BAB II
LANDASAN TEORI

2.1 Kata

Pengertian kata secara sederhana adalah sekumpulan huruf yang mempunyai arti. Dalam kamus besar bahasa indonesia (KBBI) pengertian kata adalah unsur bahasa yang diucapkan atau dituliskan yang merupakan perwujudan kesatuan perasaan dan pikiran yang dapat digunakan dalam berbahasa. Kata dasar adalah kata yang belum diberi imbuhan atau belum mengalami morfologi lainnya(Moeliono dan Anton M, 1988).

Kategori kata ada lima (Moeliono dan Anton M, 1988), yaitu :

1. Kata kerja (*verba*)

   Kata kerja adalah kata yang menjadi inti dalam frasakerja, sama ada yang berlaku atau dilakukan.

2. Kata sifat (*Adjektif*)

   Kata sifat merupakan kata-kata yang dapat diikuti dengan kata keterangan *sekali* serta dapat dibentuk menjadi kata ulang berimbuhan gabung SE-NYA.

3. Kata keterangan (*Adverbia*)

   Kata keterangan adalah kata-kata yang digunakan untuk memberikan keterangan kepada kata lain. Seperti kata kerja dan kata sifat.

4. Kata benda

   Kata benda merupakan kata-kata yang dapat diterangkan menggunakan jenis kata-kata lain.

5. Kata Tugas

   Kata tugas adalah kata yang hanya memiliki arti gramatikal dan tidak memiliki arti leksikal.

2.2 Kalimat

Kalimat adalah suatu bahasa yang terdiri dari dua kata atau lebih yang mengandung pikiran yang lengkap dan punya pola intonasi.

Ciri-ciri unsur kalimat terdiri dari (Moeliono dan Anton M, 1988) :

1. Subjek, yaitu bagian yang menjadi pangkal atau pokok pembicaraan.
2. Predikat, yaitu bagian yang menerangkan *subyek*, biasanya berdiri sesudah subyek.
3. Obyek, yaitu bagian yang menjadi tujuan.
4. Keterangan, yaitu bagian yang menunjukkan waktu (keterangan waktu), tempat (keterangan tempat), alat (keterangan alat) dan sebagainya.

Sedangkan jenis kalimat menurut fungsi isinya dapat dikategorikan sebagai berikut:

1. Kalimat berita(*deklaratif*)
2. Kalimat Tanya (*interogatif*)
3. Kalimat perintah (*imperative)*

2.3 Paragraf

Paragraf disebut juga alinea. Menurut (alwi, 2003) Kata paragraf merupakan kumpulan suatu kesatuan pikiran yang lebih tinggi dan lebih luas dari pada kalimat. Paragraf adalah seperangkat kalimat yang membicarakan suatu gagasan atau *topic*.

Paragraph merupakan inti penuang buah pikiran dalam sebuah karangan. Kalimat-kalimat dalam paragraph memperlihatkan kesatuan pikiran atau mempunyai keterkaitan dalam membentuk gagasan atau topic.

2.4 Tipe evaluasi

Metode untuk mengevaluasi hasil ringkasan merupakan topik yang cukup sulit, baik evaluasi terhadap ringkasan yang dihasilkan dari mesin peringkas otomatis ataupun ringkasan yang manual dibuat oleh *abstractor* yang professional, dikarenakan tidak terdapat definisi ringkasan ideal. Terdapat dua klasifikasi metode evaluasi (Sparck dan Galliers,1996),yaitu :

a. Ekstrinsik

   Kualitas ringkasan diukur berdasarkan bagaimana ini membantu penyelesaian tugas *user*.

b. Intrinsik

   Hanya diukur dari kualitas hasil (*output*) ringkasan yang dihasilkan. Evaluasi sistem peringkas yang ada saat ini adalah intrinsic. Pengevaluasi menciptakan sekumpulan ringkasan yang ideal, masing-masing satu untuk menguji teks. Kemudian membandingkan hasil ringkasan sistem dengan ringkasan ideal. Yang diukur adalah *overlap* dari isi, seringkali disebut dengan *recall* dan *precision* kalimat atau *frase*, tapi kadang-kadang dengan overlap kata tunggal.

$$recall = \frac{\#kalimatringkasansistem \cap kalimatringkasanideal}{\Sigma kalimatringkasanideal} \quad (2.1)$$

$$precision = \frac{\#kalimatringkasansistem \cap kalimatideal}{\Sigma kalimatringkasansistem} \quad (2.2)$$

Kombinasi antara recall dengan precision menghasilkan f-measure.

$$f-measure = 2 * \frac{precision*recall}{recall+precision} \quad (2.3)$$

$$akurasi = \frac{total jumlahringkasanbenar}{jumlahringkasanbenar+jumlahringkasansalah} \quad (2.4)$$

2.5 Pemrosesan Teks

Text preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. Inputan awal pada proses ini adalah berupa dokumen. *Text preprocessing* pada penelitian ini terdiri dari beberapa tahapan, yaitu: proses pemecahan kalimat, proses *case folding, proses filtering kalimat, proses tokenizing kata, dan proses stemming*

Gambar 2.1 Proses *text preprocessing*

2.4.1 Pemecahan Kalimat

Memecah dokumen menjadi kalimat-kalimat merupakan langkah awal tahapan *text preprocessing*. Pemecahan kalimat yaitu proses memecah string teks dokumen yang panjang menjadi kumpulan kalimat- kalimat. Dalam memecah dokumen menjadi kalimat-kalimat menggunakan fungsi split(), dengan tanda titik ".", tanda tanya "?" dan tanda tanya "!" sebagai delimiter untuk memotong *string* dokumen. Dengan menghilangkan tanda-tanda tersebut dokumen akan terpotong menjadi kalimat. Contoh hasil pemecahan dokumen menjadi kalimat dapat dilihat pada tabel 2.1.

Tabel 2.1. Contoh pemecahan kalimat

Case folding	Hasil *Tokenizing* kalimat
Taman kanak-kanak merupakan pendidikan formal. Kegiatan TK dilakukan melalui belajar sambil bermain.	- Taman kanak-kanak merupakan pendidikan formal - Kegiatan TK dilakukan melalui belajar sambil bermain

2.4.2 *Case Folding*

Case folding adalah tahapan proses mengubah semua huruf dalam teks dokumen menjadi huruf kecil semua, serta menghilangkan karakter selain a-z dan dianggap sebagai delimiter.

Tabel 2.2. Contoh case *folding*

Kalimat	*Case folding*
- Taman kanak-kanak merupakan pendidikan formal - Kegiatan TK dilakukan melalui belajar sambil bermain	- taman kanak-kanak merupakan pendidikan formal - kegiatan tk dilakukan melalui belajar sambil bermain

2.4.3 *Filtering* Kalimat

Filtering merupakan proses penghilangan *stopword*. *Stopword* adalah kata kata yang sering kali muncul dalam dokumen namun artinya tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu. Didalam bahasa Indonesia *stopword* dapat disebut sebagai kata tidak penting, misalnya “di”, ”oleh”, “pada”, ”sebuah”, ”karena” dan lain sebagianya.

Tabel 2.3 Contoh *Filtering* kalimat

Hasil *case folding* kalimat	*Filtering* kalimat
- taman kanak-kanak merupakan pendidikan formal - kegiatan tk dilakukan melalui belajar sambil bermain	- taman kanak-kanak pendidikan formal - kegiatan tk belajar bermain

2.4.4 *Tokenizing* Kata

Tokenizing adalah proses pemotongan *string* input berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan men-scan kalimat dengan pemisah (*delimiter*) *white space* (spasi, tab, dan *newline*).

Tabel 2.4. Contoh *Tokenizing* kata

Filtering kalimat	*Tokenizing* kata
- taman kanak-kanak pendidikan formal - kegiatan tk belajar bermain	taman kanak pendidikan formal kegiatan tk belajar bermain

2.4.5 *Stemming* dengan Algoritma Nazief dan Andriani

Stemming merupakan suatu proses yang terdapat dalam sistem IR yang mentransformasikan kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu. Sebagai contoh, kata bersama, kebersamaan, menyamai, akan distem ke *root word*nya yaitu "sama". Ada beberapa algoritma yang dapat digunakan untuk *stemming* dalam bahasa indonesia, yaitu algoritma Nazief dan Andriani, algoritma Arifin dan Setiono, algoritma porter . Algoritma Nazief dan Andriani adalah algoritma yang paling efektif untuk stemming bahasa Indonesia (Agusta, 2009). Algoritma Nazief dan Adriani merupakan algoritma stemming untuk teks berbahasa indonesia yang memiliki *persentase* keakuratan lebih baik dari algoritma lainnnya.

Berikut ini adalah langkah-langkah yang dilakukan oleh algoritma Nazief dan Adriani (Agusta, 2009) :

1. Cari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* ("-lah", "-kah", "-ku", "-mu", atau "-nya") dibuang. Jika berupa *particles* ("-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 dikamus, 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 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
    a. Periksa tabel kombinasi awalan-akhiran yang tidak diizinkan (tabel 2.5). 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 “tidak ada” maka berhenti. Jika tipe awalan adalah bukan “tidak ada” maka awalan dapat dilihat pada Tabel 2.6. Hapus awalan jika ditemukan.

Tabel 2.5 Kombinasi Awalan Akhiran Yang Tidak Diijinkan

Awalan	Akhiran yang tidak diijinkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan

Tabel 2.6 Cara Menentukan Tipe Awalan Untuk Kata Yang Diawali Dengan "te-"

FollowingCharact				Tipe Awalan
Set 1	Set 2	Set 3	Set 4	
"-r-"	"-r-"	-	-	none
"-r-"	Vowel	-	-	ter-luluh
"-r-"	not (vowel or "-r-")	"-er-"	vowel	ter
"-r-"	not (vowel or "-r-")	"-er-"	not vowel	ter-
"-r-"	not (vowel or "-r-")	not "-er-"	-	ter
not (vowel or "-r-")	"-er-"	vowel	-	none
not (vowel or "-r-")	"-er-"	not vowel	-	te

Tabel 2.7 Jenis Awalan Berdasarkan Tipe Awalannya

Tipe Awalan	Awalan yang harus dihapus
di-	di-
ke-	ke-
se-	se-
te-	te-
ter-	ter-
ter-luluh	ter

2.6 *Automatic Text Summarization*

Peringkasan teks otomatis (*Automatic Text Summarization*) adalah pembuatan versi yang lebih singkat dari sebuah teks dengan memanfaatkan aplikasi yang dijalankan pada komputer. Peringkasan teks otomatis berguna untuk membantu manusia dalam mendapatkan ringkasan dari suatu bacaan tanpa harus membaca semua isi dari bacaan (nugraha,2008).

Menurut (Jezek & Steinberger 2007) ada dua kriteria peringkasan teks yaitu peringkasan teks berdasarkan eksktraksi dan abstraksi Teknik ekstraksi merupakan suatu teknik untuk menyalin unit-unit teks yang paling penting atau paling informatif dari teks sumber menjadi ringkasan, sedangkan teknik abstraksi adalah mengambil intisari dari teks sumber kemudian membuat ringkasan dengan menciptakan kalimat-kalimat baru yang merepresentasikan intisari teks sumber dalam bentuk berbeda (Jezek & Steinberger 2007).

2.6.1 Riset Tentang Peringkas teks Sebelumnya

Sejumlah penelitian telah dilakukan dalam membangun sistem peringkasan dokumen otomatis dengan menggunakan beberapa metode diantaranya penelitian tentang *Penerapan Algoritma Genetika pada Peringkasan Teks Dokumen Bahasa Indonesia* oleh Aristoteles (2013), *penerapan Terms Frequency –Inverse Document Frequency pada sistem peringkas teks otomatis dokumen tunggal berbahasa indonesia* oleh Iyan Mulyana, dkk (2010), *Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode Maximum Marginal Relevance* oleh Muchammad Mustaqhfiri, dkk (2011).

Pada penelitian Aristoteles (2013) dilakukan penelitian tentang bagaimana meringkas dokumen bahasa Indonesia yang berjenis file teks dengan menggunakan algoritma genetika. algoritma genetika atau *genetic algorithm* adalah algoritma pencarian yang didasari pada mekanisme genetik alamiah dan seleksi alamiah. Terdapat 11 fitur teks yang dilakukan pada penelitian ini yaitu yaitu posisi kalimat, *positive keyword*, *negative keyword*, kemiripan antar kalimat, kalimat menyerupai judul, kalimat yang mengandung nama entiti, kalimat yang mengandung data numerik, koneksi antar-kalimat, penjumlahan bobot antar-kalimat, dan kalimat semantic. Hasil penelitian yang diperoleh bahwa algoritma genetika dapat digunakan untuk mencari tingkat kepentingan yang optimal dari tiap fitur teks. Nilai akurasi 47.46% pada *compression* 30%. Sedangkan hasil tidak optimal pada *compression* 10%.

Pada penelitian (Mulyan, iyan dkk, 2008) dilakukan penelitian tentang bagaimana penerapan *Term Frequency –Inverse Document Frequency* pada sistem peringkas dokumen tunggal. Metode ini digunakan untuk meringkas kalimat dengan cara pembobotan TF-IDF lalu menghitung bobot (w) masing-masing dokumen. Kemudian melakukan proses pengurutan nilai kumulatif dari W untuk setiap kalimat. Tiga kalimat dengan nilai W terbesar dijadikan sebagai hasil ringkasan. Hasil penelitian yang diperoleh bahwa metode TF-IDF dapat digunakan untuk meringkas *single document* dan memiliki tingkat akurasi 61%.

Pada penelitian Mustaqhfiri, dkk (2011) dilakukan penelitian tentang bagaimana meringkas dokumen menggunakan metode *Maximum Marginal Relevance*. Pada peringkasan dokumen dengan metode MMR dilakukan proses

segmentasi dokumen menjadi kalimat dan dilakukan pengelompokan sesuai dengan *gender* kalimat tersebut. MMR digunakan dengan mengkombinasikan matrik cosine similarity untuk merangking kalimat-kalimat sebagai tanggapan pada *query* yang diberikan oleh *user*. Hasil perhitungan evaluasi diurutkan berdasarkan nilai *recall, precision* dan *f-measure* dari *persentase* yang tertinggi ke urutan terendah.Dari hasil evaluasi antara ringkasan sistem dengan ringkasan manual maka menghasilkan nilai akurasi 70% pada compression 60%.

2.6.2 Hipotesa

Pada penelitian kali ini akan dilakukan riset bagaimana membangun aplikasi peringkas dokumen menggunakan metode *maximum marginal relevance*. Algoritma *Maximum Marginal Relevance* (MMR) digunakan untuk merangking kalimat-kalimat sebagai tanggapan terhadap *query* yang diberikan oleh *user*. Yang mana *Query* merupakan judul dari dokumen tersebut. Perhitungan MMR dilakukan dengan perhitungan iterasi antara kombinasi dua matrik *cosine similarity* yaitu relevansi antar *query* terhadap keseluruhan kalimat dan *similarity* antar kalimat dengan kalimat. kalimat dengan nilai MMR tertinggi dari setiap perhitungan iterasi akan diambil dan dipilih untuk dijadikan sebagai ringkasan. Iterasi akan berhenti ketika hasil mmr maksimum sama dengan 0 atau 1. Algoritma MMR dipilih karena memiliki hasil yang lebih akurat karena hasil ringkasan yang akan keluar berhubungan dengan *query* yang di inputkan (Mustaqfiri, 2011). Maka informasi yang didapatkan sesuai dengan *query* tersebut dan menampilkan *keyword* yang telah dirangking menggunakan metode TF-IDF.

2.7 *Algoritma* TF-IDF

Menurut (Mulyana dkk, 2012) *Term Frequency-Inverse Document Frequency* (TF-IDF) adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu *Term frequency* (TF) merupakan frekuensi kemunculan kata (*t*) pada kalimat (*d*). *Document frequency* (DF) adalah banyaknya kalimat dimana suatu kata (*t*) muncul.

Pembobotan dapat diperoleh berdasarkan jumlah kemunculan suatu *term* dalam sebuah dokumen *term frequency* (*tf*) dan jumlah kemunculan *term* dalam koleksi dokumen *inverse document frequency* (*idf*). Bobot suatu istilah semakin besar jika istilah tersebut sering muncul dalam suatu dokumen dan semakin kecil jika istilah tersebut muncul dalam banyak dokumen (Grossman, 1998). Nilai *idf* sebuah *term* (kata) dapat dihitung menggunakan persamaan sebagai berikut:

$$IDF = Log\left(\frac{D}{Dfi}\right) \quad (2.5)$$

D adalah jumlah dokumen yang berisi *term* (*t*) dan *dfi* adalah jumlah kemunculan (frekuensi) *term* terhadap D.

Adapun algoritma yang digunakan untuk menghitung bobot (W) masing-masing dokumen terhadap kata kunci (*query*) (Mustaqhfiri,2011), yaitu:

$$W_{d,t} = TF_{d,t} * IDF_t \quad (2.6)$$

Keterangan :

d = dokumen ke–d

t = kata ke–t dari kata kunci

tf = kata frekuensi/frekuensi kata

W = bobot dokumen ke–d terhadap kata ke-t

Setelah bobot (W) masing-masing dokumen diketahui, maka dilakukan proses pengurutan (*sorting*) dimana semakin besar nilai W, semakin besar tingkat kesamaan (*similarity)* dokumen tersebut terhadap kata yang dicari, demikian pula sebaliknya.

2.8 *Cosine Similarity*

cosine similarity adalah Suatu ukuran kemiripan yang paling umum digunakan untuk menghitung pendekatan relevansi *query* terhadap dokumen (Shasha Xie dkk,2008) Berikut adalah rumus *cosine similarity :*

$$sim\,(D_1,D_2) = \frac{\sum_i t_{1i} t_{2i}}{\sqrt{\sum_i t_{1i}^2} \times \sqrt{\sum_i t_{2i}^2}} \quad (2.7)$$

Keterangan:

T = *term* dalam kalimat

t1i = bobot term t dalam blok b1

t2i = bobot term t dalam blok b2

2.9 Teknik *text summarization* dengan *Maximum Marginal Relevance* (MMR)

Algoritma *Maximum Marginal Relevance* (MMR) merupakan salah satu metode ekstraksi ringkasan. MMR digunakan untuk meringkas dokumen dengan menghitung kesamaan (*simlarity*) antara bagian teks. Pada peringkasan dokumen dengan metode MMR dilakukan proses segmentasi dokumen menjadi kalimat dan dilakukan pengelompokan sesuai dengan gender kalimat tersebut (shasha, 2008). MMR digunakan dengan mengkombinasikan matrik *cosine similarity* untuk merangking kalimat-kalimat sebagai tanggapan pada *query* yang diberikan oleh user.

Kebanyakan mesin pencarian *information retrieval* (IR) modern menghasilkan daftar perangkingan dari dokumen yang diukur dari penurunan relevansi terhadap permintaan user (*user query*). Penaksiran pertama untuk mengukur hasil peringkasan yang *relevan* adalah dengan mengukur hubungan antar informasi yang ada dalam dokumen dengan *query* yang diberikan oleh *user* dan menambah kombinasi linier sebagai sebuah matrik

Sebuah dokumen dikatakan mempunyai *marginal relevance* yang tinggi jika dokumen tersebut relevan terhadap isi dari dokumen dan mempunyai kesamaan bobot term maksimum dibandingkan dengan *query*. Peringkasan dokumen dengan tipe ekstraktif, nilai akhir diberikan pada kalimat Si dalam MMR dihitung dengan persamaan :

$$MMR = argmax\ [\lambda * Sim_1(S_i, Q) - (1 - \lambda) * \max Sim_2(S_i, S')] \qquad (2.8)$$

Si adalah kalimat di dokumen, sedangkan *S*’ adalah kalimat yang telah dipilih atau telah diekstrak (Shasha Xie, 2008).Koefisien λ digunakan untuk mengatur kombinasi nilai untuk memberi penekanan bahwa kalimat tersebut

relevan dan untuk mengurangi redudansi. Pada penelitian ini, *Sim1* dan *Sim2* merupakan dua fungsi *similarity* yang merepresentasikan kesamaan kalimat pada seluruh dokumen dan memilih masing-masing kalimat untuk dijadikan ringkasan. *Sim1* adalah matrik *similarity* kalimat *Si* terhadap *query* yang diberikan oleh user sedangkan *Sim2* adalah matrik *similarity* kalimat *Si* terhadap kalimat yang telah diekstrak sebelumnya (Shasa Xie, 2008).

Nilai parameter λ adalah mulai dari 0 sampai dengan 1 (range [0,1]). Pada saat parameter λ = 1 maka nilai MMR yang diperoleh akan cenderung relevan terhadap dokumen asal. Ketika maka nilai *MMR* yang diperoleh cenderung relevan terhadap kalimat yang diekstrak sebelumnya. Oleh sebab itu sebuah kombinasi linier dari kedua kriteria dioptimalkan ketika nilai λ terdapat pada interval [0,1]. Untuk peringkasan *small* dokumen, seperti pada berita (*news*), menggunakan nilai parameter λ=0.7 atau λ=0.8, karena akan menghasilkan ringkasan yang baik (Jade Goldstein, 2008).

Untuk mendapatkan hasil ringkasan yang relevan maka harus menetapkan nilai λ ke nilai yang lebih dekat dengan λ. Kalimat dengan nilai MMR yang tertinggi akan dipilih berulang kali ke dalam ringkasan sampai tercapai ukuran ringkasan yang diinginkan.

2.10 Tahapan Algoritma pada *Automatic Document Summarization*

Tahapan algoritma adalah langkah-langkah dalam peringkasan dokumen. Berikut adalah tahapan algoritmanya :

1. *Text Preprocessing*

   Kalimat yang akan diringkas akan melakukan *text preprocessing* yaitu pemecahan kalimat, *case folding, filtering, tokenizing* kata dan *stemming.*
2. Pembobotan TF-IDF

   Setelah melakukan *text preprocessing* maka akan dilakukan pembobotan TF-IDF yang mana pembobotan ini berfungsi untuk menghitung bobot kata. Pembobotan kata juga digunakan untuk membuat *query expansion*.

3. Hitung bobot *relevance* dan bobot *similarity* kalimat

   Selanjutnya akan dilakukan perhitungan bobot *relevance* yang berguna untuk menghitung bobot *relevance* antar *query* terhadap seluruh kalimat dalam dokumen. Bobot *similarity* kalimat berguna untuk menghitung bobot *similarity* antar kalimat.

4. Hitung bobot MMR

   Tahap terakhir yaitu Ekstraksi ringkasan yang dilakukan dengan mengkombinasikan bobot relevansi dan *similarity* antar kalimat dengan proses *iterasi* MMR untuk mendapatkan bobot maksimum MMR. Dan dijadikan sebagai kalimat ringkasan.