BAB II
TINJAUAN PUSTAKA

II.1 Sejarah Perusahaan

Telkom Group adalah satu-satunya BUMN telekomunikasi serta penyelenggara layanan telekomunikasi dan jaringan terbesar di Indonesia. Telkom Group melayani jutaan pelanggan di seluruh Indonesia dengan rangkaian lengkap layanan telekomunikasi yang mencakup sambungan telepon kabel tidak bergerak dan telepon nirkabel tidak bergerak, komunikasi seluler, layanan jaringan dan interkoneksi serta layanan internet dan komunikasi data. Telkom Group juga menyediakan berbagai layanan di bidang informasi, media dan edutainment, termasuk *cloud-based and server-based managed services*, layanan *e-Payment* dan *IT enabler, e-Commerce* dan layanan portal lainnya.

Berikut penjelasan portofolio bisnis Telkom:

1. Telecommunication

Telekomunikasi merupakan bagian bisnis *legacy* Telkom. Sebagai ikon bisnis perusahaan, Telkom melayani sambungan telepon kabel tidak bergerak *Plain Ordinary Telephone Service* (”POTS”), telepon nirkabel tidak bergerak, layanan komunikasi data, *broadband*, satelit, penyewaan jaringan dan interkoneksi, serta telepon seluler yang dilayani oleh Anak Perusahaan Telkomsel. Layanan telekomunikasi Telkom telah menjangkau beragam segmen pasar mulai dari pelanggan individu sampai dengan Usaha Kecil dan Menengah (“UKM”) serta korporasi.

2. Information

Layanan informasi merupakan model bisnis yang dikembangkan Telkom dalam ranah *New Economy Business* (“NEB”). Layanan ini memiliki karakteristik sebagai layanan terintegrasi bagi kemudahan proses kerja dan transaksi yang mencakup *Value Added Services* (“VAS”) dan *Managed Application/IT Outsourcing* (“ITO”), *e-Payment* dan *IT enabler Services* (“ITeS”).

3. Media

Media merupakan salah satu model bisnis Telkom yang dikembangkan sebagai bagian dari NEB. Layanan media ini menawarkan *Free To Air* ("FTA") dan *Pay TV* untuk gaya hidup digital yang modern.

4. Edutaiment

Edutainment menjadi salah satu layanan andalan dalam model bisnis NEB Telkom dengan menargetkan segmen pasar anak muda. Telkom menawarkan beragam layanan di antaranya *Ring Back Tone* ("RBT"), *SMS Content,* portal dan lain-lain.

5. Sevices

Services menjadi salah satu model bisnis Telkom yang berorientasi kepada pelanggan. Ini sejalan dengan *Customer Portfolio* Telkom kepada pelanggan *Personal, Consumer/Home, SME, Enterprise, Wholesale,* dan Internasional.

Sebagai perusahaan telekomunikasi, Telkom Group terus mengupayakan inovasi di sektor-sektor selain telekomunikasi, serta membangun sinergi di antara seluruh produk, layanan dan solusi, dari bisnis legacy sampai *New Wave Business*. Untuk meningkatkan business value, pada tahun 2012 Telkom Group mengubah portofolio bisnisnya menjadi TIMES (*Telecommunication, Information, Media Edutainment & Service*). Untuk menjalankan portofolio bisnisnya, Telkom Group memiliki empat anak perusahaan, yakni PT. Telekomunikasi Indonesia Selular (Telkomsel), PT. Telekomunikasi Indonesia International (Telin), PT. Telkom Metra dan PT. Daya Mitra Telekomunikasi (Mitratel).

Visi dan Misi

Visi :

> "*To become a leading Telecommunication, Information, Media, Edutainment and Services ("TIMES") player in the region*"

Misi :

1. Menyediakan layanan "*more for less*" TIMES
2. Menjadi model pengelolaan korporasi terbaik di Indonesia.

Corporate Culture *: The Telkom Way*
Basic Belief *: Always The Best*
Core Values *: Solid, Speed, Smart*
Key Behaviours *: Imagine, Focus, Action*
Inisiatif Strategis :

1. Pusat keunggulan.
2. Fokus pada portofolio dengan pertumbuhan atau *value* yang tinggi.
3. Percepatan ekspansi internasional.
4. Transformasi biaya.
5. Pengembangan IDN (*id-Access, id-Ring, id-Con).*
6. *Indonesia Digital Solution* ("IDS") – layanan konvergen pada solusi ekosistem digital.
7. *Indonesia Digital Platform* ("IDP") – platform enabler untuk pengembangan ekosistem.
8. Eksekusi sistem pengelolaan anak perusahaan terbaik.
9. Mengelola portofolio melalui BoE dan CRO.
10. Meningkatkan sinerlagi di dalam Telkom Group.

II.2 Produk Indihome

Indihome adalah salah satu produk terbaru yang tersedia dalam bentuk *bundle* yang menggabungkan tiga jenis layanan yang disebut *Triple Play* dari Telkom yang terdiri dari Internet Speedy via jaringan serat *Optik Fiber to the Home* (FTTH) *Phone* (Telepon Rumah) dan IP TV (*UseeTV Cable*). Indie Home memerlukan bandwitdh besar agar layanannya berjalan lancar. Untuk itu, Telkom menggunakan jaringan serat optik agar koneksi internet bisa lebih kencang dan layanan lainnya bisa berjalan lancar. Jaringan serat optik FTTH masih terus dibangun secara bertahap, terutama untuk menggantikan jaringan lama yang tidak

memiliki *bandwidth* cukup untuk layanan internet berkecepatan tinggi yang dijadikan pondasi layanan IndiHome, termasuk telepon (*voice over IP*).

II.3 Analisis Sentimen

Analisis sentimen atau *opinion mining* merupakan proses memahami, mengekstrak dan mengolah data tekstual secara otomatis untuk mendapatkan informasi sentimen yang terkandung dalam suatu kalimat opini untuk menghasilkan opini yang baru [6]. Analisis sentimen dilakukan untuk melihat pendapat atau kecenderungan opini terhadap sebuah masalah atau objek oleh seseorang, apakah cenderung berpandangan atau beropini negatif atau positif. Analisis Sentimen biasanya dilakukan untuk memantau perkembangan pasar atau untuk melihat respon terhadap suatu masalah, salah satu contoh penggunaan analisis sentimen dalam dunia nyata adalah identifikasi kecenderungan pasar dan opini pasar terhadap suatu objek produk [7].

Pada dasarnya analisis sentimen merupakan klasifikasi, tetapi tidak semudah proses klasifikasi biasa karena terkait penggunaan bahasa yang terus berkembangan. Dimana media yang digunakan dalam kasus ini adalah sebuah teks yang ambigu karena tidak ada intonasi dalam sebuah teks [8].

Manfaat sentimen analisis terhadap perkembangan suatu bisnis sangat besar, sehingga banyak perusahaan menerapkan analisis sentimen sebagai media untuk melihat perkembangan pasar dalam menentukan langkah bisnis yang diambil sebagai bahan pertimbangan perusahaan tersebut.

II.4 Text Mining

Text Mining merupakan data berupa teks dimana sumber data biasanya didapatkan dari dokuman, yang bertujuan mencari kata-kata yang dapat mewaliki isi dari dokumen sehingga dapat dilakukan analisis keterhubungan antar dokumen [9].

Text mining (penambangan teks) adalah penambangan yang dilakukan oleh komputer untuk mendapatkan sesuatu yang baru, sesuatu yang tidak diketahui sebelumnya atau menemukan kembali informasi yang tersirat secara implisit,

yang berasal dari informasi yang diekstrak secara otomatis dari sumber-sumber data teks yang berbeda-beda.

Pada dasarnya proses kerja *Text Mining* banyak mengadopsi *Data Mining* namun yang menjadi perbedaan adalah pola yang digunakan *Text Mining* diambil dari sekumpulan bahasa alami yang tidak terstruktur sedangkan dalam data *Data Mining* pola yang diambil dari database terstruktur. Oleh karena itu dalam *Text Mining,* diperlukan proses pengubahan bentuk dari data yang tidak terstruktur menjadi data yang terstruktur, yang biasanya akan menjadi nilai-nilai numerik. Setelat data menjadi data yang terstruktur dan berupa nilai numerik maka data dapat dijadikan sebagai sumber data yang dapat diolah lebih lanjut.

Text mining merupakan teknik yang digunakan untuk menangani masalah *information retrival* seperti melakukan *indexing* dan mengambil informasi dari dokumen teks, menemukan kumpulan dokumen yang relevan terhadap suatu permintaan tertentu. *Text mining* dapat digunakan juga pada *Web Mining* yang diginakan untuk proses *clastering dan classification,* sedangkan *information extraction* dapat mengekstrasi sebagian informasi yang terdapat pada teks.

II.5 Regular Expression

Regular Expression atau yang biasa disingkat *regex* adalah sebuah teks khusus untuk menggambarkan pencarian sebuah pola. *Regex* biasa digunakan untuk pencarian atau manipulasi teks. *Regex* didukung oleh banyak basaha pemrograman, seperti Java, PHP, C# dan masih banyak bahasa pemrograman lainnya. Berikut adalah aturan-aturan penulisan *Regex* dalam bahasa pemrograman Java [10].

1. Pencocokan simbol umum

Pola yang bisa digunakan dalam *Regex* untuk mencocokan simbol-simbol yang umum pada suatu teks. Berikut contoh dan deskripsi simbol umum *Regex* pada Tabel II-1.

Tabel II-1 Daftar Simbol Umum *Regex*

Regular Expression	Deskripsi
.	Mencocokan dengan karakter apapun
^regex	Menemukan kata *regex* yang ada di awal baris.
regex$	Menemukan kata *regex* yang ada di akhir baris.
[abc]	Tanda kurung siku digunakan untuk mencocokan salah satu huruf yang ada di dalamnya. Contoh digunakan untuk mencocokan dengan huruf a atau b atau c.
[abc][de]	Mencocokan dengan huruf a atau b atau c kemudian diikuti dengan huruf d atau e.
[^abc]	Tanda sisipan yang muncul dalam tanda kurung siku sebagai tanda negasi. Contoh digunakan untuk mencocokan dengan huruf apapun kecuali a atau b atau c.
[a-d1-7]	Mencocokan dengan deretan huruf yang yang ada dari a hingga d dan 1 sampai 7.
a|b	Menemukan a atau b.
Ab	Menemukan a yang kemudian diikuti dengan b.
a!b	Menemukan a yang kemudian diikuti bukan dengan b

2. *Metacharacters*

Metacharacter, karakter ini memiliki arti yang ditentukan dan membuat pola umum yang lebih mudah digunakan. Berikut contoh dan deskripsi *Metacharacter Regex* pada Tabel 2-2.

Tabel II-2 Daftar *Metacharacter Regex*

Regular Expression	Deskripsi
\d	Mencocokan dengan angka, lebih sederhana dari [0-9]
\D	Mencocokan dengan bukan angka, lebih sederhana dari [^0-9]
\s	Mencocokan dengan spasi, lebih sederhana dari [\t\n\x0b\r\f]
\S	Mencocokan dengan bukan spasi, lebih sederhana dari [^\s]
\w	Mencocokan dengan alphanumerik, lebih sederhana dari [a-zA-Z_0-9]
\W	Mencocokan dengan bukan alphanumerik, lebih sederhana dari [^\w]

3. *Quantifier*

Sebuah *Quantifier* menentukan atau mendefinisikan seberapa sering sebuah elemen dapat terjadi. Berikut contoh dan deskripsi *Quantifier Regex* pada Tabel II-3.

Tabel II-3 Daftar *Quantifier Regex*

Regular Expression	Deskripsi	Contoh
*	Terjadi kemunculan tidak sama sekali atau berkali-kali. Lebih sederhana dari {0,}.	a* menemukan tidak sama sekali atau berkali-kali kemunculan huruf a
+	Terjadi kemunculan sekali atau berkali-kali. Lebih sederhana dari {1,}	a+ menemukan sekali atau berkali-kali kemunculan huruf a
?	Terjadi kemunculan tidak sama sekali atau sekali. Lebih sederhana dari {0,1}	A? menemukan tidak sama sekali atau tepat satu kali kemunculan huruf a
{x}	Terjadi kemunculan sebanyak x	\d{5} mencari untuk angka yang memiliki tiga digit.
{x,y}	Terjadi kemunculan sebanyak x hingga ke y.	\d{1-5} berarti \d harus muncul meninmal satu dan paling banyak lima kali.

4. *Backslash*

Backslash digunakan di dalam *Regex* memiliki arti yang ditentukan dalam Java. Sebelumnya telah dibahas penggunaan secara implisit penggunaan *backslah.* Dalam implementasinya untuk menentukan garis miring terbalik tunggal maka harus menggunakan *blackhash* ganda \\, dan ketika ingin mendefiniskan \w , maka harus menggunakan \\w di *Regex* yang dibuat. Jika ingin mendefiniskan *backslahes* dan tanda baca lainnya maka menggunakan *double backslashes* diikuti dengan tanda baca.

II.6 Preprocessing

Text Preprocessing yang merupakan tahap awal dari *text mining* yang akan memproses data latih dan data uji. *Text Preprocessing* ini bertujuan untuk mempersiapkan dokumen teks yang tidak terstruktur menjadi data yang terstruktur yang siap digunakan untuk proses selanjutnya. Tahapan *Text Preprocessing* dalam penelitian ini meliputi:

1. *Convert Emoticon*

Emoticon adalah kata gabungan dari "*emotion*" dan "*icon*" yang berarti icon yang digunakan untuk mengekspresikan emosi sebuah pernyataan tertulis, dan bisa mengubah serta meningkatkan interpretasi terhadap tulisan tersebut. *Emoticon* (*emotion icon*) merupakansalah satu cara pengungkapan perasaan secara tekstual. Hal initentu akan membantu dalam menentukan sentimen suatu *tweet*. Emosi yang bisa menunjukan sebagai sentimen positif ataupun negatif. Oleh karena itu setiap *emoticon* harus dikonversikan ke dalam bentuk kata yang mewakili arti dari *emoticon* tersebut agar bisa diproses. Setiap *emoticon* yang dikonversi akan diberi pemisah spasi untuk mengantisipasi *emoticon* yang berdampingan rapat oleh spasi dengan kata sebelum atau sesudah *emoticon*. Sehingga hasil konversi menjadi lebih mudah diproses. Pada Tabel II-4, dijelaskan hasil pengklasifikasian dari *emoticon* secara umum.

Tabel II-4 Daftar *Emoticon* Yang Akan Dikonversi

Emoticon	Deskripsi
:-) :) :o) :] :3 :c) :> =] 8) =) :} :^) :つ)	Senang
:-D :D 8-D 8D x-D xD X-D XD =-D =D =-3 =3 B^D	Ketawa
:-‖ :@ >:(	Benci
:$ >:[:-(:(:-c :c :-< :つC :< :-[:[:{ ;(:'-(:'(D:< D: D8 D; D= DX v.v D-':	Kecewa
<3 ;-) ;) *-) *) ;-] ;] ;D ;^) :-,	Suka

Pada Tabel II-5, dijelaskan contoh penerapan dari *convert emoticon* menjadi kata

Tabel II-5 Contoh Penerapan Dari *Convert Emoticon*

Data Latih	
Input	**Output**
Jaringan Speedy bangus banget ;)	Jaringan Speedy bangus banget senang
Koneksi Speedy lagi lambat nih :-(	Koneksi Speedy lagi lambat nih kecewa

2. *Cleansing*

Tahap ini akan menghapus semua karakter selain alfabetis dengan tujuan untuk mengurangi *nois*. Sebagaimana diketahui bahwa *emoticon* ini disimbolkan dengan kombinasi karakter khusus dan juga angka, sehingga *emoticon* ini tidak terhapus. Selain karakter khusus, , URL, *hashtag (#)*, *username* (@username), tanda

koma(,), tanda titik(.), tanda seru (!), tanda titik koma (;), tanda titik dua (:), tanda hubung (-), tanda elipsis (…), tanda tanya (?), tanda kurung ((..)), tanda kurung siku ({..}), tanda petik (".."), tanda petik tunggal ('..'), tanda garis miring (/) dan (\), dan tanda penyingkat (') akan dihilangkan.

. Hal tersebut yang menjadi alasan tahapan *cleansing* dilakukan setelah *convert emoticon*. Pada Tabel II-6, dijelaskan contoh penerapan pada *Cleansing.*

Tabel II-6 Contoh Penerapan *Cleansing*

Data Latih	
Input	**Output**
Nonton video @YouTube https://www.youtube.com/watch?v=W1VLMz3yI6A **Taylor Swift Live in Manila Love Story pake Speedy lancar.**	Nonton video **Coldplay Taylor Swift Live in Manila Love Story pake XL lancar.**
Acara di @UseeTVcom bagus nih mendidik banget	**Acara di bagus nih mendidik banget**

3. *Case Folding*

Tahapan ini merupakan tahapan merubah semua masukkan huruf menjadi huruf kecil semua (*lower case*). Karena sistem yang akan dibangun menggunakan basaha pemrograman java, maka disamakan dahulu kedalam bentuk yang sama, dalam hal ini menjadi huruf kecil semua. Hal tersebut agar mempermudah dalam proses selanjutnya di *tokenizing*, karena proses *tokenizing* akan dibandingkan dengan data pada *databases* yang semuanya menggunakan huruf kecil. Pada Tabel II-7, dijelaskan contoh dari penerapan proses *case folding*.

Tabel II-7 Contoh Penerapan *Case Folding*

Data Latih	
Input	**Output**
Cuaca lagi hujan tapi sinyal SPEEDY tetap bagus	cuaca lagi hujan tapi sinyal speedy tetap bagus
Koneksi speedy LELET banget nih	**koneksi speedy lelet banget nih**

4. *Convert Negation*

Tahap ini dikerjakan sebelum tahap *tokenizing* karena pada proses *tokenizing* dilakukan proses perulangan sebanyak data.

Kata-kata yang bersifat negasi seperti “bukan”, “bkn”, “tidak”, “enggak”, “g”, “gak”,“tidak”, “tdk”, “enggak”, “engga”, “ga”, “gk”, “jangan”, “jgn”, “nggak”, “tak” dan “gak”. *Convert negation* dilakukan jika terdapat kata negasi sebelum kata yang bernilai positif, maka kata tersebut akan diubah nilainya menjadi negatif dan begitupun sebaliknya. Pada Tabel II-8, dijelaskan contoh penerapan *Convert Negation*.

Tabel II-8 Contoh Penerapan *Convert Negation*

Sebelum Convert Negation	Setelah Convert Negation
koneksi speedy lagi gak bagus nih	koneksi speedy lagi gak_bagus nih
Acara di usee tv hari ini gak seru banget	Acara di usee tv hari ini gak_seru banget

5. *Tokenizing*

Tokenizing adalah tahap pemotongan dokumen teks bedasarkan tiap kata yang menyusunnya. Potongan kata tersebut disebut dengan token atau *term*.

Pada tahap ini akan dilakukan pengecekan tweets dari karakter pertama sampai karakter terakhir. Pada Tabel II-9, dijelaskan contoh penarapan *tokenizing.*

Tabel II-9 Contoh Penerapan *Tokenizing*

Data Latih	
Input	**Output**
Koneksi speedy lancar dipake streaming bola sama download walaupun cucaca lagi hujan	String 1 : koneksi String 2 : speedy String 3 : lancar String 4 : dipake String 5 : streaming String 6 : bola String 7 : sama String 8 : download String 9 : walaupun String 10 : cuaca String 11 : lagi String 12 : hujan

6. *Filtering*

Filtering berperan untuk membuang kata-kata yang sering muncul dan bersifat umum, kurang menunjukan relevansinya dengan teks. Proses ini akan menghilangkan kata-kata yang sering muncul namun tidak memiliki pengaruhapapun dalam ekstraksi sentimen suatu tweet. Kata-kata yang akan dihilangkan tersebut didefinisikan dalam *stopword list* Contoh beberapa kata yang sering masuk ke dalam *stopword list* adalah “sebuah”, “yang”, “di” “itu”. Proses penghilangan kata yang termasuk *stopword list* dengan cara melakukan pencarian kedalam *database* dimana *stopword list* itu disimpan. Berikut adalah beberapa daftar *stopword list* yang disimpan dalam *database*. Pada Tabel II-10, dijelaskan data *Stopword list* pada proses *Filtering.*

Tabel II-10 *Stopword* Proses *Filtering*

Masih	Dong	ke	Ada	Yoi
Malam	Ya	loe	Pada	Yang
Ini	Dan	juga	Kita	Saya
untuk	Dari	bagi	Iya	di mana
kapan	Bisa	mana	Itu	Sih
sudah	Bikin	dengan	Anda	Begitu
entah	Lalu	yuk	Aku	Adalah
gue	Nanti	gw	Tau	Kemarin

Pada Tabel II-11, berikut adalah contoh penerapan dari tahapan *Stopword* pada proses *Filtering.*

Tabel II-11 Contoh Penerapan *Filtering*

Data Latih	
Input	**Output**
Entah kenapa gue setiap lagi download film koneksi speedy selalu lelet.	Kenapa setiap download film koneksi speedy selalu lelet

7. *Stemming*

Tahap *Stemming* adalah tahap menacari *root* kata dari setiap kata hasil *filtering*. Kata-kata yang muncul didalam dokumen sering kali mengandung imbuhan. Oleh karena itu, setiap kata yang tersisa dari proses hasil tahapan *filtering* dibentuk kedalam kata dasar dengan cara menghilangkan imbuhannya.

Algoritma *Stemming* yang digunakan pada penelitian ini yaitu algoritma *Confix Stripping Stemmer*. Algoritma ini menambahkan suatu algoritma tambahan untuk mengatasi kesalahan pemenggalan akhiran yang seharusnya tidak dilakukan. Pada Tabel II-12, dijelaskan penerapan pada tahapan *stemming*. [12]

Tabel II-12 Contoh Penerapan *Stemming*

Data Latih	
Input	**Output**
lagi	lagi
asik	asik
menonton	nonton
film	film
drama	drama
streaming	streaming
pake	pake
speedy	speedy
koneksinya	koneksinya
lancar	lancar
banget	banget

II.7 *Confix Stripping Stemmer*

Confix stripping Stemmer adalah metode stemming pada Bahasa Indonesia yang diperkenalkan oleh Jelita Asian yang merupakan pengembangan dari metode stemming yang dibuat oleh Nazief dan Adriani (1996). Kata-kata yang muncul didalam dokumen sering kali mengandung imbuhan. Oleh karena itu, setiap kata yang tersisa dari proses hasil tahapan *filtering* dibentuk kedalam kata dasar dengan cara menghilangkan imbuhannya. Pada dasarnya, algoritma ini mengelompokkan imbuhan ke dalam beberapa kategori sebagai berikut: Untuk lebih jelasnya tahapan proses *stemming* adalah sebagai berikut :

1. *Inflection Suffixes* yakni kelompok-kelompok akhiran yang tidak mengubah bentuk kata dasar. Kelompok ini dapat dibagi menjadi dua:
    a. *Particle* (P) atau partikel, termasuk di dalamnya adalah partikel “-lah”, “-kah”, “-tah”, dan “-pun”.
    b. *Possessive Pronoun* (PP) atau kata ganti kepunyaan, termasuk di dalamnya adalah “-ku” , “-mu”, dan “-nya”.

2. *Derivation Suffixes* (DS) yakni kumpulan akhiran yang secara langsung dapat ditambahkan pada kata dasar. Termasuk di dalam tipe ini adalah akhiran "-i", "-kan", dan "-an".

3. Tahapan *Derivation Prefixes* (DP) yakni kumpulan awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah mendapatkan penambahan sampai dengan 2 awalan. Termasuk di dalamnya adalah awalan yang dapat bermorfologi ("me-", "be-", "pe-", dan "te-") dan awalan yang tidak bermorfologi ("di-", "ke-" dan "se-"). Berdasarkan pengklasifikasian imbuhan-imbuhan tersebut, maka bentuk kata dalam bahasa Indonesia dapat dimodelkan sebagai berikut:

   [DP+[DP + [DP+]]] Kata Dasar [[+DS][+PP][+P]] (II-1)

   Dengan batasan-batasan sebagai berikut :

   a. Tidak semua kombinasi imbuhan diperbolehkan. Kombinasi imbuhan yang dilarang dapat dilihat pada Tabel 1.
   b. Penggunaan imbuhan yang sama secara berulang tidak diperkenankan.
   c. Jika suatu kata hanya terdiri dari satu atau dua huruf, maka proses stemming tidak dilakukan.
   d. Penambahan suatu awalan tertentu dapat mengubah bentuk asli kata dasar, ataupun awalan yang telah diberikan sebelumnya pada kata dasar bersangkutan (bermorfologi).

Tabel II-13 Kombinasi Awalan dan Akhiran

Awalan (Prefiks)	Akhiran (Suffiks)
be-	-i
di-	-an
ke-	-i –kan
me-	-an
se-	-i –kan
te-	-an

Tabel II-14 Aturan Peluruhan Kata Dasar

Aturan	Bentuk Awalan	Peluruhan
1	berV…	ber-V… | be-rV…
2	belajar…	bel-ajar
3	beC$_1$erC$_2$…	be-C$_1$erC$_2$… dimana C$_1$!={'r'|'l'}
4	terV…	ter-V… |te-rV…
5	terCer…	ter-Cer… dimana C!='r'
6	teC$_1$erC$_2$	te-C$_1$erC$_2$… 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}…	men-{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}

Algoritma *CS stemmer* bekerja sebagai berikut:

1. Kata yang hendak di-*stemming* dicari terlebih dahulu pada kamus. Jika ditemukan, berarti kata tersebut adalah kata dasar, jika tidak maka langkah 2 dilakukan.
2. Cek *rule precedence*. Apabila suatu kata memiliki pasangan awalan-akhiran "be-lah", "be-an", "me-i", "di-i", "pe-i", atau "te-i" maka langkah *stemming* selanjutnya adalah (5, 6, 3, 4, 7). Apabila kata tidak memiliki pasangan awalan-akhiran tersebut, langkah *stemming* berjalan normal (3, 4, 5, 6, 7).
3. Hilangkan *inflectional particle P* ("-lah", "-kah", "-tah", "-pun") dan kata ganti kepunyaan atau *possessive pronoun PP* ("-ku", "-mu", "-nya").
4. Hilangkan *Derivation Suffixes DS* ("-i", "-kan", atau "-an").
5. Hilangkan *Derivational Prefixes DP* {"di-","ke-","se-","me-","be-","pe", "te-"} dengan iterasi maksimum adalah 3 kali:
   a. Langkah 5 ini berhenti jika :
      1. Terjadi kombinasi imbuhan terlarang
      2. Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
      3. Tiga awalan telah dihilangkan.
   b. Identifikasikan tipe awalan dan hilangkan. Awalan ada dua tipe:

1. Standar: “di-”, “ke-”, “se-” yang dapat langsung dihilangkan dari kata.
2. Kompleks: “me-”, “be-”, “pe”, “te-” adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel II-14 untuk mendapatkan pemenggalan yang tepat.

c. arti kata yang telah dihilangkan awalannya ini di dalam kamus. Apabila tidak ditemukan, maka langkah 5 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.

6. Apabila setelah langkah 5 kata dasar masih belum ditemukan, maka proses *recoding* dilakukan dengan mengacu pada aturan pada Tabel II-14. *Recoding* dilakukan dengan menambahkan karakter *recoding* di awal kata yang dipenggal. Pada Tabel II-14, karakter *recoding* adalah karakter setelah tanda hubung (’-’) dan terkadang berada sebelum tanda kurung. Sebagai contoh, pada kata “menangkap” (aturan 15), setelah dipenggal menjadi “nangkap”. Karena tidak valid, maka *recoding* dilakukan dan menghasilkan kata “tangkap”. Perlu diperhatikan bahwa aturan ke-22 tidak ditemukan dalam tesis Jelita Asian
7. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

Apabila pada kata yang hendak di-*stemming* ditemukan tanda hubung (’-’), maka kemungkinan kata yang hendak di-*stemming* adalah kata ulang. *Stemming* untuk kata ulang dilakukan dengan memecah kata menjadi dua bagian yakni bagian kiri dan kanan (berdasarkan posisi tanda hubung ’-’) dan lakukan stemming (langkah 1-7) pada dua kata tersebut. Apabila hasil *stemming* keduanya sama, maka kata dasar berhasil didapatkan.

Pada Tabel II-14, simbol C merupakan konsonan, V menandakan vokal, A merupakan vokal atau konsonan, dan P merupakan partikel.[16]

II.8 Term Weighting

Term Weighting adalah teknik pembobotan pada setiap *term* atau kata. Tahapan ini sebagian besar teknik pembobotan pada *text mining* menggunakan TF.IDF. TF.IDF menerapkan pembobotan kombinasi keduanya berupa perkalian bobot lokal(*term frequency)* dan bobot global (*global inverse document frequency*). [13] Metode TF-IDF dapat dirumuskan sebagai berikut:

$$idf = \log\left(\frac{N}{df}\right) \qquad \text{(II-2)}$$

Dimana :

N = Banyaknya Data

df = *document frequency*

$$\mathbf{w(t, d) = tf\ (t, d) * IDF} \qquad \text{(II-3)}$$

Dimana :

tf = *term frequency*

IDF = *Inverse Document Frequency*

d = dokumen ke-d

t = kata ke-t dari kata kunci

w(t,d) = bobot dokumen ke-d terhadap kata ke-t

II.9 *Imvroved K-Nearest Neighbor*

Penentuan *k-values* yang tepat diperlukan agar didapatkan akurasi yang tinggi dalam proses kategorisasi dokumen uji. Algoritma *Improved k-Nearest Neighbors* melakukan modifikasi dalam penentuan *k-values*. Dimana penetapan *k-values* tetap dilakukan, hanya saja tiap-tiap kategori memiliki *k-values* yang berbeda. Perbedaan *k-values* yang dimiliki pada tiap-tiap kategori disesuaikan dengan besar-kecilnya jumlah dokumen latih yang dimiliki kategori tersebut. Sehingga ketika *k-values* semakin tinggi, hasil kategori tidak terpengaruh pada kategori yang memiliki jumlah dokumen latih yang lebih besar.

Untuk menghitung similaritas antara dua dokumen menggunakan metode Cosine Similarity (CosSim). Dipandang sebagai pengukuran (*similarity measure*) antara vector dokumen (D) dengan vector query (Q). Semakin sama suatu *vector* dokumen dengan *vector query* maka dokumen dapat dipandang semakin sesuai dengan *query*. [13] Rumus yang digunakan untuk menghitung *cosine similarity* adalah sebagai berikut:

$$Cos\left(\theta_{QD}\right) = \frac{\sum_{i=1}^{n} Q_i \cdot D_i}{\sqrt{\sum_{i=1}^{n}(Q_i)^2} \cdot \sqrt{\sum_{i=1}^{n}(D_i)^2}} \qquad \text{(II-4)}$$

Dimana :

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

Q = Data Uji

D = Data Latih

n = Banyaknya data

Perhitungan penetapan k-values pada algoritma *Improved k-Nearest Neighbor* dilakukan dengan menggunakan persamaan (4), dengan terlebih dahulu mengurutkan secara menurun hasil perhitungan similaritas pada setiap kategori.

Selanjutnya pada algoritma *Improved k-Nearest Neighbor*, *k-values* yang baru disebut dengan n. Persamaan (4) menjelaskan mengenai proporsi penetapan *k-values* (n) pada setiap kategori.

$$n = \left\lceil \frac{k * N(c_m)}{maks\{N(c_m) \mid j=1.....N_c\}} \right\rceil \qquad \text{(II-5)}$$

Dimana :

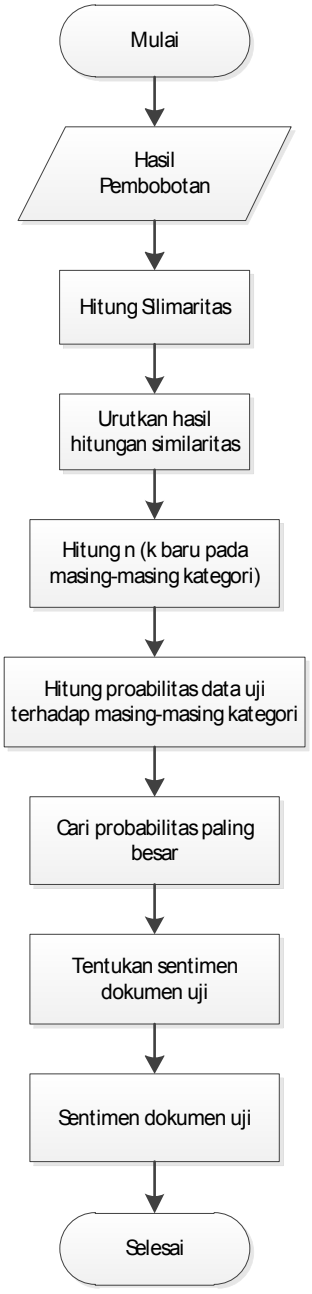
n = *k-values* baru

k = *k-values* yang ditetapkan

$N(c_m)$ = Jumlah dokumen latih di kategori / kategori m

$maks\{N(c_m) \mid j=1.....N_c\}$ = jumlah dokumen latih terbanyak pada semua kategori

Sejumlah n dokumen yang dipilih pada tiap kategori adalah *top* n dokumen atau dokumen teratas yaitu dokumen yang mempunyai similaritas paling besar di setiap kategorinya.



Gambar II-1. ***Flowchart Improved K-Nearest Neighbor***

II.10 *Unified Modeling Language*

Unified modeling language (UML) adalah bahasa pemodelan standar untuk pembangunan perangkat lunak ataupun sistem yang dibangun dengan pendekatan *object-oriented.* UML adalah metodelogi untuk mengembangkan sistem *Object Oriented Programming* (OOP) dan sekelompok tools untuk mendukung pengembangan sistem tersebut tidak hanya merupakan sebuah bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti JAVA, C++, Visual Basic, atau bahkan dihubungkan secara langsung ke dalam sebuah *object-oriented database* [14].

UML itu sendiri sudah mengalami beberapa generasi, dimulai dari UML 1.0, 1.5, dan yang terakhir 2.0. Versi pertama dari UML memberikan komuniasi rancangan secara jelas, menyampaikan esensi dari suat perancangan, bahkan menangkap peta kebutuhan fungsional dari solusi perangkat lunak. Secara keseluruhan ada 13 diagram yang dapat dimanfaatkan dari UML 2.0 ini diantaranya :

1. *Use Case Diagram*
2. *Activity Diagram*
3. *Sequence Diagram*
4. *Communication Diagram (Collaboration diagram in versi 1.x)*
5. *Class Diagram*
6. *State Machine Diagram (Statechart diagram in versi 1.x)*
7. *Component Diagram*
8. *Deployment Diagram*
9. *Composite Structure Diagram*
10. *Interaction Overview Diagram*
11. *Object Diagram*
12. *Package Diagram*
13. *Timing Diagram*

Namun dalam penelitian ini hanya akan menerapkan empat diagram. Pada Tabel II-13, menjelaskan mengenai diagram yang akan digunakan.

Tabel II-15 Diagram-Diagram Uml 2.0 yang Digunakan

Nama Diagram	Deskripsi
Use Case Diagram	Menggambarkan interaksi antara sistem yang akan dibangun dengan pengguna atau sistem eksternal. Juga membantu dalam pemetaan kebutuhan ke dalam sistem
Activity Diagram	Menggambarkan aktivitas secara sekuensial dan pararel dalam sistem.
Class Diagram	Menggambarkan hubungan *class, interface* dalam sistem.
Sequence Diagram	Menggambarkan interaksi diantara *object* di mana urutan dari interaksinya sangat penting

II.11 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek (*Object-Oriented Programming*) atau sering disingkat OOP adalah merupakan paradigma pemrograman yang berorientasikan kepada objek, jadi semua data dan fungsi didalam paradigma ini disatukan kedalam *class-class* atau objek-objek. Definisi *class* yaitu *tamplate* untuk membuat objek. *Class* merupakan prototipe atau *blue prints* yang mendefinisikan variabel-variabel dalam method-method secara umum. Sedangkan Objek dibangun dari sekumpulan data (atribut) yang disebut variabel untuk menjelaskan karakteristik khusus objek, dan juga terdiri dari sekumpulan method yang menjelaskan tingkah laku dari objek. Bisa dikatakan bahwa objek adalah sebuah perangkat lunak yang berisi sekumpulan *variable* dan *method* yang berhubungan. Sebuah objek adalah sebuah *instance* (keturunan) dari class.

Class dan *Object*

Class adalah cetak biru (rancangan) atau *prototype* atau template dari objek. Kita bisa membuat banyak objek dari satu macam *class. Class* mendefinisikan sebuah tipe dari objek. Di dalam class kita dapat mendeklarasikan variabel dan menciptakan objek (instansiasi). Sebuah class mempunyai anggota yang terdiri dari atribut dan method. Atribut adalah semua field identitas yang kita berikan pada suatu *class*, misal *class* manusia memiliki *field* atribut berupa nama dan umur. Method dapat kita artikan sebagai semua fungsi ataupun prosedur yang merupakan perilaku (*behaviour*) dari suatu *class.*

Contructor

Tipe khusus *method* yang digunakan untuk menginstansiasi atau menciptakan sebuah objek. Nama *constructor* = nama kelas. *Constructor* TIDAK BISA mengembalikan nilai. Tanpa membuat *constructor* secara eksplisit-pun, Java akan menambahkan *constructor default* secara implisit. Tetapi jika kita sudah mendefinisikan minimal sebuah *constructor*, maka Java tidak akan menambah *constructor* default. *Constructor* default tidak punya parameter. *Constructor* bisa digunakan untuk membangun suatu objek, langsung mengeset atribut-atributnya. *Construktor* seperti ini harus memiliki parameter masukkan untuk mengeset nilai atribut. *Access Modifier constructor* selayaknya adalah *public*, karena *constructor* akan diakses di luar kelasnya.

Method

Sebuah *method* adalah bagian-bagian kode yang dapat dipanggil oleh kelas, badan program atau *method* lainnya untuk menjalankan fungsi yang spesifik di dalam kelas. Secara umum *method* dalam java adalah sebuah fungsi.

Didalam OOP, objek memiliki data dan *behaviour.* Data adalah atribut yang melekat pada objek, yang akan merepresentasikan state objek tersebut. Sedangkan behavior adalah fungsi-fungsi yang dapat dilakukan oleh objek, yang selanjutnya merepresentasikan sifat-sifat atau perilaku objek. OOP memiliki beberapa konsep dasar yang digunakan dalam pengembangan aplikasi berbasis OO. Konsep konsep tersebut antara lain adalah *encapsulation*, *inheritance*, dan *polymorphism*.

1. Enkapsulasi (*Encapsulation*)

Enkapsulasi adalah metode untuk menggabungkan data dengan fungsi. Dalam konsep ini data dan fungsi digabung menjadi satu kesatuan yaitu *class*. Konsep ini erat kaitannya dengan konsep penyembunyian informasi (*information hiding*). Dalam OOP, konsep enkapsulasi sebenarnya merupakan perluasan dari struktur dalam bahasa C.

Contoh Enkapsulasi (*Encapsulation*) : Mobil adalah sebuah *class.* Seorang pengendara mobil tidak harus tahu bagaimana proses kerja mesin mobilnya, pembakaran bahan bakar, proses pemindahan transmisi gigi, dan seterusnya. Yang dia tahu adalah bahwa setir untuk mengendalikan jalannya mobil, pedal gas untuk menambah kecepatan, pedal rem untuk mengurangi kecepatan, dan seterusnya. Singkatnya dia hanya tahu bahwa mobilnya berjalan dengan baik.

2. Pewarisan (*Inheritance*)

Pewarisan merupakan pewarisan atribut dan *method* dari sebuah *class* ke *class* lainnya. *Class* yang mewarisi disebut *superclass* dan *Class* yang diwarisi disebut *subclass*. *Subclass* bisa berlaku sebagai s*uperclass* bagi *class* lainya, disebut sebagai *multilevel inheritance*.

Dari konsep penurunan ini suatu *class* bisa diturunkan menjadi *class* baru yang masih mewarisi sifat-sifat *class* orang tuanya. Hal ini dianalogikan dengan class manusia. Manusia merupakan turunan dari class orang tuanya dan sifat-sifat orang tua diwarisi olehnya. Bisa ditarik kesimpulan bahwa semua *class* di dunia selalu memiliki hirarki yang menggambarkan silsilah *class* tersebut.

Contoh Pewarisan (*Inheritance*) : Dari *class* kendaraan bermotor dapat diturunkan beberapa *class* baru, misalnya: sepeda motor, sedan, pick-up, mini-bus, dan kendaraan roda banyak. *Class* sepeda motor mungkin masih bisa dibagi lagi menjadi dua *class*, yaitu sepeda motor 2 tak dan sepeda motor 4 tak. Sepeda motor 2 tak mungkin masih dibagi lagi menurut pabrik pembuatnya, seperti Honda, Suzuki, Yamaha, dan lain-lain. Masing-masing pabrik mungkin masih mempunyai beberapa *class* turunan lagi seperti Honda Astrea Star, Honda Beat, Honda Vario dan lain-lain.

Keuntungan dari Pewarisan, *Subclass* menyediakan state/behaviour yang spesifik yang membedakan dengan *superclass*, sehingga memungkinkan programmer untuk menggunakan ulang *source code* dari *superclass* yang telah ada. Programmer dapat mendefinisikan *superclass* khusus yang bersifat generik, yang disebut *abstract class* (abstraksi), untuk mendefinisikan c*lass* dengan tingkah laku dan state secara umum.

3. Polimorfisme (*Polymorphism*)

Polimorfisme (*Polymorphism*) adalah kemampuan suatu objek untuk mempunyai lebih dari satu bentuk. Polimorfisme berarti *class-class* yang berbedatetapi berasal dari satu orang tua dapat mempunyai metode yang sama tetapi cara pelaksanaannya berbeda-beda. Atau dengan kata lain, suatu fungsi akan memiliki perilaku berbeda jika dilewatkan ke *class* yang berbeda-beda.

Contoh Polimorfisme (*Polymorphism*) : jika sebuah burung menerima pesan "gerak cepat", dia akan menggerakan sayapnya dan terbang. Bila seekor singa menerima pesan yang sama "gerak cepat", dia akan menggerakkan kakinya dan berlari. Keduanya menjawab sebuah pesan yang sama, namun yang sesuai dengan kemampuan hewan tersebut. Ini disebut polimorfisme karena sebuah *variabel* tungal dalam program dapat memegang berbagai jenis objek yang berbeda selagi program berjalan, dan teks program yang sama dapat memanggil beberapa metode yang berbeda di saat yang berbeda dalam pemanggilan yang sama. [15]

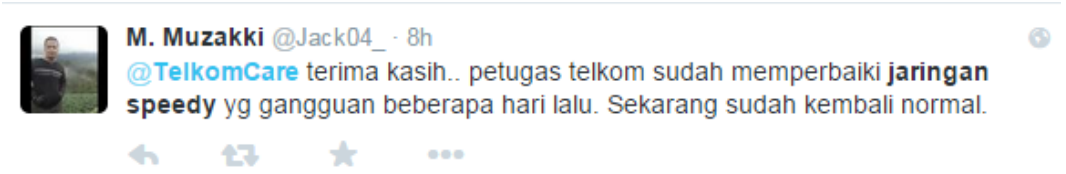
II.12 Twitter

Data yang digunakan pada penelitian ini diambil dari situs media sosial twitter. Twitter adalah sebuah situs dimiliki dan dan dioprasikan oleh twitterInc., yang menawarkan jaringan sosial berupa *mikroblogging* sehingga memungkinkan penggunanya mengirim kucauan sebanyak 140 karakter dan membaca pesan yang disebut *tweets*. Kicauan yang diambil adalah kicauan yang berhubungan dengan tema penelitian ini yaitu kicauan yang membahas Telkom IndiHome di Indonesia.

Adapun karakteristik dari twitter adalah sebagai berikut :

1. Kicauan/*Tweets*

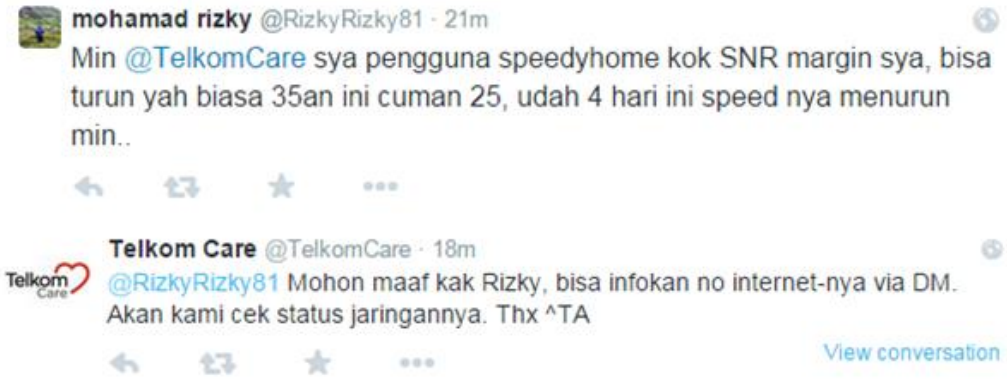
Kicauan atau tweets adalah pesan singkat yang berisikan informasi yang berisi maksimal 140 karakter. Kicauan yang dibuat atau dikirim oleh suatu akun akan muncul pada halaman utama atau *timeline* akun itu sendiri dan akun yang mengikuti akun tersebut.



Gambar II-2. Contoh Kicauan/*tweets*

2. Balas/*Reply*

Balas/*reply* atau sering disebut dengan *mention* adalah kicauan yang bertujuan untuk menanggapi atau membalas kicauan lainnya. yang bertujuan agar sesama pengguna bisa langsung menandai orang yang akan diajak berinteraksi.



Gambar II-3. Contoh *Mention Reply*

3. *Retweet*

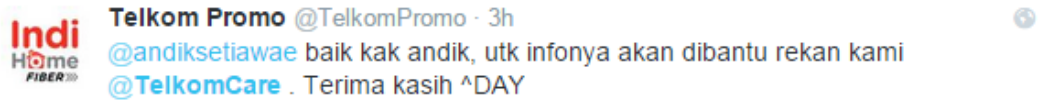
Retweet pada twitter digunakan apabila seseorang ingin membagikan *tweets* ke pengikut/*follower* akun tersebut. *Retweet* biasanya disingkat dengan RT yang artinya mengulang, menulis kembali apa yang dilutis seseorang. RT ditambahkan pada *tweets* diikuti dengan @(*username*) ditambah dengan kata-kata yang ingin diulang, maksudnya agar orang yang membaca *tweets* tersebut tahu darimana sumber *tweets* yang sebenarnya.



Gambar II-4. Contoh *Retweet*

4. @(*username*)

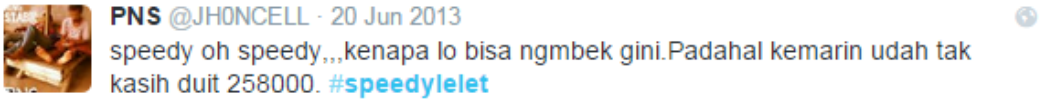
Simbol @ diikuti dengan *username*, digunakan untuk menyebutkan atau menandai akun lain pada sebuah kucauan, sering juga disebut *mention*.



Gambar II-5. Contoh *Mention*

5. Hastag (#)

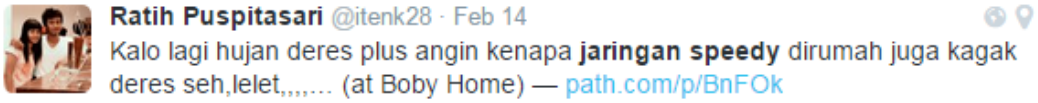
Tanda # pada twitter digunakan untuk menjadikan sebuah kata menjadi kata kunci dalam sebuah kicauan. Tanda # diikuti sebuah kata yang akan dijadikan kata kunci, seperti #TelkomSpeedy #IndiHome #JaringanSpeedy yang bertujuan agar pengguna lain bisa mencari topik yang sejenis yang ditulis oleh orang lain.



Gambar II-6. Contoh Kicauan *hastag*

6. URL

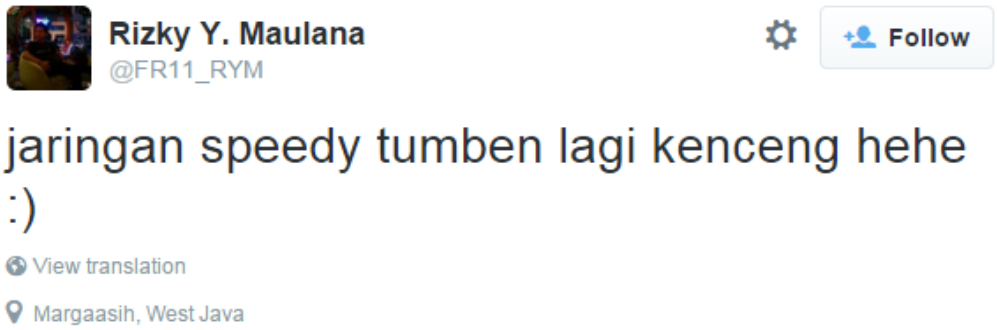
Twitter sebagai media sosial yang populer, banyak dimanfaatkan oleh aplikasi lain untuk publikasi konten dari aplikasi tersebut atau biasa disebut dengan *share* yang berupa URL yang ditautkan di dalam kicauan. Dari sekian banyak URL yang ditautkan seperti path.com/p/XXokQ, bit.ly/1plRcT9, goo.gl/GmAQVS , dan ow.ly/2NSfQI, dapat disumpulkan bahwa setiap *string* yang menyatu dan terdapat *string* ".com/" maka string tersebut dipastikan URL. Selain itu setiap dua *string* yang sebelumnya terdapat titik, kemudian diikuti *backslahes* dibagian selanjutnya, bisa dipastikan juga string tersebut adalah URL. Hal tersebut diamati untuk memudahkan menghapus URL dari dokumen untuk mengurangi *noise*.



Gambar II-7. Contoh Kicauan Yang Mengandung URL

7. Lokasi

Lokasi/*location* adalah salah satu fitur yang terdapat pada twitter yang merupakan lokasi dari akun yang membuat suatu kicauan. Fitur lokasi bersifat dinamis mengikuti lokasi pengguna twitter tersebut. Lokasi ditampilkan tepat dibawah kicauan yang terkait.



Gambar II-8. Contoh Kicauan Mengandung Lokasi

8. Tata Bahasa

Twitter membatasi setiap kicauan dengan maksimal hanya 140 karakter saja dalam satu kali kicauan. Ini membuat pengguna harus sebisa mungkin mengoptimalkan cara agar informasi yang disampaikan menjadi jelas. Sehingga menyingkat suatu kata menjadi biasa, biasanya kata yang sering disingkat adalah kata penghubung, seperti "yang" menjadi "yg" dan masih banyak kata lainnya yang sering disingkat. Dalam penggunaan tanda bacapun biasanya para pengguna sudah tidak memperhatikan lagi aturan yang benar. Sebagai contoh pengguna menggunakan tanda titik lebih dari satu kali (…..) atau tanda tanya lebih dari dua kali (???) yang sebenarnya cukup dengan diwakili satu tanda titik dan tanda tanya. Penulisan penyingkatan kata penghubung, penyingkatannya, dan juga penggunaan tanda baca perlu dihilangkan dalam dokumen untuk mengurangi *noise*.

9. Akses Data Twitter

Data twitter dalam penelitian ini dibutuhkan sebagai bahan baku utama untuk melakukan analisis sentimen. Data twitter bisa didapatkan dengan berbagai cara, salah satunya adalah dengan memanfaatkan *service/library* dari pihak ketiga seperti Twitter4J. Twitter4J adalah *library* untuk bahasa pemrograman Java untuk dapat mengakses API (Application Programming Interface), dengan *library* tersebut dapat dengan mudah mengintegrasikan aplikasi java dengan layanan dari twitter.

II.13 Precision, Recall dan F-Measure

Sistem temu kembali informasi mengembalikan sekumpulan dokumen sebagai jawaban dari *query* pengguna. Terdapat dua kategori dokumen yang dihasilkan oleh sistem temu kembali informasi terkait pemrosesan *query*, yaitu relevant documents (dokumen yang relevan dengan *query)* dan *retrieved* documents (dokumen yang diterima pengguna). Ukuran umum yang digunakan untuk mengukur kualitas dari data *retrieval* adalah kombinasi precision dan *recall*.

Precision mengevaluasi kemampuan sistem temu kembali informasi untuk menemukan kembali data *top-ranked* yang paling relevan, dan didefinisikan sebagai persentase data yang dikembalikan yang benar-benar relevan terhadap *query* pengguna. *Precision* merupakan proporsi dari suatu set yang diperoleh yang relevan. *Precision* dapat dirumuskan persamaan (II-6).

Tabel II-16 Tabel Kontingensi

	Predicted Class		
Actual Class		+	-
	+	TP	FN
	-	FP	TN

$$Precision = \frac{TP}{TP + FP} \quad \text{(II-6)}$$

$$Recall = \frac{TP}{TP + FN} \quad \text{(II-7)}$$

Dengan menjabarkan tabel II.16 diatas maka kita bisa mendapatkan persamaan (II-6) dan (II-7) untuk mendapatkan nilai precision dan recall. Dengan TP adalah *true positive* yaitu jumlah dokumen yang di hasilkan aplikasi sesuai dengan jumlah dokumen yang diberi oleh pakar. FP adalah *false positive* yaitu jumlah dokumen yang bagi pakar dianggap salah akan tetapi oleh aplikasi dianggap benar (hasil yang tidak diinginkan). FN adalah *false negative* yaitu jumlah dokumen yang bagi pakar dianggap benar akan tetapi oleh aplikasi dianggap salah (missing result).

Kombinasi precision dan recall biasa dikombinasikan sebagai harmonic mean, biasa disebut F-measure yang mana dapat di formulasikan seperti persamaan (2.17).

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \quad \text{(II-8)}$$

F-measure biasa digunakan pada bidang sistem temu kembali informasi untuk mengukur klasifikasi pencarian dokumen dan performa *query classification*. Pada penelitian terdahulu *F-measure* lebih difokuskan untuk menghitung nilai, namun seiring dengan perkembangan mesin pencari dengan skala besar, kini *F-measure* lebih menekankan pada kinerja *precision* dan *recall* itu sendiri. Sehingga lebih bisa dilihat pada aplikasi secara keseluruhan.