BAB II
TINJAUAN PUSTAKA

II.1. Machine Learning

Machine learning merupakan kemampuan komputer untuk melakukan pembelajaran tanpa harus menjelaskan atau memprogram ulang secara eksplisit kepada komputer atau kita hanya membuat program hanya dengan satu kali membuat algoritma lalu program akan belajar dari kasus-kasus yang sudah pernah diselesaikan sebelumnya tidak perlu memprogram ulang kembali. Dengan kata lain *machine learning* adalah ilmu yang mempelajari bagaimana mengajarkan komputer cara belajar berdasarkan data-data yang ada.

Machine learning dibagi menjadi empat jenis, yaitu *supervised*, *unsupervised*, *semi-unsupervised* dan *reinforcment learning*. Dalam penelitian ini *machine learning* yang digunakan adalah *supervised learning*. Karakteristik dari *supervised learning* adalah masalah yang diselesaikan biasanya berupa klasifikasi, data set yang dimiliki kasus yang berbentuk klasifikasi biasanya sudah memiliki label untuk setiap datanya. Contoh algoritma *supervised learning* adalah *naïve bayes, hidden markov models, C4.5, random forest, support vector machines, linear regression, artificial neural network* dan lain sebagainya.

II.2. Web Mining

Web Mining bertujuan untuk menemukan informasi ataupun pengetahuan yang terdapat pada struktur Webhyperlink, halaman konten, dan penggunaan data. Walaupun *web mining* menggunakan banyak teknik *data mining*, namun *web mining* tidak dapat disamakan dengan *data mining* karena sifat dan karakteristik dari data web yang tidak beraturan. Berdasarkan dari jenis data yang digunakan, teknik *web mining* dikategorikan menjadi tiga jenis, yaitu *web structure mining, web content mining,* dan *web usage mining* [11].

a) *Web Structure Mining*: *web structure mining* menggali pengetahuan dari *hyperlink* yang merepresentasikan struktur web. Contohnya, dari sebuah link dapat diperoleh halaman web yang penting, selain itu dapat juga untuk menemukan sekelompok user yang saling berbagi kegemaran. *Data Mining*

Tradisional tidak menangani hal yang seperti itu karena struktur tabel relasional tidak ada yang seperti itu.

b) *Web Content Mining*: *web content mining* mengekstrak atau menggali informasi atau *knowledge* dari konten sebuah halaman website. Sebagai contoh, kita dapat otomatis mengklasifikasi dan mengkluster sebuah halaman website termasuk ke dalam kategori apa. Bagaimanapun kita juga dapat memperoleh data yang berguna seperti deskripsi dari sebuah produk, seluruh post dari forum dan lain sebagainya. Kita juga dapat menggali informasi review pelanggan dan komentar di forum untuk mengetahui opini dari pelanggan.

c) *Web Usage Mining*: *web usage mining* digunakan untuk mengtahui pola penggunaan atau akses dari log penggunaan website. *Web usage mining* mengaplikasikan banyak algoritma data mining. Contoh penggunaan *web usage mining* adalah mendeteksi setiap klik yang dilakukan di halaman web, penggnaan bandwitch website dan lain sebagainya.

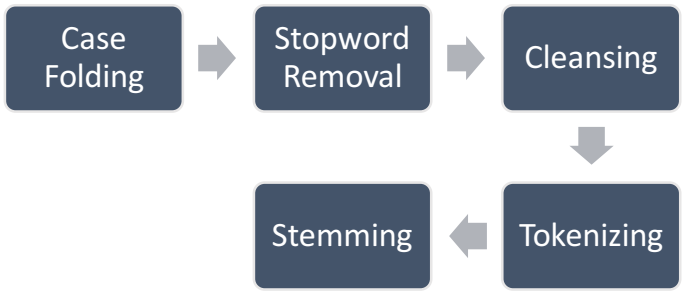
II.3. Text Mining

Text mining yang juga dikenal dengan *text data mining* atau pencarian pengetahuan di basis data textual adalah sebuah proses untuk melakukan pencarian pengetahuan yang berfokus kepada data yang berbentuk dokumen atau teks, dengan tujuan untuk mengekstrak informasi yang berguna dan mengidentifikasinya. *Text mining* mempunyai kesamaan dengan *data mining*. Keduanya memiliki tujuan yang sama yaitu untuk memperoleh pengetahuan dan informasi dari sekumpulan data yang besar. Yang membedakan antara *text mining* dan *data mining* adalah datanya. Untuk *data mining* masukkan datanya sudah terstruktur sedangkan *text mining* masukkan datanya tidak terstruktur [12].

Pada kasus *text mining*, dokumen atau teks yang menjadi sumber data masih belum terstruktur, maka proses transformasi data dari data yang masih belum terstruktur menjadi lebih terstruktur sangat penting. Pada kasus dengan sumber data yang sudah terstruktur, normalisasi data merupakan proses yang menjadi fokus pada *preprocessing.* Sedangkan pada *text mining* yang sumber datanya masih belum terstruktur, *preprocessing* fokus pada mentransformasi data yang belum terstruktur pada dokumen menjadi lebih terstruktur.

II.4. Text Preprocessing

Tahapan *text preprocessing* diperlukan untuk membersihkan sumber data dari data yang tidak diperlukan [12]. Proses ini bertujuan agar data yang digunakan nantinya bersih dari *noise* atau ciri-ciri yang tidak berpengaruh pada proses-proses selanjutnya seperti *link*, "@", "RT", *stopword.* Proses preprocessing juga mempunyai tujuan agar data yang digunakan memiliki dimensi yang lebih kecil dan lebih terstruktur, sehingga dapat diolah lebih lanjut. Tahap *preprocessing* yang digunakan dapat dilihat pada Gambar II-1.



Gambar II-1 Tahap Preprocessing

Text preprocessing pada umummnya terdiri dari tahapan *tokenizing, case folding, stopword removing*. Pada penelitian ini, tahapan *preprocessing*-nya adalah *case folding, stopword removal, cleansing, tokenizing, stemming* dan *un-escape HTML*. Keseluruhan tahapan memiliki fungsi dan perannya masing-masing. Untuk mendapatkan dataset yang berdimensi lebih kecil dari data sebelumnya, terstruktur, serta bersih dari *noise,* maka semua tahap tersebut harus berkesinambungan.

III.4.1 Case Folding

Case folding merupakan proses *text preprocessing* yang dilakukan untuk menyeragamkan karakter pada dokumen atau teks. Pada proses ini semua huruf besar(*uppercase*) dijadikan huruf kecil(*lowercase*). Berikut adalah contoh penerapan *case folding* dapat dilihat pada Tabel II-1.

Tabel II-1 Contoh Penerapan *Case Folding*

Input	*Output*
Enak benar masakan ini! Coba kalau kamu jual, pasti laris!	enak benar masakan ini! coba kalau kamu jual, pasti laris!
Usaha mereka hingga sekitar tahun 1917.	usaha mereka hingga sekitar tahun 1917.
“Maneh keur ngagoreng naon?”	“maneh keur ngagoreng naon?”
"Tahu" berasal dari Bahasa Mandarin *dòufu*(豆腐)	"tahu" berasal dari bahasa mandarin *dòufu*(豆腐)
Dapat memuat 25 – 100 buah tahu goreng	dapat memuat 25 – 100 buah tahu goreng

III.4.2 Stopword Removal

Stopword removal merupakan tahap selanjutnya pada proses *text preprocessing*. Tahap ini bertujuan untuk menghilangkan kata atau *term* yang dianggap tidak berpengaruh dalam menentukan kategori dalam suatu dokumen. Proses ini dilakukan karena *term* tersebut sering muncul di setiap dokumen sehingga dianggap tidak dapat menjadi pembeda.

Sebelum dilakukan proses *stopword removal*, terlebih dahulu dibuat kata-kata yang termasuk ke dalam *stopwords*. Kata-kata yang termasuk *stopwords* biasanya kata ganti orang, kata penghubung, dan lain sebagainya. Berikut adalah contoh penerapan *stopword removal* dapat dilihat pada Tabel II-2.

Tabel II-2 Contoh Penerapan *Stopword Removal*

Input	*Output*
stelah kehabisan bakso gak kebagian kebab dan gagal main game master at shifu ramen the miracle of taste	stelah kehabisan bakso kebagian kebab dan gagal main game master at shifu ramen the miracle of taste
walaupun tak seenak masakan mama dan papa tapi hasil karya sendiri	seenak masakan mama papa hasil karya
hobi sepak bola main game basket snowboarding	hobi sepak bola main game basket snowboarding
hujan nih mari kita merayakannya dengan indomie cabe rawit bakso	hujan merayakannya indomie cabe rawit bakso
taman wisata alam kawah gunung tangkuban prahu	taman wisata alam kawah gunung tangkuban prahu

III.4.3 Cleansing

Cleansing merupakan tahap untuk menghapus komponen yang tidak berpengaruh dalam menentukan kategori. Komponen yang dapat dihapus dalam *tweet* yaitu *username*, *url,* “RT” (tanda retweet), dan *hashtag*. Sedangkan untuk artikel *wikipedia* komponen yang akan dihapus adalah simbol. Berikut adalah contoh penerapan *cleansing* dapat dilihat pada Tabel II-3.

Tabel II-3 Contoh Penerapan *Cleansing*

Input	*Output*
stelah kehabisan bakso, gak kebagian kebab dan gagal main game master (at shifu ramen "the miracle of taste")	stelah kehabisan bakso gak kebagian kebab dan gagal main game master at shifu ramen the miracle of taste
walaupun tak seenak masakan mama dan papa tapi hasil karya sendiri #sarapanseru	walaupun tak seenak masakan mama dan papa tapi hasil karya sendiri
hobi: sepak bola, main game, basket & snowboarding	hobi sepak bola main game basket snowboarding
hujan nih. mari kita merayakannya dengan indomie cabe rawit bakso! \o/	hujan nih mari kita merayakannya dengan indomie cabe rawit bakso
taman wisata alam kawah gunung tangkuban prahu	taman wisata alam kawah gunung tangkuban prahu

III.4.4 Tokenizing

Tokenizing merupakan proses memisahkan kata atau *word* dari sebuah kalimat. Pada penelitian ini *tokenizing* yang dilakukan adalah memisahkan setiap kata yang dipisahkan oleh spasi ataupun simbol untuk artikel *wikipedia* ataupun *tweets*. Gambaran proses *tokenizing* dapat dilihat pada gambar Tabel II-4.

Tabel II-4 Gambaran Proses *Tokenizing*

Input	*Output*
stelah kehabisan bakso gak kebagian kebab dan gagal main game master at shifu ramen the miracle of taste	[stelah, kehabisan, bakso, gak, kebagian, kebab, dan, gagal, main, game, master, at, shifu, ramen, the, miracle, of, taste]
walaupun tak seenak masakan mama dan papa tapi hasil karya sendiri	[walaupun, tak, seenak, masakan, mama, dan, papa, tapi, hasil, karya, sendiri]
hobi sepak bola main game basket snowboarding	[hobi, sepak, bola, main, game, basket, snowboarding]
hujan nih mari kita merayakannya dengan indomie cabe rawit bakso	[hujan, nih, mari, kita, merayakannya, dengan, indomie, cabe, rawit, bakso]
taman wisata alam kawah gunung tangkuban prahu	[taman, wisata, alam, kawah, gunung, tangkuban prahu]

III.4.5 Unescape HTML

Unescape HTML merupakan proses untuk menghapus semua *tag HTML*, karena yang dibutuhkan hanya di dalam *tak html* saja, maka atribut yang terdapat di dalamanya juga akan dihapus. Berikut adalah contoh penerapan *unescape html* dapat dilihat pada Tabel II-5.

Tabel II-5 Contoh Penerapan *Unescape* HTML

Input	*Output*
Enak benar masakan ini! Coba kalau kamu jual, pasti laris!	Enak benar masakan ini! Coba kalau kamu jual, pasti laris!
Usaha mereka hingga sekitar tahun 1917.	Usaha mereka hingga sekitar tahun 1917.

“Maneh keur ngagoreng naon?”	“Maneh keur ngagoreng naon?”
"Tahu" berasal dari Bahasa Mandarin *dòufu*(豆腐)	"Tahu" berasal dari Bahasa Mandarin *dòufu*(豆腐)
Dapat memuat 25 – 100 buah tahu goreng	Dapat memuat 25 – 100 buah tahu goreng

III.4.6 Stemming

Stemming merupakan proses untuk mengubah setiap kata yang sudah dipisahkan atau sudah dilakukan proses *tokenizing* menjadi kata dasar. Algoritma *stemming* untuk setiap bahasa berbeda. Proses *stemming* pada teks berbahasa Indonesia lebih rumit karena terdapat variasi imbuhan yang harus dibuang untuk mendapatkan *root word* dari sebuah kata. Algoritma *stemming* yang digunakan pada penelitian ini adalah algoritma Confix-stripping Stemmer. Contoh penerapan *stemming* dapat dilihat pada Tabel II-6.

Tabel II-6 Contoh Penerapan *Stemming*

Input	***Output***
stelah kehabisan bakso gak kebagian kebab dan gagal main game master at shifu ramen the miracle of taste	stelah habis bakso bagi kebab dan gagal main game master at shifu ramen the miracle of taste
walaupun tak seenak masakan mama dan papa tapi hasil karya sendiri	enak masak mama papa hasil karya
hobi sepak bola main game basket snowboarding	hobi sepak bola main game basket snowboarding
hujan nih mari kita merayakannya dengan indomie cabe rawit bakso	hujan raya indomie cabe rawit bakso
taman wisata alam kawah gunung tangkuban prahu	taman wisata alam kawah gunung tangkuban prahu

II.5. Regular Expression

Regular expression atau yang biasa disingkat dengan regex adalah sebuah teknik yang biasa digunakan untuk pencarian atau memanipulasi sebuah string teks seperti karakter tertentu, kata-kata, dan pola karakter. Pola yang dibentuk oleh regex mungkin akan cocok sekali atau tidak sama sekali dari teks yang diberikan. Regex didukung oleh banyak bahasa pemrograman, seperti JavaScript, Java, Ruby, Python dan masih banyak lagi bahasa pemrograman lainnya yang dapat menggunakan teknik regex. Berikut adalah aturan-aturan penulisan Regex dalam bahasa pemrograman JavaScript [13].

1. Character Classes

Character Classes berikut memiliki arti yang ditentukan dan membuat pola umum lebih mudah digunakan. Berikut contohnya pada tabel Tabel II-7.

Tabel II-7 Contoh Teknik Regular Expression

Regular Expression	Deskripsi
.	Mencocokkan dengan karakter apapun, kecuali baris baru atau terminator unicode yang lain, seperti (\n, \r, \u2028 atau \u2029)
\w	Mencocokkan dengan alfanumerik karakter, termasuk *underscore.* Ini ekuivalen dengan [a-zA-Z0-9_]
\W	Mencocokkan dengan beberapa huruf yang bukan alfanumerik. Ini ekuivalen dengan [^a-zA-Z0-9_]
\d	Mencocokkan dengan digit angka. Ini ekuivalen dengan [0-9]
\D	Mencocokkan dengan non digit angka. Ini ekuivalen dengan [^0-9]
\s	Mencocokkan dengan spasi apapun. Ini ekuivalen dengan [\t\r\n\v\f]
\S	Mencocokkan dengan non spasi. Ini ekuivalen dengan [\t\r\n\v\f]

2. *Literals*

Literals berikut memiliki arti untuk mencari pola karakter literal, dimana memberitahukan regex untuk mencocokkan dengan spesial karakter.

Tabel II-8 Contoh Penerapan *Literals*

Regular Expression	Deskripsi
Alphanumeric	Mencocokkan diri secara literal
\0	Mencocokkan karakter NUL
\n	Mencocokkan karakter baris baru atau enter
\f	Mencocokkan karakter *form feed*
\r	Mencocokkan karakter *carriage return*
\t	Mencocokkan karakter *tab*
\v	Mencocokkan karakter *vertical tab*
[\b]	Mencocokkan karakter *backpasce*
\xxx	Mencocokkan karakter ASCII basis *octal number*
\xdd	Mencocokkan karakter ASCII basis *hex number*
\uxxxx	Mencocokkan karakter ASCII basis UNICODE
\	Mencocokkan apakah karakter berikutnya adalah karakter khusus atau tidak

3. *Character Sets*

Character Sets berikut memiliki arti untuk mencari pola untuk kumpulan dari beberapa karakter, di mana regex memberitahu untuk mencocokkan hanya satu karakter atau lebih dari beberapa karakter.

Tabel II-9 Contoh Penerapan *Character Sets*

Regular Expression	Deskripsi
`[xyz]`	Mencocokkan satu karakter yang berada pada sekumpulan karakter. Untuk contoh, /[a-z]/ mencocokkan huruf dalam alfabet yang cocok dan mencocokkan /[0-9]/ untuk setiap angka
`[^xyz]`	Mencocokkan satu karakter yang tidak terapit dalam kumpulan karakter.

4. *Boundaries*

Boundaries berikut memiliki arti untuk mencari batasan dari kumpulan karakter, dimana regex akan memberitahukan posisi yang cocok dengan batasannya.

Tabel II-10 Contoh Penerapan *Boundaries*

Regular Expression	Deskripsi
`^`	Mencocokkan awalan dari sebuah input.
`$`	Mencocokkan akhiran dari sebuah input
`\b`	Mencocokkan setiap batas kata.
`\B`	Mencocokkan apa saja selain batas kata

5. *Quantifiers*

Quantifiers berguna untuk menentukan berapa banyak contoh dari karakter, grup, atau seberapa sering sebuah elemen dapat terjadi.

Tabel II-11 Contoh Penerapan *Quantifiers*

Regular Expression	Deskripsi
`{n}`	Mencocokkan jumlah sama persis dengan n karakter. Contoh /\d{5}/ cocok dengan "12345" (lima digit angka) dalam "123456789"
`{n, }`	Mencocokkan jumlah minimal n karakter. Contoh /\d{5, }/ cocok dengan "123456789" (minimal lima digit angka) dalam "123456789"
`{n, m}`	Mencocokkan jumlah minimal n dan maksimal m karakter. Contoh /\d{5,7}/ cocok dengan "1234567" (minimal lima digit dan maksimal tujuh digit angka) dalam "123456789"
`*`	Mencocokkan kemunculan tidak sama sekali atau berkali-kali. Sama seperti {0, }. Contoh /fo*/ cocok dengan "foo" dalam "foo" dan cocok dengan "fooooo" dalam "fooooold".

Regular Expression	Deskripsi
+	Mencocokkan kemunculan minimal sekali atau berkali-kali. Sama seperti {1, }. Contoh /o+/ cocok dengan "oo" dalam "foo"
?	Mencocokkan kemunculan tidak sama sekali atau sekali. Sama seperti {0,1}. Contoh /fo?/ cocok dengan "fo" dalam "foo" dan cocok dengan "f" dalam "fairy"

II.6. Term Weighting

Term weighting ialah proses pemberian bobot terhadap semua kata pada dokumen, metode *term weighting* yang digunakan pada penelitian ini adalah tf-idf.

Tf-idf adalah kependekan dari *term frequency – inverse document frequency. Term frequency* adalah salah satu metode pembobotan yang paling sederhana. Pada metode ini, setiap *term* diasumsikan memiliki proporsi kepentingan sesuai dengan jumlah terjadinya atau munculnya *term* tersebut dalam dokumen [14]. Persamaan TF adalah sebagai berikut:

$$W(d,t) = TF(d,t)$$

Dimana *TF(d,t)* adalah frekuensi kemunculan *term t* pada dokumen *d*

Inverse document frequency memperhatikan kemunculan *term* pada kumpulan dokumen. Pada metode ini, *term* yang dianggap bernilai atau berharga adalah *term* yang jarang muncul pada koleksi atau kumpulan dokumen. Persamaan IDF adalah sebagai berikut:

$$IDF(t) = \log \frac{N}{df(t)}$$

Dimana *df(t)* adalah banyaknya dokumen yang mengandung *term* t.

$TF.IDF$ merupakan kombinasi metode TF dan IDF. Sehingga persamaan $TF.IDF$ adalah sebagai berikut:

$$TF.IDF(d,t) = TF(d,t) * IDF(t)$$

II.7. Algoritma K-Nearest Neighbor

Algoritma k-nearest neighbor (KNN) adalah algoritma yang digunakan untuk melakukan klasifikasi terhadap suatu objek, berdasarkan *k* buah data latih yang jaraknya paling dekat dengan objek tersebut. Syarat nilai *k* adalah tidak boleh lebih besar dari jumlah data latih, dan nilai *k* harus ganjil dan lebih dari satu. Dekat

atau jauhnya jarak data latih yang paling dekat dengan objek yang akan diklasifikasi dapat dihitung dengan menggunakan metode *cosine similiarity* [8] [15].

Cosine similiarity merupakan salah satu cara atau metode yang dapat digunakan untuk melihat sejauh mana kemiripan isi antar dokumen. Dalam hal ini *cosine similiarity* berfungsi untuk menguji ukuran yang dapat digunakan sebagai interpretasi kedekatan jarak berdasarkan kemiripan dokumen.

Berikut ini adalah rumus untuk menghitung jarak pada algoritma KNN dengan metode *cosine similiarity*:

$$\cos(\theta_{QD}) = \frac{\sum_{i=1}^{n} Q_i D_i}{\sqrt{\sum_{i=1}^{n}(Q_i)^2} \cdot \sqrt{\sum_{i=1}^{n}(D_i)^2}}$$

Dimana,

$\cos(\theta_{QD})$ = Kemiripan Q terhadap dokumen D

Q = Data Uji

D = Data Latih

n = Banyaknya Data

II.8. Algoritma Confix-Stripping Stemmer

Algoritma *confix-stripping stemmer* adalah algoritma yang digunakan untuk melakukan proses *stemming* terhadap kata-kata berimbuhan [16]. Algoritma *confix-stripping stemmer* mempunyai aturan sendiri dengan model sebagai berikut:

[[[AW +] AW +] AW +] Kata-Dasar [[+ AK][+KK][+P]

AW	: Awalan	AK	: Akhiran
KK	: Kata ganti kepunyaan	P	: Partikel

Langkah-langkah algoritma *confix-stripping stemmer* adalah sebagai berikut:

1. Kata yang belum di-*stemming* dibandingkan ke dalam database kamus kata dasar. Jika ketemu, maka kata tersebut diasumsikan sebagai kata dasar dan algoritma berhenti. Jika kata tidak sesuai dengan kata dalam kamus maka lanjut ke langkah 2.
2. Jika kata di-*input* memiliki pasangan awalan-akhiran “be-lah”, “be-an”, “me-i”, “di-i”, “pe-i” atau “te-i” maka langkah stemming selanjutnya adalah 5, 3, 4,

5, 6, tetapi jika kata yang di-*input* tidak memiliki pasangan awalan-akhiran tersebut, langkah *stemming* berjalan normal yaitu 3, 4, 5, 6, 7.

3. Hilangkan partikel dan kata ganti kepunyaan. Pertama hilangkan partikel ("-lah", "-kah", "-tah", "-pun"). Setelah itu hilangkan juga kata ganti kepunyaan ("-ku", "-mu", atau "-nya"). Contoh: kata "bajumulah", proses stemming kedua menjadi "baju". Jika kata "baju" ada dalam kamus maka algoritma berhenti. Sesuai dengan model imbuhan, menjadi:

   [[[AW+]AW+]AW+] Kata Dasar [+AK]

4. Hilangkan juga akhiran ("-i", "-an" dan "-kan") sesuai dengan model imbuhan menjadi:

   [[[AW+]AW+]AW+] Kata Dasar

   Contoh: kata "membelikan" di-*stemming* menjadi "membeli", jika tdak ada dalam *database* kata maka dilakukan penghilangan awalan.

5. Penghilangan awalan ("be-", "di-", "ke-", "me-", "pe-", "se-" dan "te-") mengikuti langkah berikut:

   a. Algoritma akan berhenti jika:

   i. Awalan diidentifikasi bentuk sepasang imbuhan yang tidak diperbolehkan dengan akhiran berdasarkan Tabel II-13 ***Kombinasi Perfiks dan Suffiks Tidak Diperbolehkan***yang dihapus pada langkah 3.

   ii. Diidentifikasi awalan yang sekarang identic dengan awalan yang telah dihapus sebelumnya atau,

   iii. Kata tersebut sudah tidak memiliki awalan.

   b. Identifikasi jenis awalan dan peluruhannya bila diperlukan jenis awalan ditentukan dengan aturan di bawah ini:

   i. Jika awalan dari kata adalah "di-", "ke-" atau "se-" maka awalan dapat langsung dihilangkan.

   ii. Hapus awalan "te-", "be-", "me-" atau "pe-" yang menggunakan aturan peluruhan yang dijelaskan pada Tabel II-12**.**

   Sebagai contoh kata "menangkap", setelah menghilangkan awalan "me-" kata yang didapat adalah "nangkap". Karena kata "nangkap" tidak ditemukan dalam *database* kata dasar maka karakter "n" diganti dengan karakter "t" sehingga dihasilkan kata "tangkap" dan kata "tangkap"

merupakan kata yang sesuai dengan kata yang ada di *database* kata dasar, maka algoritma berhenti.

6. Jika semua langkah gagal, maka kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

Tabel II-12 Perfiks Suffiks

Awalan (perfiks)	Akhiran (suffiks)
Be-	-i
di-	-an
Ke-	-i -kan
Me-	-an
Se-	-i -kan
Te-	-an

Tabel II-13 Kombinasi Perfiks dan Sufiks yang tidak diperbolehkan

Tabel II-13 Kombinasi Perfiks dan Suffiks Tidak Diperbolehkan

Aturan	Bentuk Awalan	Peluruhan
1	berV…	ber-V… | be-rV
2	belajar…	bel-ajar
3	beC_1erC_2…	be-C_1erC_2… dimana C_1 != {'r'|'l'}
4	terV…	ter-V… | te-rV…
5	terCer…	ter-Cer… dimana C != 'r'
6	teC_1erC_2	te-C_1erC_2... dimana C_1 != 'r'
7	me{l|r|w|y}V…	me-{l|r|w|y}V…
8	mem{b|f|v}…	mem-{b|f|v}…
9	mempe…	mem-pe
10	mem{rV|V}…	me-m{rV|V}… | me-p{rV|V}
11	men{c|d|j|z}…	me-n{c|d|j|z}…
12	menV…	me-nV… | me-tV…
13	meng{g|h|q|k}…	meng-{g|h|q|k}…
14	mengV…	meng-V… | meng-kV…
15	mengeC…	menge-C
16	menyV…	me-ny… | meny-sV…
17	mempV…	mem-pV…
18	pe{w|y}V…	pe-{w|y}V…
19	perV…	per-V… | pe-rV…
20	pem{b|f|v}…	pem-{b|f|v}…
21	pem{rV|V}…	pe-m{rV|V}… | pe-p{rV|V}…
22	pen{c|d|j|z}…	pen-{c|d|j|z}…
23	penV…	pe-nV… | pe-tV…
24	peng{g|h|q}…	peng-{g|h|q}…
25	pengV…	peng-V… | peng-kV
26	penyV…	pe-nya | peny-sV
27	pelV…	pe-lV…; kecuali untuk kata "pelajar" menjadi "ajar"
28	peCP…	pe-CP… dimana C !={r|w|y|l|m|n} dan P!={'er'}
29	perCerV…	per-CerV… dimana C!={r|w|y|l|m|n}

II.9. Web Service

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. *Web service* memiliki *interface* yang menggambarkan kumpulan operasi yang diakses melalui jaringan, menjalankan sebuah atau kumpulan tugas secara spesifik, dideskripsikan menggunakan standar notasi XML formal yang disebut sebagai *service description* yang menyediakan semua rincian kebutuhan untuk interaksi dengan *service* (layanan) termasuk *message* format (rincian tentang operasi), transportasi protokol, dan lokasinya [17].

II.10. RESTful Web Service

REpresentational State Transfer (REST) merupakan sebuah metode dalam menyampaikan *resource* melalu media web. *Resource* sendiri bisa didefinisikan sebagai segala sesuatu yang dapat disimpan didalam sebuah computer dan ditampilkan sebagai urutan bit, misalnya dokumen, *row* dalam *database*, atau hasil dari sebuah perhitungan algoritma [18]. REST adalah model arsitekur yang pada dasarnya memanfaatkan teknologi dan protokol yang sudah ada seperti HTTP (Hypertext Transfer Protocol) dan XML.

Arsitektur REST dibangun dengan sifat sebagai berikut:

1. Addressability

Dalam prinsip ini seluruh *resource* harus tersedia melalui sebuah alamat unik, pengalamatan ini dilakukan dengan menggunakan URI (*Unique Resource Identifier*s).

2. Uniform Interface

RESTful service menampilkan semua *resource* dan interaksinya dengan *interface* yang seragam, dalam metode REST antarmuka yang digunakan adalah dengan menggunakan HTTP. HTTP menawarkan semua operasi yang diperlukan, dikenal, dan tersebar luas. Semua interaksi antara klien dan *resource* didasarkan pada metode dasar HTTP. Metode-metode interaksi yang digunakan dengan dasar HTTP ditunjukkan pada Tabel II-14.

3. Representation-oriented

Representasi menjelaskan dalam bentuk apa data sedang dipertukarkan antara *client* dan *server*. Pada umumnya data dipertukarkan dalam bentuk XML, JSON, dan HTML.

4. Statelessness

Setiap interaksi antara client dan server harus memiliki state sendiri. Jadi server hanya akan memantau *resource state* bukan *client session*.

Berikut merupakan penggunaan metode HTTP dalam REST Web Service:

Tabel II-14 Metode HTTP REST Web Service

Method	Deskripsi
GET	Mendapatkan (read) sebuah sumber daya (resource) yang diidentifikasi dengan URI (Uniform Resource Identifier)
POST	Mengirimkan sumber daya (*resource*) ke server. Digunakan untuk membuat (*create*) sumber daya baru
DELETE	Menghapus (*delete*) sumber daya (*resource*) yang diidentifikasi dengan URI
PUT	Mengirimkan sumber daya (*resource*) ke server. Digunakan untuk memasukkan (*insert*) atau memperbarui (*update*) sumber daya yang tersimpan
HEAD	Mendapatkan metadata (*response header*) dari sumber daya(*resource*) yang diidentifikasi dengan URI

II.11. JSON

JSON atau kependekan dari JavaScript Object Notation adalah standar terbuka untuk merepresentasikan data sebagai atribut dengan value. Awalnya berasal dari *syntax* JavaScript untuk digunakan dalam aplikasi web sebagai alternatif yang lebih ringan dan terstruktur dibandingkan dengan XML (*Extensible Markup Language*). Dan sekarang JSON sering digunakan untuk serialisasi data dan transportasi data di banyak platform dan web aplikasi. JSON memberikan cara yang ideal untuk merangkum data antara klien dan server [19]. Kriteria bentuk JSON adalah sebagai berikut:

1. Data disusun dalam atribut dan value, masing-masing dipisahkan oleh titik dua (:).

2. Atribut muncul sebagai karakter string dengan diapit tanda petik ganda (“) sebelum titik dua (:).
3. Nilai sebelah kanan titik dua (:) dapat berbentuk karakter string, angka, *array*, *object* dan dapat diisi juga oleh nilai *boolean* yaitu *true* atau *false.*

II.12. Object Oriented Analysis and Design

Analisis dan Desain Berorientasi Objek (*Object Oriented Analysis and Design*) adalah cara baru dalam memikirkan suatu masalah dengan menggunakan model yang dibuat menurut konsep. Dasar pembuatannya sendiri adalah objek yang merupakan kombinasi antara struktur data dan perilaku dalam satu entitas. Alasan mengapa harus memakai metode berorientasi objek yaitu karena perangkat lunak itu sendiri yang bersfat dinamis, di mana hal ini disebabkan karena kebutuhan pengguna berubah dengan cepat. Selain itu bertujuan untuk menghilangkan kokmpleksitas transisi antar tahap pada pengembangan perangkat lunak, karena pada pendekatan berorientasi objek, notasi yang digunakan pada tahap análisis perancangan dan impelemntasi relatif sama tidak seperti pendekatan konvensional yang dikarenakan notasi yang digunakan pada tahap analisanya berbeda-beda. Hal itu menyebabkan transisi antar tahap pengembangan menjadi kompleks. Di samping itu dengan pendekatan berorientasi objek membawa pengguna kepada abstraksi atau istilah yang lebih dekat dengan dunia nyata, karena di dunia nyata itu sendiri yang sering pengguna lihat adalah objeknya bukan fungsinya. Beda ceritanya dengan pendekatan terstruktur yang hanya mendukung abstraksi pada level fungsional. Adapun dalam pemrograman berorientasi objek menekankan berbagai konsep seperti: *Class, Object, Abstract, Encapsulation, Polymorphism, Inheritance* dan tentunya UML (*Unified Modeling Language*). UML (*Unified Modeling Language*) sendiri merupakan salah satu alat bantu yang dapat digunakan dalam Bahasa pemrograman berorientasi objek. Selain itu UML merupakan *standard modeling language* yang terdiri dari kumpulan-kumpulan diagram, dikembangkan untuk membantu para pengembang sistem dan software agar bias menyelesaikan tugas-tugas seperti: Spesifikas, Visualisasi, Desain Arsitektur, Konstruksi, Simulasi dan Testing. Dapat disimpulkan bahwa UML (*Unified Modeling Language*) adalah sebuah Bahasa yang berdasarkan grafik atau gambar untuk memvisualisasikan, menspesfikasikan, membangun dan pendokumentasian

dari sebuah sistem pengembangan perangkat lunak berbasis objek (*Object Oriented Programming*) [9].

Dokumentasi UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek yang 4 diantaranya adalah [20]:

1. *Activity Diagram*

Activity Diagram adalah sebuah tahapan yang lebih focus kepada menggambarkan proses bisnin dan urutan aktivitas dalam sebuah proses. Di mana biasanya dipakai pada *business modeling* untuk memperlihatkan urutan aktifitas proses bisnis. *Activity diagram* ini sendiri memiliki struktur yang mirip dengan *flowchart* atau data flow diagram pada perancangan terstruktur. *Activity diagram* dibuat berdasarkan sebuah atau beberapa *use case* pada *use case diagram* [21].

2. *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Di dalam *use case diagram* ini sendiri lebih ditekankan kepada apa yang diperbuat sistem dan bagaimana sebuah sistem itu bekerja. Sebuah *use case* merepresentasikan sebuah interaksi antara actor dengan sistem. *Use case* merupakan bentuk dari sebuah pekerjaan tertentu, misalnya *login* ke dalam sistem, *posting* dan sebagainya, sedangkan seorang actor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu [21]. Adapaun komponen-komponen dalam *use case diagram* diantaranya:

a. Actor

Actor merupakan suatu entitas yang berkaitan dengan sistem tapi bukan dari bagian dalam sistem itu sendiri. *Actor* berada diluar sistem namum berkaitan erat dengan fungsionalitas di dalamnya. *Actor* memiliki hubungan secara langsung terhadap fungsi utama baik terhadap salah satu atau semua fungsionalitas utama. *Actor* juga dapat dibagi terhadap berbagai jenis atau tingkatan dengan cara digeneralisasi atau dispesifikasi tergantung kebutuhan sistemnya. *Actor* biasanya dapat berupa pengguna atau *database* yang secara pandang berada dalam suatu ruang lingkup sistema.

b. Use Case

Use case merupakan gambaran umum dari fungsi atau proses utama yang menggambarkan tentang salah satu perilaku sistem. Perilaku sistem ini

terdefinisi dari proses bisnis sistem yang akan dimodelkan. Tidak semua proses bisnis digambarkan secara fungsional pada *use case*, tetapi yang digambarkan hanya fungsionalitas utama yang berkaitan dengan sistem. *Use case* menitik-beratkan bagaimana sautu sistem dapat berinteraksi baik antar sistem maupun di luar sistem.

3. *Sequance Diagram*

Sequance diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram jenis ini memberikan kejelasan sejumlah objek dan pesan-pesan yang diletakkan diantaranya di dalam sebuah *use case*. Komponen utamanya adalah objek yang digambarkan dengan kotak segi empat atau bulat, *message* yang digambarkan dengan gari penus dan waktu yang ditunjukkan dengan *progress vertical*. Manfaat dari *sequence* diagram adalah memberikan gambaran detail dari setiap *use case diagram* yang dibuat sebelumnya [21].

4. *Class Diagram*

Class diagram adalah sebuah *class* yang menggambarkan struktur dan penjelasan *class,* paket dan objek serta hubungan satu sama lain. *Class diagram* juga menjelaskan hubungan antar *class* secara keseluruhan di dalam sebuah sistem yang sedang dibuat dan bagaimana caranya agar mereka saling berkolaborasi untuk mencapai sebuah tujuan [21].

II.13. Confussion Matrix

Confussion Matrix merupakan sebuah tabel yang terdiri atas banyaknya baris data uji yang diprediksi benar dan tidak benar oleh model klasifikasi, tabel ini diperlukan untuk menentukan kinerja suatu model klasifikasi [22]. Metode ini sering digunakan dengan kasus *multiple classifiers* atau kelas yang lebih dari dua. Maka dari itu metode ini cocok untuk digunakan dalam penelitian ini untuk mengukur seberapa akurat hasil klasifikasi dari model yang telah dibuat.

Berikut adalah contoh dari tabel *confussion matrix* dapat dilihat pada Tabel II-15

Tabel II-15 Tabel Confussion Matrix

		Predicted Class	
		Class = 1	*Class = 0*
Actual Class	*Class = 1*	F 11	F 10
	Class = 0	F 01	F 00

Perhitungan akurasi dengan menggunakan tabel *confussion matrix* adalah sebagai berikut:

$$Akurasi = \frac{\mathrm{F11} + \mathrm{F}\ 00}{\mathrm{F}\ 11 + \mathrm{F}\ 10 + \mathrm{F}\ 01 + \mathrm{F}\ 00}$$