Penggunaan *Named Entity Recognition* dan *Artificial Intelligence Markup Language* untuk Penerapan Chatbot Berbasis Teks

David Christianto[#1], Elisafina Siswanto[#2], Ria Chaniago[#3]

[#]*Faculty of Informatics Engineering, Institut Teknologi Harapan Bangsa Bandung, Indonesia*

[1]david.christianto1@gmail.com

[2]elisafina@ithb.ac.id

[3]rk.chaniago@gmail.com

***Abstrak*— Aplikasi *chatbot* dapat digunakan untuk membantu memberikan kebutuhan informasi pada sistem layanan *operator service*. Sistem *chatbot* yang digunakan adalah sistem *chatbot* yang berbasiskan pada teks. Pada penelitian ini, *chatbot* dibuat untuk memenuhi kebutuhan informasi di ITHB dengan menggunakan *Named Entity Recognition* (NER) dan *Artificial Intelligence Markup Language* (AIML). NER digunakan untuk membantu mengenali pola (kata kunci) kalimat dari bahasa sehari-hari manusia (*Natural Language Processing*). AIML digunakan untuk memberikan jawaban yang relevan dan sesuai dengan pola (kata kunci) kalimat yang telah ditemukan di dalam bahasa manusia. Selain itu, pada penelitian ini juga dilakukan beberapa optimasi seperti optimasi pada proses perhitungan *Naïve Bayes* pada NER, proses *spelling correction*, dan proses *pattern matching* yang terbukti dapat mempercepat dan meningkatkan akurasi sistem *chatbot* dalam proses pencarian jawaban. Berdasarkan hasil pengujian, sistem *chatbot* ini dapat mengenali pola kalimat bahasa manusia dengan akurasi (NER) hingga 97% dan sistem dapat memberikan jawaban yang tepat dengan akurasi hingga 90% berdasarkan pola yang telah ditemukan tersebut.**

***Kata kunci* — *Chatbot*, NER, AIML, *Natural Language Processing, Naïve Bayes, Spelling Correction, Pattern Matching*.**

Abstract*—*In operator service system area, information is an essential needs for every individuals. Chatbot application can be used to support the fulfilment of information in operation service system. Chatbot system that will be implemented is a text-based chatbot system. In this paper, chatbot was made in order to fulfil the information needs in ITHB by using Named Entity Recognition (NER) and Artificial Intelligence Markup Language (AIML). NER is used to recognize the sentence pattern (keyword) in human natural language (Natural Language Processing). AIML is used to process relevant responses based on the keyword patterns found in human natural language which then will be transformed into data which can be processed and understood by system. This research also covers several optimizations, such as Naïve Bayes calculation optimization in NER, spelling correction optimization, and pattern matching optimization that has been proven to hasten and increase chatbot system's accuracy in finding answers as response. Based on the empirical examination, this chatbot system can recognize human sentence pattern (NER process) with accuracy of 97% and system can provide suitable response with accuracy of 90% based on the recognized patterns from NER process.

Keywords—Chatbot, NER, AIML, Natural Language Processing, Naïve Bayes, Spelling Correction, Pattern Matching.

I. PENDAHULUAN

Chatterbot (atau yang dikenal sebagai *Chatbot*) merupakan sebuah program komputer yang dirancang untuk mensimulasikan sebuah percakapan atau komunikasi yang interaktif kepada *user* (manusia) melalui bentuk teks, *audio*, maupun *video*. Respon yang dihasilkan merupakan hasil pemindaian kata kunci pada *input* yang dilakukan oleh *user* dan menghasilkan respon balasan yang dianggap paling cocok, sehingga percakapan yang terjadi seakan-akan dilakukan oleh dua pribadi manusia yang saling berkomunikasi.

Dalam pembuatan sistem *chatbot* terdapat beberapa metode yang dapat digunakan, yaitu metode *scripts*, metode *shallow parsing*, dan metode *deep syntactic processing*. Berdasarkan penelitian yang dilakukan oleh Ioannis Doumanis & Serengul Smith [1] terhadap *performance* dari metode *scripts*, *shallow parsing*, dan *deep syntactic processing*, dapat dihasilkan bahwa metode *scripts* vs *shallow parsing* memiliki perbandingan akurasi sebesar 59% vs 57%, metode *scripts* vs *deep syntactic processing* memiliki perbandingan akurasi sebesar 59% vs 57%, dan metode *shallow parsing* vs *deep syntactic processing* memiliki perbandingan akurasi sebesar 57% vs 40%.

Berdasarkan metode-metode tersebut, maka pada penelitian ini digunakan metode *scripts* dalam mengenali pola *input* yang dilakukan oleh *user* karena metode *scripts* memiliki *performance* akurasi yang paling besar. Metode *scripts* yang akan digunakan adalah metode *Artificial Intelligence Markup Language* (*AIML*). Untuk membantu meningkatkan akurasi, maka pada penelitian ini juga akan ditambahkan proses *Text Preprocessing* dan *Named Entity Recognition* (*NER*) untuk membantu mengenali pola bahasa Indonesia yang beraneka ragam sehingga dapat membantu memudahkan dalam mengenali pola *input*. Metode *scripts* memiliki salah satu kelemahan yaitu memiliki *knowledge based* yang tidak bisa berubah. Oleh karena itu, juga akan dilakukan penelitian agar *knowledge based* dapat bertambah atau berubah secara dinamis.

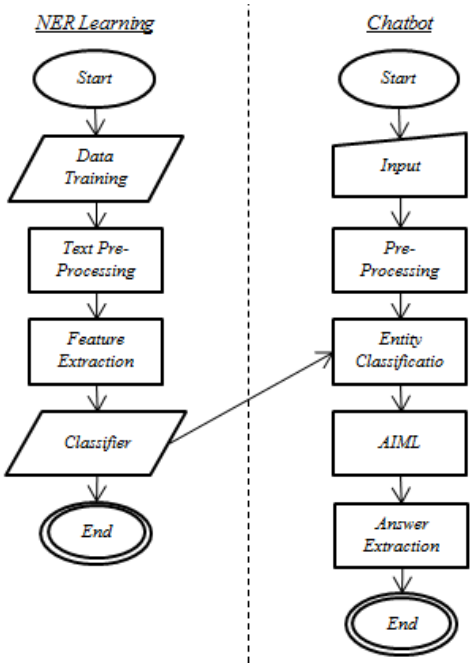
II. IMPLEMENTASI CHATBOT

Sistem yang akan dibangun sangat dipengaruhi oleh pengenalan pola *input* yang tepat, *keywords* yang sesuai dan relevan dengan pola *input* tersebut, kemudian memberikan jawaban yang sesuai dengan *input* tersebut. Langkah-langkah sistem *chatbot* dapat terlihat pada Gambar 1 sebagai berikut:

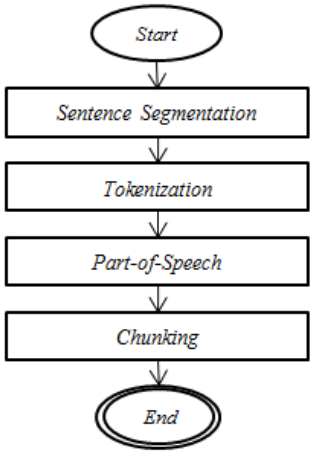
A. Text Preprocessing

Proses ini dilakukan untuk mengolah sebuah teks menjadi data yang mudah diolah oleh sistem saat proses utama dilakukan. Langkah-langkah untuk melakukan proses *text preprocessing* terlihat pada Gambar 2.

1) *Sentence Segmentation* adalah proses yang digunakan untuk melakukan proses segmentasi pada kumpulan kalimat. Pada penelitian ini, proses *sentence segmentation* dilakukan dengan cara mensegmentasikan kumpulan kalimat berdasarkan *linebreak*.
2) *Tokenization* adalah proses yang digunakan untuk memotong sebuah kalimat berdasarkan tiap kata penyusunnya. Berikut langkah-langkah untuk melakukan proses *tokenization*:
   a. Ganti semua *unambiguous separators* [?!()";/|] dengan spasi.
   b. Berikan spasi pada tanda koma, tetapi jangan berikan spasi pada tanda koma apabila tanda koma tersebut berada di dalam angka. Hal ini perlu dilakukan karena mungkin saja tanda koma tersebut menunjukkan sebuah harga.
   c. Berikan spasi pada tanda titik, tetapi jangan berikan spasi pada tanda titik apabila tanda titik tersebut berada dalam kata gelar atau singkatan.
3) *Part-of-Speech* adalah proses yang digunakan untuk memberikan *tagset* pada sebuah kalimat. Pada penelitian ini, proses *part-of-speech* dilakukan dengan menggunakan bantuan *library* iPostagger.jar (*Indonesian Postagger*). *Library* ini adalah *library* yang dibuat pada penelitian Alfan Farizki Wicaksono dan Ayu Purwarianti (2010) [2].
4) *Chunking* adalah proses yang digunakan untuk mentransformasikan makna dari sebuah kalimat ke konteks yang berbeda, dapat menjadi konteks yang lebih besar (*chunk up*), lebih kecil (*chunk down*) atau serupa (*chunk laterally*) [3]. Proses *chunking* ini menggunakan skema *BILOU* (*Begin, Inside, Last, Outside, Union*) untuk mentransformasikan entitas *NER*.



Gambar 1. Alur kerja sistem



Gambar 2. Alur kerja *text preprocessing*

B. Feature Extraction

Feature Extraction adalah proses yang digunakan untuk mengektraksi atribut dari sebuah teks atau *data training* yang telah diolah. Satu kata dapat didefinisikan sebagai satu *feature* karena keberadaan suatu kata dapat mempengaruhi kata lainnya dan satu kata dapat memiliki arti yang berbeda. Hasil dari *feature extraction* ini digunakan pada proses selanjutnya untuk membuat sebuah pemodelan yang disebut *classifier*.

Untuk melakukan proses *feature extraction* ini dibutuhkan beberapa analisis. Analisis ini dilakukan dengan memperhatikan setiap kata penyusunnya karena setiap kata penyusunnya dapat memberikan pengaruh dalam menentukan sebuah entitas. Analisis-analisis atribut tersebut dapat terlihat pada Tabel I.

C. Classifier

Classifier adalah sebuah pemodelan yang berisi kumpulan klasifikasi atribut yang sudah dianalisis berdasarkan setiap kata penyusunnya. Pemodelan c*lassifier* ini dibuat dengan cara mengubah hasil *feature extraction* menjadi sebuah pemodelan yang dapat diproses oleh *machine learning*. *Classifier* inilah yang akan digunakan sebagai proses *learning* untuk mengenali pola *input user* pada saat proses *chat* dilakukan.

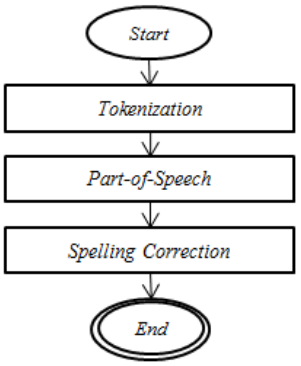
TABEL I
DAFTAR ANALISIS ATRIBUT

No	Atribut	Keterangan
1	$Word_{i-1}$	Atribut yang menunjukkan 1 kata sebelumnya.
2	$Word_{i-2}$	Atribut yang menunjukkan 2 kata sebelumya.
3	$Word_{i+1}$	Atribut yang menunjukkan 1 kata sesudahnya.
4	$Word_{i+2}$	Atribut yang menunjukkan 2 kata sesudahnya.
5	$Tagset_i$	Atribut yang menunjukkan *tagset* saat ini.
6	$Tagset_{i-1}$	Atribut yang menunjukkan 1 *tagset* sebelumnya.
7	$Tagset_{i-2}$	Atribut yang menunjukkan 2 *tagset* sebelumnya.
8	$Tagset_{i+1}$	Atribut yang menunjukkan 1 *tagset* sesudahnya.
9	$Tagset_{i+2}$	Atribut yang menunjukkan 2 *tagset* sesudahnya.
10	$Shape_i$	Atribut yang menunjukkan bentuk dari sebuah kata. Jenis *shape* yang digunakan adalah: d. *Uppercase*: bentuk kata *uppercase*. e. *Lowercase*: bentuk kata *lowercase*. f. *Mixedcase*: bentuk kata *uppercase* dan *lowercase*. g. *Capital*: bentuk kata yang huruf awalnya diikuti oleh huruf besar. h. *Number*: kata yang termasuk angka. i. *Numeric*: kata yang termasuk angka yang disertai dengan tanda titik atau koma.
11	*Bigram*	Atribut yang menunjukkan keberadaan 2 buah kata secara bersamaan.
12	*Keyword Title*	Atribut yang menunjukkan bahwa apakah suatu kata termasuk ke dalam *keyword title* atau tidak.
13	*Keyword Organization*	Atribut yang menunjukkan bahwa apakah suatu kata termasuk ke dalam *keyword organization* atau tidak.
14	$Entity_{i-1}$	Atribut yang menunjukkan 1 entitas sebelumnya.
15	$Entity_{i-2}$	Atribut yang menunjukkan 2 entitas sebelumnya.

Pada penelitian ini, sistem akan melakukan proses *learning* dengan menggunakan metode *Naive Bayes* [4]. Karena proses *Naive Bayes* tidak memerlukan sebuah pemodelan data dari hasil *feature extraction*, maka proses ini akan menghitung semua probabilitas suatu entitas dan menyimpannya ke dalam *hashmap* sehingga sistem tidak perlu lagi melakukan proses perhitungan secara terus-menerus.

D. Preprocessing

Proses ini dilakukan untuk mengolah sebuah *input* kalimat yang dimasukkan oleh *user* menjadi data yang mudah diolah oleh sistem *chatbot*. Langkah-langkah untuk melakukan proses *preprocessing* dapat terlihat pada Gambar 3 sebagai berikut.



Gambar 3. Alur kerja *preprocessing*

1) *Tokenization*, proses ini sama seperti saat proses *NER Learning*.
2) *Part-of-Speech*, proses ini sama seperti saat proses *NER Learning*.
3) *Spelling Correction* adalah proses yang digunakan untuk memperbaiki ejaan dari sebuah kata. Pengaruh dengan adanya proses *spelling correction* ini adalah ketika *user* melakukan kesalahan dalam memasukkan sebuah kata, maka sistem akan dapat memperbaiki kata yang salah tersebut sehingga kata tersebut dapat dikenali oleh sistem dalam proses selanjutnya.

Proses ini menggunakan metode *Lavenshtein Distance* sebagai proses untuk menghitung nilai kedekatan dari suatu kata. Proses ini mengharuskan system untuk memeriksa semua kata. Oleh karena itu, untuk mempercepat proses ini, maka dilakukan optimasi dengan menggunakan langkah-langkah sebagai berikut:

a. Lakukan pengecekan terhadap kata, apakah kata tersebut terdapat di dalam KBBI [5] atau tidak. Jika berhasil ditemukan, proses akan langsung mengembalikan kata tersebut dan proses selesai.
b. Jika tidak berhasil ditemukan, maka lakukan pemeriksaan pada *tagset* yang dimiliki oleh kata tersebut. Jika *tagset* dari kata tersebut tidak sesuai dengan *tagset* yang telah ditentukan (NN, VBI, VBT, NNC, NNP, IN dan WP), maka proses akan langsung mengembalikan kata tersebut dan proses selesai.
c. Jika *tagset* dari kata tersebut sesuai, maka selanjutnya lakukan proses pengecekan apakah kata tersebut termasuk ke dalam kamus data yang dimiliki oleh sistem atau tidak. Jika kata tersebut sesuai dengan kata di dalam kamus data, maka proses akan langsung mengembalikan kata tersebut dan proses selesai.
d. Jika kata tersebut tidak ditemukan dalam kamus data, maka barulah lakukan proses *spelling correction*. Proses ini akan melakukan pengecekan terhadap kata yang memiliki panjang lebih besar dan lebih kecil 2 huruf dari kata yang dimasukkan.

E. ntity Classification

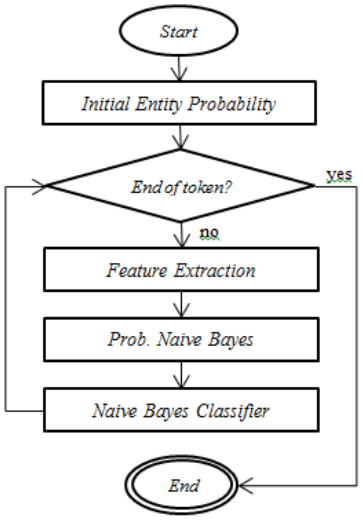
Entity classification adalah proses yang digunakan untuk memberikan entitas pada sebuah kata atau kalimat. Proses ini bertujuan untuk mengenali jenis entitas yang berguna untuk proses pengenalan pola (*pattern*) yang terkandung di dalam suatu *input*. Dalam melakukan proses *entity classification*, diperlukan sebuah *machine learning* untuk dapat menentukan sebuah entitas dari suatu kalimat.

Dalam prosesnya *machine learning* ini akan menggunakan metode *Naive Bayes*. Langkah-langkah sistem *chatbot* dalam menentukan suatu entitas *NER* dapat terlihat pada gambar 4 sebagai berikut.

1) Lakukan proses perhitungan probabilitas awal suatu entitas. Perhitungan ini dilakukan terhadap semua entitas yang dimiliki oleh sistem.
2) Lakukan pemeriksaan apakah token sudah selesai diperiksa semua atau tidak. Jika sudah habis, maka proses selesai. Jika belum habis, maka lakukan proses 3.
3) Lakukan proses *feature extraction* seperti pada proses *NER Learning*.

   Lakukan proses perhitungan probabilitas *Naive Bayes* dengan menggunakan data pada proses *learning* yang sudah disimpan ke dalam *hashmap*. Proses ini mengharuskan sistem untuk mengecek semua entitas. Oleh karena itu, dilakukan optimasi dengan menggunakan langkah-langkah sebagai berikut:

   a. Ambil dan lakukan pemeriksaan terhadap entitas sebelumnya (entityMin1).
   b. Jika entitas sebelumnya memiliki skema (o atau null) atau memiliki skema u (union) dan l (last), maka lakukan proses Naïve Bayes terhadap semua entitas.
   c. Jika entitas sebelumnya ada dan memiliki entitas dengan skema b (begin) atau i (inside), maka lakukan proses Naïve Bayes hanya pada entitas yang memiliki skema i (inside) dan l (last) saja.



Gambar 4. Alur kerja proses *entity classification*

4) Lakukan proses perhitungan *Naive Bayes Classifier* dengan menggunakan data hasil perhitungan *Naïve Bayes*.

F. Artificial Intelligence Markup Language

Entity classification adalah proses yang digunakan untuk memberikan entitas pada sebuah kata atau kalimat. Proses ini bertujuan untuk mengenali jenis entitas yang berguna untuk proses pengenalan pola (*pattern*) yang tekandung di dalam suatu *input*. Dalam melakukan proses *entity classification*, diperlukan sebuah *machine learning* untuk dapat menentukan sebuah entitas dari suatu kalimat.

AIML adalah proses utama sistem *chatbot* dalam memberikan respon kepada *user*. Dalam memberikan respon kepada *user*, proses ini menggunakan *knowledge based* sebagai dasar dalam melakukan percakapan.

Knowledge based dalam proses *AIML* adalah sekumpulan data yang digunakan untuk membuat sebuah *model* percakapan untuk proses *chat*. Data-data ini berisi sekumpulan skenario percakapan dan jawaban yang akan diberikan kepada *user*. Skenario ini dibuat berdasarkan *AIML language* yaitu berupa sebuah percakapan yang sudah diberi *tag-tag* tertentu. *Tag-tag* inilah yang akan digunakan sebagai sumber untuk memberikan sebuah respon atau jawaban kepada *user*. Contoh *knowledge based* yang digunakan pada proses *AIML* dapat terlihat pada tabel II.

Untuk dapat menemukan jawaban yang relevan sesuai dengan yang ditanyakan oleh *user*, maka diperlukan proses *pattern matching* untuk mencari *pattern* yang paling sesuai. Sebelum melakukan proses *pattern matching*, maka akan dilakukan optimasi dengan cara mengelompokkan *category* dari *knowledge based* berdasarkan panjang *keyword*-nya sehingga ketika proses *pattern matching* berlangsung, maka sistem hanya perlu melakukan proses pencocokan *pattern* berdasarkan jumlah *keyword* yang ditemukan. Dengan demikian, sistem tidak perlu lagi melakukan pencocokan terhadap semua *pattern* yang terdapat di dalam *knowledge based*. Berikut langkah-langkah optimasi proses *pattern matching*:

1) Ambil *category* berdasarkan panjang *keyword* yang ditemukan dalam *input*. Misalnya terdapat *keyword* “question:dimana, others:lokasi, organization:ithb”, berdasarkan data tersebut dapat terlihat adanya 3 *keyword* yang ditemukan, sehingga sistem akan mengambil semua data *category* yang mempunyai panjang *keyword* 3.
2) Lakukan proses pencocokan *pattern* berdasarkan *category* yang telah diambil. Proses pencocokan dilakukan dengan cara mencocokkan setiap *keyword* yang ada di dalam *pattern*. Misalnya “question:dimana, others:lokasi, organization:ithb” dengan *pattern* “others:lokasi, organization:ithb, question:dimana” memiliki persentase kecocokan 100%.

TABEL II
CONTOH *KNOWLEDGE BASED*

```
<aiml version="2.0" encoding="UTF-8"?>
    <category>
        <pattern>
            Question:DIMANA,Others:LOKASI,
            Organization:ITHB
        </pattern>
        <template>
            Lokasi Universitas ITHB berada
            di Jalan Dipatiukur 82-84,
            Bandung.
        </template>
    </category>
    <category>
        <pattern>
            Question:DIMANA,Others:LOKASI,
            Organization:INSTITUT TEKNOLOGI
            HARAPAN BANGSA
        </pattern>
        <template>
            <srai>DIMANA LOKASI ITHB</srai>
        </template>
    </category>
</aiml>
```

3) Jika hasil persentase kecocokan menunjukkan 100%, maka proses *pattern matching* selesai dan kemudian mengembalikan *category* tersebut.
4) Jika hasil persentase kecocokan tidak sama dengan 100%, maka proses akan dilanjutkan dengan menghitung kembali proses pencocokan *pattern* dengan menggunakan *pattern* yang lebih panjang 1 dan lebih pendek 1 dari *pattern* yang ditemukan pada *input*.
5) Berdasarkan pengecekan pada *pattern* dengan panjang yang sama, *pattern* yang lebih panjang 1 dan lebih pendek 1, maka proses *pattern matching* akan mengembalikan *category* dengan nilai persentase kecocokan yang terbesar dengan syarat bahwa nilai kecocokan tersebut lebih dari 75%.

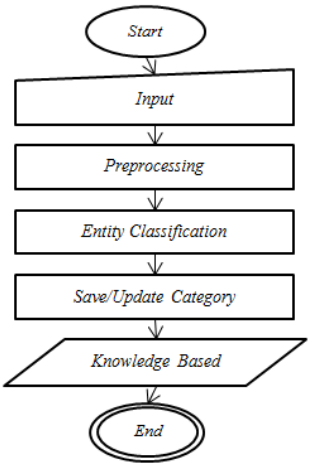
G. Answer Extraction

Answer Extraction adalah proses yang digunakan untuk mengekstraksi jawaban yang akan diberikan kepada *user*. Secara umum, proses ini hanya membaca *tag* <template> dari *tag* <category> yang didapatkan. Apabila *tag* <template> ini mengandung *tag* lainnya, maka sistem akan membaca *tag-tag* tersebut. Apabila semua *tag* telah selesai diproses, maka sistem akan memberikan *output* kepada *user* dengan menggabungkan semua *output tag* yang telah dihasilkan tersebut.

III. IMPLEMENTASI REVERSE AIML

Proses *reverse AIML* [6] merupakan proses yang digunakan untuk mengubah *chat* ke dalam *language AIML*, kemudian menyimpannya ke dalam *knowledge based* sistem *chatbot*. Langkah-langkah sistem dalam melakukan proses Reverse AIML dapat terlihat pada gambar 5.

Dengan adanya proses ini, maka sistem dapat bertambah pintar seiring dengan bertambahnya pengetahuan baru. Proses *Reverse AIML* ini dapat dijelaskan dengan langkah-langkah sebagai berikut:



Gambar 5. Alur kerja proses *reverse AIML*

1) Lakukan proses *preprocessing* pada *input* berupa sebuah kalimat yang akan dilakukan proses *reverse AIML*.
2) Lakukan proses *entity classification* menggunakan data yang sudah dilakukan proses *preprocessing*.
3) Lakukan proses *chunking* (*chunk up*) untuk menemukan *keywords* apa saja yang ditemukan.
4) Berdasarkan *keywords* yang telah ditemukan, lakukan proses pengecekan apakah *keywords* tersebut sudah ada di dalam *knowledge based* atau tidak.
5) Jika sudah ada di dalam *knowledge based*, maka proses hanya akan melakukan *update* ke dalam *knowledge based*.
6) Jika tidak ditemukan, maka proses akan langsung memasukkan *category* tersebut ke dalam *knowledge based* sistem.

IV. PENGUJIAN

Tujuan dilakukannya pengujian terhadap aplikasi ini, di antaranya adalah sebagai berikut :

1) Untuk menentukan konfigurasi *feature* dan *entity* yang terbaik dalam proses *NER*, sehingga sistem dapat bekerja dengan cepat dan memiliki akurasi yang baik.
2) Untuk menentukan konstanta kecocokan yang terbaik dan apakah sistem *chatbot* sudah bekerja dengan baik dalam memberikan jawaban yang tepat.

A. Pengujian NER

Dalam melakukan pengujian *NER* terdapat 2 jenis pengujian yang dilakukan, yaitu sebagai berikut:

1) *Self testing* adalah pengujian yang dilakukan dengan menggunakan semua *data training* sebagai *data testing*, kemudian mengujinya dengan *data training* itu sendiri.
2) Pengujian sebagian adalah pengujian yang dilakukan dengan cara menggunakan 20% dari *data training* sebagai *data testing* (persentase ini dapat dipilih secara bebas tergantung dari kebutuhan pengujian,

dalam pengujian ini digunakan 20%), kemudian mengujinya dengan menggunakan 80% sisanya.

Pengujian ini dilakukan dengan menggunakan *data training* yang terdiri dari kurang lebih 2500 kalimat. Pada Tabel III sampai Tabel VI dapat dilihat hasil pengujian yang dilakukan.

TABEL III
DAFTAR KONFIGURASI *FEATURE*

Jenis Konfigurasi	Jenis *Feature* yang Digunakan
Featureset 1	Menggunakan semua *feature* yang ada.
Featureset 2	$Word_i$, $Word_{i-1}$, $Word_{i+1}$, $Tagset_i$, $Tagset_{i-1}$, $Tagset_{i+1}$, $Shape_i$, *Bigram,Title Keyword, Title Organization,* $Entity_{i-1}$
Featureset 3	$Word_i$, $Word_{i-1}$, $Word_{i+1}$, $Shape_i$, *Bigram, Title Keyword, Title Organization,* $Entity_{i-1}$
Featureset 4	$Word_i$, $Word_{i-1}$, $Word_{i+1}$, $Shape_i$, *Bigram, Title Keyword, Title Organization*
Featureset 5	$Word_i$, $Word_{i-1}$, $Word_{i+1}$, *Bigram, Title Keyword*

TABEL IV
DAFTAR KONFIGURASI *ENTITY*

Jenis Konfigurasi	Jenis *Entity* yang Digunakan
Entity 1	Pengecekan yang dilakukan terhadap semua entitas.
Entity 2	Pengecekan yang dilakukan berdasarkan skema. (optimasi yang dilakukan)

TABEL V
KESIMPULAN SELF TESTING

Konfigurasi		Akurasi (%)			Time (detik)
Feature	Entity	Precision	Recall	F1	
Featureset 1	*Entity* 1	97.29	90.13	93.16	8.18
Featureset 1	*Entity* 2	**97.35**	**90.40**	**93.36**	**7.13**
Featureset 2	*Entity* 1	**97.47**	93.01	95.02	5.92
Featureset 2	*Entity* 2	97.28	**93.56**	**95.24**	**5.26**
Featureset 3	*Entity* 1	**97.78**	93.86	95.63	4.72
Featureset 3	*Entity* 2	97.73	**94.56**	**96.03**	**4.08**
Featureset 4	*Entity* 1	**98.16**	94.23	96.03	4.29
Featureset 4	*Entity* 2	98.08	**95.77**	**96.83**	**3.77**
Featureset 5	*Entity* 1	97.98	96.56	97.21	3.47
Featureset 5	*Entity* 2	**98.20**	**97.67**	**97.91**	**3.09**

TABEL VI
KESIMPULAN PADA PENGUJIAN MENGGUNAKAN 20% DATA *TRAINING* SEBAGAI DATA *TESTING*

Konfigurasi		Akurasi (%)			Time (detik)
Feature	Entity	Precision	Recall	F1	
Featureset 1	*Entity* 1	**96.18**	82.60	87.38	1.87
Featureset 1	*Entity* 2	96.13	**82.73**	**87.40**	**1.72**
Featureset 2	*Entity* 1	**96.47**	86.19	90.02	1.47
Featureset 2	*Entity* 2	96.31	**87.10**	**90.45**	**1.36**
Featureset 3	*Entity* 1	**97.31**	**89.90**	**92.93**	1.23
Featureset 3	*Entity* 2	96.89	87.96	91.18	**1.14**
Featureset 4	*Entity* 1	**97.64**	91.58	94.20	1.16
Featureset 4	*Entity* 2	97.50	**93.54**	**95.24**	**1.07**
Featureset 5	*Entity* 1	**97.59**	95.50	96.49	1.03
Featureset 5	*Entity* 2	97.38	**96.98**	**97.14**	**0.94**

Berdasarkan hasil pengujian yang dilakukan terhadap proses *NER*, dapat disimpulkan bahwa konfigurasi dengan *Featureset* 5 merupakan konfigurasi yang terbaik, baik dalam pengujian *self testing* dan pengujian yang menggunakan sebagian *data training*. Berdasarkan pengujian tersebut juga dapat terlihat bahwa konfigurasi dengan menggunakan *Entity* 2 memiliki hasil akurasi yang juga tinggi dan tidak berbeda jauh dengan konfigurasi *Entity* 1, namun konfigurasi dengan *Entity* 2 memiliki waktu proses yang lebih cepat dibandingkan dengan konfigurasi dari *Entity* 1. Jadi, berdasarkan hasil pengujian tersebut, maka proses *NER* pada sistem *chatbot* ini akan menggunakan konfigurasi *Featureset* 5 dan *Entity* 2.

B. Pengujian Chatbot

Pengujian ini dilakukan dengan cara memperhatikan sistem *chatbot* dalam memberikan setiap jawaban atau respon ketika diberikan sebuah *input* oleh *user*. Pengujian ini dilakukan dengan 2 tujuan yaitu untuk mengetahui konstanta kecocokan yang terbaik ketika proses *pattern matching* sedang berlangsung dan untuk menentukan apakah sistem *chatbot* sudah dapat memberikan jawaban atau respon dengan tepat atau tidak.

Berdasarkan hasil pengujian yang dilakukan terhadap proses *chatbot*, dapat disimpulkan bahwa secara garis besar sistem *chatbot* sudah dapat memberikan jawaban yang relevan/tepat sesuai dengan apa yang telah dijawab dan ditanyakan oleh *user* yaitu:

1. Konstanta 50% = benar 64 dari 75 pertanyaan = 85.33%.
2. Konstanta 75% = benar 72 dari 75 pertanyaan = 96%.

Terjadinya kesalahan dalam memberikan jawaban kepada *user* disebabkan karena sistem saat ini tidak memiliki *knowledge based* dan *data training* yang cukup banyak. Tetapi *knowledge based* ini dapat bertambah dengan adanya proses *Reverse AIML* sehingga sistem *chatbot* dapat bertambah pintar seiring dengan adanya penambahan pengetahuan yang baru.

Berdasarkan hasil pengujian tersebut, dapat disimpulkan juga bahwa konstanta kecocokan yang terbaik adalah 75%. Konstanta 50% dan 75% sama-sama dapat memberikan jawaban dengan tepat, namun perbedaannya adalah pada saat sistem melakukan kesalahan dalam mengenali *keyword* karena tidak memiliki *data training*, konstanta 50% masih dapat memberikan jawaban karena konstanta kecocokannya hanya 50%. Ini berarti bahwa jawaban tersebut bisa berarti 50% benar dan 50% salah. Oleh karena itu, daripada menampilkan jawaban yang salah, maka akan lebih baik jika sistem memberikan jawaban *default* seperti yang terlihat pada sistem jika diberikan konstanta 75%.

V. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian yang telah dilakukan dalam membuat sistem *chatbot*, terdapat beberapa kesimpulan yang dapat diambil, yaitu sebagai berikut:

1) Berdasarkan hasil pengujian proses *Named Entity Recognition* (*NER*), dapat disimpulkan bahwa sistem dapat mengenali berbagai pola kalimat dengan nilai

akurasi yang mencapai lebih dari 97%. Hal ini dipengaruhi karena adanya proses *preprocessing* untuk membantu mengenali *feature-feature* apa saja yang terdapat di dalam suatu kalimat. Berdasarkan hasil pengujian, konfigurasi *featureset* yang terbaik adalah dengan menggunakan *feature*: $Word_i$, $Word_{i-1}$, $Word_{i+1}$, *Bigram*, *Title Keyword*.

2) Berdasarkan hasil pengujian *NER*, dapat disimpulkan juga bahwa pengecekan entitas berdasarkan pada skemanya dapat memberikan akurasi yang tinggi dan tidak berbeda jauh dari pengecekan yang dilakukan terhadap semua entitas, namun memiliki waktu proses yang jauh lebih cepat.
3) Berdasarkan hasil pengujian terhadap sistem *chatbot*, dapat disimpulkan bahwa secara garis besar sistem sudah dapat memberikan jawaban atau respon yang relevan sesuai dengan apa yang sedang dibicarakan oleh *user*. Hasil terbaik yaitu dengan menggunakan konstanta kemiripan atau kecocokan 75% yang menghasilkan akurasi sebesar 96%.
4) Pengujian ini menggunakan data *AIML* yang dibuat secara *manual* berdasarkan data *prospectus* ITHB 2015, berdasarkan hasil pengujian *chatbot* tersebut, dapat disimpulkan juga bahwa *knowledge based* sistem *chatbot* saat ini dirasa sudah cukup baik.
5) *Knowledge based* dari sistem *chatbot* dapat bertambah atau berubah, tanpa harus mengubahnya secara *manual* karena adanya proses *Reverse AIML* yang membantu mengubah *chat* menjadi *AIML language*.

Berdasarkan pengujian yang telah dilakukan, proses perhitungan *Naïve Bayes* akan menjadi sangat lama apabila sistem menggunakan data yang sangat besar. Hal ini terjadi karena proses pencarian nilai probabilitas ke dalam data yang sangat besar sehingga prosesnya menjadi lebih lama. Oleh karena itu, untuk mempercepat proses ini, maka sistem akan menghitung semua kemungkinan probabilitas dan menyimpannya ke dalam sebuah *hashmap* sehingga ketika sistem melakukan proses *Naïve Bayes*, maka sistem hanya akan mengambil nilai yang sudah disediakan. Dan berdasarkan penelitian terbukti bahwa sistem dapat menjadi lebih cepat.

Berdasarkan hasil penelitian yang telah dilakukan dalam membuat sistem *chatbot*, terdapat beberapa saran yang dapat dilakukan, yaitu sebagai berikut:

1) Proses *Reverse AIML* masih menggunakan proses yang sederhana yaitu mengubah sebuah *chat* menjadi *AIML Language* sehingga tidak ada pengetahuan mengenai unsur percakapan yang dapat disimpan. Berdasarkan masalah tersebut, maka disarankan supaya pengembangan selanjutnya sistem *reverse AIML* dapat mengenali semua unsur percakapan dari kumpulan beberapa *chat* yang dilakukan kemudian menyimpannya ke dalam *know-ledge based*. Untuk dapat mengenali semua unsur percakapan tersebut, maka diperlukan proses semantik untuk menarik sebuah kesimpulan dalam memahami sekumpulan kalimat percakapan.
2) Penggunaan *machine learning Naïve Bayes* memang dapat memberikan hasil akurasi yang cukup tinggi, namun dalam penelitian ini penggunaan *Naïve Bayes* pada proses *entity classification* dirasa kurang efektif karena *Naïve Bayes* hanya menghitung probabilitas setiap *feature* saja, *Naïve Bayes* tidak dapat menghitung probabilitas hubungan antar *feature* sehingga tingkat kepentingan antar *feature* seperti tidak terlihat. Oleh karena itu, disarankan untuk menggunakan *machine learning* lainnya seperti *Bayes Network* yang menghitung probabilitas hubungan antar *feature*.
3) *Knowledge based* sebaiknya dibuat dengan lebih lengkap lagi untuk beberapa istilah yang memiliki arti yang sama karena sistem saat ini tidak memiliki proses *synonym* yang mengubah kata yang bermakna yang sama menjadi kata yang benar. Misalnya:
    a. "profil lulusan harapan bangsa" bisa saja ditulis dengan kalimat "profil lulusan ithb", "profil lulusan harapan", "profil lulusan bangsa".
    b. "Ir. Inge Martina, M.T." bisa saja ditulis dengan kalimat "Ibu Inge".

Oleh karena itu, *knowledge based* sebaiknya memiliki semua kombinasi yang mungkin dari *keyword* tersebut sehingga sistem dapat memberikan jawaban dengan lebih baik. Solusi lain untuk mengatasi masalah ini adalah selanjutnya bisa ditambahkan proses *synonym* untuk menangani masalah-masalah tersebut atau dengan menggunakan pengecekan kedekatan sebuah *keyword* seperti "profil lulusan harapan bangsa" dengan "profil lulusan bangsa" memiliki persentase kedekatan sebesar 75%.

DAFTAR REFERENSI

[1] I. Doumanis and S. Smith. *Beyond Artificial Intelligence Markup Language (AIML) Rapid Prototyping of a Q&A system*. Middlesex University, The Burroughs Hendon, London NW4 4BT, 2013.

[2] A. F. Wicaksono, A. Purwarianti. "HMM based part-of-speech tagger for Bahasa Indonesia," in *Proceeding of the Fourth International MALINDO Workshop*, 2010.

[3] T. Nudd BA MSc. 2010. *Introduction To Improving Your Communication Through Chunking Information*. http://www.chunkmaps.com/nlp/ introduction. [21 Maret 2015]

[4] D. Xhemali, Christopher J. Hinde and Roger G. Stone. 2009. "Naive Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages," in *IJCSI International Journal of Computer Science Issues*, Vol. 4, No. 1, 2009.

[5] Kamus Besar Bahasa Indonesia Online. http://pusatbahasa.diknas.go.id [10 Juni 2015]

[6] Charlix. 2010. "Reverse AIML". Internet: http://www.charlix.sourceforge.net. [7 Februari 2015]

David Christianto, mahasiswa jurusan Teknik Informatika di Institut Teknologi Harapan Bangsa yang lulus pada tahun 2015.

Elisafina Siswanto, lahir di Bandung pada tahun 1989, menerima gelar Sarjana Teknik dari Institut Teknologi Harapan Bangsa pada tahun 2011 jurusan Teknik Informatika, dan menyelesaikan pendidikan Magister Informatika di Institut Teknologi Bandung pada tahun 2014 Saat ini aktif sebagai pengajar di Departement Teknik Informatika, Institut Teknologi Harapan Bangsa di Bandung. Minat penelitian adalah pada bidang Pembelajaran Mesin dan Pemrosesan Bahasa Alami.

Ria Chaniago, Sarjana Teknik Informatika dengan fokus bidang Kecerdasan Buatan. Menyelesaikan pendidikan S1 di Institut Teknologi Harapan Bangsa pada tahun 2013. Saat ini sedang melanjutkan pendidikan Magister Teknik Informatika di Institut Teknologi Bandung (sejak tahun 2014). Memiliki minat dan pengalaman bekerja sebagai peneliti dibidang Mesin Pembelajaran dan Pemrosesan Bahasa Alami.