f_similar()*

No	Input	Skor
1	apa kabar	1
2	apa kabarmu	0,8888
3	apa kabar kamu	0,8571
4	bagaimana kabarmu	0,5416
5	aku mau tanya	0

Berdasar Tabel 4 di atas skor tertinggi adalah input nomor 1 karena memiliki nilai yang sama persis dengan *pattern*. Demikian sebaliknya untuk input nomor 5 yang menghasilkan skor 0.

Fungsi *f_similar()* selanjutnya digunakan di dalam *core* proses *pattern-matching* untuk mencari skor tertinggi di antara *pattern* yang ada. *Template* pada *pattern* dengan skor tertinggi inilah yang nantinya akan dijadikan respon untuk dikembalikan kepada *user*. Proses *pattern-matching* akan dilakukan oleh fungsi **get_template()**. Gambar 5 berikut ini adalah *flowchart* proses *pattern-matching*.



Gambar 5. *Flowchart Pattern-matching*

Pseudocode fungsi *get_template()* berdasar *flowchart* pada Gambar 5 di atas adalah sebagai berikut.

```
 1 FUNC get_template(input):
 2 BEGIN
 3  RS ← SELECT idpatt, pattern
 4       FROM pattern
 5  idmax ← 0
 6  simmax ← 0
 7  WHILE NOT EOF(RS) DO
 8   sim ← f_similar(input, RS.pattern)
 9   IF sim > simmax THEN
10     simmax ← sim
11     idmax ← RS.idpatt
12   END IF
13  END WHILE
14  templ ← SELECT template FROM pattern
15          WHERE idpatt = idmax
16  get_template ← templ
17 END
```

Berdasar *pseudocode* di atas fungsi *get_template()* memiliki argumen *input* bertipe *string* sebagai input dari *user*. Pertama kali fungsi akan mengambil seluruh data *idpatt* dan *pattern* dan di-*assign* ke variabel *RS* sebagai *recordset*. Mula – mula sebelum perulangan variabel *idmax* sebagai penampung *idpatt* dari *pattern* dengan skor *similarity* tertinggi di-*assign* dengan nilai 0 dan variabel *simmax* sebagai penampung skor *similarity* tertinggi di-*assign* dengan nilai 0. Kemudian akan dilakukan perulangan selama *recordset RS* masih belum mencapai akhir. Selama perulangan akan ada proses pengambilan skor *similarity* untuk *input* dan *pattern* pada *recordset* yang sedang diambil dengan memanggil fungsi *f_similar()*. Skor di-*assign* ke variabel *sim*. Bila nilai *sim* sekarang lebih tinggi dari nilai *simmax* maka nilai *simmax* sekarang digantikan oleh *sim* dilanjutkan dengan nilai *idmax* sekarang digantikan oleh nilai *id* pada *recordset* yang sedang diambil. Setelah selesai perulangan akan diambil *template* dari tabel *pattern* dengan nilai *idpatt = idmax* yang hasilnya di-*assign* ke variabel *templ*. Langkah terakhir fungsi akan mengembalikan nilai variabel *templ*.

Fungsi *get_template()* diimplementasikan sebagai *stored function* di database *SSMTes*. Pengujian dilakukan melalui *console* MySQL menggunakan perintah berikut ini.

```
SELECT get_template(input);
```

Tabel 5 berikut ini menunjukkan hasil pengujian yang berupa input *user* dan respon yang diberikan berdasar pengetahuan yang sudah dipaparkan pada Tabel 2 sebelumnya.

Tabel 5. Hasil Pengujian Fungsi *get_template()*

No	Input	Respon
1	assalamualaikum	Wa'alaikum salam
2	hai	Hai juga
3	hai apa kabar	Kabar baik
4	bolehkan aku tanya	OK, silakan
5	siapa namamu	Namaku TIARA
6	namamu siapa sih	Namaku TIARA
7	di mana ibu kota indonesia	Ibu kota Indonesia adalah Jakarta
8	jakarta adalah ibu kota indonesia	Betul sekali
9	jakarta bukan ibu kota indonesia	Betul sekali
10	Ibu kota indonesia adalah yogyakarta	Betul sekali
11	siapa soekarno	Presiden pertama Indonesia
12	siapa presiden pertama indonesia	Soekarno
13	siapa presiden kedua indonesia	Soekarno
14	Hari ini sungguh cerah	Betul, Soekarno adalah presiden pertama Indonesia
15	uhui	*NULL*

Tabel 5 di atas menunjukkan bahwa pengujian nomor 9, 10, 13, 14 bisa dikatakan tidak sesuai dengan fakta yang sebenarnya. Namun, berdasar skor *similarity* tertinggi proses *pattern-matching* dikatakan berhasil karena jumlah *pattern-template* yang digunakan masih sedikit. Sedangkan pengujian nomor 15 menghasilkan skor 0 sehingga tidak ada satupun *pattern-template* yang terpilih.

3. Kesimpulan

Berdasar uraian pada bab sebelumnya dapat ditarik beberapa kesimpulan seperti berikut ini.

1. Penggunaan *sentence-similarity measurement* dalam proses *pattern-matching* di *chatbot* adalah dengan mencari skor tertinggi pada *pattern*.
2. Beberapa input berbeda yang mengandung arti sama dapat diwakili oleh satu *pattern* dalam pengetahuan.
3. Diperlukan keragaman fakta yang dimasukkan sebagai *pattern-template* untuk dapat memberikan hasil yang lebih natural.

Berikut ini adalah beberapa saran untuk pengembangan penelitian selanjutnya.

1. Kode SQL untuk pembuatan *stored function* dapat dibuat lebih *portable* sehingga memungkinkan untuk diimplementasikan juga ke RDBMS lain selain MySQL.
2. Penelitian selanjutnya sebaiknya menggunakan sampel yang beragam sehingga diharapkan dapat menghasilkan standar skala validitas skor *similarity*.

Daftar Pustaka

[1] N.Bush, "Artificial Intelligence Markup Language (AIML) Version 1.0. 1", A.L.I.C.E AI Foundation Working Draft, June 25 2001, (rev 006).
[2] E.Utami, S.Hartati, "Aplikasi BotQA Untuk Meningkatkan Cara Interaksi Manusia dan Mesin", Seminar Nasional Teknologi Informasi 2007 (SNATI 2007), Juni 16, 2007, Yogyakarta.
[3] B.Setiaji, "Membangun Chatbot Berbasis AIML dengan Arsitektur Pengetahuan Modular", Semnasteknomedia 2013, pp. 18-15, Januari 19, 2013.
[4] http://en.wikipedia.org/wiki/Pattern_matching, diakses pada November 28, 2013, 18.58
[5] P.Achananuparp, X.Hu, X.Shen, "The Evaluation of Sentence Similarity Measures", Lecture Notes in Computer Science Volume 5182, pp.306, 2008.
[6] http://www.nairaland.com/1150055/simple-very-useful-word-sentence, diakses pada November 25, 14.23

Biodata Penulis

Ayu Mutiara Oktavia Dewi, mahasiswa Jurusan Teknik Informatika STMIK AMIKOM Yogyakarta.

Bayu Setiaji, M.Kom, memperoleh gelar Sarjana Komputer (S.Kom) Jurusan Teknik Informatika STMIK AMIKOM Yogyakarta, lulus tahun 2006. Memperoleh gelar Magister Komputer (M.Kom) Program Pasca Sarjana Magister Teknik Informatika pada perguruan yang sama, lulus tahun 2012. Saat ini menjadi dosen tetap di STMIK AMIKOM Yogyakarta.