Hak cipta dan penggunaan kembali:

Lisensi ini mengizinkan setiap orang untuk menggubah, memperbaiki, dan membuat ciptaan turunan bukan untuk kepentingan komersial, selama anda mencantumkan nama penulis dan melisensikan ciptaan turunan dengan syarat yang serupa dengan ciptaan asli.

Copyright and reuse:

This license lets you remix, tweak, and build upon work non-commercially, as long as you credit the origin creator and license it on your new creations under the identical terms.

BAB II

LANDASAN TEORI

2.1 Requirements Engineering

Requirements engineering merupakan bagian dari tahapan *software engineering* sebelum dimulainya tahap perancangan dan tahap selanjutnya bertujuan untuk mengumpulkan, mengerti, menganalisa serta mendokumentasikan permasalahan yang ada (Pressman, 2010; Sommerville, 2010). Para *software engineer* harus paham apa permasalahan yang ada, mengapa masalah tersebut harus selesaikan serta siapa saja yang terlibat (Lamsweerde, 2009).

Terdapat beberapa basis kegiatan dalam RE yaitu *elicitation, elaboration, negotiation, specification, validation, dan management* (Pressman, 2010).

2.1.1 Requirements Elicitation

Requirements elicitation disebut juga *requirements gathering* dan merupakan salah satu aktivitas dari *requirement engineering*. *Elicitation* berhubungan erat dengan para *stakeholder,* pada aktivitas ini akan diidentifikasi masalah atau kelemahan dari sistem yang telah ada dan akan dilakukan pemecahan masalah yaitu dengan merumuskan asumsi kebutuhan-kebutuhan atau layanan dari sistem yang nanti akan dibangun (Lamsweerde, 2009; Pressman, 2010).

Maka dari itu pertama yang dilakukan adalah mengidentifikasi siapa *stakeholder* yang berkaitan, karena keterlibatan *stakeholder* berpengaruh pada sistem atau *software* yang nantinya akan dibuat (Pressman, 2010) dimana

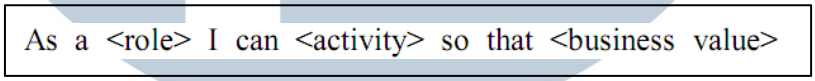
Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

stakeholder atau orang yang terlibat bermacam-macam dan asumsi dari setiap orang berbeda-beda (Sommerville, 2010).

2.1.2 User stories

User stories termasuk *agile development method* yang pertama kali dikenalkan pada *extreme programming* untuk mengumpulkan *requirements* pengguna yang dibutuhkan oleh sistem (O'hEocha and Conboy, 2010) dan menjadi kunci dalam proses pengembangan *software* (Rees, 2002). *User stories* menekankan interaksi dengan pengguna atau *customer* (Lamsweerde, 2009) dan mereka akan mengekspresikan fungsionalitas-fungsionalitas yang menurut mereka dibutuhkan dalam sistem dalam bentuk *story* (O'hEocha and Conboy, 2010).

Contoh bentuk dari *user story* (Zeaaraoui et al., 2013):



Gambar 2.1 Contoh Bentuk *User story* (Zeaaraoui et al., 2013)

a. <role> pada Gambar 2.1 merepresentasikan tipe atau siapa penggunanya (Zeaaraoui et al., 2013).
b. <activity> pada Gambar 2.1 merepresentasikan kebutuhan dari sistem (Zeaaraoui et al., 2013).
c. <business value> pada Gambar 2.1 merepresentasikan alasan dari <activity> baik nilai dari bisnis maupun alasan lainnya (Zeaaraoui et al., 2013).

2.2 Chatbot

Chatbot merupakan suatu teknologi atau *agent* yang dirancang melalui pemrograman komputer untuk dapat membuat mesin berinteraksi dengan manusia menggunakan *natural language processing* sehingga membuat orang tersebut

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

seolah-olah berinteraksi dengan manusia (Lokman et al., 2009; Lokman and Zain, 2010).

Kemampuan yang dimiliki *chatbot* selayaknya manusia adalah *chatbot* dapat memahami *input* percakapan yang dimasukkan manusia ke dalam program *chatbot* tersebut (Abdul-Kader and John, 2015). Masukkan pengguna berupa bahasa natural yang nantinya akan diproses dan dicocokkan *pattern*-nya dengan *knowledge* yang telah dimasukkan *developer* ke *brain file* sebelumnya hingga *chatbot* akan membalas percakapan tersebut (Dahiya, 2017; Lokman et al., 2009). Selama percakapan terjadi maka proses ini akan terus berulang (Abdul-Kader and John, 2015).

Knowledge based dibangun dan disimpan pada *brain file (*AIML *file*) oleh *developer* sehingga *chatbot* yang baik sebenarnya memiliki *database* yang besar yang menampung semua kemungkinan percakapan yang kan diajukan oleh pengguna sehingga *chatbot* tersebut dapat merespon semua percakapan tersebut dengan jawaban yang masuk akal dan berkaitan dengan percakapan yang terjadi (Abdul-Kader and John, 2015).

Menurut Samera dan John berdasarkan penelitian sebelumnya, *chatbot* terdiri dari 3 bagian yaitu:

a. Responder

Merupakan bagian dimana yang mengatur masukan dan keluaran dari *chatbot* tersebut dan melakukan pen-*transfer*-an data dari masukkan pengguna ke *classifier.*

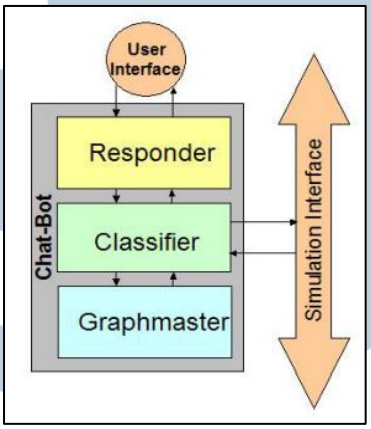
UNIVERSITAS MULTIMEDIA NUSANTARA

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

b. Classifier

Merupakan suatu bagian yang melakukan pemrosesan pada masukkan pengguna yang telah ditransfer melalui *responder* tadi. Pemrosesan *input* dilakukan sedemikian rupa baik dengan *tokenizing, filtering* dan sebagainya sehingga nantinya dapat dilempar ke *graphmaster* dan dilakukan pencocokan dari hasil normalisasi *input* tadi dengan *knowledge* yang telah ada (AIML).

c. Graphmaster



Gambar 2.2 Komponen *Chatbot* (Stoner et al., 2004)

Pada tahap ini akan dilakukan proses pencocokan dari *input* yang telah dinormalisasi pada tahap *classifier*. Pencocokan dilakukan dengan mencocokkan *pattern* dari *brain file* yang telah dimasukkan atau diprogram sebelumnya oleh *developer*.

2.3 Artificial Intellegence Markup Language (AIML)

AIML merupakan turunan dari XML yang digunakan pada A.L.I.C.E (Wallace, 2003) dan memiliki data objek bernama AIML *object*. AIML *object* berfungsi untuk mengatur *flow (pattern)* dari percakapan atau *behavior* percakapan *chatbot* tersebut (Marietto et al., 2013; Wallace, 2003).

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

Unit dasar dari pengetahuan pada AIML *object* adalah kategori (<category>) dan setiap kategori terdiri dari masukan pengguna (<pattern>) maupun keluaran sistem (<template>) dan konteks opsional (<that>, <topic>) (Wallace, 2003). Akan tetapi topik merupakan elemen opsional yang bersifat *high-level* dan memiliki nama serta kategori yang berkaitan dengan topik tersebut (Abu Shawar and Atwell, 2007).

2.3.1 Tipe-Tipe AIML Categories

Menurut Shawar dan Atwell (2007) terdapat 3 kategori AIML**:**

a. *Atomic Categories*

Pada *atomic categories pattern* tidak memiliki simbol *wildcard* dan *underscore* seperti yang telihat pada Gambar 2.3 (Abu Shawar and Atwell, 2007).

```
<category>
  <pattern>10 Dollars</pattern>
  <template>Wow, that is cheap. </template>
</category>
```

Gambar 2.3 Contoh *Atomic Categories* (Abu Shawar and Atwell, 2007)

b. *Default Categories*

```
<category>
  <pattern>10 *</pattern>
  <template>It is ten.</template>
</category>
```

Gambar 2.4 Contoh *Default Categories* (Abu Shawar and Atwell, 2007)

Pada *default categories*, *pattern* memiliki simbol *wildcard* atau *underscore* seperti pada Gambar 2.4 *Wildcard* menunjukkan bahwa *pattern* akan cocok dengan *input* apapun (Abu Shawar and Atwell, 2007).

c. *Recursive Categories*

Pada *Recursive Categories*, terdapat *tag* <srai> dan <sr> (*symbol reduction*) di dalam *tag* <templates>. Terdapat banyak deskripsi dari <srai> yaitu

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

symbol reduction untuk mengurangi kompleksitas dari bentuk *grammar* ke bentuk yang lebih sederhana, *divide and conquer* untuk memisahkan *input* menjadi 2 atau lebih bagian kecil dan mengkombinasikan *response* <template> dengan masing-masing bagian kecil tadi, dan untuk mendeteksi kata kunci pada masukan pengguna (Abu Shawar and Atwell, 2007).

Beberapa contoh <srai>:

- Contoh penggunaan <srai> *Symbolic Reduction* dapat dilihat pada Gambar 2.5.

```
<category>
  <pattern>DO YOU KNOW WHAT THE * IS</pattern>
  <template>
    <srai>What is <star/></srai>
  </template>
</category>
```

Gambar 2.5 Contoh *Symbolic Reduction* (Abu Shawar and Atwell, 2007)

- Contoh penggunaan <srai> *Divide and Conquer* dapat dilihat pada Gambar 2.6.

```
<category>
  <pattern>YES*</pattern>
  <template>
    <srai>YES</srai>
    <sr/>
  <template>
</category>
```

Gambar 2.6 Contoh *Divide and Conquer* (Abu Shawar and Atwell, 2007)

- Contoh penggunaan <srai> pada sinonim dapat dilihat pada Gambar 2.7

```
<category>
  <pattern>HALO</pattern>
  <template>
    <srai>Hello</srai>
  </template>
</category>
```

Gambar 2.7 Contoh Sinonim (Abu Shawar and Atwell, 2007)

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

2.3.2 Language's Tag

Terdapat beberapa *tag* pada AIML (Marietto et al., 2013):

a. *Tag* <aiml>

Tag ini digunakan ketika ini memulai AIML *file* untuk pertama kalinya (Marietto et al., 2013). Contoh *tag* ini dapat dilihat pada Gambar 2.8.

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

Gambar 2.8 Contoh *Tag* <aiml> (Marietto et al., 2013)

b. *Tag* <category>

Tag ini merupakan unit dasar dalam AIML dialog. Di dalam *tag* <category> terdapat masukan pengguna <pattern>, keluaran *bot* <templates> dan konteks tambahan <that> <topic> (Marietto et al., 2013).

```
<category>
   <pattern> HELLO BOT </pattern>
   <template>
      Hello my new friend!
   </template>
</category>
```

Gambar 2.9 Contoh Tag <category> (Marietto et al., 2013)

c. *Tag* <pattern>

Tag ini berisi kemungkinan masukan yang diberikan oleh pengguna. Untuk setiap <category> hanya terdapat 1 <pattern> yang terletak pada elemen awal setelah *tag* <category>(Marietto et al., 2013) seperti pada Gambar 2.9 dan Gambar 2.10.

UNIVERSITAS MULTIMEDIA NUSANTARA

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

d. *Tag* <template>

Tag ini berisi keluaran *bot* yang mungkin dan sesuai dengan masukan

```
<aiml version="1.0">
<topic name="the topic">
<category>
<pattern>PATTERN</pattern>
<that>THAT</that>
<template>Template</template>
</category>
        ..
        ..
</topic>
</aiml>
```

Gambar 2.10 Contoh *Tag* <category> dengan konteks opsional

(Marietto et al., 2013)

pengguna. *Tag* ini juga harus berada di dalam <category> dan letaknya setelah <pattern> (Marietto et al., 2013) seperti yang terlihat pada gambar 2.9 dan gambar 2.10.

e. *Tag* <star index = "n"/>

Tag ini berguna untuk mengambil *fragment text* tertentu yang berada pada masukan pengguna. Indeks n di dalam *tag* tersebut menunjukkan frase komponen (*wildcard* dari masukkan pengguna) (TutorialsPoint, n.d.) yang akan dipetakan dan diambil. *Tag* </star> bisa menggantikan <star index="1"/>, karena jika n diabaikan maka akan diasumsikan nilainya 1 (Marietto et al., 2013).

f. *Tag* <random> dan *tag*

Tag <random> ini digunakan untuk memberikan tanggapan dengan cara yang berbeda terhadap masukan dari pengguna, sedangkan *tag* digunakan untuk sebagai pembatas tiap tanggapan, hal ini dapat dilihat pada Gambar 2.11. Tanggapan dipilih secara acak oleh AIML *interpreter*.

```
<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 2.11 Contoh <random> dan <li> (Marietto et al., 2013)

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

g. *Tag* <set> dan *tag* <get>

Kedua *tag* ini berguna agar *variable* dapat digunakan untuk data pada *chatbot. Tag* <set> berguna untuk menetapkan nilai variabelnya.

Contoh *syntax* (Marietto et al., 2013):

<set name = “variableName”> VariableValue</set>

Sedangkan <get> berguna untuk mengambil nilai dari variabel yang telah ditetapkan dengan *tag* <set>.

Contoh *syntax* (Marietto et al., 2013):

<get name =”variableName”>

h. *Tag* <that>

Tag ini merupakan *tag* opsional yang ada di dalam <category>. *Tag* ini digunakan untuk analisis *pragmatic* yaitu untuk menganalisa kalimat terakhir dan bagaimana meresponnya (Marietto et al., 2013). Contoh dapat dilihat pada Gambar 2.12

```
<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 2.12 Contoh <that> (Marietto et al., 2013)

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

i. *Tag* <topic>

Tag ini digunakan untuk menentukan topik percakapan yang akan dilakukan oleh *chatbot* tersebut. Hal ini membuat waktu pencarian *pattern* menjadi lebih singkat karena telah diberikan batasan pencarian pada <category> berupa topik yang masuk melalui masukan dari pengguna (Marietto et al., 2013; TutorialsPoint, n.d.). Contoh dapat dilihat pada Gambar 2.13.

```
<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 2.13 Contoh <topic> (Marietto et al., 2013)

j. *Tag* <think>

Tag ini digunakan pada *chatbot* untuk melakukan pemrosesan *data*, *conditional statement*, atau untuk *testing*, yang dimana proses ini tidak akan terlihat oleh pengguna. Contoh dapat dilihat pada Gambar 2.14.

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

Gambar 2.14 Contoh <think> (Marietto et al., 2013)

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

```
<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 2.15 Contoh <condition> (Marietto et al., 2013)

k. *Tag* <condition>

Tag ini digunakan untuk memberikan respon sesuai dengan analisa dari percakapan yang terjadi antara pengguna dan *chatbot* (Marietto et al., 2013).

Contoh *Tag* ini dapat dilihat pada Gambar 2.15.

```
<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 2.16 Contoh <bot> (Marietto et al., 2013)

l. *Tag* <*bot*>

Tag ini berguna untuk konfigurasi *properties* dari *chatbot* dan dapat dilihat oleh pengguna. Contoh *Tag* ini dapat dilihat pada Gambar 2.16.

2.4 Natural Language Processing

NLP merupakan sistem komputer yang bertujuan untuk mengerti, menganalisa dan mengubah bahasa manusia menjadi suatu representasi bahasa yang dapat dimengerti oleh mesin sehingga mesin dapat memproduksi konten bahasa manusia (Cambria and White, 2014; Collobert and Weston, 2008; Liddy et al., 2003; Winston, 1991). Terdapat 2 komponen dari NLP yaitu *natural language understanding* (NLU) dan *natural language generation* (NLG). NLU bertugas

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

untuk memetakan masukan dari pengguna (bahasa natural) ke bentuk yang dapat dimengerti oleh mesin (Chopra et al., 2013; TutorialsPoint, n.d.), sedangkan NLG bertugas men-*generate* kalimat atau teks (TutorialsPoint, n.d.).

2.5 Pre-processing Methods

Pre-processing method atau *text pre-processing* merupakan sebuah teknik untuk mengubah suatu data yang tidak terstruktur menjadi bentuk data yang terstruktur (Prasidhatama, Adhi; Suryanngrum, 2018). Terdapat beberapa tahap utama dalam *pre-processing* (Imbar et al., 2014; Prasidhatama, Adhi; Suryanngrum, 2018):

a. *Casefolding/Normalization*

Pada tahap ini akan diterapkan *lower case* pada semua kata atau kalimat.

b. *Tokenization*

Pada tahap ini akan dilakukan pemotongan atau pemecahan kalimat (pemisahan karakter seperti spasi) menjadi kumpulan *token* atau kata.

c. *Filtering/Stop-word Removal*

Pada tahap ini kata-kata hasil dari *tokenizing* yang tidak memiliki makna penting akan dihilangkan seperti “ini”, “itu”, “adalah”, dan sebagainya.

d. *Stemming*

Merupakan proses penghilangan semua imbuhan (*affixes*) baik yang terdiri dari awalan (*prefix*), pertengahan (*infixes*), akhiran (*suffixes*), ataupun kombinasi dari *affixes* dan *suffixes* sehingga didapat kata dasarnya.

2.6 Algoritma Stemming Nazief & Adriani

Algoritma *Stemming* Nazief & Adriani merupakan salah satu algoritma *stemming* bahasa Indonesia yang dibuat dan dikembangkan oleh Bobby Nazief

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

dan Mirna Adriani untuk mencari kata dasar dalam bahasa Indonesia (Prasidhatama, Adhi; Suryanngrum, 2018).

Menurut Nazief & Adriani berdasarkan jurnal Moeliono dan Dardjowidjojo (1988) *inflectional suffixes* pada bahasa Indonesia terbagi menjadi 2 jenis:

a. *Particles* (P)

Terdiri dari: "-kah", "-lah", "-tah", "-pun" (merujuk pada kata yang bukan kata benda ataupun kata kerja) (Adriani et al., 2007).

Contoh : "duduklah"

b. *Possessive Pronouns* (PP)

Terdiri dari: "-ku", "-mu", "-nya" (menunjukkan kepunyaan) (Adriani et al., 2007).

Contoh: "bukuku"

Derivational prefix terdiri dari (Adriani et al., 2007):

"be-," "di-," "ke-," "me-," "pe-," "se-," dan "te-".

Derivational suffixes terdiri dari (Adriani et al., 2007):

"-i," "-kan," "-an," .

Contoh *derivational confixes* (Adriani et al., 2007):

"be-an," "di-i," "ke-," "me-kan,".

Akan tetapi tidak semua bentuk *confix* merupakan hasil kombinasi dari *prefix* (awalan) dan *suffix* (akhiran) tergantung dari akar katanya (Adriani et al., 2007).

Tahapan dari algoritma Nazief & Adriani (Adriani et al., 2007) adalah sebagai berikut:

UNIVERSITAS MULTIMEDIA NUSANTARA

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

1. Pertama-tama cari kata yang akan di-*stem* dalam kamus. Jika kata ditemukan, maka kata tersebut diasumsikan sebagai *root word* dan algoritma pun akan berhenti (Adriani et al., 2007).
2. Hapuslah *inflection suffixes* ("-lah", "-kah", "-ku", "-mu", atau "-nya"). Pertama hapus *inflectional particle* (P) ("-lah", "-kah", "-tah", atau "-pun") baru setelah itu menghapus *possesive pronouns* (PP) ("-ku", "-mu", atau "-nya"), jika ada c.
3. Hapus *derivation suffixes* ("-i", "-an", atau "-kan") (Adriani et al., 2007).
4. Hapus *derivational prefix* ("be-", "di-", "ke-", "me-", "pe-", "se-", dan "te-").
    a. Proses akan berhenti jika:
        - *Prefix* yang diidentifikasi merupakan bentuk yang tidak *valid* dan telah dihapus pada langkah 3.
        - *Prefix* yang diidentifikasi identik dengan *prefix* yang dihapus sebelumnya.
        - 3 *prefix* telah dihapus.
    b. Identifikasikan tipe *prefix* dan hapus *prefix* tersebut. 2 tipe dari *prefix*:
        - *Plain*: *prefix* berupa {"di-", "ke-", "se-"} dapat dihapus langsung.
        - *Complex*: *prefix* {"me-", "be-", "pe-", dan "te-"} memiliki variasi yang berbeda-beda maka dari itu digunakanlah aturan yang terdapat pada Tabel 2.2.
    c. Jika kata belum dapat ditemukan pada kamus maka langkah ini akan dilakukan kembali (*rekursif*) sedangkan jika kata ditemukan maka proses akan berhenti.

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

5. Melakukan *recoding* (Adriani et al., 2007).
6. Jika semua langkah telah selesai tetapi tidak juga berhasil, maka kata awal diasumsikan sebagai kata dasar(Adriani et al., 2007).

Jika kombinasi imbuhan merupakan "be-…-lah", "be-…-an", "me…-i", "di…-i", "pe…-i", dan "te…-i" maka akan dilakukan penghilangan prefix terlebih dahulu setelah itu baru suffix akan dihilangkan.

Tabel 2.2 Prefix & suffix yang tidak diijinkan (Adriani et al., 2007)

Prefix	Disallowed suffixes
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

Tabel 2.1 Derivation Prefix rules (Adriani et al., 2007)

Rule	Construct	Return
1	berV...	ber-V... | be-rV...
2	berCAP...	ber-CAP... where C!='r' and P!='er'
3	berCAerV...	ber-CAerV... where C!='r'
4	belajar...	bel-ajar...
5	beC_1erC...	be-C_1erC ... where C_1!={'r'| 'l'}
6	terV...	ter-V... | te-rV...
7	terCP...	ter-CP... where C!='r' and P!='er'
8	terCer...	ter-Cer... where C!='r'
9	teC_1erC_2...	te-C_1erC_2... where C_1!='r'
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe...	mem-pe...
13	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14	men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV... | me-tV...
16	meng{g|h|q|k}...	meng-{g|h|q|k}...
17	mengV...	meng-V... | meng-kV...
18	menyV...	meny-sV...
19	mempV...	mem-pV... where V!='e'
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... | pe-rV...
22	perCAP...	per-CAP... where C!='r' and P!='er'
23	perCAerV...	per-CAerV... where C!='r'
24	pem{b|f|v}...	pem-{b|f|v}...
25	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
26	pen{c|d|j|z}...	pen-{c|d|j|z}...
27	penV...	pe-nV... | pe-tV...
28	peng{g|h|q}...	peng-{g|h|q}...
29	pengV...	peng-V... | peng-kV...
30	penyV...	peny-sV...
31	pelV...	pe-lV...; Exception: for "pelajar", return ajar
32	peCP...	pe-CP... where C!={r|w|y|l|m|n} and P!='er'
33	peCerV...	per-CerV... where C!={r|w|y|l|m|n}

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

Terdapat 1 tipe kata berulang yaitu *plurals* dan *composite word.* Plural adalah dibentuk dari 1 kata dasar seperti buku-buku dan *composite* dibentuk dari berasal kata umum dan dipisahkan dengan "-" seperti sebaik-baiknya (Adriani et al., 2007).

2.7 Chatbot Metric Efficiency

Terdapat beberapa *chatbot metric* yang digunakan untuk mengevaluasi *chatbot* (Hung et al., 2009). *Metric* tersebut diambil dari *metric* PARADISE atau *PARAdigm for Dialogue System Evaluation* yang merupakan suatu *framework* atau model umum untuk mengevaluasi *spoken dialogue agent* (Walker et al., 1997). *Metric* tersebut dapat dilihat pada Tabel 2.3.

Tabel 2.3 *Chatbot Metrics Efficiency* (Hung et al., 2009)

Metric	**Type**	**Data Collection Method**
Total elapsed time	Efficiency	Quantitative Analysis
Total number of user/system turns	Efficiency	Quantitative Analysis
Total number of system turns	Efficiency	Quantitative Analysis
Total number of turns per task	Efficiency	Quantitative Analysis
Total elapsed time per turn	Efficiency	Quantitative Analysis

2.8 Technology Acceptance Model

Technology Acceptance Model (TAM) merupakan suatu model yang digunakan untuk penerimaan *user* dari suatu sistem informasi. TAM diajukan oleh Davis pada tahun 1986 (Van Eeuwen, 2017). Menurut Davis (1989), TAM mengukur *Perceived Usefulness* (PU), *perceived ease of use* (PEOU), *user attitude towards* (A), dan *behavior intention to use* (BI). (Van Eeuwen, 2017). BI menunjukkan probabilitas pengguna akan menggunakan *chatbot* tersebut, dimana

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019

BI bergantung pada PU dan A. A menunjukkan respon dari perasaan pengguna ketika menggunakan chatbot tersebut. PU menunjukkan perspektif *user* mengenai probabilitas meningkatnya *performance* dari suatu *job* jika menggunakan sistem tersebut. PEOU menunjukkan perspektif *user* mengenai *effort* yang dikeluarkan untuk menggunakan sistem tersebut (Van Eeuwen, 2017).

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

Rancang bangun aplikasi..., Ferliana Dwitama, FTI UMN, 2019