BAB II

LANDASAN TEORI

2.1 Chatterbot

Chatterbot atau *chatbot* merupakan program yang bertujuan untuk mensimulasikan percakapan antara manusia dengan komputer seolah seperti percakapan manusia dengan manusia. *Chatbot* menerima *input user* berupa bahasa alami (*Natural Language*) dan memberikan respon yang tepat sesuai dengan pengetahuan atau *knowledge base* yang dimiliki program tersebut (Chantarotwong, 2006).

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 (Weizenbaum, 1966). 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 (2005) mengatakan bahwa program *chatbot* terdiri dari dua komponen utama, yaitu *bot program* dan *brain file*. *Bot program* merupakan program utama pada *chatbot* yang akan mengakses input dari pengguna, melakukan *parsing* dan kemudian membawanya ke *brain file* untuk kemudian diberikan respon. *Bot program* terdiri dari *parser* dan *scanner*. *Brain file* merupakan otak dari *chatbot* itu sendiri yang menentukan bagaimana cara *chatbot* berpikir dan akan memberikan respon. Semakin banyak pengetahuan yang dimiliki *chatbot* maka akan semakin besar ukuran file dari *brain file* tersebut.

Shawar dan Atwell (2007) mengatakan bahwa *chatbot* dapat berfungsi sebagai *entertainment* atau media hiburan, alat bantu pembelajaran bahasa, alat bantu pengambilan informasi (*information retrieval*), dan asisten dalam *e-commerce*, bisnis, dan bidang lainnya.

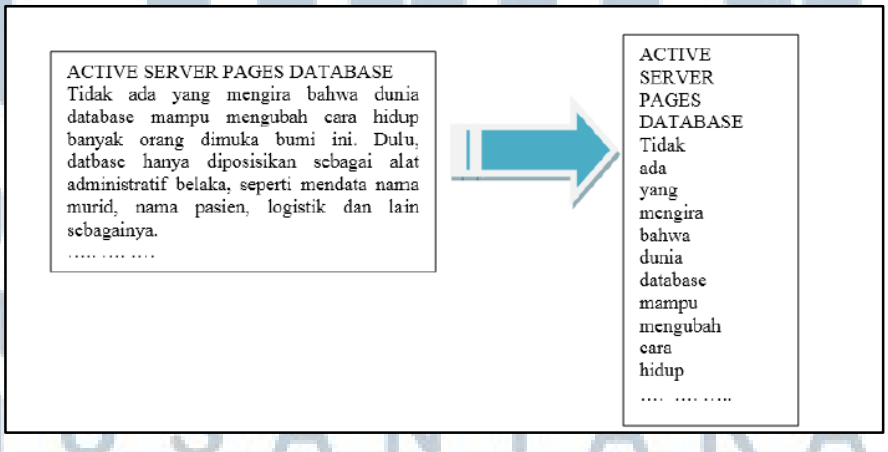
2.2 Natural Language Processing (NLP)

Natural Language merupakan bahasa yang digunakan oleh manusia untuk berkomunikasi general. *Natural Language Processing* merupakan prosedur yang bertujuan untuk membuat komputer dapat mengerti bahasa atau kata yang diucapkan atau ditulis oleh manusia (Chopra, Prashar, dan Sain, 2013).

Langkah-langkah *Natural Language Processing* (NLP) adalah sebagai berikut.

1. Tokenization

Tokenization atau *segmentation* merupakan tahap pemecahan paragraph menjadi kalimat atau kalimat menjadi kata. Dalam konsep *Natural Language Processing*, hasil dari pemecahan disebut sebagai *token*. *Token* ini akan digunakan untuk proses komputasi selanjutnya (Indurkhya dan Damerau, 2010). Contoh proses *tokenization* ditunjukkan pada Gambar 2.1.

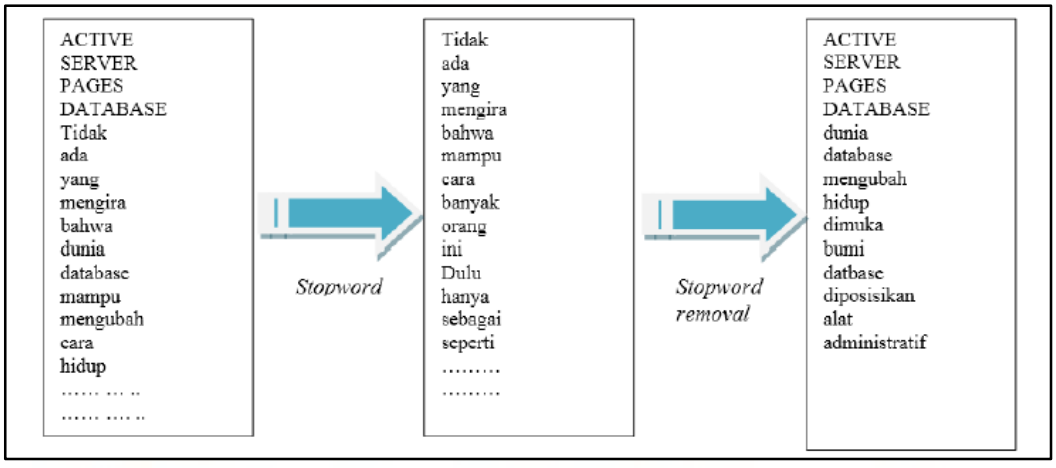


Gambar 2.1 Contoh Tahap Tokenization (Indriyono dkk., 2015)

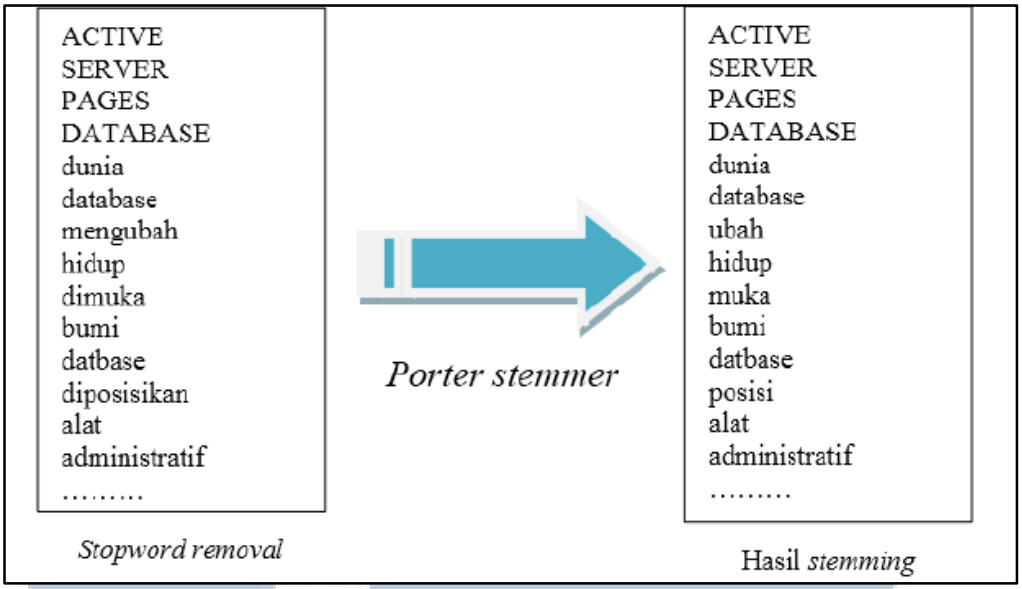
2. Lexical Analysis (Scanning)

Scanner merupakan salah satu bagian dari komponen *chatbot* yang berfungsi menjalankan analisis leksikal (Rudiyanto, 2005). *Lexical Analysis* merupakan tahap identifikasi dan analisis struktur dari kata per kata. Analisis menghasilkan *mapping* antara kata dengan *dataset* atau *dictionary* kata tersebut (Indurkhya dan Damerau, 2010).

Sebelum analisis dilakukan, proses *stopword removal* dan *stemming* dilakukan. Pada proses *stopword removal*, sistem akan membuang kata-kata yang tidak berguna dari *token* yang dihasilkan oleh tahap *tokenizing* (Indriyono dkk., 2015). Algoritma Porter Stemmer digunakan pada kata-kata yang akan dibuang imbuhannya sehingga kata akan diproses dalam bentuk kata dasar. Contoh pembuangan kata ditunjukkan pada Gambar 2.2 dan proses *stemming* ditunjukkan pada Gambar 2.3.



Gambar 2.2 Contoh Stopword Removal (Indriyono dkk., 2015)



Gambar 2.3 Stemming Token (Indriyono dkk., 2015)

3. Syntactic Analysis

Syntactic Analysis merupakan tahap analisa hubungan antara kata-kata di dalam suatu kalimat untuk menggambarkan struktur dari kalimat tersebut. Dalam bahasa Inggris terdapat aturan *grammar* yang dianalisa pada tahap ini. Kata ditransformasi ke dalam suatu struktur untuk mengetahui hubungan antar kata (Chopra, Prashar, dan Sain, 2013).

4. Semantic Analysis

Semantic Analysis merupakan tahap analisa pengertian dari kalimat berdasarkan hubungan antara objek-objek sintaktik. *Semantic Analysis* mendukung konsep *Ontologies* pada *Natural Language Processing* (Indurkhya dan Damerau, 2010).

5. Pragmatic Analysis

Pragmatic Analysis merupakan tahap analisis pengertian dari kalimat yang sebenarnya berdasarkan konteks situasi kalimat tersebut. Pada tahap ini pengertian dari kalimat akan diinterpretasikan oleh komputer sebagai pengertian yang

sebenarnya (Chopra, Prashar, dan Sain, 2013). Pada penelitian ini, *tag* AIML <that> dan <topic> dapat digunakan untuk mendefinisikan konteks pembicaraan yang sedang dilakukan.

2.3 Artificial Intelligence Markup Language (AIML)

Artificial Intelligence Markup Language merupakan turunan dari *markup language* XML berbasis *tag element* yang bertujuan untuk mempermudah permodelan dialog komunikasi berdasarkan paradigma *stimulus-response*. AIML mendefinisikan data objek yang disebut objek AIML yang digunakan untuk memodelkan pola *input* dan *output* dari suatu percakapan (Marietto dkk., 2013). AIML merepresentasikan *knowledge* yang digunakan oleh *chatbot* berdasarkan pengembangan teknologi *bot* ALICE (Kader, 2015).

```
<command> ListOfParameters </command>
```

Gambar 2.4 Sintaks Dasar AIML (Marietto dkk., 2013)

Sintaks dasar dari objek AIML ditunjukkan pada Gambar 2.4. Menurut Marietto (2013) terdapat tiga tag utama dalam objek AIML yang perlu diperhatikan, yaitu *category* yang mendefinisikan satu unit *knowledge* dari suatu percakapan, *pattern* yang mengidentifikasi *input* dari *user*, dan *template* yang digunakan untuk mencatat respon yang akan diberikan berdasarkan *input* tertentu dari *user*. *Pattern* ditulis dalam huruf kapital untuk standarisasi dan simplifikasi proses *pattern matching* dari *input user*.

Tag yang digunakan pada AIML dijelaskan sebagai berikut.

1. Tag <aiml>

Setiap *file* AIML dimulai dengan *tag* <aiml> dan diakhiri dengan *closing tag* </aiml>. *Tag* ini mengandung atribut *version* dan *encoding*. Atribut *version* mengindentifikasikan versi AIML yang digunakan pada *knowledge base chatbot*. Apabila atribut ini tidak didefinisikan, program tidak akan menghasilkan *error*, tetapi dapat menimbulkan masalah ketika proses *maintenance* atau *upgrade*. Atribut *encoding* digunakan untuk mendefinisikan tipe *encoding* karakter yang digunakan pada dokumen AIML ini. Pada contoh yang ditunjukkan pada Gambar 2.5, atribut *version* yang digunakan adalah 1.0.1 dan *encoding* UTF-8 (Marietto dkk., 2013).

```
<aiml version="1.0.1" encoding="UTF-8"?>
<category>
   <pattern> HELLO BOT </pattern>
   <template>
      Hello my new friend!
   </template>
</category>
</aiml>
```

Gambar 2.5 Penggunaan Tag <aiml> (Marietto dkk., 2013)

2. Tag <category>

Unit dasar dari dialog AIML adalah *category*. *Category* terdiri dari tiga bagian, yaitu *user input* dalam bentuk kalimat, respon dari *chatbot* terhadap *user input*, dan konteks opsional. *Knowledge base* yang dibuat dengan AIML merupakan kumpulan dari *category*. *Category* diorganisir dalam bentuk *file* ".aiml". *Category* dimulai dengan *tag* <category> dan diakhiri *closing tag* </category>. *Tag* <category></category> harus berada di dalam *tag* <aiml></aiml> dan memiliki *tag*

pattern dan *template* di dalamnya. Gambar 2.6 menunjukkan contoh penggunaan *tag category*.

```
<category>
<pattern>MOTHER</pattern>
<template> Tell me more about your family. </template>
</category>
```

Gambar 2.6 Penggunaan AIML Tag <category>, <pattern>, dan <template> (Wallace, 2003)

3. Tag <pattern>

Tag ini berisi *input user* yang mungkin atau pola kalimat dari *input user* yang telah melalui proses dengan NLP. *Tag* <pattern> hanya didefinisikan satu kali pada setiap *tag* <category> dan harus menjadi elemen pertama yang didefinisikan. Kata-kata di dalam *tag* <pattern> dipisahkan dengan satu spasi dan penggunaan *wildcards* dapat mensubtitusi bagian dari kalimat tersebut. Pada contoh yang ditunjukkan oleh Gambar 2.5, *tag* <pattern> berisi kalimat "HELLO BOT" yang merupakan kalimat *input* dari *user*. Kalimat ditulis dalam huruf kapital untuk menyederhanakan proses *matching* (Marietto dkk., 2013).

4. Tag <template>

Tag <template> berisi respon yang diberikan oleh *chatbot* untuk menjawab kalimat pada *tag* <pattern>. *Tag* ini dapat menyimpan data, mengaktifkan program lain, dan memberikan jawaban kondisional atau jawaban dari *category* lain (Marrietto dkk., 2016). Pada Gambar 2.5, respon dari *input* "HELLO BOT" adalah "Hello my new friend!".

5. Tag <star index = ”n”>

Tag ini menyimpan dan memetakan komponen bagian dari kalimat pada *user input*. Indeks “n” mengindikasikan komponen yang dipetakan oleh *star index*. *Tag* ini merupakan konteks opsional pada *category*. Untuk memetakan *input user* pada *tag pattern*, digunakan simbol atau *wildcard* “*” (Marietto dkk., 2013). Gambar 2.7 menunjukkan contoh penggunaan *tag* <star index = “n”>. Apabila *user input* adalah kalimat “A rose is a flower”, maka respon yang diberikan oleh *chatbot* adalah “When a rose is not a flower?”.

```
<category>
   <pattern> I LIKE * </pattern>
   <template>
      I like <star/> too.
   </template>
</category>

<category>
   <pattern> A * IS A * </pattern>
   <template>
      When a <star index="1"/> is not a <star index="2"/>?
   </template>
</category>
```

Gambar 2.7 Penggunaan Tag <star index = “n”> (Marietto dkk., 2013)

6. Tag <srai>

Salah satu properti yang sangat berguna dalam AIML adalah kemampuan untuk menargetkan *pattern input* yang berbeda untuk respon pada *tag template*. Dengan demikian, *interpreter* AIML dapat mencari jawaban dari *input user* yang berbeda. Hal ini dilakukan dengan menggunakan *tag* <srai> (Marietto dkk., 2013).

Pada Gambar 2.8 terdapat dua kategori yang membicarakan Alan Turing (baris 1-8) dan Albert Sabin (baris 10-16). *Pattern* pada kategori Alan Turing dan Albert Sabin menggunakan kalimat “WHO IS ALAN TURING?” dan “WHO IS ALBERT SABIN?”. Namun, ada kemungkinan *user* bertanya dengan kalimat yang berbeda,

sebagai contoh “DO YOU KNOW WHO ALAN TURING IS?” dan mengharapkan respon yang sama. Kategori baru dapat dibuat untuk menangani *input* tersebut dengan *pattern* “DO YOU KNOW WHO * IS?”. *Wildcard* “*” menyatakan nama orang yang ditanyakan. Respon dibuat dengan “<srai> WHO IS <star/> </srai>” yang melakukan *redirect pattern* ke kategori yang lain.

```
<category>
   <pattern> WHO IS ALAN TURING? </pattern>
   <template>
      Alan Turing was a British mathematician, cryptographer,
      and computer scientist often credited as
      the founder of modern Computer Science.
   </template>
</category>

<category>
   <pattern> WHO IS ALBERT SABIN? </pattern>
   <template>
      Albert Sabin was the researcher who developed
      the vaccine that is the main defense against polio.
   </template>
</category>

<category>
   <pattern> DO YOU KNOW WHO * IS? </pattern>
   <template>
      <srai> WHO IS <star/> </srai>
   </template>
</category>
```

Gambar 2.8 Tag <srai> Untuk Symbolic Reduction (Marietto dkk., 2013)

```
<category>
   <pattern> BYE </patter>
      <template> Goodbye friend! </template>
</category>

<category>
   <pattern> BYE * </pattern>
   <template> <srai> BYE </srai> </template>
</category>
```

Gambar 2.9 Tag <srai> Divide and Conquer (Marietto dkk., 2013)

Pada Gambar 2.9 ditunjukkan penggunaan *tag* <srai> untuk konsep *divide and conquer*. Sebagai contoh kalimat “BYE *” mengharapkan respon yang sama

dengan kalimat “BYE”, maka *tag* <srai> dapat digunakan sehingga apabila kalimat *input* “BYE RICO” atau karakter apapun setelah kata “BYE” akan mendapatkan respon “Goodbye friend!”.

```
category>
   <pattern> INDUSTRY </pattern>
   <template>
      It is a development center.
   </template>
</category>

<category>
   <pattern> FACTORY </pattern>
   <template>
      <srai> INDUSTRY </srai>
   </template>
</category>
```

Gambar 2.10 Tag <srai> Untuk Sinonim (Marietto dkk., 2013)

Sebagai contoh, kategori pertama pada Gambar 2.10 menerima *input* “INDUSTRY” dengan respon “It is a development center.”. Kata “FACTORY” merupakan sinonim dari “INDUSTRY” sehingga apabila *user input* berupa kata “FACTORY”, maka respon yang diharapkan adalah “It is a development center.”. *Tag* <srai> dapat digunakan untuk memanggil respon dari “INDUSTRY” sehingga *pattern* “FACTORY” akan direspon dengan respon pada *category* “INDUSTRY”.

Tag <srai> juga dapat digunakan untuk mendeteksi kata kunci tertentu seperti yang ditunjukkan pada Gambar 2.11. Penggunaan *wildcard* “_” dan “*” menandakan kalimat terdiri dari kata dan salah satu diantaranya adalah kata “FAMILY”. *Tag* <srai> digunakan untuk menargetkan respon “Family is an important institution.” untuk setiap kalimat yang mengandung kata kunci “FAMILY”.

```
<category>
   <pattern> FAMILY </pattern>
   <template>
      Family is an important institution.
   </template>
</category>

<category>
   <pattern> _ FAMILY </pattern>
   <template>
      <srai> FAMILY </srai>
   </template>
</category>

<category>
   <pattern> FAMILY * </pattern>
   <template>
      <srai> FAMILY </srai>
   </template>
</category>

<category>
    <pattern> _ FAMILY * </pattern>
    <template>
        <srai> FAMILY </srai>
    </template>
</category>
```

Gambar 2.11 Tag <srai> Untuk Deteksi Keyword (Marietto dkk., 2013)

7. Tag <random> dan

Tag <random> digunakan untuk memberikan respon dengan cara yang berbeda. Setiap kalimat respon yang mungkin ditulis dalam *tag* . *Interpreter* AIML membaca respon sebagai *list* dan memilih secara acak salah satu kalimat respon dalam *list* (Marietto dkk., 2013). Gambar 2.12 menunjukkan contoh penggunaan *tag* <random> dan <li>.

```
<category>
   <pattern> HI </pattern>
   <template>
      <random>
         <li> Hi! Nice to meet you </li>
         <li> Hello, How are you? </li>
         <li> Hello! </li>
      </random>
   </template>
</category>
```

Gambar 2.12 Tag <random> dan <li> (Marietto dkk., 2013)

8. Tag <set> dan <get>

Tag <set> dan <get> memungkinkan *chatterbot* bekerja dengan menggunakan variabel. *Tag* <set> digunakan untuk menyimpan *value* ke dalam variabel. *Tag* ini harus berada di dalam *tag* <template>. Contoh penggunaan *tag* <set> ditunjukkan pada Gambar 2.13. *Value* yang direpresentasikan dengan *wildcard* "*" disimpan ke dalam variabel bernama "nameUser".

```
<category>
   <pattern> MY NAME IS * </pattern>
   <template>
      Hello <set name="nameUser"> <star/> </set>
   </template>
</category>
```

Gambar 2.13 Tag <set> (Marietto dkk., 2013)

```
<category>
   <pattern> GOOD NIGHT </pattern>
   <template>
      Good night <get name="nameUser"/>
   </template>
</category>
```

Gambar 2.14 Tag <get> (Marietto dkk., 2013)

Value yang tersimpan di dalam variabel dapat diambil dengan menggunakan *tag* <get>. Pada Gambar 2.14, variabel "nameUser" yang telah di-*set* pada Gambar 9 digunakan pada kalimat respon dengan *tag* "<get name="nameUser">". Apabila variabel "nameUser" berisi "RICO", maka kalimat respon dari "GOOD NIGHT" yaitu "GOOD NIGHT RICO".

9. Tag <that>

```
User: Make some question
Bot: Do you like movies?
User: No
Bot: OK. But I like movies.
```

Gambar 2.15 Skenario Percakapan (Marietto dkk., 2013)

```
<category>
   <pattern> MAKE SOME QUESTION </pattern>
   <template>
      Do you like movies?
   </template>
</category>

<category>
   <pattern> YES </pattern>
   <that> Do you like movies? </that>
   <template>
      Nice, I like movies too.
   </template>
</category>

<category>
   <pattern> NO </pattern>
   <that> Do you like movies? </that>
   <template>
      OK. But I like movies.
   </template>
</category>
```

Gambar 2.16 Penggunaan Tag <that> (Marietto dkk., 2013)

Tag <that> memungkinkan sistem untuk menganalisis kalimat respon *chatbot* sebelumnya. Percakapan dapat berjalan sesuai dengan konteks yang sedang dibicarakan. *Tag* ini harus berada di dalam *tag* <category> (Marietto dkk., 2013). Gambar 2.15 menunjukkan sebuah skenario percakapan dan Gambar 2.16 menunjukkan penggunaan *tag* <that> pada percakapan tersebut. *User* memberikan *input* "NO". Kemudian, Sistem akan melihat kalimat respon sebelumnya, yaitu "Do you like movies?" dan memberikan respon "OK. But I like movies".

10. Tag <topic>

Tag <topic> digunakan untuk menyatakan topik atau subjek pembicaraan *chatbot*. Topik ini mengelompokkan *tag category* sehingga mempercepat proses pencarian respon (Marietto dkk., 2013). Contoh penggunaan *tag* <topic> ditunjukkan pada Gambar 2.17. Pada pembicaraan "LET TALK ABOUT FLOWER", *chatbot* akan mengisi variabel *topic* dengan *flowers*. Pembicaraan

selanjutnya dengan *input* “I LIKE IT SO MUCH” yang diterima, *chatbot* akan merespon dengan “I like flowers too” berdasarkan variabel *topic* yang telah didefinisikan.

```
<category>
   <pattern> LET TALK ABOUT FLOWERS. </pattern>
   <template>
      Yes <set name="topic">flowers</set>
   </template>
</category>

<topic name="flowers">
   <category>
      <pattern> * </pattern>
      <template>
         Flowers have a nice smell.
      </template>
   </category>

   <category>
      <pattern> I LIKE IT SO MUCH! </pattern>
      <template>
         I like flowers too.
      </template>
   </category>
</topic>
```

Gambar 2.17 Penggunaan Tag <topic> (Marietto dkk., 2013)

11. Tag <think>

```
<category>
   <pattern> MY NAME IS * </pattern>
   <template>
      <think> <set name="nameUser"> * </set> </think>
</template>
</category>
```

Gambar 2.18 Penggunaan Tag <think> (Marietto dkk., 2013)

Tag <think> digunakan untuk memproses data, *conditional statements* dan *tests* yang tidak ditampilkan kepada *user*. Isi dari *tag* ini diproses oleh *chatbot* tanpa memperlihatkan proses tersebut kepada *user* (Marietto dkk., 2013). Pada Gambar 2.18, *chatbot* mengisi variabel “nameUser” dan *user* tidak mengetahui proses tersebut.

12. Tag <condition>

Tag <condition> digunakan ketika ada lebih dari satu respon yang dapat diberikan kepada *user* dan pemilihan respon tersebut dilakukan berdasarkan analisis *value* dari suatu variabel yang berubah seiring dengan berjalannya proses *chatting*. *Tag* <condition> dapat dikatakan ekuivalen dengan *command* "Case" pada bahasa pemrograman (Marietto dkk., 2013). Gambar 2.19 menunjukkan contoh penggunaan *tag* <condition>. Atribut "name=" berisi nama variabel yang dianalisis dan "value=" berisi *value* yang dibandingkan dengan *value* variabel tersebut.

```
<category>
   <pattern> HOW ARE YOU? </pattern>
   <template>
      <condition name="state" value="happy">
         It is nice being happy.
      </condition>
      <condition name="state" value="sad">
         Being sad is not nice.
      </condition>
   </template>
</category>
```

Gambar 2.19 Penggunaan Tag <condition> (Marietto dkk., 2013)

13. Tag <bot>

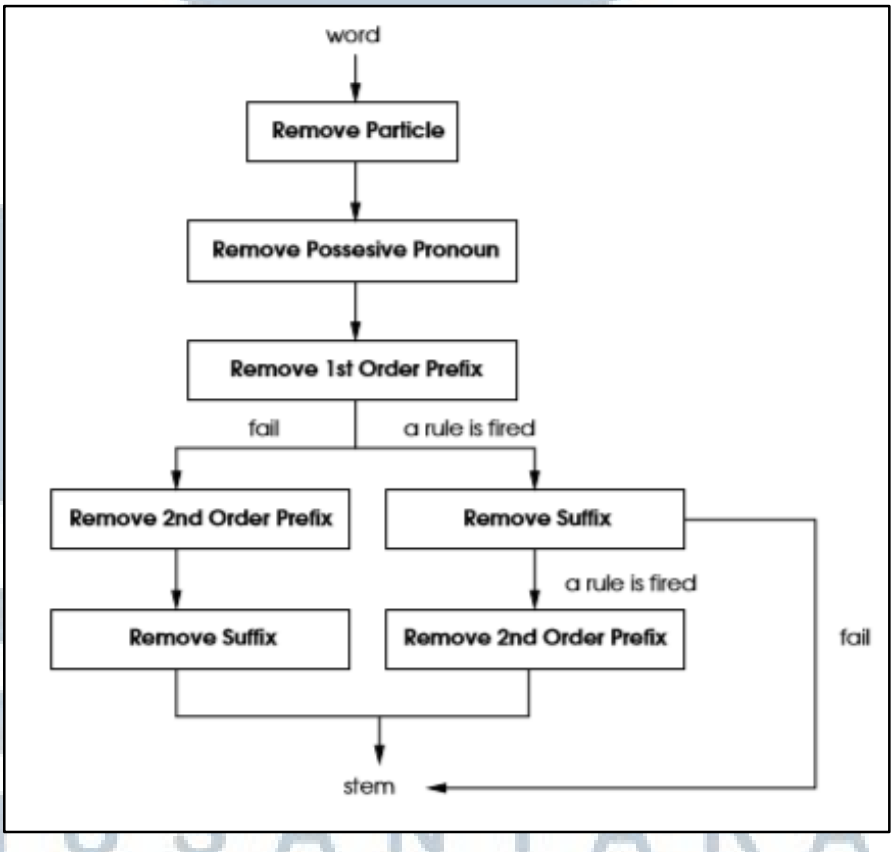
```
<category>
   <pattern> BOT'S PROPERTIES </pattern>
   <template>
      <bot name="age"/>
      <bot name="gender"/>
      <bot name="location"/>
      <bot name="nationality"/>
      <bot name="birthday"/>
      <bot name="sign"/>
      <bot name="botmaster"/>
   </template>
</category>
```

Gambar 2.20 Penggunaan Tag <bot> (Marietto dkk., 2013)

Tag <bot> digunakan untuk mendefinisikan properti *chatbot* dan properti ini dapat dilihat oleh *user* saat percakapan berlangsung (Marietto dkk., 2013). Pada Gambar 2.20 ditunjukkan contoh penggunaan *tag* <bot>. Properti yang didefinisikan dalam contoh di atas adalah "age", "gender", "location", "nationalitiy", "birthday", "sign", dan "botmaster".

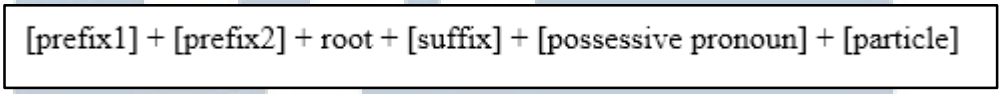
2.4 Porter Stemmer

Stemming adalah proses pemetaan dan penguraian berbagai bentuk (*variants*) dari suatu kata menjadi bentuk kata dasarnya (Porter, 1980). Proses ini disebut juga sebagai *conflation* (Gupta, 2014). Salah satu keuntungan menggunakan *stemming* adalah efisiensi dan indeks *file* yang sudah terkompresi. Dengan proses *stemming*, setiap kata yang memiliki akar kata yang sama masih dapat disamakan. Tanpa proses *stemming*, kata "stemmed", "stem", dan "stemming" merupakan kata yang berbeda (Wiguna dan Hartono, 2013). Menurut Tala (2003), tujuan dari proses *stemming* adalah menghilangkan imbuhan-imbuhan baik itu berupa prefiks, sufiks, maupun konfiks yang ada pada setiap kata. Pada penelitian ini, *stemmer* Tala digunakan untuk mengubah kalimat *user input* ke dalam bentuk *token* yang berisi kata dasar (*root*).



Gambar 2.21 Desain Porter Stemmer Untuk Bahasa Indonesia (Tala, 2003)

Fadillah Z Tala (2003) melakukan penelitian mengenai pengaruh proses *stemming* pada temu kembali informasi (*information retrieval*) dalam Bahasa Indonesia. Gambar 2.21 menampilkan model rancangan algoritma Porter Stemmer untuk digunakan pada Bahasa Indonesia. Algoritma ini disebut juga sebagai *stemmer* Tala. Tala merancang algoritma ini berdasarkan struktur kata pada Bahasa Indonesia yang dirumuskan dengan rumusan pada Gambar 2.22 berikut.



Gambar 2.22. Struktur Kata Bahasa Indonesia (Tala, 2003)

Indriyono, Utami, dan Sunyoto (2015) memaparkan langkah-langkah yang dilakukan dalam algoritma *stemmer* Tala dalam penelitian klasifikasi jenis buku. Wiguna dan Hartono (2013) juga memaparkan langkah-langkah penggunaan algoritma *stemmer* Tala. Langkah-langkah algoritma *stemmer* Tala dijabarkan sebagai berikut.

1. Menghapus partikel seperti: -kah, -lah, -tah.
2. Menghapus kata ganti (*Possessive Pronoun*) seperti: -ku, -mu, -nya.
3. Menghapus awalan pertama. Jika tidak ditemukan, maka lanjut ke langkah *remove 2nd order prefix* (4a), dan jika ada maka lanjut ke langkah *remove suffix* (4b).
4. a. Menghapus awalan kedua, dan dilanjutkan pada langkah ke *remove suffix* (5a).

   b. Menghapus akhiran, jika tidak ditemukan maka kata tersebut diasumsikan sebagai kata dasar (*root word*). Jika ditemukan maka lanjut ke langkah *remove 2nd order prefix* (5b).

5. a. Menghapus akhiran dan kata akhir diasumsikan sebagai kata dasar (*root word*).

   b. Menghapus awalan kedua dan kata akhir diasumsikan sebagai kata dasar (*root word*).

Terdapat lima aturan pada algoritma *stemmer* Tala. Aturan-aturan tersebut dijabarkan pada Tabel 2.1, Tabel 2.2, Tabel 2.3, Tabel 2.4, dan Tabel 2.5 (Tala, 2003).

Tabel 2.1 Kelompok Rule Pertama: *Inflectional Particles*

Suffix	***Replacement***	***Additional Condition***	***Examples***
-kah	NULL	NULL	Apakah ➔ Apa
-lah	NULL	NULL	Adalah ➔ Ada
-pun	NULL	NULL	Bukupun ➔ Buku

Tabel 2.2 Kelompok Rule Kedua: *Inflectional Possessive Pronouns*

Suffix	***Replacement***	***Additional Condition***	***Examples***
-ku	NULL	NULL	Bukuku ➔ Buku
-mu	NULL	NULL	Bukumu ➔ Buku
-nya	NULL	NULL	Bukunya ➔ Buku

Tabel 2.3 Kelompok Rule Ketiga: *First Order of Derivational Prefixes*

Prefix	***Replacement***	***Additional Condition***	***Examples***
meng-	NULL	NULL	Mengukur ➔ ukur
meny-	S	V…	Menyapu ➔ sapu
men-	NULL	NULL	Menduga ➔ duga
mem-	P	V…	Memilah ➔ pilah
mem-	NULL	NULL	Membaca ➔ baca
me-	NULL	NULL	Merusak ➔ rusak
peng-	NULL	NULL	Pengukur ➔ ukur
peny-	S	V…	Penyapu ➔ sapu
pen-	NULL	NULL	Penduga ➔ duga
pem-	P	V…	Pemilah ➔ pilah
pem-	NULL	NULL	Pembaca ➔ baca
di-	NULL	NULL	Diukur ➔ ukur

Tabel 2.4 Kelompok Rule Keempat: *Second Order of Derivational Prefixes*

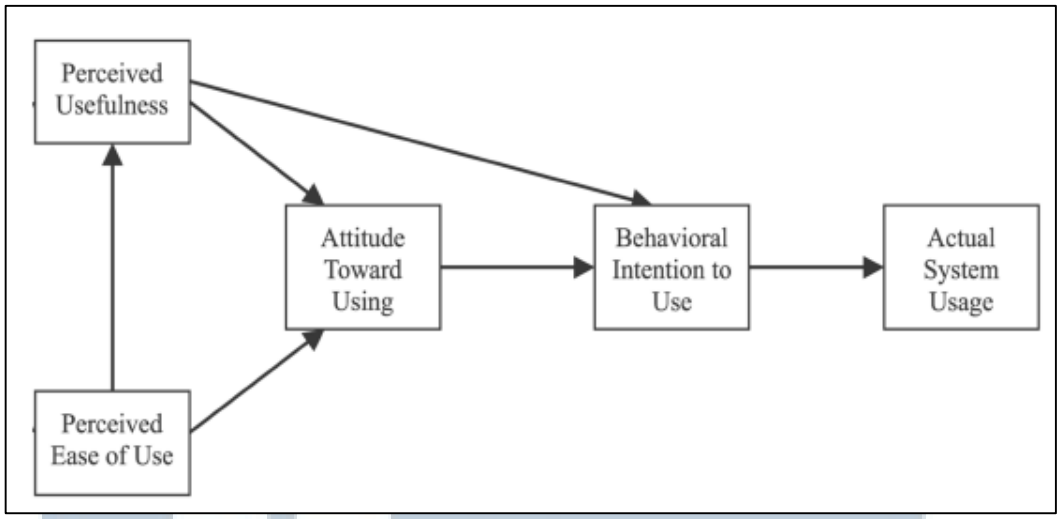
Prefix	***Replacement***	***Additional Condition***	***Examples***
ber-	NULL	NULL	Berlari ➔ lari
bel-	NULL	Ajar	Belajar ➔ ajar
be-	NULL	K*er	Bekerja ➔ kerja
per-	NULL	NULL	Perjelas ➔ jelas
pel-	NULL	Ajar	Pelajar ➔ ajar
pe-	NULL	NULL	Pekerja ➔ kerja

Tabel 2.5 Kelompok Rule Kelima: *Derivational Suffixes*

Suffix	***Replacement***	***Additional Condition***	***Examples***
-kan	NULL	*Prefix* ∉{ke, peng}	Tarikkan ➔ tarik (Meng)ambilkan ➔ ambil
-an	NULL	*Prefix* ∉{di, meng, ter}	Makanan ➔ makan (Per)janjian ➔ janji
-i	NULL	V|K...c1c1, c1≠i,c2≠i *and* Prefix ∉{ber, ke, peng}	Tandai ➔ tanda (Men)dapati ➔ dapat

2.5 Technology Acceptance Model (TAM)

Technology Acceptance Model (TAM) merupakan *model* evaluasi sistem yang digunakan untuk mengetahui apakah *user* menerima dan berkehendak untuk menggunakan teknologi baru (Davis, 1989). TAM menggunakan 2 (dua) variabel utama sebagai faktor yang mempengaruhi *user*, yaitu *perceived ease of use* dan *perceived usefulness*. *Perceived ease of use* merupakan tingkat kepercayaan *user* bahwa sistem dapat dipahami dan digunakan tanpa membutuhkan banyak usaha (*effort*) untuk mempelajarinya (Davis, 1989). *Perceived usefulness* merupakan tingkat kepercayaan *user* bahwa penggunaan sistem dapat meningkatkan kinerja (Davis, 1989). *Ease of use* dan *usefulness* merupakan faktor penting dalam proses evaluasi kualitas dari *online service* (Rod dkk., 2009). TAM telah banyak dipelajari dan dikembangan (Sharma dkk., 2014). TAM dapat dilihat pada Gambar 2.23.



Gambar 2.23 Technology Acceptance Model (Davis, 1989)

Beberapa penerapan metode modifikasi TAM (*Extended Technology Acceptance Model*) berhubungan dengan *trust* dan *user satisfaction* telah dilakukan oleh Al-Gahtani (2011), Shawar dkk. (2014), dan Aufar (2009). Menurut Zeithaml (2002), *user satisfaction* merupakan tingkat atau tolak ukur dari *product service* apakah *product* atau *service* tersebut telah memenuhi kebutuhan dan ekspektasi *user*. *Trust* merupakan tingkat kepercayaan *user* untuk menggunakan sistem dengan mempertimbangkan karakteristik sistem (Al-Gahtani, 2011).

2.6 Skala Likert

Menurut Djaali (2008), Skala Likert merupakan skala yang digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang suatu gejala atau fenomena pendidikan. Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam kuesioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei. Skala ini menggunakan pertanyaan positif untuk mengukur skala positif dan pertanyaan negative untuk mengukur skala

negatif. Jawaban dari skala Likert adalah Sangat Setuju (SS), Setuju (S), Netral (N), Tidak Setuju (TS), dan Sangat Tidak Setuju (STS).

Tabel 2.6 Interval Skor 5 (Lima) Tingkat Skala Likert

Pernyataan	Skor	Bobot Interval
Sangat Setuju	5	Skor >= 80%
Setuju	4	80% > Skor >= 60%
Netral / Biasa Saja	3	60% > Skor >= 40%
Tidak Setuju	2	40% > Skor >= 20%
Sangat Tidak Setuju	1	Skor < 20%

Persentasi skor pada suatu kuesioner dihitung berdasarkan rumus yang dijelaskan oleh Sugiyono (2012).

Persentase Skor = (((Sangat Setuju * 5) + (Setuju * 4) + (Netral * 3) + (Tidak Setuju * 2) + (Sangat Tidak Setuju * 1)) / (5 * Jumlah Responden)) * 100% …(2.1)

2.7 Validitas dan Reliabilitas Kuesioner

Validitas adalah ukuran yang menunjukkan sejauh mana instrument pengukur mampu mengukur apa yang ingin diukur (Purbayu dan Ashari, 2005). Uji validitas dilakukan untuk memastikan seberapa baik suatu instrumen digunakan untuk mengukur konsep yang seharusnya diukur. Menurut Sugiyono (2010) untuk menguji validitas konstruk dilakukan dengan cara mengkorelasikan antara skor butir pertanyaan dengan skor totalnya. *Corelation Product Moment* adalah rumus yang digunakan untuk menguji validitas dari sebuah kuesioner dan hasil perhitungan akan dibandingkan dengan r-tabel (Arikunto, 2010).

$$r_{xy} = \frac{N \sum XY - (\sum X)(\sum Y)}{\sqrt{\{N \sum X^2 - (\sum X)^2\}\{N \sum Y^2 - (\sum Y)^2\}}} \quad \ldots(2.2)$$

Keterangan:

r_{xy} = koefisien korelasi suatu pertanyaan

N = jumlah subyek

X = skor suatu pertanyaan

Y = skor total

Menurut Arikunto (2006), reliabilitas menunjuk pada suatu pengertian bahwa suatu instrumen dapat dipercaya atau diandalkan untuk digunakan sebagai alat pengumpul data karena instrument tersebut baik. *Alpha Cronbach* adalah rumus yang digunakan untuk menguji reliabilitas dari suatu instrument.

Nilai reliabilitas di atas 0,90 merupakan nilai untuk reliabilitas sempurna. Jika nilai reliabilitas di antara 0,70 – 0,90, nilai tersebut menunjukkan reliabilitas tinggi. Nilai reliabilitas di antara 0,50 – 0,70 menunjukan nilai reliabilitas moderat atau sedang. Nilai reliabilitas di bawah 0,50 menunjukkan reliabilitas yang rendah (Riskawati, 2013).

$$r_{11} = \left[\frac{k}{(k-1)}\right]\left[1 - \frac{\sum Si}{St}\right] \quad \ldots(2.3)$$

Keterangan:

r_{11} = nilai realibilitas

Σ Si = jumlah varians skor setiap pertanyaan

St = varians total

k = jumlah pertanyaan