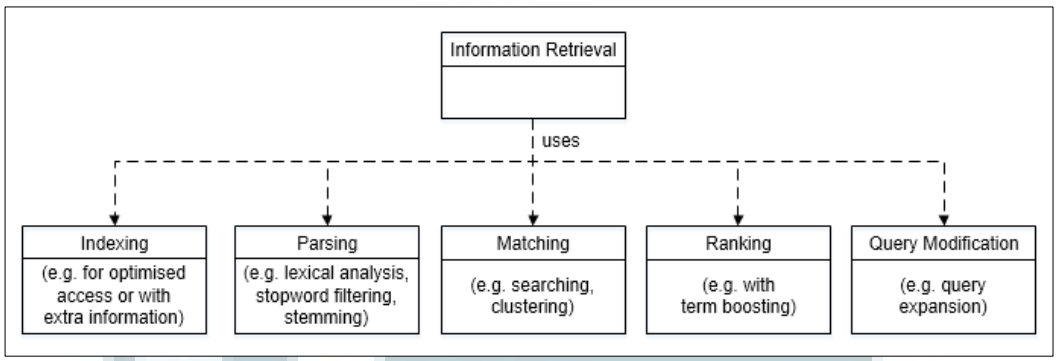
BAB II

LANDASAN TEORI

2.1 Information Retrieval

Information Retrieval (IR) adalah proses mencocokkan *query* dengan objek informasi yang telah terindeks (Goker & Davies, 2009). Indeks adalah struktur data yang telah dioptimasi yang dibuat berdasarkan objek informasi, sehingga mempercepat proses pencarian. Pengguna mengekspresikan kebutuhan akan informasi dalam bentuk *query*. Sistem informasi kemudian akan meresponnya dengan mencocokkan objek-objek informasi dengan *query* yang dimasukkan oleh pengguna, dimana proses pencocokan tidak hanya sekedar memilih objek informasi yang mengandung kata yang sama, namun juga mencakup objek informasi yang relevan dengan *query* dari pengguna.

Secara umum, proses IR dapat dibagi menjadi lima bagian, yaitu *indexing*, *parsing*, *matching*, *ranking*, dan *query modification* (Goker & Davies, 2009). Proses-proses tersebut sering ditemukan dalam *search engine* seperti Google, Bing, Yahoo, dan sebagainya. Dalam implementasinya, beberapa proses tersebut dapat digabungkan bersama. Misalnya, *parsing* dapat menjadi bagian dari proses *indexing*.



Gambar 2.1 *Infomation Retrieval Process*
(Sumber: Goker & Davies, 2009)

2.2 Stemming

Stemming merupakan suatu proses yang terdapat dalam sistem IR yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu. Sebagai contoh, kata bersama, kebersamaan, menyamai memiliki *root word* yang sama yaitu "sama". Proses *stemming* pada teks berbahasa Indonesia berbeda dengan teks berbahasa Inggris, dimana dalam bahasa Inggris, proses yang diperlukan hanya proses menghilangkan sufiks. Sedangkan pada teks berbahasa Indonesia, selain sufiks, prefiks dan konfiks juga dihilangkan (Agusta, 2009).

2.3 Algoritma Nazief-Adriani

Algoritma stemming Nazief-Adriani (1996) dikembangkan berdasarkan aturan morfologi bahasa Indonesia yang mengelompokkan imbuhan menjadi awalan (prefiks), sisipan (infiks), akhiran (sufiks), dan gabungan awalan akhiran (konfiks). Algoritma ini menggunakan kamus kata dasar dan mendukung

recoding, yakni penyusunan kembali kata-kata yang mengalami proses *stemming* berlebih (Tahitoe & Purwitasari, 2010).

$$\left[\mathrm{DP} + \left[\mathrm{DP} + [\mathrm{DP}]\right]\right] root\ word\ [[+\mathrm{DS}][+\mathrm{PP}][+\mathrm{P}]] \ldots\ldots\text{Rumus 2.1}$$

DP : *Derivation Prefix*

DS : *Derivation Suffix*

PP : *Possesive Pronoun* (*Inflection*)

P : *Particle* (*Inflection*)

Tahap-tahap algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani adalah sebagai berikut:

1) Cari kata yang akan di-*stem* dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tersebut adalah *root word*.
2) *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa *particle* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3) Hapus *Derivation Suffixes* (“-i”, “-an”, atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak, maka ke langkah 3a.
   a) Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k” maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.

b) Akhiran yang dihapus (“-i”, “-an”, atau “-kan”) dikembalikan, lanjut ke langkah 4.

4) Hapus *Derivation Prefix*. Jika pada langkah 3b ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   a) Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   b) *For* i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama, algoritma berhenti.
5) Melakukan *recoding*.
6) Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

1) Jika awalannya adalah: “di-”, “ke-“, atau “se-“ maka tipe awalannya secara berturut-turut adalah “di-”, “ke-“, atau “se-“.
2) Jika awalannya adalah “te-”, “me-“, “be-“, atau “pe-“ maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3) Jika dua karakter pertama bukan “di-”, “ke-“, “se-“, “te-”, “be-“, “me-“, atau “pe-“ maka berhenti.
4) Jika tipe awalan adalah “*none*” maka berhenti. Jika tipe awalan adalah bukan “*none*” maka awalan dapat dilihat pada tabel. Hapus awalan jika ditemukan.

Untuk mengatasi keterbatasan pada algoritma di atas, maka Ledy (Agusta, 2009) menambahkan aturan-aturan di bawah ini:

1) Aturan untuk reduplikasi.
   a) Jika kedua kata yang dihubungkan oleh kata penghubung adalah kata yang sama maka *root word* adalah bentuk tunggalnya, contoh: "buku-buku" *root word*-nya adalah "buku".
   b) Kata lain, misalnya "bolak-balik", "berbalas-balasan", dan "seolah-olah". Untuk mendapatkan *root word*-nya, kedua kata diartikan secara terpisah. Jika keduanya memiliki *root word* yang sama maka diubah menjadi bentuk tunggal, contoh: kata "berbalas-balasan", "berbalas", dan "balasan" memiliki *root word* yang sama yaitu "balas", maka *root word* "berbalas-balasan" adalah "balas". Sebaliknya, pada kata "bolak-balik", "bolak" dan "balik" memiliki *root word* yang berbeda, maka *root word*-nya adalah "bolak balik".
2) Tambah bentuk awalan dan akhiran serta aturannya.
   a) Untuk tipe awalan "mem-", kata yang diawali dengan awalan "mengp-" memiliki tipe awalan "mem-"
   b) Tipe awalan "meng-", kata yang diawali dengan awalan "mengk-" memiliki tipe awalan "meng-".

Berdasarkan algoritma di atas, maka dapat dibuat pseudocode algoritma Nazief-Adriani sebagai berikut.

1) Jika kata terdapat di dalam kamus, asumsi kata tersebut adalah kata dasar. Proses selesai. Jika tidak, lanjut ke langkah 2.
2) Jika kata tersebut adalah kata ulang, lanjut ke langkah 3. Jika tidak, lanjut ke langkah 5.
3) Kata ulang tersebut dipisah. Jika kedua kata adalah kata yang sama, kata dasarnya adalah bentuk tunggal dari kata ulang tersebut. Jika tidak, lanjut ke langkah 4.
4) Kata ulang dipisah dan dicari kata dasarnya secara terpisah. Jika kata dasar kedua kata yang telah dipisah tersebut adalah sama, kata dasar dari kata ulang tersebut adalah bentuk tunggal dari kata dasar dari kata ulang. Jika tidak, maka kata dasar dari kata ulang tersebut adalah gabungan dari kedua kata dasar yang telah dicari kata dasarnya secara terpisah. Proses selesai.
5) Hapus *particle* ("-lah", "-kah", "-tah", "-pun") yang terdapat pada kata, jika ada. Hapus juga *possessive pronouns* ("-ku", "-mu", "-nya"), jika ada.
6) Hapus *derivation suffixes* ("-i", "-an", "-kan"), jika ada. Jika kata ditemukan di kamus, proses selesai. Jika "-an" telah dihapus dan huruf terakhir dari kata tersebut adalah "k", maka "-k" juga dihapus. Jika kata ditemukan, proses selesai. Jika tidak, akhiran yang dihapus dikembalikan.
7) Jika pada langkah 6 ada akhiran yang dihapus, lanjut ke langkah 8. Jika tidak ada, lanjut ke langkah 9.
8) Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan, proses selesai. Jika tidak, lanjut ke langkah 9.

9) Dari i = 1 hingga 3, hapus awalan pada kata. Jika kata dasar belum juga ditemukan lakukan langkah 10. Jika kata ditemukan di kamus, proses selesai. Catatan: jika awalan kedua dan awalan pertama sama, proses selesai.

10) Lakukan *recoding*. Jika kata ditemukan di kamus, proses selesai. Jika belum ditemukan juga, asumsi kata awal sebagai kata dasar. Proses selesai.

Tabel 2.1 Imbuhan pada bahasa Indonesia
(Sumber: Marsya & Abidin, 2011)

Imbuhan		**Kata dasar**	**Alomorf**	**Bunyi hilang**	**Contoh**		**Sifat**
					Kata dasar	**Diimbuhkan**	
awalan	me-	berawalan huruf ‘m’, ‘n’, ‘l’, ‘r’, ‘ng’, ‘ny’, ‘w’, ‘y’		‘k’, ‘p’, ‘t, dan ‘s’	nganga nyanyi	menganga menyanyi	derivatif
		berawalan huruf ‘p’, ‘b’, ‘f’	mem-		paksa protes fitnah	memaksa memprotes memfitnah	
		berawalan huruf ‘t’, ‘d’, ‘j’, ‘c’	men-		tulis capai	menulis mencapai	
		berawalan huruf ‘s’	meny-		sapu	menyapu	
		bersuku satu	menge-		bom	mengebom	
		berawalan huruf ‘a’, ‘i’, ‘u’, ‘e’, ‘o’, ‘g’, ‘h’, dan ‘k’	meng-		ambil olah kunci	mengambil mengolah mengunci	
	pe-	sama dengan imbuhan me-		sama dengan imbuhan me-	rusak nyanyi	perusak penyanyi	
			pem-		paksa protes fitnah	pemaksa pemprotes pemfitnah	
			pen-		tulis capai	penulis pencapai	
			peny-		sapu	penyapu	
			penge-		bom	pengebom	

Tabel 2.1 Imbuhan pada bahasa Indonesia (lanjutan)
(Sumber: Marsya & Abidin, 2011)

Imbuhan		Kata dasar	Alomorf	Bunyi hilang	Contoh		Sifat
					Kata dasar	Diimbuhkan	
awalan	pe-	sama dengan imbuhan me-	peng-	sama dengan imbuhan me-	ambil olah kunci	pengambil pengolah pengunci	
	per-	berawalan huruf ‘r’	pe-		redam	peredam	
		kata ‘ajar’	pel-		ajar	pelajar	
	ber-	berawalan ‘r’ atau bersuku kata awal ‘er’	be-		kerja runding	bekerja berunding	
		kata ‘ajar’	bel-		ajar	belajar	
	ter-	berawalan huruf ‘r’ atau (terkadang) bersuku awal ‘er’	te-		rasa pergok	terasa terpergok	
		yang lainnya			kadang sudut	terkadang tersudut	
	di-, ke-, se-				makan tua sesama	dimakan ketua sesama	
akhiran	-i, -an, -kan				main teman buat	mainan temani buatkan	inflektif
	-kah, -lah, -tah, -pun				bukan kapan	bukankah kapanpun	
	-ku, -mu, -nya				milik	milikku	

2.4 Algoritma Successor Variety

Successor Variety (SV) dari sebuah kata adalah jumlah karakter berbeda pada *substring* kata tersebut (Alajrami, 2009). Ide dalam algoritma ini adalah jumlah karakter berbeda yang mengikuti *substring* sebuah kata akan berkurang

setiap penambahan karakter pada *substring* hingga suatu batas tercapai, misalnya ketika jumlah *sucessor variety* meningkat tajam.

Test word : READABLE

Corpus: ABLE, BEATABLE, FIXABLE, READ, READS,

READABLE, READING, RED, ROPE, RIPE

Tabel 2.2 Contoh *Successor Variety*
(Sumber: Al-Shalabi, et al, 2005)

Prefix	Successor Variety	Letters
R	3	E, O, I
RE	2	A, D
REA	1	D
READ	3	A, I, S
READA	1	B
READAB	1	L
READABL	1	E
READABLE	1	blank

Algoritma ini dibagi menjadi empat metode (Al-Shalabi, et al, 2005), yaitu

1) Metode *Cutoff*

Dalam metode ini, beberapa nilai *cutoff* (*threshold*) dipilih kemudian pecah kata tersebut ketika nilai SV mencapai atau melebihi nilai *cutoff* tersebut. Walaupun mudah diimplementasikan, metode ini mengharuskan pemilihan nilai *cutoff* yang cermat. Jika terlalu kecil, akan banyak karakter yang terbuang. Jika terlalu besar, potongan kata yang benar bisa terlewatkan.

2) Metode *Peak and Plateau*

Pemecahan kata dalam metode ini dilakukan ketika SV dari sebuah *substring* lebih besar daripada SV *substring* sebelumnya dan SV *substring*

selanjutnya. Menggunakan contoh di atas, pemecahan kata dilakukan pada saat *substring* = "READ" karena nilai SV-nya lebih besar daripada nilai SV *substring* sebelumnya dan selanjutnya.

3) Metode *Complete Word*

Dalam metode ini, pemecahan kata dilakukan ketika *segment* (*substring*) merupakan sebuah kata yang terdapat dalam corpus. Menggunaka contoh di atas, pemecahan kata dilakukan pada saat *substring* = "READ" karena kata "READ" terdapat dalam corpus.

4) Metode *Entropy*

Berbeda dengan metode lainnya, penentuan *stem* dalam metode ini didasarkan atas distribusi karakter-karakter yang mengikuti *substring*. Cara kerja metode ini adalah sebagai berikut:

a) Asumsi $|D_{ai}|$ sebagai jumlah kata dalam corpus yang dimulai dengan karakter-karakter a sepanjang i.

b) Asumsi $|D_{aij}|$ sebagai jumlah kata dalam $|D_{ai}|$ dengan *successor* j yang perhitungan nilai $|D_{aij}|$ mengikuti langkah a.

c) Probabilitas sebuah kata memiliki *successor* j adalah:

Entropy dari $|D_{ai}| = \frac{|D_{aij}|}{|D_{ai}|}$.....Rumus 2.2

$$H_{ai} = \sum_{j=1}^{26} - \frac{|D_{aij}|}{|D_{ai}|} . \log 2 \frac{|D_{aij}|}{|D_{ai}|}$$.....Rumus 2.3

Contoh : menggunakan contoh di atas dengan i = 2

untuk i = 2, a = "RE", $|D_{ai}|$ = 5

untuk j = ‘A’, $|D_{aij}|$ = 4

untuk j = ‘D’, $|D_{aij}|$ = 1

maka $H_{aij} = -\frac{1}{5} * \log 2 \left(\frac{1}{5}\right) - \frac{4}{5} * \log 2 \left(\frac{4}{5}\right)$

Hasil perhitungan di atas adalah 0.46 + 0.26 = 0.72. Nilai tersebut dianggap kecil karena terdapat empat kata yang mengikuti *substring* “REA...”.

Metode algoritma *Successor Variety* yang digunakan dalam penelitian ini adalah metode *Cutoff* yang berdasarkan hasil eksperimen Hafer dan Weiss (1974) memiliki nilai presisi paling tinggi. Berikut adalah pseudocode algoritma tersebut.

1) Tentukan *successor value* dan *predecessor value* dari kata yang akan di-*stem*.
2) Dari i = 1 hingga panjang kata, jika *string* sepanjang i memenuhi salah satu syarat di bawah ini, maka ambil string tersebut sebagai *stem*. Proses selesai.
   a) *Successor value* bernilai negatif*, *predecessor value* ≥ 5
   b) *Successor value* ≥ 2, *predecessor value* ≥ 17
3) Jika syarat di atas tidak terpenuhi setelah i = panjang kata, asumsi kata awal sebagai *stem*. Proses selesai.

   Catatan: *successor value* bernilai negatif ketika *string* sepanjang i tersebut adalah *complete word*.

2.5 Algoritma N-gram

Kemiripan antar *string* digunakan dalam algoritma yang tidak bergantung pada bahasa tertentu (*language independent*) ini sebagai cara untuk mengubah kata menjadi *root word*. N-gram adalah sebuah *string* yang terdiri atas n karakter, biasanya bersebelahan, yang diekstrak dari bagian teks yang berkelanjutan. Misalnya, kata 'INTRODUCTIONS' jika diekstrak dengan nilai n = 2 akan menghasilkan:

I, IN, NT, TR, RO, OD, DU, UC, CT, TI, IO, ON, NS, S.

Sedangkan jika kata 'INTRODUCTIONS' diekstrak dengan nilai n = 3 akan menghasilkan:

I, *IN, INT, NTR, TRO, ROD, ODU, DUC, UCT, CTI, TIO, ION, ONS, NS*, S.

2.6 Algoritma N-gram Termodifikasi

Algoritma N-gram yang digunakan dalam penelitian ini adalah algoritma N-gram termodifikasi yang diajukan oleh B.P. Pande, Pawan Tamta, dan H. S. Dhami (2013). Penelitian Mayfield James dan McNamee Paul (2003) terhadap sejumlah kata menggunakan *Snowball stems, pseudo stems* dan 4-grams menyebutkan bahwa 4-grams mempunyai performa yang paling bagus dan menyimpulkan 4 karakter awal dari sebuah kata adalah representasi terbaik dari sebuah *stem*. Oleh karena itu, 4-grams diambil sebagai perkiraan awal dalam menentukan *stem* sebuah kata.

Berdasarkan 4-gram tersebut, maka bisa didapatkan frekuensi dari kata dalam *corpus*. Nilai tersebut cenderung menurun (atau tidak berubah) dengan penambahan karakter pada *substring*. Kemungkinan penurunan nilai secara tajam sangat besar terjadi pada saat transisi penambahan bagian sufiks sebuah kata. Namun, bagaimanakah cara mendapatkan nilai penurunan maksimum tersebut? Salah satu caranya adalah dengan menghitung selisih frekuensi dari dua *substring* yang berurutan. Maka dibuatlah sebuah variabel yang bernilai

$$\lambda_i = \begin{cases} abs(F_i - F_{i-1}) & if\ i > 4 \\ 0 & else \end{cases}$$ .....Rumus 2.4

dimana F_i adalah frekuensi N-gram ke-i. Kemudian hitung nilai deviasi tingkat dua menggunakan

$$\Delta_i = \lambda_i - \lambda_{i-1}$$ .....Rumus 2.5

Prosedur metode ini dibagi dalam dua tahap:

Tahap 1:

1) Inisialisasi

   $\lambda_i = M$ dimana M adalah nilai *integer* yang sangat besar

   $\psi_N = 4$

2) Rekursif

   Hitung λ_i , $4 < i \leq |\text{word}|$

   Jika $\lambda_i > \Gamma$

$$\psi_N = \underset{4<i\leq|word|}{argmax}[F_i, F_{i-1}]$$ .....Rumus 2.6

Jika tidak,

$\psi_N = i$……Rumus 2.7

3) Terminasi

a) Jika $N = |word|$, berhenti. Jika tidak, hitung Δ_i.

b) Jika $\Delta_i > 0$, berhenti.

dimana Γ adalah *threshold*, yang mendefinisikan berapa deviasi frekuensi dari dua N-gram yang bisa diterima. Nilai tersebut di-set nol atau pada nilai minimum. Jika pada akhir tahap 1, $\psi_N = |w|$, lanjutkan ke tahap 2.

Tahap 2:

Jika frekuensi tiga N-gram terakhir identik, hapus tiga karakter terakhir dari kata jika setelah dihapus, *stem* masih mempunyai lebih dari tiga karakter.

$\psi_N = \psi_N - 3 \quad if\ \psi_N - 3 > 3$……..Rumus 2.8

Berdasarkan algoritma di atas, maka dapat dibuat pseudocode algoritma N-gram termodifikasi sebagai berikut.

1) Dari i = 1 hingga panjang kata,

a) Jika i < 4, maka λ_i = 0.

Jika tidak, λ_i = absolut(frekuensi i-gram – frekuensi [i-1]-gram)

b) Jika λ_i > 0,

- Jika frekuensi i -gram > frekuensi [i-1]-gram, $\psi_N = i$.
- Jika tidak, $\psi_N = i - 1$

Jika tidak, $\psi_N = i$.

c) Jika ψ_N = panjang kata, berhenti. Jika tidak, hitung Δ_i dengan rumus frekuensi i-gram – frekuensi [i-1]-gram. Jika $\Delta_i > 0$, berhenti.

2) Jika ψ_N = panjang kata dan frekuensi tiga N-gram terakhir identik, maka $\psi_N = \psi_N - 3$, jika $\psi_N - 3 > 3$.

3) Jadikan *string* sepanjang ψ_N sebagai *stem*. Proses selesai.

2.7 Metode Evaluasi Paice

Metode evaluasi Paice berfungsi untuk mengevaluasi kualitas dari algoritma *stemming* dengan menghitung kesalahan-kesalahan yang terjadi selama proses *stemming*. Idealnya, sebuah *stemmer* yang baik akan mengubah semua kata dari kelompok semantik yang sama dalam kategori *root word* yang sama. Namun, karena faktor ketidakteraturan dalam semua bahasa alami, semua *stemmer* tidak dapat menghindari melakukan kesalahan, baik yang menggunakan daftar kosakata maupun tidak (Tala, 2003).

Terdapat dua masalah yang sering muncul ketika menggunakan *stemming* sebagai proses standardisasi. Pertama, terkadang pasangan kata yang berhubungan secara etimologis memiliki arti yang berbeda, contoh : "*author*" dan "*authoritarian*". Kedua, transformasi kata dengan penambahan atau penghilangan imbuhan memiliki banyak faktor ketidakteraturan dan kasus-kasus spesial. Terdapat dua macam *stemming error*, yaitu *understemming* dan *overstemming*. *Understemming error* adalah kesalahan dimana kata-kata yang seharusnya dikelompokkan dalam *root word* yang sama, namun pemenggalan imbuhan tidak mereduksinya hingga *root word* yang sama. *Overstemming error* adalah kesalahan

dimana kata-kata yang dikonversi ke *root word* yang sama memiliki *root word* yang berbeda (Paice, 1994).

Paice mendefinisikan tiga kelas dari hubungan antara pasangan-pasangan kata, yaitu sebagai berikut (Tala, 2003).

1) Tipe 0 adalah dua kata yang identik dalam bentuk dan sudah tergabung, dengan mengabaikan kemungkinan homograf (duplikasi).
2) Tipe 1 adalah dua kata yang berbeda dalam bentuk tapi memiliki semantik yang sama.
3) Tipe 2 adalah dua kata yang berbeda dalam bentuk dan memiliki semantik yang berbeda.

Paice kemudian menghitung kesalahan *understemming* dan *overstemming* menggunakan dua parameter baru yang disebut *Understemming Index* (UI) dan *Overstemming Index* (OI). UI didefinisikan sebagai proporsi pasangan tipe 1 yang tidak berhasil digabung oleh algoritma *stemming*, sedangkan OI didefinisikan sebagai proporsi pasangan tipe 2 yang berhasil digabung oleh prosedur *stemming* (Tala, 2003).

Jika semua kata dalam sampel teks dikelompokkan secara semantik, maka untuk kelompok semantik tertentu, sebut kelompok semantik g, penggabungan semua kata dalam kelompok tersebut didefinisikan sebagai

$$DMT_g = 0.5\, N_g\, (N_g - 1)$$ .....Rumus 2.9

dimana N_g adalah jumlah kata dalam kelompok g. Untuk kelompok yang hanya berisi satu bentuk, nilai DMT = 0 karena tidak ada pasangan yang bisa dibentuk.

Gabungan nilai dalam semua kelompok pada sampel teks dinamakan *Global Desired Merge Total* yang didefinisikan sebagai berikut.

$$GDMT = \sum_{i \in n_g} DMT_{g_i} \text{.....Rumus 2.10}$$

dimana n_g adalah jumlah total kelompok semantik yang ada dalam sampel teks.

Setelah proses *stemming*, semua kata telah direduksi menjadi *root word*. Dalam kelompok yang tidak tergabung secara keseluruhan, akan ada lebih dari satu bentuk *root word* dalam kelompok tersebut. Ini berarti tidak semua kata dalam kelompok tersebut dikelompokkan pada *root word* (*stem*) yang sama. Algoritma *stemming* yang digunakan tidak dapat mengabungkan kata-kata tersebut. Ketidakmampuan sebuah *stemmer* dalam menggabungkan kata-kata dalam sebuah kelompok semantik g pada *stem* yang sama dapat dihitung dengan menggunakan parameter yang dinamakan *Unachieved Merge Total*,

$$UMT_g = 0.5 \sum_{i \in [1..f_g]} n_{g_i} (N_g - n_{g_i}) \text{.....Rumus 2.11}$$

dimana f_g adalah jumlah *stem* yang berbeda dalam kelompok semantik g, dan n_{g_i} adalah jumlah kata dalam kelompok tersebut yang telah menjadi *stem* i.

Nilai total UMT dari semua kelompok dari sampel teks dinamakan *Global Unachieved Merge Total*,

$$GUMT = \sum_{i \in n_g} UMT_{g_i} \text{.....Rumus 2.12}$$

Berdasarkan rumus 2.10 dan 2.12, *Understemming Index* (UI) dapat didefinisikan sebagai berikut

$$UI = \frac{GUMT}{GDMT} \text{.....Rumus 2.13}$$

Sebuah *stemmer* mungkin mengubah banyak pasangan kata yang berasal dari kelompok-kelompok semantik yang berbeda ke dalam *stem* yang identik. Setiap *stem* mendefiniskan sebuah kelompok *stem* yang anggotanya mungkin berasal dari sejumlah kelompok semantik yang berbeda. Jika semua kata dari kelompok *stem* tertentu berasal dari kelompok semantik awal yang sama, maka kelompok *stem* tidak mengandung kesalahan, sebaliknya jika kelompok *stem* tertentu berisi anggota yang berasal dari kelompok-kelompok semantik yang berbeda, ini berarti bahwa telah terjadi kesalahan penggabungan (*wrongly-merged*). Jumlah kesalahan penggabungan dalam sebuah kelompok *stem* s, yang mengandung *stem* yang berasal dari f_s kelompok semantik awal yang berbeda, dinamakan *Wrongly-Merged Total*,

$$WMT_g = 0.5 \sum_{i \in [1..f_s]} n_{s_i} (N_s - n_{s_i}) \text{.....Rumus 2.14}$$

dimana N_s adalah jumlah *item* dalam kelompok *stem*, n_{s_i} adalah jumlah *stem* yang berasal dari kelompok semantik awal ke- i . Jumlah *wrongly-merged* untuk semua kata-kata dalam sampel teks setelah proses *stemming* dinamakan *Global Wrongly-Merged Total*,

$$GWMT = \sum_{i \in n_s} WMT_{s_i} \text{.....Rumus 2.15}$$

dimana n_s adalah jumlah kelompok *stem* setelah proses *stemming*.

Setiap kata dalam kelompok semantik tertentu memiliki kemungkinan untuk digabungkan dengan kata-kata dari kelompok semantik yang berbeda, yang harus dihindari. Untuk sebuah kelompok g tertentu, nilai ini disebut *Desired Non-merged Total*,

$$DNT_g = 0.5\, N_g\, (W\, -\, N_g)$$ ……Rumus 2.16

dimana W adalah jumlah total kata dalam sampel teks. Jumlah kemungkinan DNT untuk semua kata dalam sampel teks dinamakan *Global Desired Non-Merge Total*,

$$GDNT = \sum_{i \in [1..N_g]} DNT_g$$ ……..Rumus 2.17

Sama seperti *Understemming Index* (UI), *Overstemming Index* (OI) dapat didefinisikan menggunakan rumus 2.15 dan 2.17 sebagai

$$OI = \frac{GWMT}{GDNT}$$ ……Rumus 2.18