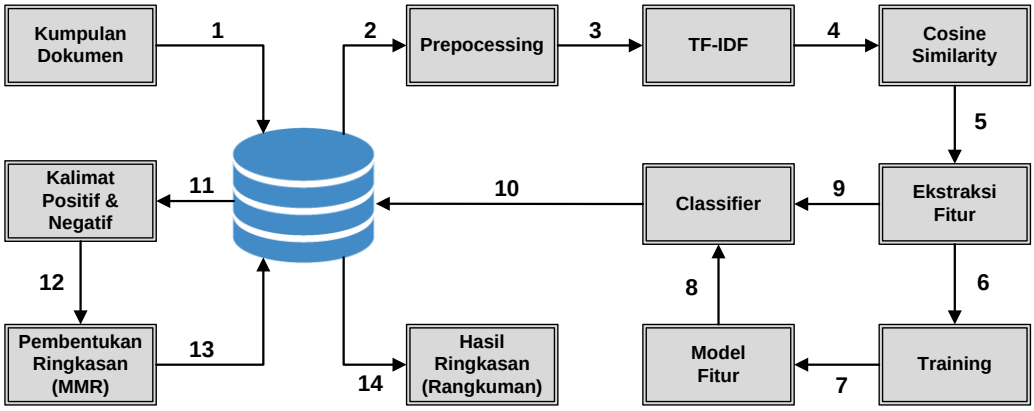
BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Analisis masalah adalah suatu gambaran masalah yang diangkat dalam menghasilkan aplikasi yang mampu menampilkan sebuah rangkuman dari kumpulan dokumen berita dengan satu topik sejenis, serta mengimplementasikan metode *Support Vector Machine* (SVM) dan metode *Maximum Marginal Relevance* (MMR) agar rangkuman yang dihasilkan relevan. Berdasarkan informasi yang mempunyai pandangan yang berbeda-beda tentang suatu topik yang sama, serta tidak hanya bersumber dari satu dokumen saja, melainkan dari beberapa dokumen yang ditulis dengan tata bahasa dan cara penyampaiannya yang berbeda-beda oleh media pemuat berita. Hal ini dapat menyulitkan dalam menyerap informasi penting, serta tidak ada waktu untuk membaca semua dokumen berita dari setiap media tersebut. Oleh karena itu, diperlukan suatu proses dengan menerapkan metode SVM dan MMR, sehingga dapat menghasilkan sebuah ringkasan berupa rangkuman relevan yang diperoleh dari kumpulan dokumen berita berbahasa Indonesia dengan satu topik sejenis.

3.2 Analisis Sistem

Analisis sistem berupa penguraian dari sistem utama ke dalam sub-sub sistem, dimana proses dalam menghasilkan rangkuman diuraikan dan diidentifikasi menjadi beberapa tahapan dalam sistem. Adapun gambaran umum sistem dari aplikasi yang akan dibangun dapat dilihat pada Gambar 3.1 berikut:



Gambar 3.1 Gambaran Umum Sistem

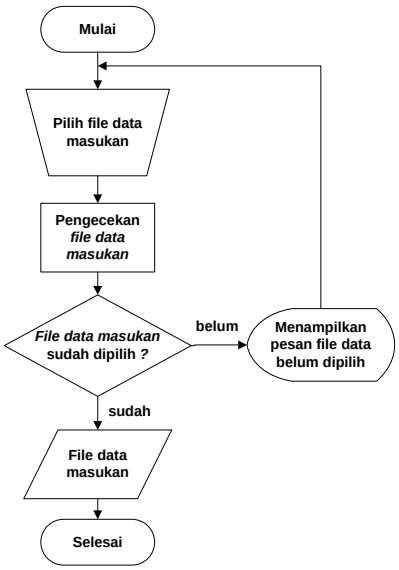
Berikut penjelasan lengkap dari Gambar 3.1 di atas:

1. Kumpulan dokumen diambil dari media *online* pemuat berita bahasa Indonesia yang dipilih berdasarkan satu topik sejenis dengan format *file* .doc.
2. Kumpulan dokumen yang dipilih pada tahap 1 dilakukan proses *Preprocessing* berupa (pemisahan kalimat, *case folding*, *filtering*, *tokenizing, stopword removal* dan *stemming*).
    a. Pemisahan Kalimat merupakan proses memecah dokumen menjadi perkalimat.
    b. *Case folding* merupakan proses mengubah semua huruf dalam dokumen menjadi huruf kecil semua.
    c. *Filtering* merupakan proses menghilangkan text selain "a" sampai dengan "z" dan hanya menerima spasi.
    d. *Tokenizing* merupakan proses pemotongan *string* text berdasarkan tiap kata yang menyusunnya.
    e. *Stopword removal* merupakan proses penghilangan *stopword* yang masuk dalam ke daftar *stopword*.
    f. *Stemming* merupakan proses mengembalikan kata yang berimbuhan ke dalam bentuk kata dasarnya.
3. TF-IDF merupakan proses pembobotan setiap *term* (kata).
4. *Cosine Similarity* merupakan proses perhitungan nilai kemiripan pada setiap kalimat dalam banyaknya dokumen
5. Ekstraksi Fitur dengan mengekstraksi dokumen dengan cara mengubah seluruh kalimat ke dalam vektor fitur.
6. Dilakukan proses SVM *Training* menggunakan vektor fitur untuk mendapatkan model fitur.
7. Model fitur yang terbentuk hasil dari tahap proses SVM *Training*.
8. Tahap ini masuk ke dalam tahap SVM *Classifier* yaitu mengklasifikasikan data uji.
9. Tahap ini merupakan tahap SVM *Classifier* yaitu vektor fitur yang terbentuk diuji dan diklasifikasikan dengan menggunakan model fitur yang telah diperoleh pada proses SVM *Training*.

10. Data hasil SVM *Classifier* yang dibagi ke dalam kelas positif dan kelas negatif.
11. Data hasil klasifikasi kalimat kelas positif dan kalimat kelas negatif.
12. Tahap dimana proses pembentukan ringkasan dengan metode MMR dengan mengambil data kalimat kelas positif yang akan menjadi kandidat kalimat ringkasan.
13. Menampilkan hasil ringkasan berupa rangkuman singkat beserta nilai performansi *recall*, *precision*, dan *f-measure.*

3.3 Analisis Data Masukan

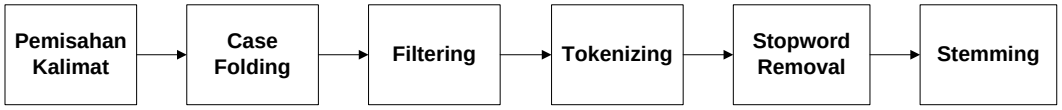
Analisis data masukan adalah dokumen-dokumen berita yang diambil dari media *online* pemuat berita berbahasa Indonesia, yang dipilih berdasarkan satu topik sejenis dengan format *file* .doc. Adapun contoh berita yang dimuat media online pemuat berita berbahasa Indonesia dengan satu topik sejenis, tetapi hanya ditulis dengan tata bahasa dan cara penyampaiannya yang berbeda-beda. Topik berita yang dipilih berupa "Malaysia meliburkan sekolah selama dua hari karena udara tidak sehat", serta dipilih 4 contoh media *online* yang memiliki kemiripan dalam memuat berita dengan topik tersebut, dapat dilihat pada lampiran data masukan. Adapun *flowchart* pemilihan data masukan dapat dilihat pada Gambar 3.2 berikut:



Gambar 3.2 *Flowchart* Pemilihan Data Masukan

3.4 Analisis Proses *Preprocessing*

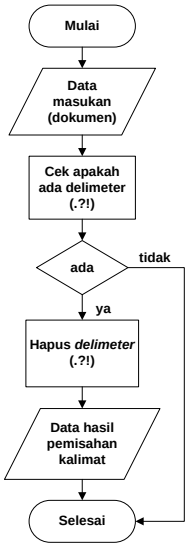
Analisis tahap *Preprocessing* merupakan tahap awal yang dilakukan. *Preprocessing* adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. *Input*-an awal pada proses ini adalah berupa dokumen. *Preprocessing* pada penelitian ini terdiri dari beberapa tahapan, yaitu: proses pemisahan kalimat, proses *case folding*, proses *filtering*, proses *tokenizing*, proses *stopword removal* dan proses *stemming*. Berikut gambaran tahap *preprocessing* dapat dilihat pada Gambar 3.3 berikut:



Gambar 3.3 Proses *Preprocessing*

3.4.1 Pemisahan Kalimat

Pemisahan kalimat adalah proses memecah teks pada dokumen menjadi kumpulan kalimat-kalimat yang merupakan langkah awal tahapan *text preprocessing*. Teknik yang digunakan dalam pemisahan kalimat adalah memisahkan kalimat dengan tanda titik (.), tanda tanya (?), dan tanda seru (!) sebagai pemisah (*delimiter).* Menghilangkan *delimeter* tersebut dokumen akan terpotong menjadi kalimat[6]. Adapun *flowchart* dari proses *preprocessing* pemisahan kalimat dapat dilihat pada Gambar 3.4 berikut:



Gambar 3.4 *Flowchart* Proses *Preprocessing* Pemisahan Kalimat

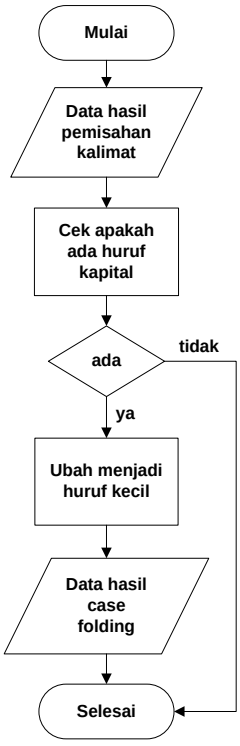
Adapun contoh proses *preprocessing* untuk pemisahan kalimat dapat dilihat pada Tabel 3.1 berikut:

Tabel 3.1 Contoh Pemisahan Kalimat

Sebuah Paragraf	Hasil Pemisahan Kalimat	
Malaysia memerintahkan penutupan sebagian sekolah selama 2 hari, akibat kabut asap tebal menyelimuti kawasan Asia Tenggara. Akibat kebakaran hutan di Indonesia.	No	Kalimat
	1	Malaysia memerintahkan penutupan sebagian sekolah selama 2 hari, akibat kabut asap tebal menyelimuti kawasan Asia Tenggara
	2	Akibat kebakaran hutan di Indonesia

3.4.2 *Case Folding*

Proses *case folding* adalah tahapan pemrosesan teks dimana semua teks diubah ke dalam *case* yang sama. Pada penelitian ini semua huruf dalam teks dokumen diubah representasinya menjadi huruf kecil semua[6]. Adapun *flowchart* dari proses *preprocessing case folding* dapat dilihat pada Gambar 3.5 berikut:



Gambar 3.5 *Flowchart* Proses *Preprocessing Case Folding*

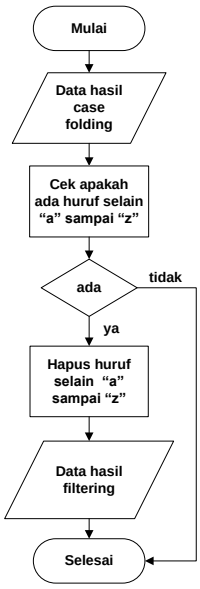
Adapun contoh proses *preprocessing* untuk *case folding* dapat dilihat pada Tabel 3.2 berikut:

Tabel 3.2 Contoh *Case Folding*

Sebelum	Sesudah *Case Folding*
Malaysia memerintahkan penutupan sebagian sekolah selama 2 hari, akibat kabut asap tebal menyelimuti kawasan Asia Tenggara	malaysia memerintahkan penutupan sebagian sekolah selama 2 hari akibat kabut asap tebal menyelimuti kawasan asia tenggara

3.4.3 *Filtering*

Proses *filtering* adalah tahapan pemrosesan teks dimana semua teks selain karakter "a" sampai "z" akan dihilangkan dan hanya menerima spasi[6]. Adapun *flowchart* dari proses *preprocessing filtering* dapat dilihat pada Gambar 3.6 berikut:



Gambar 3.6 *Flowchart* Proses *Preprocessing Filtering*

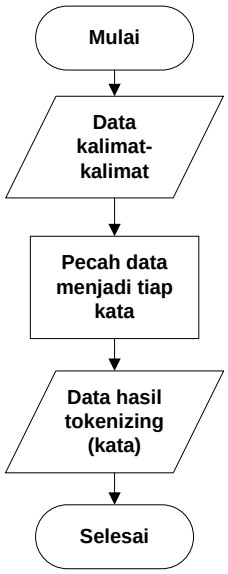
Adapun contoh proses *preprocessing* untuk *filtering* dapat dilihat pada Tabel 3.3 berikut:

Tabel 3.3 Contoh *Filtering*

Sebelum	Sesudah *Case Folding*
malaysia memerintahkan penutupan sebagian sekolah selama 2 hari, akibat kabut asap tebal menyelimuti kawasan asia tenggara	malaysia memerintahkan penutupan sebagian sekolah selama hari akibat kabut asap tebal menyelimuti kawasan asia tenggara

3.4.4 *Tokenizing*

Proses *Tokenizing* adalah proses pemotongan *string input* berdasarkan tiap kata yang menyusunnya, memecahkan data kalimat dan memisahkannya menjadi setiap kata. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan men-*scan* kalimat dan setiap kata terindentifikasi atau terpisahkan dengan kata yang lain oleh pemisah spasi[7]. Adapun *flowchart* dari proses *preprocessing tokenizing* dapat dilihat pada Gambar 3.7 berikut:



Gambar 3.7 *Flowchart* Proses *Preprocessing Tokenizing*

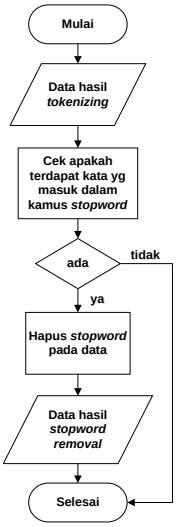
Adapun contoh proses *preprocessing* untuk *tokenizing* dapat dilihat pada Tabel 3.4 berikut:

Tabel 3.4 Contoh *Tokenizing*

Sebelum	Sesudah *Tokenizing*	
malaysia memerintahkan penutupan sebagian sekolah selama hari akibat kabut asap tebal menyelimuti kawasan asia tenggara	• malaysia • memerintahkan • penutupan • sebagian • sekolah • selama • hari • akibat	• kabut • asap • tebal • menyelimuti • kawasan • asia • tenggara

3.4.5 *Stopword Removal*

Proses *stopword removal* merupakan proses penghilangan *stopword*, menghilangkan kata-kata yang masuk ke dalam daftar kamus *stopword*. *Stopword* adalah kata-kata yang sering kali muncul berupa kata sambung, kata depan, kata ganti, kata penghubung, dll. Namun artinya tidak deskriptif dan tidak memiliki keterkaitan dengan topik tertentu[6]. Adapun *flowchart* dari proses *preprocessing stopword removal* dapat dilihat pada Gambar 3.8 berikut:



Gambar 3.8 *Flowchart* Proses *Preprocessing Stopword Removal*

Adapun contoh proses *preprocessing* untuk *stopword removal* dapat dilihat pada Tabel 3.5 berikut:

Tabel 3.5 Contoh *Stopword Removal*

Sebelum		Sesudah *Stopword Removal*	
• malaysia • memerintahkan • penutupan • sebagian • sekolah • selama • hari • akibat	• kabut • asap • tebal • menyelimuti • kawasan • asia • tenggara	• malaysia • memerintahkan • penutupan • sekolah • hari • kabut • asap • tebal	• menyelimuti • kawasan • asia • tenggara

3.4.6 *Stemming*

Proses *Stemming* merupakan proses pencarian akar kata (*root word*) dari tiap kata yaitu dengan mengembalikan suatu kata berimbuhan atau yang memiliki imbuhan ke bentuk dasarnya (*stem*). Untuk pemrosesan pada bahasa Indonesia, proses *stemming* dilakukan dengan menghilangkan imbuhan yang mengawali dan mengakhiri kata sehingga diperoleh bentuk dasar dari kata tersebut[7]. Adapun *flowchart* dari proses *preprocessing stemming* dapat dilihat pada Gambar 3.9 berikut:



Gambar 3.9 *Flowchart* Proses *Preprocessing Stemming*

Adapun contoh proses *preprocessing* untuk *stemming* dapat dilihat pada Tabel 3.6 berikut:

Tabel 3.6 Contoh *Stemming* Algoritma *Porter Stemmer* Bahasa Indonesia

No	Tahap	Kata	Keterangan
1	Menghapus partikel (*particle*)	mempersiapkan	Tidak ada di kamus dan tidak ada partikel.
2	Menghapus kata ganti (*possesive pronoun*)	mempersiapkan	Tidak ada di kamus dan tidak ada *possesive pronoun*.
3	Menghapus awalan pertama (*first order prefix*), jika ditemukan maka lanjutkan ke langkah 4a, jika tidak maka lanjutkan ke langkah 4b	persiapkan	Tidak ada di kamus tetapi terdapat *first order prefix* "mem-" dihapus.
4a	Menghapus awalan kedua (*second order prefix*) kemudian lanjutkan ke langkah 5a	siapkan	Tidak ada di kamus tetapi terdapat *second order prefix* "per-" dihapus
4b	Menghapus akhiran (*suffix*), jika ditemukan lanjutkan ke langkah 5b, jika tidak maka kata tersebut diasumsikan sebagai kata dasar		
5a	Menghapus akhiran (*suffix*) dan kata akhir diasumsikan sebagai kata dasar	siap	Tidak ada di kamus tetapi terdapat *suffix* "-kan" dihapus dan kata akhir diasumsikan sebagai kata dasar
5b	Menghapus awalan kedua (*second order prefix*) dan kata akhir diasumsikan sebagai kata dasar		

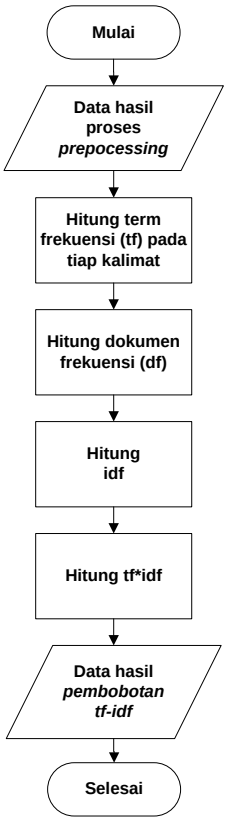
Hasil proses *stemming* tersebut digunakan dalam melakukan pembobotan tf-idf dan perhitungan *cosine similarity* untuk *similarity* kalimat. *Similarity* kalimat merupakan bobot hasil perbandingan kemiripan antar kalimat.

3.5 Analisis Pembobotan TF-IDF

Term Frequency – Inverse Document Frequency digunakan untuk menentukan nilai frekuensi sebuah kata di dalam dokumen. Perhitungan statistik numerik yang dimaksudkan untuk mencerminkan betapa pentingnya dan seberapa

relevannya sebuah kata di dalam sebuah dokumen. Prosedur dalam implementasi TF-IDF terdapat perbedaan kecil di dalam semua aplikasinya seperti pada banyaknya kalimat atau banyaknya dokumen, tetapi pendekatannya kurang lebih sama[11].

Pada penelitian ini pembobotan diperoleh dari frekuensi sebuah kata yang terdapat di dalam sebuah kalimat atau jumlah kemunculan *term* dalam satu dokumen *term frequency* (tf) dan sebuah kata di dalam kumpulan dokumen atau jumlah kemunculan *term* dalam koleksi dokumen *inverse document frequency* (idf). Nilai idf sebuah *term* (kata) dapat dihitung menggunakan persamaan (2.1). Untuk menghitung bobot (W) masing-masing dokumen terhadap setiap *term* (kata) dapat menggunakan persamaan (2.2). Adapun *flowchart* dari proses pembobotan tf-idf dapat dilihat pada Gambar 3.10 berikut:



Gambar 3.10 *Flowchart* Proses Pembobotan TF-IDF

Sebagai contoh dilakukan peringkasan terhadap dokumen pertama, diasumsikan *term-term* sudah didapatkan dari dokumen tersebut, berikut adalah contoh pembobotan tf-idf dengan S_i adalah kalimat seperti pada tabel Tabel 3.7:

Tabel 3.7 *Term* dari Kalimat pada Dokumen Pertama

Kalimat	*Term*
S1	pemerintah malaysia libur sekolah dua hari lindung anak kabut asap asal indonesia senin
S2	indeks polusi udara wilayah negeri jiran sentuh level sangat tidak sehat
S3	otoritas paksa tutup sekolah hari selasa besok
S4	shah alam selangor indeks polusi udara turun cukup tajam
S5	angka indeks aman moderat tidak sehat sangat bahaya
S6	dapat area malaysia utama wilayah utara level tidak sehat buruk penang level
S7	kabut asap paksa berapa bandara malaysia tutup minggu
S8	acara lari maraton kuala lumpur batal

Proses pertama lakukan perhitungan kata (*term)* yang ada pada setiap kalimat, hasilnya dapat dilihat pada lampiran tabel perhitungan Tabel 1 *Term Frequency*. Kemudian hitung df, karena perhitungan dilakukan pada sebuah dokumen dan yang dihitung *term* pada kalimat dalam dokumen tersebut, maka df = jumlah *term* yang ada pada setiap kalimat, hasilnya dapat dilihat pada lampiran tabel perhitungan Tabel 2 Nilai DF. Selanjutnya menghitung idf, maka hasil perhitungan dapat dilihat pada lampiran tabel perhitungan Tabel 3 Nilai IDF.

Kemudian hasil tf dan idf kembali dilakukan perhitungan menggunakan persamaan (2.2). Adapun hasil yang didapatkan dari perhitungan pembobotan TF-IDF dapat dilihat pada Tabel 3.8 berikut:

Tabel 3.8 Hasil Pembobotan TF-IDF

Term	*Weight*							
	Kalimat							
	S1	S1	S1	S1	S1	S1	S1	S1
pemerintah	0.903	0	0	0	0	0	0	0
malaysia	0.426	0	0	0	0	0.426	0.426	0
libur	0.903	0	0	0	0	0	0	0
sekolah	0.602	0	0.602	0	0	0	0	0
dua	0.903	0	0	0	0	0	0	0
hari	0.602	0	0.602	0	0	0	0	0
lindung	0.903	0	0	0	0	0	0	0

Term	*Weight*							
	Kalimat							
	S1	S1	S1	S1	S1	S1	S1	S1
anak	0.903	0	0	0	0	0	0	0
kabut	0.602	0	0	0	0	0	0.602	0
asap	0.602	0	0	0	0	0	0.602	0
asal	0.903	0	0	0	0	0	0	0
indonesia	0.903	0	0	0	0	0	0	0
senin	0.903	0	0	0	0	0	0	0
indeks	0	0.426	0	0.426	0.426	0	0	0
polusi	0	0.602	0	0.602	0	0	0	0
udara	0	0.602	0	0.602	0	0	0	0
wilayah	0	0.602	0	0	0	0.602	0	0
negeri	0	0.903	0	0	0	0	0	0
jiran	0	0.903	0	0	0	0	0	0
sentuh	0	0.903	0	0	0	0	0	0
level	0	0.426	0	0	0	0.852	0	0
sangat	0	0.602	0	0	0.602	0	0	0
tidak	0	0.426	0	0	0.426	0.426	0	0
sehat	0	0.426	0	0	0.426	0.426	0	0
otoritas	0	0	0.903	0	0	0	0	0
paksa	0	0	0.602	0	0	0	0.602	0
tutup	0	0	0.602	0	0	0	0.602	0
selasa	0	0	0.903	0	0	0	0	0
besok	0	0	0.903	0	0	0	0	0
shah	0	0	0	0.903	0	0	0	0
alam	0	0	0	0.903	0	0	0	0
selangor	0	0	0	0.903	0	0	0	0
turun	0	0	0	0.903	0	0	0	0
cukup	0	0	0	0.903	0	0	0	0
tajam	0	0	0	0.903	0	0	0	0
angka	0	0	0	0	0.903	0	0	0
aman	0	0	0	0	0.903	0	0	0
moderat	0	0	0	0	0.903	0	0	0
bahaya	0	0	0	0	0.903	0	0	0
dapat	0	0	0	0	0	0.903	0	0

Term	*Weight*							
	Kalimat							
	S1	S1	S1	S1	S1	S1	S1	S1
area	0	0	0	0	0	0.903	0	0
utama	0	0	0	0	0	0.903	0	0
utara	0	0	0	0	0	0.903	0	0
buruk	0	0	0	0	0	0.903	0	0
penang	0	0	0	0	0	0.903	0	0
berapa	0	0	0	0	0	0	0.903	0
bandara	0	0	0	0	0	0	0.903	0
minggu	0	0	0	0	0	0	0.903	0
acara	0	0	0	0	0	0	0	0.903
lari	0	0	0	0	0	0	0	0.903
maraton	0	0	0	0	0	0	0	0.903
kuala	0	0	0	0	0	0	0	0.903
lumpur	0	0	0	0	0	0	0	0.903
batal	0	0	0	0	0	0	0	0.903

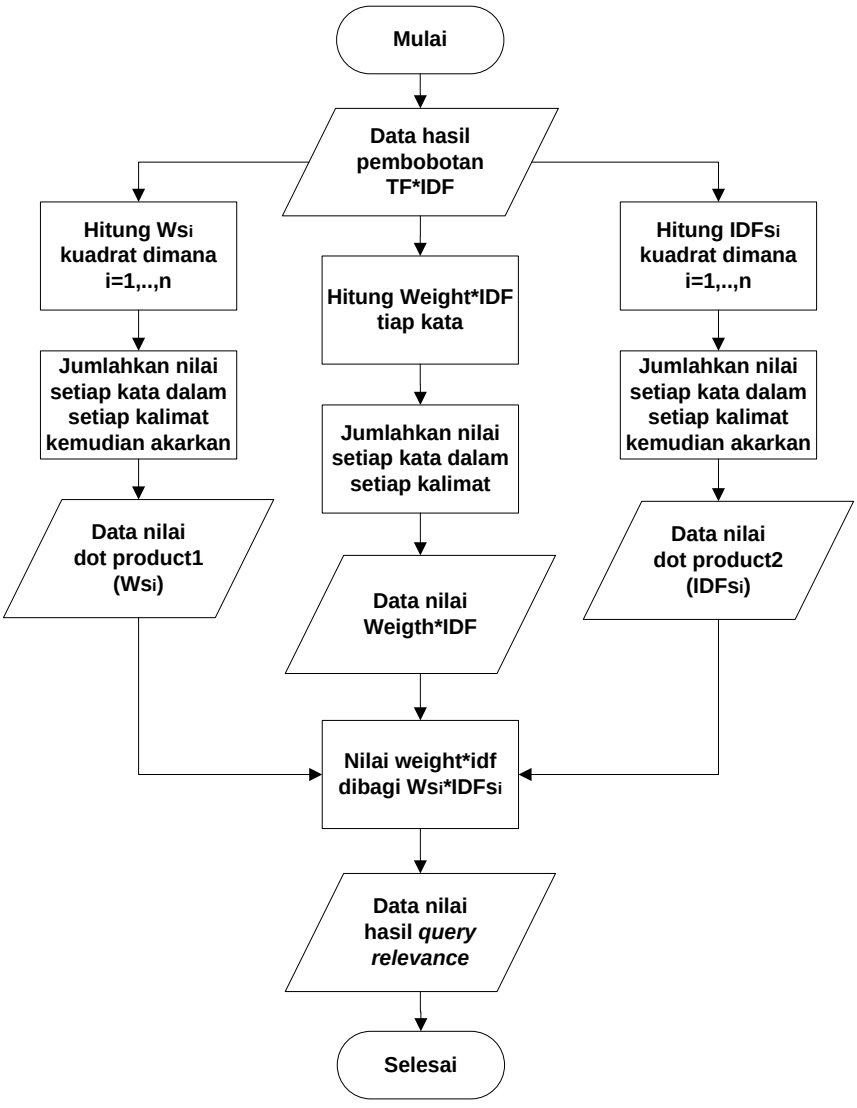
Hasil pembobotan tf-idf dilakukan untuk memberi bobot pada setiap kata (*term*). Kemudian hasil dari pembobotan tf-idf tersebut digunakan sebagai langkah dalam melakukan perhitungan *cosine similarity* untuk bobot *query relevance*. *Query relevance* merupakan bobot hasil perbandingan kemiripan (similaritas) antara vektor *query* terhadap setiap kalimat.

3.6 Analisis Perhitungan *Cosine Similarity*

Cosine Similarity digunakan untuk menghitung pendekatan relevansi *query* terhadap dokumen. Semakin banyak *term* yang muncul dalam dokumen, maka *term* tersebut dipandang sebagai vektor *query*. *Cosine similarity* digunakan dalam ruang positif, dimana hasilnya dibatasi dengan *range* nilai 0 sampai 1[11].

Untuk mendapatkan nilai *query relevance*, pertama dilakukan perkalian antara nilai idf dengan *term* pada setiap kalimat (S_i) dari hasil pembobotan tf-idf, kemudian jumlahkan. Hasil pada langkah pertama dapat dilihat pada lampiran tabel perhitungan Tabel 4 Hasil Perhitungan *Cosine Similarity* Langkah Pertama. Langkah kedua, hitung panjang setiap kalimat. Caranya, kuadratkan bobot setiap

term dalam setiap kalimat, kemudian jumlahkan nilai kuadrat dan terakhir akarkan. Hasil pada langkah kedua merupakan W_{Si} (*dot product1*) dimana $i = 1,..,n$ dapat dilihat pada lampiran tabel perhitungan Tabel 5 Hasil Perhitungan *Cosine Similarity* Langkah Kedua. Langkah ketiga, cari IDF_{Si} (*dot product2*) dimana $i = 1,..,n$ dengan cara kuadratkan setiap nilai IDF_{Si}, kemudian jumlahkan nilai kuadrat dan terakhir akarkan. Hasil pada langkah ketiga dapat dilihat pada lampiran tabel perhitungan Tabel 6 Hasil Perhitungan *Cosine Similarity* Langkah Ketiga. Langkah terakhir, terapkan rumus *cosine similarity* menggunakan persamaan (2.3). Adapun *flowchart* dari proses perhitungan *cosine similarity* untuk *query relevance* dapat dilihat pada Gambar 3.11 berikut:



Gambar 3.11 *Flowchart* Perhitungan *Query Relevance*

Sebagai contoh dilakukan perhitungan pada kalimat pertama (S1) berikut:

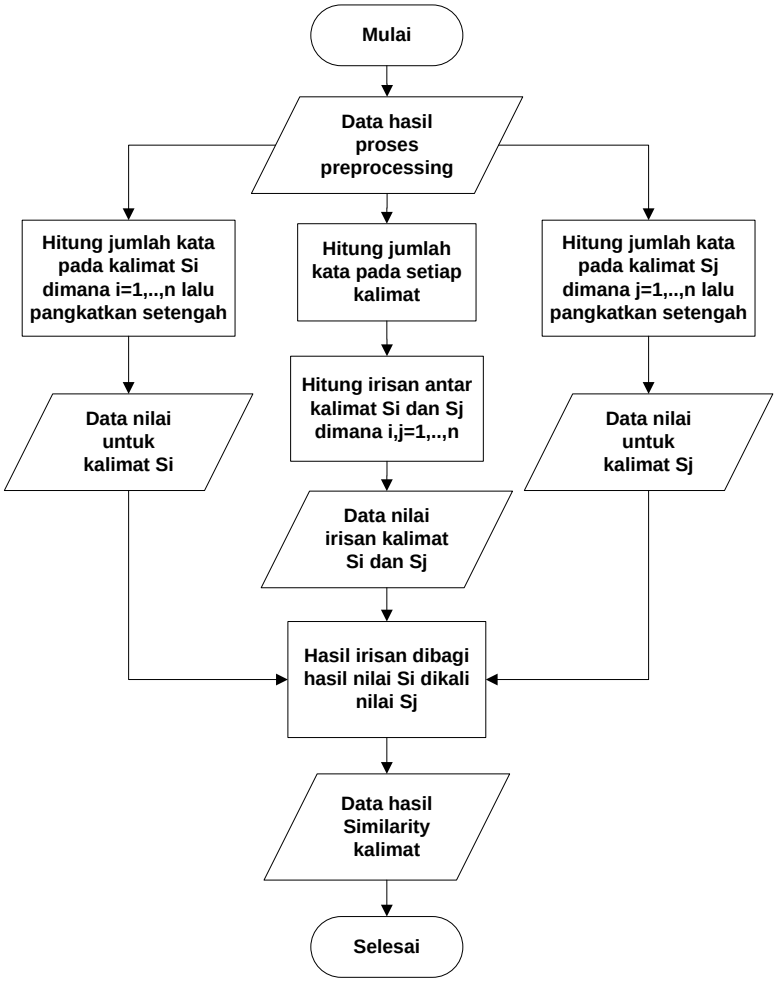
CS = $(W_{S1}, IDF_{S1}) = \frac{8.155927}{(2.855858 * 6.028209)} = 0.473749$

Setelah perhitungan dilakukan pada seluruh kalimat, maka hasil dari perhitungan *query relevance* didapatkan hasilnya pada Tabel 3.9 berikut:

Tabel 3.9 Bobot *Query Relevance*

Kalimat	S1	S2	S3	S4	S5	S6	S7	S8
CS = (d1,d2)	0.474	0.357	0.327	0.400	0.339	0.400	0.335	0.367

Untuk mendapatkan bobot matriks *similarity* kalimat digunakan persamaan (2.4) dengan data jumlah *term* pada setiap kalimat sesuai Tabel 3.7. Adapun *flowchart* dari proses perhitungan *cosine similarity* untuk *similarity* kalimat dapat dilihat pada Gambar 3.12 berikut:



Gambar 3.12 *Flowchart* Perhitungan *Similarity* Kalimat

Sebagai contoh perhitungan dilakukan pada kalimat pertama (S1) sebagai berikut:

$$CS(S1,S1) = \frac{13}{13^{\frac{1}{2}} * 13^{\frac{1}{2}}} = 1$$

Setelah dilakukan perhitungan pada seluruh kalimat, maka hasil dari perhitungan *similarity* kalimat dapat dilihat pada Tabel 3.10 berikut:

Tabel 3.10 Bobot Matriks *Similarity* Kalimat

Kalimat	S1	S2	S3	S4	S5	S6	S7	S8
S1	1	0	0.21	0	0	0.08	0.29	0
S2	0	1	0	0.3	0.43	0.35	0	0
S3	0.21	0	1	0	0	0	0.27	0
S4	0	0.3	0	1	0.12	0.19	0	0
S5	0	0.43	0	0.12	1	0.2	0	0
S6	0.08	0.35	0	0.19	0.2	1	0.1	0
S7	0.29	0	0.27	0	0	0.1	1	0
S8	0	0	0	0	0	0	0	1

Perhitungan *cosine similarity* akan menghasilkan nilai kemiripan pada setiap kalimat dalam banyaknya dokumen, dengan melihat kesamaan vektor *query* dengan vektor dokumen. Nilai hasil bobot *query relevance* akan digunakan sebagai acuan untuk memberi label dalam klasifikasi kalimat menggunakan metode SVM pada perhitungan SVM *Training* dan perhitungan pembobotan MMR awal, sedangkan nilai hasil dari perhitungan matriks *similarity* kalimat ini digunakan dalam perhitungan iterasi max pembobotan mmr untuk menghasilkan rangkuman

Klasifikasi kalimat menggunakan Metode SVM dilakukan untuk memilih kandidat kalimat yang akan dimasukkan ke dalam suatu ringkasan berupa rangkuman. Untuk kelas positif diberi label +1 yang artinya kalimat tersebut berpengaruh besar terhadap hasil ringkasan, sedangkan untuk kelas negatif diberi label -1 yang artinya kalimat tersebut tidak berpengaruh besar terhadap hasil ringkasan. Sedemikian sehingga kalimat yang masuk ke dalam kelas positif lah yang masuk kedalam sebuah ringkasan. Klasifikasi kalimat dengan metode ini dilihat dari data uji dengan vektor kalimat, jika nilai vektor melebihi nilai *hyperplane* maka kalimat tersebut masuk ke dalam kelas positif dan jika tidak maka kalimat tersebut masuk ke dalam kelas negatif.

3.7 Analisis Klasifikasi Kalimat Menggunakan Metode SVM

Tahap klasifikasi menggunakan metode SVM dalam mengklasifikasikan kalimat ke dalam kelas positif dan negatif. Data yang akan dihitung merupakan data kalimat dalam setiap dokumen. Proses klasifikasi terbagi ke dalam 2 proses, yaitu proses *training* dan *classifier*. Proses *training* dilakukan untuk menghasilkan model fitur yang nantinya akan digunkan sebagai acuan perhitungan untuk mengklasifikasikan kalimat ke dalam dua kelas positif atau negatif.

Data masukan secara bersamaan akan diubah menjadi data vektor. Pada penelitian ini untuk format representasi data menggunakan format SVM *light*. Format data *input* untuk klasifikasi SVM dalam penelitian ini adalah 1 1:0.39794. Masukan yang pertama +1 atau -1 menyatakan dua label awalan yang diberikan. Angka 1 pada karakter pertama menyatakan data tersebut masuk dalam label positif. Angka 1 berikutnya sebelum tanda ":" menyatakan indeks dan angka 0.39794 setelah karakter ":" menyatakan bobot tf-idf dari *term* tersebut. Sebagai contoh dalam pengubahan data teks dalam sebuah kalimat menjadi data vektor dapat dilihat pada Tabel 3.11 berikut:

Tabel 3.11 Pengubahan Data Kalimat Menjadi Format Data Vektor

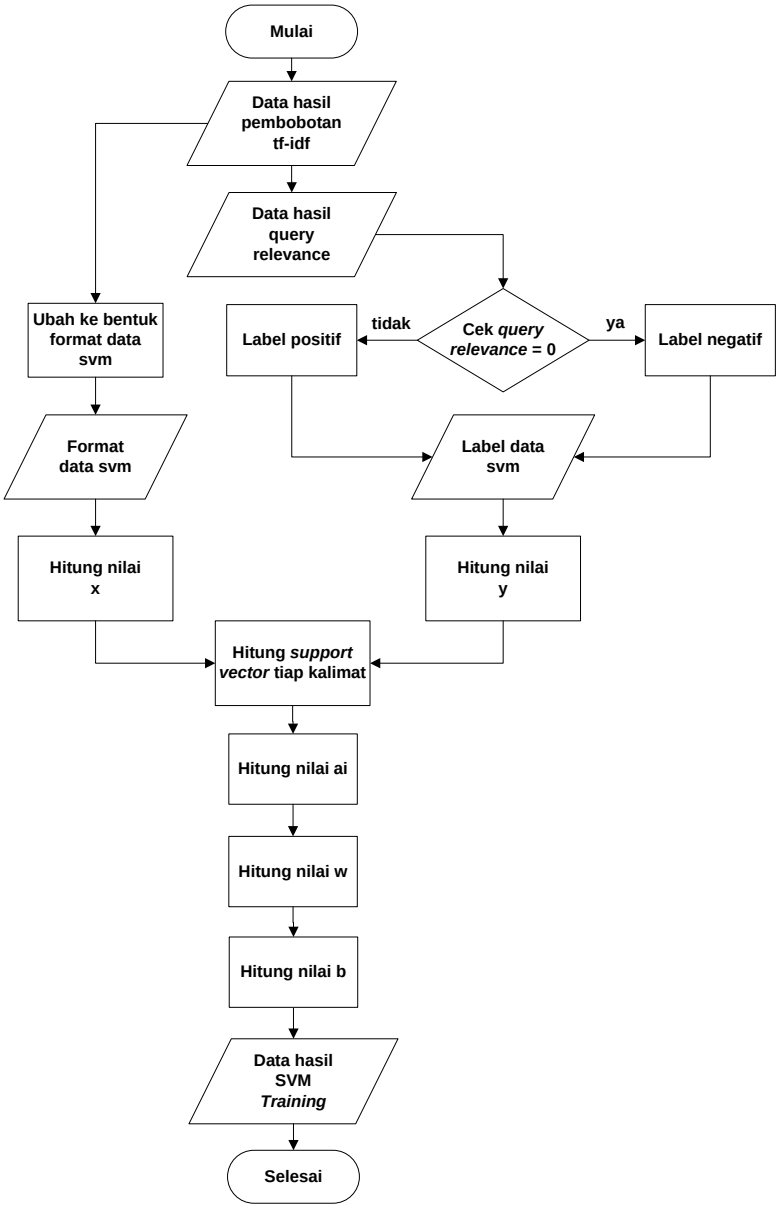
Kalimat	pemerintah malaysia libur sekolah dua hari lindung anak kabut asap asal indonesia senin
Vektor	[1 1:0.903 2:0.426 3:0.903 4:0.602 5:0.903 6:0.602 7:0.903 8:0.903 9:0.602 10:0.602 11:0.903 12:0.903 13:0.903]

Begitu seterusnya sehingga semua kata dalam kalimat terwakili oleh format data vektor. Untuk kata yang sama muncul lebih dari sekali dalam sebuah kalimat akan diwakili oleh sebuah data vektor saja dengan nilai tf-idf yang bersesuaian.

3.7.1 Analisis SVM *Training*

Pada proses pelatihan dibutuhkan label positif dan negatif, dimana penilaian positif atau tidaknya suatu kalimat dilihat dari hasil perhitungan bobot *query relevance* pada Tabel 3.9. Kalimat dengan nilai bobot *query relevance* > 0 diberi label positif, sedangkan kalimat dengan bobot *query relevance* = 0 diberi label negatif, hal ini dikarenakan kalimat tersebut tidak memiliki kerelevanan dengan

query. Kelas positif berlabel +1 yang artinya kalimat tersebut berpengaruh besar terhadap hasil ringkasan, sedangkan untuk kelas negatif berlabel -1 yang artinya kalimat tersebut tidak berpengaruh besar terhadap hasil ringkasan. Adapun *flowchart* klasifikasi kalimat menggunakan metode SVM pada proses *training* dapat dilihat pada Gambar 3.13 berikut:



Gambar 3.13 *Flowchart* SVM *Training*

Untuk contoh kasus pada tahap ini data disesuaikan dengan perhitungan yang melalui pembobotan tf-idf dan *cosine similarity* sebelumnya, sehingga hasilnya didapat sesuai Tabel 3.12 berikut:

Tabel 3.12 Proses SVM *Training*

Kalimat	*Term*	Label
S1	pemerintah malaysia libur sekolah dua hari lindung anak kabut asap asal indonesia senin	1
S2	indeks polusi udara berbagai wilayah negeri jiran sentuh level sangat tidak sehat	1
S3	otoritas paksa tutup sekolah hari selasa besok	1
S4	shah alam selangor indeks polusi udara turun cukup tajam	1
S5	angka indeks aman moderat tidak sehat sangat bahaya	1
S6	dapat area malaysia utama wilayah utara level tidak sehat buruk penang level	1
S7	kabut asap paksa berapa bandara malaysia tutup minggu	1
S8	acara lari maraton kuala lumpur batal	1

Dari Tabel 3.8 didapat hasil pembobotan tf-idf diketahui bobot setiap kata (*term*) dari semua kalimat. Selanjutnya data tersebut diubah formatnya ke dalam format data SVM, dapat dilihat pada lampiran tabel perhitungan Tabel 7 Format Data SVM. Untuk label dalam setiap kalimat dapat dilihat pada lampiran tabel perhitungan Tabel 8 Label Data SVM. Kemudian tahap selanjutnya adalah melakukan perhitungan $x_i x_j^T$ dimana $i, j = 1, .., n$. Untuk x_1 adalah seluruh nilai yang diambil dari kolom S1, x_2 = S2 sampai x_8 = S8. Sedemikian sehingga setiap kalimat untuk nilai x_1 dapat diubah sesuai hasil pembobotan tf-idf menjadi bentuk sebagai berikut, sebagai contoh bentuk pada kalimat pertama:

x_1 = [0.903, 0.426, 0.903, 0.602, 0.903, 0.602, 0.903, 0.903, 0.602, 0.602, 0.903, 0.903, 0.903, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Kemudian data vektor setiap kalimat sesuai dengan hasil pembobotan tf-idf pada setiap kalimatnya. Dilakukan perhitungan $x_i x_j^T$ pada setiap kalimat, sebagai contoh dilakukan perhitungan pada kalimat pertama sebagai berikut:

$x_1 x_1^T$ = [0.903, 0.426, 0.903, 0.602, 0.903, 0.602, 0.903, 0.903, 0.602, 0.602, 0.903, 0.903, 0.903, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [0.903, 0.426, 0.903, 0.602,

0.903, 0.602, 0.903, 0.903, 0.602, 0.602, 0.903, 0.903, 0.903, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T

= 0.816 + 0.181 + 0.816 + 0.362 + 0.816 + 0.362 + 0.816 + 0.816 + 0.362 + 0.362+ 0.816 + 0.816 + 0.816

= 8.156

Setelah dilakukan perhitungan pada seluruh kalimat, maka matriks yang terbentuk dari hasil perhitungan $x_i x_j^T$ adalah sebagai berikut:

$$x_i x_j^T = \begin{bmatrix} 8.156 & 0 & 0.725 & 0 & 0 & 0.181 & 0.906 & 0 \\ 0 & 4.622 & 0 & 0.906 & 0.907 & 1.088 & 0 & 0 \\ 0.725 & 0 & 3.987 & 0 & 0 & 0 & 0.725 & 0 \\ 0 & 0.906 & 0 & 5.8 & 0.181 & 0 & 0 & 0 \\ 0 & 0.907 & 0 & 0.181 & 4.169 & 0.363 & 0 & 0 \\ 0.181 & 1.088 & 0 & 0 & 0.363 & 6.526 & 0.181 & 0 \\ 0.906 & 0 & 0.725 & 0 & 0 & 0.181 & 4.078 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4.893 \end{bmatrix}$$

Kemudian tahap selanjutnya adalah melakukan perhitungan $y_i y_j^T$ dimana $i,j = 1,..,n$. Untuk nilai y adalah nilai dari label yang diberikan. Sesuai pada lampiran tabel perhitungan Tabel 8 Label Data SVM, sehingga matriks yang terbentuk dari hasil perhitungan $y_i y_j^T$ adalah hasilnya sebagai berikut:

$$y_i y_j^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Kemudian tahap selanjutnya adalah mencari nilai a_i, didapatkan dengan menggunakan persamaan (2.25). Proses mendapatka nilai a_i diawali dengan mengubah setiap kalimat menjadi nilai vektor (*support vector*) = $\binom{x}{y}$. Nilai x didapatkan menggunakan persamaan (2.23) sehingga didapatkan untuk nilai x pada setiap kalimat pada Tabel 3.13 berikut:

Tabel 3.13 Nilai x Setiap Kalimat

Kalimat	S1	S2	S3	S4	S5	S6	S7	S8
X	9.968	7.523	5.437	6.887	5.62	8.339	5.89	4.893

Nilai y didapatkan menggunakan persamaan (2.24), sehingga didapatkan untuk nilai x pada setiap kalima pada Tabel 3.14 berikut:

Tabel 3.14 Nilai y Setiap Kalimat

Kalimat	S1	S2	S3	S4	S5	S6	S7	S8
Y	8	8	8	8	8	8	8	8

Seteleh nilai x dan y didapatkan, substitusikan nilai tersebut ke persamaan (2.22). Sebagai contoh perhitungan dilakukan pada kalimat pertama (S1), proses perhitungan sebagai berikut:

$$S1 = \phi\begin{bmatrix}9.968\\8\end{bmatrix} = \begin{bmatrix}11.313 - 9.968 + (9.968 - 8)^2\\11.313 - 8 + (9.968 - 8)^2\end{bmatrix} = \begin{bmatrix}5.218\\7.186\end{bmatrix}$$

Setelah dilakukan perhitungan terhadap seluruh kalimat, maka didapatkan hasilnya pada Tabel 3.15 sebagai berikut:

Tabel 3.15 *Support Vector* Setiap Kalimat

Kalimat	S1	S2	S3	S4	S5	S6	S7	S8
Support Vector	$\begin{bmatrix}5.218\\7.186\end{bmatrix}$	$\begin{bmatrix}4.018\\3.541\end{bmatrix}$	$\begin{bmatrix}12.445\\9.882\end{bmatrix}$	$\begin{bmatrix}5.665\\4.552\end{bmatrix}$	$\begin{bmatrix}11.357\\8.977\end{bmatrix}$	$\begin{bmatrix}3.089\\3.428\end{bmatrix}$	$\begin{bmatrix}9.875\\7.765\end{bmatrix}$	$\begin{bmatrix}16.073\\12.966\end{bmatrix}$

Setelah itu masing-masing *support vector* diberi nilai bias 1. Untuk mendapatkan jarak tegak lurus yang optimal dengan mempertimbangkan vektor positif, serta membantu mendapatkan nilai b atau nilai *hyperplane*[3], hasilnya dapat dilihat pada Tabel 3.16 berikut:

Tabel 3.16 *Support Vector* Bias

Kalimat	S1	S2	S3	S4	S5	S6	S7	S8
Support Vector Bias	$\begin{bmatrix}5.218\\7.186\\1\end{bmatrix}$	$\begin{bmatrix}4.018\\3.541\\1\end{bmatrix}$	$\begin{bmatrix}12.445\\9.882\\1\end{bmatrix}$	$\begin{bmatrix}5.665\\4.552\\1\end{bmatrix}$	$\begin{bmatrix}11.357\\8.977\\1\end{bmatrix}$	$\begin{bmatrix}3.089\\3.428\\1\end{bmatrix}$	$\begin{bmatrix}9.875\\7.765\\1\end{bmatrix}$	$\begin{bmatrix}16.073\\12.966\\1\end{bmatrix}$

Kemudian kalikan setiap kalimat menggunakan persamaan (2.25), sebagai contoh dilakukan perhitungan pada kalimat pertama (S1) berikut:

$$a_1 \begin{bmatrix} 5.218 \\ 7.186 \\ 1 \end{bmatrix}^T * \begin{bmatrix} 5.218 \\ 7.186 \\ 1 \end{bmatrix} = 79.867a_1$$

Setelah dilakukan perhitungan pada seluruh kalimat. Kemudian cari parameter a_i menggunakan persamaan (2.26), dengan cara substitusikan nilai hasil dari perhitungan menggunakan persamaan (2.25), sehingga bentuknya dapat dilihat sebagai berikut:

$79.867a_1+47.406a_2+136.950a_3+63.268a_4+124.775a_5+41.751a_6+108.329a_7+178.049a_8 = 1$
$47.406a_1+29.676a_2+85.985a_3+39.874a_4+78.413a_5+25.546a_6+68.166a_7+111.484a_8 = 1$
$136.590a_1+85.985a_2+253.531a_3+116.478a_4+231.057a_5+73.316a_6+200.630a_7+329.168a_8 = 1$
$63.268a_1+39.874a_2+116.478a_3+53.808a_4+106.200a_5+34.101a_6+92.285a_7+151.073a_8 = 1$
$124.775a_1+78.413a_2+231.057a_3+106.200a_4+210.584a_5+66.856a_6+182.866a_7+299.958a_8 = 1$
$41.751a_1+25.546a_2+73.316a_3+34.101a_4+66.856a_5+22.292a_6+58.122a_7+95.098a_8 = 1$
$108.329a_1+68.166a_2+200.630a_3+92.285a_4+182.866a_5+58.122a_6+158.814a_7+260.413a_8 = 1$
$178.049a_1+111.484a_2+329.168a_3+151.073a_4+299.958a_5+95.098a_6+260.413a_7+427.485a_8 = 1$

Sedemikian sehingga didapatlah nilai untuk a_1 sampai a_8 berikut:

a_1 = 0.005, a_2 = 3.143, a_3 = 4.398, a_4 = -4.550, a_5 = -1.661, a_6 = -0.064, a_7 = 2.958, a_8 = -3.216

Setelah didapatkan nilai a_i masukkan ke persamaan (2.26) sesuai Tabel 3.16 seperti berikut:

$$\widetilde{W} = 0.005 \begin{bmatrix} 5.218 \\ 7.186 \\ 1 \end{bmatrix} + 3.143 \begin{bmatrix} 4.018 \\ 3.541 \\ 1 \end{bmatrix} + 4.398 \begin{bmatrix} 12.445 \\ 9.882 \\ 1 \end{bmatrix} + (-4.550) \begin{bmatrix} 5.665 \\ 4.552 \\ 1 \end{bmatrix} +$$
$$(-1.661) \begin{bmatrix} 11.357 \\ 8.977 \\ 1 \end{bmatrix} + (-0.064) \begin{bmatrix} 3.089 \\ 3.428 \\ 1 \end{bmatrix} + 2.958 \begin{bmatrix} 9.875 \\ 7.765 \\ 1 \end{bmatrix} + (-3.216) \begin{bmatrix} 16.073 \\ 12.966 \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} 0.043 \\ 0.029 \\ 0.005 \end{bmatrix} + \begin{bmatrix} 12.627 \\ 11.128 \\ 3.143 \end{bmatrix} + \begin{bmatrix} 54.733 \\ 43.461 \\ 4.398 \end{bmatrix} + \begin{bmatrix} -25.775 \\ -20.711 \\ -4.550 \end{bmatrix} + \begin{bmatrix} -18.874 \\ -14.882 \\ -1.661 \end{bmatrix} + \begin{bmatrix} -0.198 \\ -0.219 \\ -0.064 \end{bmatrix} + \begin{bmatrix} 29.211 \\ 22.969 \\ 2.958 \end{bmatrix} +$$
$$\begin{bmatrix} -51.592 \\ -41.700 \\ -3.216 \end{bmatrix}$$
$$= \begin{bmatrix} 0.075 \\ 0.075 \\ 1.013 \end{bmatrix}$$

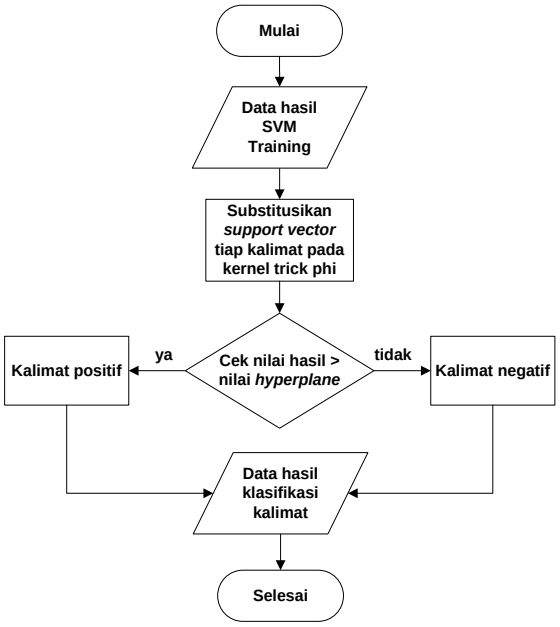
Kemudian hasil yang didapatkan melalui perhitungan dengan menggunakan persamaan (2.27), digunakan persamaan (2.28) dengan hasil yang didapatkan:

$$w = \begin{bmatrix} \frac{0.075}{0.075} \\ \frac{0.075}{0.075} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{dan} \quad b = \frac{1.013}{0.075} = 13.507$$

Sedemikian sehingga didapatkanlah nilai *hyperplane* untuk mengklasifikasikan kedua kelas, yaitu 13.507.

3.7.2 Analisis SVM *Classifier*

Setelah mendapatkan nilai w dan b atau *hyperplane*, selanjutnya dapat menentukan kalimat masuk dalam kelas positif atau negatif dengan nilai w dan *hyperplane* tersebut. Jika nilai hasil uji lebih besar dari nilai *hyperplane* maka kalimat tersebut masuk dalam kalimat kelas positif, jika lebih kecil dari nilai *hyperplane* maka kalimat tersebut masuk dalam kalimat kelas negatif. Adapun *flowchart* klasifikasi kalimat menggunakan metode SVM pada proses *classifier* dapat dilihat pada Gambar 3.14 berikut:



Gambar 3.14 *Flowchart* SVM *Classifier*

Sebagai contoh data uji dilakukan pada kalimat pertama (S1) dengan *support vector* (5.218, 7.186). Masukkan nilai vektor tersebut ke persamaan (2.22) kemudian kalikan dengan nilai dari w *transpose* dan bandingkan dengan nilai *hyperplane* seperti berikut:

$$S_1 = \phi \begin{bmatrix} x \\ y \end{bmatrix} = \phi \begin{bmatrix} 5.218 \\ 7.186 \end{bmatrix} = \begin{bmatrix} 11.313 - 5.218 + (5.218 - 7.186)^2 \\ 11.313 - 7.186 + (5.218 - 7.186)^2 \end{bmatrix} = \begin{bmatrix} 9.968 \\ 8 \end{bmatrix}$$

$$w^T . \phi(S1) = [1 \quad 1] * \begin{bmatrix} 9.968 \\ 8 \end{bmatrix} = 17.968 > 13.507$$

Hasil uji kalimat pertama dengan nilai vektor (5.218, 7.186) tersebut adalah 17.968 dimana 17.968 > 13.507. Berarti kalimat pertama termasuk kalimat kelas positif. Setelah seluruh kalimat diuji, maka didapatkan hasil klasifikasi kelas tiap kalimat dapat dilihat pada Tabel 3.17 berikut:

Tabel 3.17 Hasil Uji Kalimat

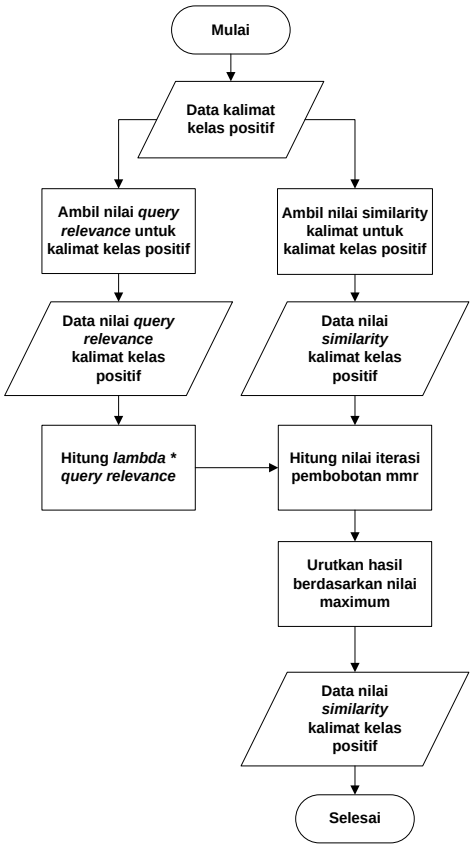
Kalimat	S1	S2	S3	S4	S5	S6	S7	S8
Nilai Uji	17.968	15.523	13.437	14.887	13.620	16.339	13.890	12.893
Kelas	Positif	Positif	Negatif	Positif	Positif	Positif	Positif	Negatif

Setelah hasil uji setiap kalimat didapatkan, serta setiap kalimat sudah masuk ke dalam kelas positif atau negatif. Hanya kalimat positiflah yang menjadi kandidat ringkasan untuk dimasukkan ke dalam sebuah rangkuman. Didalam aplikasi yang dibuat untuk menghitung nilai-nilai di atas dalam penelitian ini menggunakan sebuah *library* bernama LibSVM. Selanjutnya data kalimat-kalimat kelas positif dilakukan pembobotan mmr untuk membentuk dan menghasilkan sebuah ringkasan.

3.8 Analisis Menghasilkan Rangkuman Menggunakan Metode MMR

Dari hasil klasifikasi kalimat menggunakan metode SVM, kalimat-kalimat yang memiliki kelas positif tersebut kembali dibobotkan melalui perhitungan pembobotan MMR, untuk dinilai kembali dengan mengkombinasikan bobot *query relevance* dan bobot matriks *similarity* kalimat.

Kalimat dengan nilai *MMR* tertinggi dari setiap perhitungan iterasi akan diambil, kemudian diurutkan berdasarkan nilai maximum dan dipilih sebagai ringkasan. Nilai bobot MMR memiliki *range* nilai antara 0 sampai 1. Nilai bobot maksimum 1 dan nilai bobot minimum 0. Adapun *flowchart* pembobotan mmr menggunakan metode MMR dapat dilihat pada Gambar 3.15 berikut:



Gambar 3.15 *Flowchart* Pembentukan Ringkasan

Diambil kalimat-kalimat berkelas positif yang sebelumnya dihitung pada perhitungan metode SVM. Kemudian diambil nilai dari bobot *query relevance* dan *similarity* kalimat. Maka didapatkanlah Tabel 3.18 berikut:

Tabel 3.18 Bobot *Query Relevance* dan *Similarity* Kalimat Kelas Positif

Similarity Kalimat							*Query*
Kalimat	S1	S2	S4	S5	S6	S7	*Relevance*
S1	1	0	0	0	0.08	0.29	0.474
S2	0	1	0.3	0.43	0.35	0	0.357
S4	0	0.3	1	0.12	0.19	0	0.400
S5	0	0.43	0.12	1	0.2	0	0.339
S6	0.08	0.35	0.19	0.2	1	0.1	0.400
S7	0.29	0	0	0	0.1	1	0.335

Perhitungan pembobotan MMR diawali dengan perhitungan λ * Bobot *Query Relevance*. Nilai λ yang digunakan untuk peringkasan dokumen yang paling

efektif menggunakan nilai parameter $\lambda = 0.7$, untuk fokus pada kalimat yang paling relevan dan menghasilkan ringkasan yang baik[1][11]. Hasil dari perhitungan dapat dilihat pada Tabel 3.19:

Tabel 3.19 Pembobotan MMR Awal

Kalimat	Pembobotan MMR Awal		
	λ=0.7	*QueryRelevance*	λ*QueryRelevance
S1	0.7	0.474	0.332
S2	0.7	0.357	0.250
S4	0.7	0.400	0.280
S5	0.7	0.339	0.237
S6	0.7	0.400	0.280
S7	0.7	0.335	0.235

Hasil dari perhitungan Pembobotan MMR Awal merupakan iterasi pertama, yang akan disubstitusikan ke persamaan (2.29) pada nilai *similarity* kalimat. Untuk setiap kalimat sampai iterasi antar kalimat terpenuhi, tanpa nilai diagonal 1. Sedemikian sehingga setiap kalimat memiliki 5 iterasi, karena jumlah kalimat yang akan diproses adalah 6. Sebagai contoh dilakukan perhitungan untuk iterasi ke 2 pada kalimat pertama (S1) sebagai berikut:

Iterasi ke 2 untuk S1 terhadap kalimat S2 = 0.332 – ((1-0.7)* 0)

= 0.332

Setelah dilakukan perhitungan iterasi terhadap seluruh kalimat, hasilnya dapat dilihat pada Tabel 3.20 berikut:

Tabel 3.20 Hasil Pembobotam MMR

Iterasi ke-	Kalimat					
	S1	S2	S4	S5	S6	S7
1	0.332	0.250	0.280	0.237	0.280	0.235
2	0.332	0.250	0.280	0.237	0.256	0.148
3	0.332	0.160	0.190	0.108	0.151	0.148
4	0.332	0.031	0.154	0.072	0.094	0.148
5	0.308	0	0.097	0.012	0.034	0.148
6	0.221	0	0.097	0.012	0.004	0.118

Hasil pembobotan mmr menunjukkan bahwa, setelah iterasi pada seluruh kalimat dilakukan kalimat kedua (S2) bernilai 0, sehingga kalimat kedua tidak termasuk dalam hasil ringkasan. Kemudian kalimat kesatu (S1), keempat (S4), kelima (S5), keenam (S6) dan ketujuh (S7) masih memiliki nilai, sehingga termasuk ke dalam ringkasan. Kalimat-kalimat yang termasuk kedalam ringkasan kemudian diurutkan berdasarkan nilai maksimum. Sedemikian sehingga didapatkan Tabel 3.21 sebagai berikut:

Tabel 3.21 Hasil Bobot MMR Maksimum

MMR MAX ke-	Kalimat ke-	Bobot ArgMax MMR
1	S1	0.221
2	S7	0.118
3	S4	0.097
4	S5	0.012
5	S6	0.004

Tahap terakhir dari pembobotan mmr adalah pembentukan ringkasan. Dari hasil pembobotan mmr, dapat diketahui kalimat yang menjadi ringkasan berdasarkan urutan bobot mmr maksimum dapat dilihat pada Tabel 3.22 berikut:

Tabel 3.22 Pembentukan Ringkasan

Kalimat ke-	Kalimat
S1	Pemerintah Malaysia meliburkan hampir semua sekolah selama dua hari untuk melindungi anak-anak dari kabut asap asal Indonesia, Senin (5/10/2015)
S7	Kabut asap juga memaksa beberapa bandara di Malaysia ditutup pada Minggu
S4	Di Shah Alam, Selangor, indeks polusi udara menurun cukup tajam dari 308 menjadi 95
S5	Angka 50 dari indeks ini berarti aman, 51-100 moderat, 101-200 tidak sehat, 201-300 sangat tidak sehat dan di atas 300 berbahaya
S6	Terdapat 11 area di Malaysia, terutama di wilayah utara, yang berada di level "Tidak Sehat," dengan yang terburuk berada di Penang di level 164

Sesuai pada Gambar 2.1, maka hasil ringkasan dari setiap dokumen akan dilakukan pemilihan kalimat kembali. Perhitungan akan dilakukan pada setiap kalimat yang mewakili ringkasan pada setiap dokumen, dengan nilai frekuensi

adalah banyaknya dokumen. Sedemikian sehingga ringkasan akhir yang dihasilkan tidak memiliki kesamaan kalimat antar dokumen. Sedemikian sehingga ringkasan yang dihasilkan berupa rangkuman relevan yang diambil dari kumpulan dokumen dengan satu topik sejenis.

3.9 Spesifikasi Kebutuhan Perangkat Lunak

Spefisikasi kebutuhan perangkat lunak untuk implementasi metode SVM dan MMR untuk menhasilkan rangkuman dari kumpulan berita dengan topik sejenis, terdiri dari Spesifikasi Kebutuhan Perangkat Lunak-Fungsional (SKPL-F) dan Spesifikasi Kebutuhan Perangkat Lunak-Non Fungsional (SKPL-NF). Berikut adalah tabel dari SKPL-F dan SKPL-NF dapat dilihat pada Tabel 3.23 dan Tabel 3.24.

Tabel 3.23 Spesifikasi Kebutuhan Perangkat Lunak Fungsional

Kode	Kebutuhan
SKPL-F-01	Aplikasi dapat mengambil data masukan.
SKPL-F-02	Aplikasi dapat melakukan tahap *preprocessing.*
SKPL-F-03	Aplikasi dapat melakukan pembobotan TF-IDF untuk setiap *term* (kata).
SKPL-F-04	Aplikasi dapat melakukan perhitungan *Cosine Similarity* (*query relevance* dan *similarity* kalimat).
SKPL-F-05	Aplikasi dapat melakukan proses klasifikasi kalimat dengan menggunakan metode *Support Vector Machine.*
SKPL-F-05	Aplikasi dapat menghasilkan rangkuman dengan mennggunakan metode *Maximum Marginal Relevance* serta menampilkannya.
SKPL-F-06	Aplikasi dapat menghitung nilai performansi (*recall*, *precision* dan *f-measure*) dari rangkuman yang dihasilkan serta menampilkannya.

Tabel 3.24 Spesifikasi Kebutuhan Perangkat Lunak Non Fungsional

Kode	Kebutuhan
SKPL-NF-01	Perangkat keras yang digunakan adalah laptop dengan spesifikasi proses processor 3.10 GHz, RAM 4 GB, Harddisk 1TB, resolusi layar 1366 x 768 pixel, IBM keyboard dan mouse.
SKPL-NF-02	Perangkat lunak yang digunakan adalah OS Windows 7 Ultimate 32 Bit, NetBeans IDE, JDK, XAMPP, MySQL dan Google Chrome.

3.10 Analisis Kebutuhan Non Fungsional

Analisis non fungsional yang dilakukan dibagi ke dalam 2 tahap, yaitu analisis kebutuhan perangkat keras dan analisis kebutuhan perangkat lunak.

3.10.1 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat keras pada penelitian ini merupakan kebutuhan perangkat keras yang digunakan dalam pembangunan aplikasi. Adapun perangkat keras yang digunakan sebagai berikut:

a. Laptop dengan *processor* 3.10 GHz
b. RAM 4 GB
c. Harddisk 1TB
d. Resolusi layar 1366 x 768 pixel
e. IBM *keyboard* dan *mouse.*

3.10.2 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak pada penelitian ini merupakan *tools* yang digunakan dalam pembangunan aplikasi. Adapun tools yang digunakan sebagai berikut:

a. OS Windows 7 *Ultimate* 32 Bit
b. NetBeans IDE dan *Java Development Kit*
c. *Web server* XAMPP
d. DBMS MySQL
e. *Web browser* Google Chrome

3.11 Analisis Pengguna

Karakteristik pengguna yang dapat menjalankan sistem yang akan dibangun hanya terdapat satu jenis *user* yaitu seorang penguji. Penguji bekerja untuk menjalankan serta mengetahui hasil dari aplikasi yang dijalankan. Adapun spesifikasi pengguna yang dibutuhkan adalah sebagai berikut :

1. Menguasai penggunaan komputer.
2. Mengerti secara teknis *tools* dan *software* pendukung dalam menjalankan aplikasi.
3. Mengerti tahap-tahapan dalam menjalankan aplikasi.
4. Memahami proses dan kebutuhan dalam menggunakan aplikasi.

3.12 Analisis Kebutuhan Fungsional

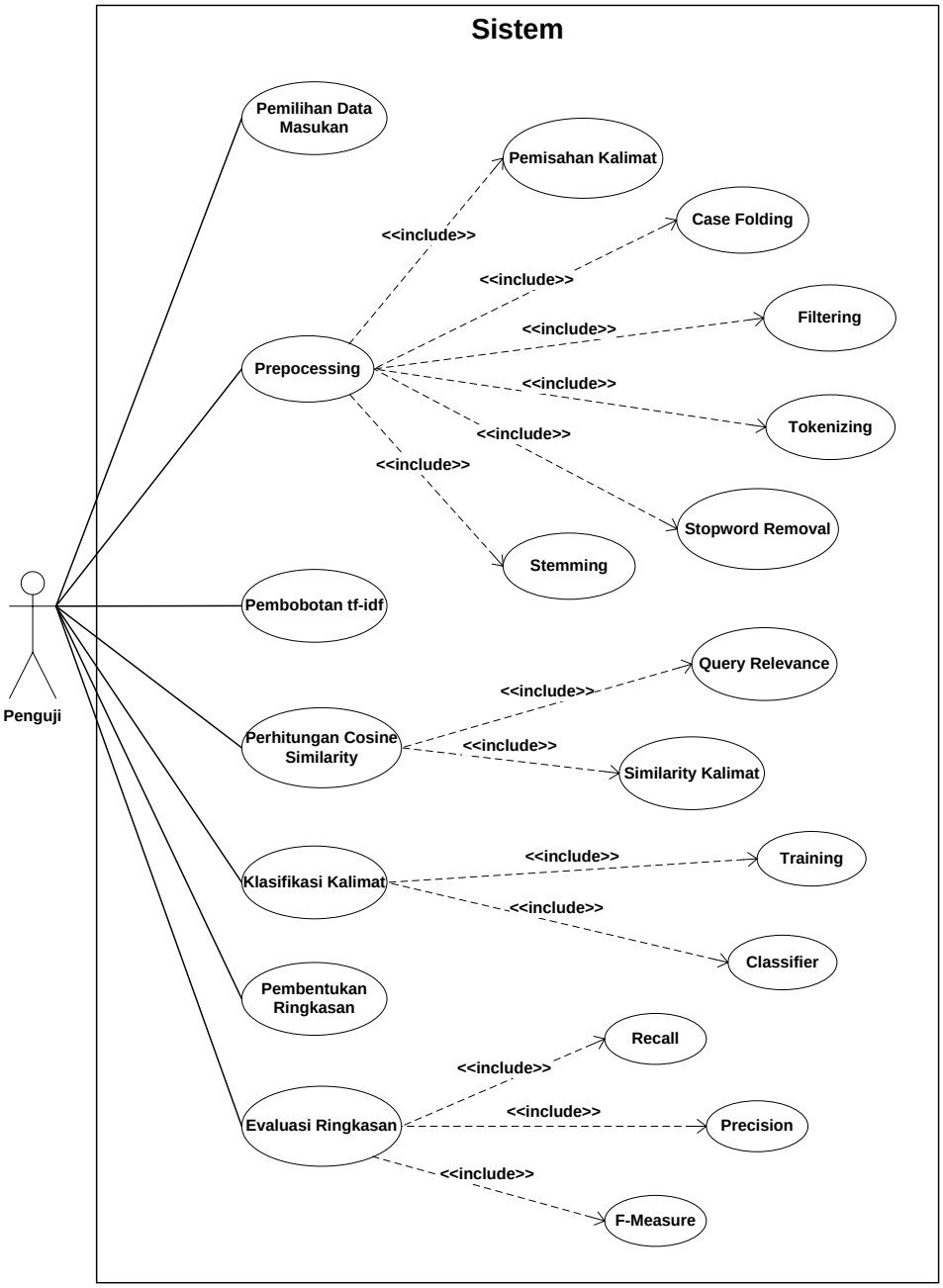
Analisis kebutuhan fungsional bertujuan untuk menganalisis proses yang diterapkan dalam sistem dan menjelaskan kebutuhan yang diperlukan. Menganalisis operasional sistem dengan mendefinisikan skenario penggunaan aplikasi. Analisis yang dilakukan dimodelkan dengan menggunakan UML (*Unified Modeling Language*).

UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Bagian-bagian yang dilakukan dalam analisis tersebut antara lain *use case diagram, use case scenario, activity diagram, class diagram* dan *sequence diagram*.

3.12.1 *Use Case Diagram*

Use case diagram adalah gambaran umum sistem dari sudut pandang pengguna sistem. Tujuan dari *use case* adalah untuk menggambarkan apa yang sistem dapat lakukan. *Use case diagram* dibentuk dari skenario tentang kegunaan sistem yang dinotasikan dengan sebuah *use case.* Setiap skenario menjelaskan suatu alur kegiatan, dapat diinisialisai oleh pengguna sistem yang disebut aktor.

Diagram *use case* dapat memperlihatkan hubungan-hubungan yang terjadi antara aktor-aktor dengan *use case* dalam sistem. Pengguna dapat mengamati diagram *use case* untuk mendapatkan pemahaman yang utuh tentang pembangunan aplikasi mengenai, implementasi metode *support vector machine* dan metode *maximum marginal relevance* untuk menghasilkan rangkuman dari kumpulan dokumen berita dengan topik sejenis. Adapun *use case diagram* dapat dilihat pada Gambar 3.16 berikut:



Gambar 3.16 ***Use Case Diagram***

Berikut ini merupakan deskripsi aktor yang dapat dilihat pada Tabel 3.25 berikut:

Tabel 3.25 Deskripsi Aktor

Aktor	Deskripsi
Penguji	Penguji dapat melakukan pemilihan data masukan, serta melihat data hasil proses *Preprocessing*, pembobotan tf-idf, perhitungan *Cosine Similarity*, Klasifikasi Kalimat, Pembentukan Ringkasan dan Evaluasi Ringkasan.

Berikut merupakan deskripsi *use case* yang dapat dilihat pada Tabel 3.26 berikut:

Tabel 3.26 Deskripsi *Use Case*

No	*Use Case*	Deskripsi
1	Pemilihan Data Masukan	Fungsionalitas ini digunakan penguji untuk memilih data masukan berupa dokumen dengan format *file* .doc.
2	*Preprocessing*	Fungsionalitas ini digunakan penguji untuk memecahkan data dokumen, sehingga layak untuk diproses selanjutnya.
3	Pemisahan kalimat	Fungsionalitas ini digunakan penguji untuk memisahkan data dokumen menjadi data perkalimat.
4	*Case folding*	Fungsionalitas ini digunakan penguji untuk mengubah data kalimat yang memiliki huruf kapital menjadi huruf kecil
5	*Filtering*	Fungsionalitas ini digunakan untuk menghilangkan karakter selain "a" sampai "z" dan hanya menerima spasi.
6	*Tokenizing*	Fungsionalitas ini digunakan penguji untuk memecahkan data kalimat dan memisahkannya menjadi data perkata.
7	*Stopword Removal*	Fungsionalitas ini digunakan penguji untuk menghilangkan kata-kata yang masuk ke dalam daftar *stopword*.
8	*Stemming*	Fungsionalitas ini digunakan penguji untuk mengubah kata yang memiliki imbuhan kedalam kata dasarnya.
9	Pembobotan tf-idf	Fungsionalitas ini digunakan penguji untuk memberi nilai bobot pada tiap *term* (kata).
10	Perhitungan Cosine Similarity	Fungsionalitas ini digunakan penguji untuk menghitung nilai bobot *query relevance* dan *similarity* kalimat.
11	*Query relevance*	Fungsionalitas ini digunakan penguji untuk mendapatkan nilai kesamaan setiap kalimat dengan vektor *query*.
12	*Similarity* kalimat	Fungsionalitas ini digunakan penguji untuk mendapatkan nilai kesamaan antar kalimat.
13	Klasifikasi Kalimat	Fungsionalitas ini digunakan penguji untuk mengklasifikasi data kalimat.
14	Training	Fungsionalitas ini digunakan untuk membentuk model fitur.
15	Classifier	Fungsionalitas ini digunakan untuk mengklasifikasikan kalimat ke dalam kelas positif dan negatif.
16	Pembentukan Ringkasan	Fungsionalitas ini digunakan penguji untuk menghasilkan sebuah ringkasan berupa rangkuman singkat.

No	*Use Case*	Deskripsi
17	Evaluasi Ringkasan	Fungsionalitas ini digunakan penguji untuk mengevaluasi hasil ringkasan yang dihasilkan untuk menilai kerelevananya.
18	*Recall*	Fungsionalitas ini digunakan penguji untuk mengetahui nilai performansi *recall* dari ringkasan yang dihasilkan.
19	*Precision*	Fungsionalitas ini digunakan penguji untuk mengetahui nilai performansi *precision* dari ringkasan yang dihasilkan.
20	*F-Measure*	Fungsionalitas ini digunakan penguji untuk mengetahui nilai performansi *f-measure* dari ringkasan yang dihasilkan.

3.12.2 *Use Case Scenario*

Bagian ini menjelaskan skenario untuk tiap *use case* yang menggambarkan urutan interaksi aktor dengan *use case* tersebut dari awal sampai akhir.

Tabel 3.27 *Use Case Scenario* Pemilihan Data Masukan

Nama *Use Case*	Pemilihan Data Masukan
Tujuan	*Input* data.
Deskripsi	Fungsionalitas ini digunakan penguji untuk memilih data masukan berupa dokumen dengan format *file* .doc.
Aktor	Penguji
Related Case	-
Kondisi Sukses	Data masukan masuk ke proses selanjutnya.
Kondisi Gagal	Data masukan gagal masuk ke proses selanjutnya.
Include Case	-
Trigger	Menekan tombol *Choose File*.
Skenario Utama	
Aksi aktor	Reaksi Sistem
1. Memilih data masukan sesuai dengan format *file* yang dipilih (*Choose File*).	
2. Aktor menekan tombol Proses.	
	3. Data masukan masuk ke proses selanjutnya.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
1. Aktor menekan tombol Proses.	
	2. *File* data masukan belum di pilih.

Tabel 3.28 *Use Case Scenario Preprocessing*

Nama *Use Case*	*Preprocessing*
Tujuan	Preproses Data.
Deskripsi	Fungsionalitas ini digunakan penguji untuk memecahkan data dokumen, sehingga layak untuk diproses selanjutnya.

Aktor	Penguji
Related Case	-
Kondisi Sukses	Data masukan berhasil di preproses.
Kondisi Gagal	Data masukan gagal di preproses.
Include Case	Pemisahan Kalimat, *Case Folding, Filtering, Tokenizing, Stopword Removal, Stemming.*
Trigger	Terdapat data masukan dan menekan tombol Lanjut kemudian pilih menu *preprocessing.*
Skenario Utama	
Aksi aktor	Reaksi Sistem
1. Aktor menekan tombol Lanjut.	
	2. Sistem melakukan proses *preprocessing*.
	3. Memisahkan data dokumen menjadi data perkalimat.
	4. Mengubah huruf kapital menjadi huruf kecil.
	5. Menghilangkan karakter selain "a" sampai "z" dan hanya menerima spasi.
	6. Memisahkan data kalimat menjadi data perkata.
	7. Menghilangkan kata yang terdapat dalam kamus stopword.
	8. Mengubah kata berimbuhan menjadi kata dasar.
	9. Data masukan berhasil di preproses.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
1. Aktor menekan tombol Lanjut.	
	2. Data masukan gagal dipreproses.

Tabel 3.29 *Use Case Scenario* Pemisahan Kalimat

Nama *Use Case*	Pemisahan Kalimat
Tujuan	Menjadikan data perkalimat.
Deskripsi	Fungsionalitas ini digunakan penguji untuk memisahkan data dokumen menjadi data perkalimat.
Aktor	Penguji
Related Case	*Preprocessing*
Kondisi Sukses	Data masukan berhasil melalui proses *preprocessing* pemisahan kalimat.
Kondisi Gagal	Data masukan gagal melalui proses *preprocessing* pemisahan kalimat.
Include Case	-
	Terdapat data masukan yang akan dipreproses pemisahan kalimat.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan *preprocessing* pemisahan kalimat.
	2. Data masukan berhasil melalui proses *preprocessing* pemisahan kalimat.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data masukan gagal melalui proses *preprocessing* pemisahan kalimat.

Tabel 3.30 *Use Case Scenario Case Folding*

Nama *Use Case*	*Case Folding*
Tujuan	Mengubah huruf kapital menjadi huruf kecil.
Deskripsi	Fungsionalitas ini digunakan penguji untuk mengubah data kalimat yang memiliki huruf kapital menjadi huruf kecil.
Aktor	Penguji
Related Case	*Preprocessing*
Kondisi Sukses	Data masukan berhasil melalui proses *preprocessing case folding.*
Kondisi Gagal	Data masukan gagal melalui proses *preprocessing case folding.*
Include Case	-
Trigger	Terdapat data hasil proses pemisahan kalimat.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan proses *preprocessing case folding.*
	2. Data masukan berhasil melalui proses *preprocessing case folding.*
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data masukan gagal melalui proses *preprocessing case folding.*

Tabel 3.31 *Use Case Scenario Filtering*

Nama *Use Case*	*Filtering*
Tujuan	Menghilangkan kata selain “a” sampai “z”. dan hanya menerima spasi
Deskripsi	Fungsionalitas ini digunakan penguji untuk menghilangkan karakter selain “a” sampai “z” dan hanya menerima spasi.
Aktor	Penguji
Related Case	*Preprocessing*
Kondisi Sukses	Data masukan berhasil melalui proses *preprocessing filtering.*
Kondisi Gagal	Data masukan gagal melalui proses *preprocessing filtering.*
Include Case	-
Trigger	Terdapat data hasil proses *case folding.*
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan proses *preprocessing filtering.*
	2. Data masukan berhasil melalui proses *preprocessing filtering.*
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data masukan gagal melalui proses *preprocessing filtering.*

Tabel 3.32 *Use Case Scenario Tokenizing*

Nama *Use Case*	*Tokenizing*
Tujuan	Memisahkan data kalimat menjadi data perkata
Deskripsi	Fungsionalitas ini digunakan penguji untuk memecahkan data kalimat dan memisahkannya menjadi data perkata.
Aktor	Penguji
Related Case	*Preprocessing*
Kondisi Sukses	Data masukan berhasil melalui proses *preprocessing tokenizing.*
Kondisi Gagal	Data masukan gagal melalui proses *preprocessing tokenizing.*
Include Case	-
Trigger	Terdapat data hasil proses *case folding*.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan proses *preprocessing tokenizing.*
	2. Data masukan berhasil melalui proses *preprocessing tokenizing.*
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data masukan gagal melalui proses *preprocessing tokenizing.*

Tabel 3.33 *Use Case Scenario Stopword Removal*

Nama *Use Case*	*Stopword Removal*
Tujuan	Menghilangkan kata yang terdapat dalam kamus stopword.
Deskripsi	Fungsionalitas ini digunakan penguji untuk menghilangkan kata-kata yang masuk ke dalam daftar *stoplist.*
Aktor	Penguji
Related Case	*Preprocessing*
Kondisi Sukses	Data masukan berhasil melalui proses *preprocessing stopword removal.*
Kondisi Gagal	Data masukan gagal melalui proses *preprocessing stopword removal.*
Include Case	-
Trigger	Terdapat data hasil proses *tokenizing.*
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan proses *preprocessing stopword removal.*
	2. Data masukan berhasil melalui proses *preprocessing stopword removal.*
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data masukan gagal melalui proses *preprocessing stopword removal.*

Tabel 3.34 *Use Case Scenario Stemming*

Nama *Use Case*	*Stemming*
Tujuan	Mengubah kata berimbuhan menjadi kata dasar.
Deskripsi	Fungsionalitas ini digunakan penguji untuk mengubah kata yang memiliki imbuhan kedalam kata dasarnya.
Aktor	Penguji
Related Case	*Preprocessing*
Kondisi Sukses	Data masukan berhasil melalui proses *preprocessing stemming.*
Kondisi Gagal	Data masukan gagal melalui proses *preprocessing stemming..*
Include Case	-
Trigger	Terdapat data hasil proses *stopword removal*
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan proses *preprocessing stemming.*
	2. Data masukan berhasil melalui proses *preprocessing stemming.*
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data masukan gagal melalui proses *preprocessing stemming.*

Tabel 3.35 *Use Case Scenario* Pembobotan TF-IDF

Nama *Use Case*	Pembobotan TF-IDF
Tujuan	Memberi nilai bobot pada tiap kata.
Deskripsi	Fungsionalitas ini digunakan penguji untuk memberi nilai bobot pada tiap *term* (kata).
Aktor	Penguji
Related Case	-
Kondisi Sukses	Data hasil *preprocessing* berhasil diberi nilai bobot tf-idf.
Kondisi Gagal	Data hasil *preprocessing* gagal diberi nilai bobot tf-idf.
Include Case	-
Trigger	Terdapat data hasil proses *preprocessing* dan menekan tombol menu pembobotan tf-idf.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan pembobotan tf-idf.
	2. Data hasil *preprocessing* berhasil diberi nilai bobot tf-idf.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data hasil *preprocessing* gagal diberi nilai bobot tf-idf.

Tabel 3.36 *Use Case Scenario* Perhitungan *Cosine Similarity*

Nama *Use Case*	Perhitungan *Cosine Similarity*
Tujuan	Mendapatkan nilai bobot *query relevance* dan *similarity* kalimat.
Deskripsi	Fungsionalitas ini digunakan untuk menghitung nilai bobot *query relevance* dan *similarity* kalimat.
Aktor	Penguji
Related Case	-
Kondisi Sukses	Data hasil pembobotan tf-idf berhasil didapatkan nilai bobot *query relevance* dan *similarity* kalimat.
Kondisi Gagal	Data hasil pembobotan tf-idf gagal didapatkan nilai bobot *query relevance* dan *similarity* kalimat.
Include Case	*query relevance*, *similarity* kalimat.
Trigger	Terdapat data hasil pembobotan tf-idf dan menekan tombol menu perhitungan *cosine similarity*.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan perhitungan *cosine similarity.*
	2. Data hasil pembobotan tf-idf berhasil dihitung dan didapatkan nilai bobot *query relevance*.
	3. Data hasil pembobotan tf-idf berhasil dihitung dan didapatkan nilai bobot *similarity* kalimat.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data hasil pembobotan tf-idf gagal dihitung untuk mendapatkan nilai bobot *query relevance*.
	2. Data hasil pembobotan tf-idf gagal dihitung untuk mendapatkan nilai bobot *similarity* kalimat.

Tabel 3.37 *Use Case Scenario Query Relevance*

Nama *Use Case*	*Query Relevance*
Tujuan	Mendapatkan nilai bobot *query relevance*.
Deskripsi	Fungsionalitas ini digunakan penguji untuk mendapatkan nilai kesamaan setiap kalimat dengan vektor *query*.
Aktor	Penguji
Related Case	Perhitungan *Cosine Similarity*
Kondisi Sukses	Data hasil pembobotan tf-idf berhasil dihitung dan didapatkan nilai bobot *query relevance.*
Kondisi Gagal	Data hasil pembobotan tf-idf gagal dihitung untuk mendapatkan nilai bobot *query relevance.*
Include Case	-
Trigger	Terdapat data hasil pembobotan tf-idf.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan perhitungan *query relevance.*
	2. Data hasil pembobotan tf-idf berhasil dihitung dan didapatkan nilai bobot *query relevance.*
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data hasil pembobotan tf-idf gagal dihitung untuk mendapatkan nilai bobot *query relevance.*

Tabel 3.38 *Use Case Scenario Similarity* Kalimat

Nama *Use Case*	*Similarity* Kalimat
Tujuan	Mendapatkan nilai bobot *similarity* kalimat.
Deskripsi	Fungsionalitas ini digunakan penguji untuk mendapatkan nilai kesamaan antar kalimat.
Aktor	Penguji
Related Case	Perhitungan *Cosine Similarity*
Kondisi Sukses	Data hasil pembobotan tf-idf berhasil dihitung dan didapatkan nilai bobot *similarity* kalimat.
Kondisi Gagal	Data hasil pembobotan tf-idf gagal dihitung untuk mendapatkan nilai bobot *similarity* kalimat.
Include Case	-
Trigger	Terdapat data hasil pembobotan tf-idf.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan perhitungan *similarity* kalimat.
	2. Data hasil pembobotan tf-idf berhasil dihitung dan didapatkan nilai bobot *similarity* kalimat.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data hasil pembobotan tf-idf gagal dihitung untuk mendapatkan nilai bobot *similarity* kalimat.

Tabel 3.39 *Use Case Scenario* Klasifikasi Kalimat

Nama *Use Case*	Klasifikasi Kalimat
Tujuan	Mendapatkan hasil klasifikasi kalimat.
Deskripsi	Fungsionalitas ini digunakan penguji untuk mengklasifikasi data kalimat menggunakan metode s*upport vector machine*.
Aktor	Penguji
Related Case	-
Kondisi Sukses	Berhasil mendapatkan hasil klasifikasi kalimat.
Kondisi Gagal	Gagal mendapatkan hasil klasifikasi kalimat.
Include Case	*Training, Classifier*
Trigger	Terdapat data hasil pembobotan tf-idf dan perhitungan *cosine similarity* dan menekan tombol menu klasifikasi kalimat.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan proses klasifikasi kalimat.
	2. Data kalimat diklasifikasi kedalam kelas positif dan negatif.
	3. Berhasil mendapatkan hasil klasifikasi kalimat.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Gagal mendapatkan hasil klasifikasi kalimat.

Tabel 3.40 *Use Case Scenario Training*

Nama *Use Case*	*Training*
Tujuan	Membentuk model fitur.
Deskripsi	Fungsionalitas ini digunakan untuk membentuk model fitur.
Aktor	Penguji
Related Case	Klasifikasi Kalimat
Kondisi Sukses	Proses SVM *Training* berhasil dan membentuk model fitur.
Kondisi Gagal	Proses SVM *Training* gagal dalam membentuk model fitur.
Include Case	-
Trigger	Terdapat data hasil pembobotan tf-idf dan perhitungan *cosine similarity*.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan pembentukan model fitur.
	2. Proses SVM *Training* berhasil dan membentuk model fitur.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Proses SVM *Training* gagal dalam membentuk model fitur.

Tabel 3.41 *Use Case Scenario Classifier*

Nama *Use Case*	*Classifier*
Tujuan	Mengklasifikasikan kalimat ke dalam kelas positif dan negatif.
Deskripsi	Fungsionalitas ini digunakan untuk mengklasifikasikan kalimat ke dalam kelas positif dan negatif.
Aktor	Penguji
Related Case	Klasifikasi Kalimat
Kondisi Sukses	Data kalimat berhasil diklasifikasi ke dalam kelas positif dan negatif.
Kondisi Gagal	Data kalimat gagal diklasifikasi ke dalam kelas positif dan negatif.
Include Case	-
Trigger	Terdapat data hasil pembobotan tf-idf dan perhitungan *cosine similarity*.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan klasifikasi kalimat ke dalam kelas positif dan negatif.
	2. Data kalimat berhasil diklasifikasi ke dalam kelas positif dan negatif.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data kalimat gagal diklasifikasi ke dalam kelas positif dan negatif.

Tabel 3.42 *Use Case Scenario* Pembentukan Ringkasan

Nama *Use Case*	Pembentukan Ringkasan
Tujuan	Mendapatkan sebuah ringkasan.
Deskripsi	Fungsionalitas ini digunakan penguji untuk menghasilkan sebuah ringkasan .
Aktor	Penguji
Related Case	-
Kondisi Sukses	Data kalimat positif dengan nilai bobot *query relevance* dan *similarity* kalimat, berhasil diberi nilai bobot mmr dan menghasilkan sebuah ringkasan.
Kondisi Gagal	Data kalimat positif dengan nilai bobot *query relevance* dan *similarity* kalimat, gagal diberi nilai bobot mmr sehingga tidak menghasilkan sebuah ringkasan.
Include Case	-
Trigger	Terdapat data hasil perhitungan *cosine similarity* dan klasifikasi kalimat serta menekan tombol menu pembentukan ringkasan.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan pembobotan mmr.
	2. Data kalimat berhasil diberi nilai bobot mmr dan menghasilkan sebuah ringkasan.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Data kalimat gagal diberi nilai bobot mmr sehingga tidak menghasilkan sebuah ringkasan.

Tabel 3.43 *Use Case Scenario* Evaluasi Ringkasan

Nama *Use Case*	Evaluasi Ringkasan
Tujuan	Mendapatkan hasil evaluasi ringkasan.
Deskripsi	Fungsionalitas ini digunakan penguji untuk mengevaluasi hasil ringkasan yang dihasilkan untuk menilai kerelevananya.
Aktor	Penguji
Related Case	-
Kondisi Sukses	Berhasil mendapatkan hasil evaluasi ringkasan.
Kondisi Gagal	Gagal mendapatkan hasil evaluasi ringkasan.
Include Case	*recall, precision, f-measure*
Trigger	Terdapat data hasil pembentukan ringkasan, menekan tombol menu evaluasi ringkasan, kemudian memilih file ringkasan manual dan klik Proses.
Skenario Utama	
Aksi aktor	Reaksi Sistem
1. Memilih file ringkasan manual (*Choose File*)	
2. Aktor menekan tombol Proses	
	3. Melakukan perhitungan nilai performansi *recall.*
	4. Melakukan perhitungan nilai performansi *precision.*
	5. Melakukan perhitungan nilai performansi *f-measure.*
	6. Berhasil mendapatkan hasil evaluasi ringkasan.

Skenario Alternatif	
Aksi aktor	Reaksi Sistem
1. Memilih file ringkasan manual (*Choose File*)	
2. Aktor menekan tombol Lanjut	
	1. Gagal mendapatkan hasil evaluasi ringkasan.

Tabel 3.44 *Use Case Scenario Recall*

Nama *Use Case*	*Recall*
Tujuan	Mendapatkan nilai performansi *recall.*
Deskripsi	Fungsionalitas ini digunakan penguji untuk mengetahui nilai performansi *recall* dari ringkasan yang dihasilkan.
Aktor	Penguji
Related Case	Evaluasi Ringkasan
Kondisi Sukses	Berhasil mendapatkan nilai performansi *recall* dari ringkasan yang dihasilkan.
Kondisi Gagal	Gagal mendapatkan nilai performansi *recall* dari ringkasan yang dihasilkan.
Include Case	-
Trigger	Terdapat data hasil pembentukan ringkasan.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan perhitungan nilai performansi *recall.*
	2. Berhasil mendapatkan nilai performansi *recall* dari ringkasan yang dihasilkan.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Gagal mendapatkan nilai performansi *recall* dari ringkasan yang dihasilkan.

Tabel 3.45 *Use Case Scenario Precision*

Nama *Use Case*	*Precision*
Tujuan	Mendapatkan nilai performansi *precision.*
Deskripsi	Fungsionalitas ini digunakan penguji untuk mengetahui nilai performansi *precision* dari ringkasan yang dihasilkan.
Aktor	Penguji
Related Case	Evaluasi Ringkasan
Kondisi Sukses	Berhasil mendapatkan nilai performansi *precision* dari ringkasan yang dihasilkan.
Kondisi Gagal	Gagal mendapatkan nilai performansi *precision* dari ringkasan yang dihasilkan.
Include Case	-
Trigger	Terdapat data hasil pembentukan ringkasan
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan perhitungan nilai performansi *precision.*
	2. Berhasil mendapatkan nilai performansi *precision* dari ringkasan yang dihasilkan.

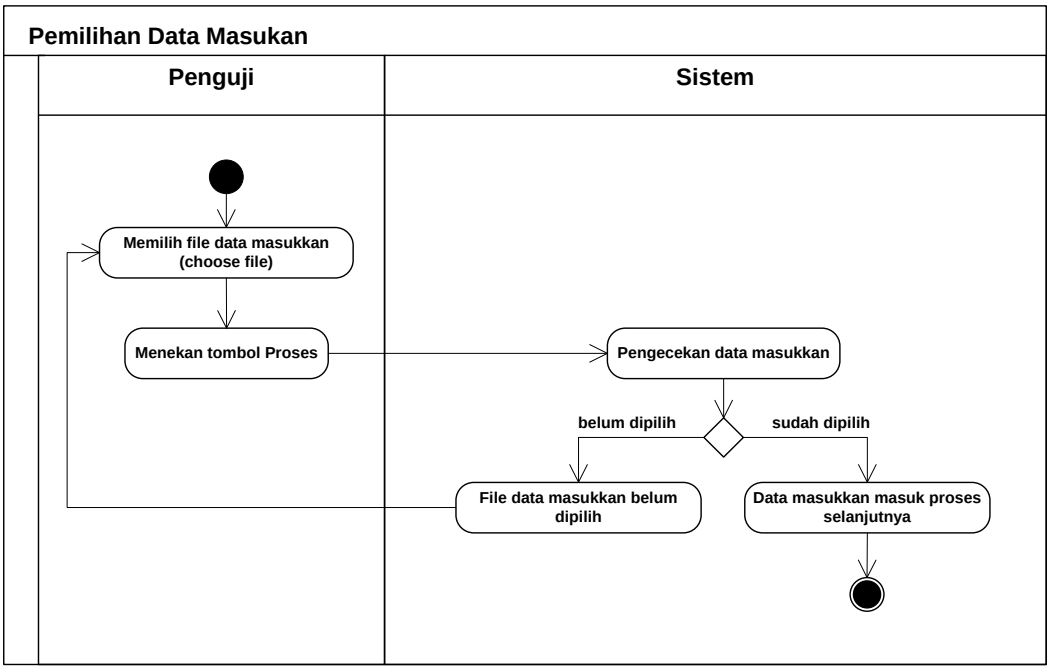
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Gagal mendapatkan nilai performansi *precision* dari ringkasan yang dihasilkan.

Tabel 3.46 *Use Case Scenario F-Measure*

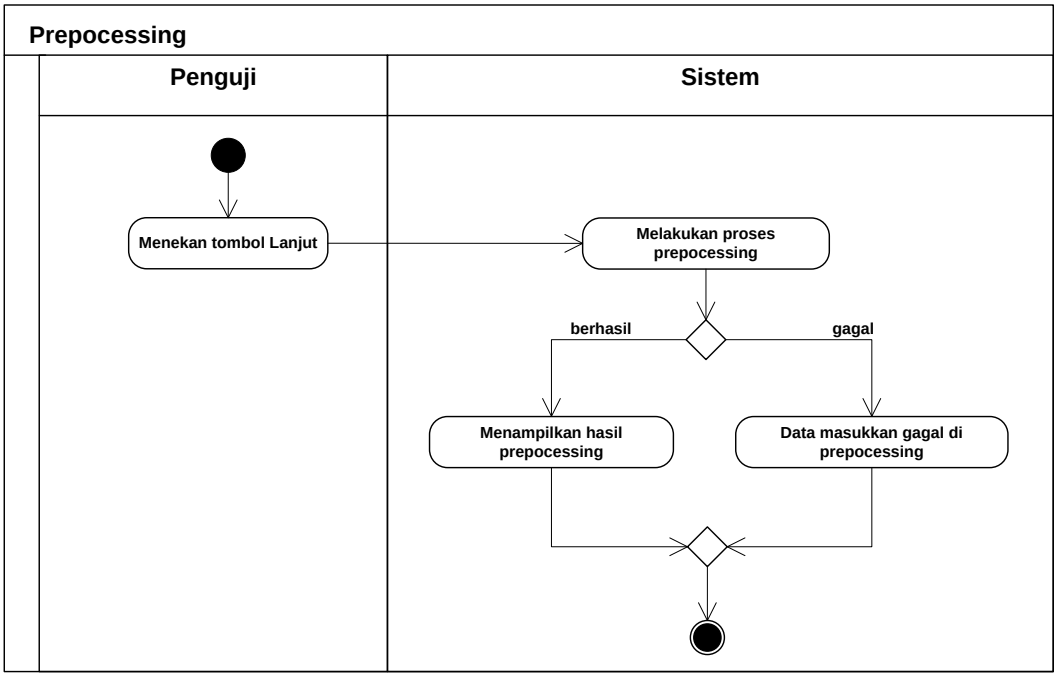
Nama *Use Case*	*F-Measure*
Tujuan	Mendapatkan nilai performansi *f-measure.*
Deskripsi	Fungsionalitas ini digunakan penguji untuk mengetahui nilai performansi *f-measure* dari ringkasan yang dihasilkan.
Aktor	Penguji
Related Case	Evaluasi Ringkasan
Kondisi Sukses	Berhasil mendapatkan nilai performansi *f-measure* dari ringkasan yang dihasilkan.
Kondisi Gagal	Gagal mendapatkan nilai performansi *precision* dari ringkasan yang dihasilkan.
Include Case	-
Trigger	Terdapat data hasil pembentukan ringkasan.
Skenario Utama	
Aksi aktor	Reaksi Sistem
	1. Sistem melakukan perhitungan nilai performansi *f-measure.*
	2. Berhasil mendapatkan nilai performansi *f-measure* dari ringkasan yang dihasilkan.
Skenario Alternatif	
Aksi aktor	Reaksi Sistem
	1. Gagal mendapatkan nilai performansi *f-measure* dari ringkasan yang dihasilkan.

3.12.3 *Activity Diagram*

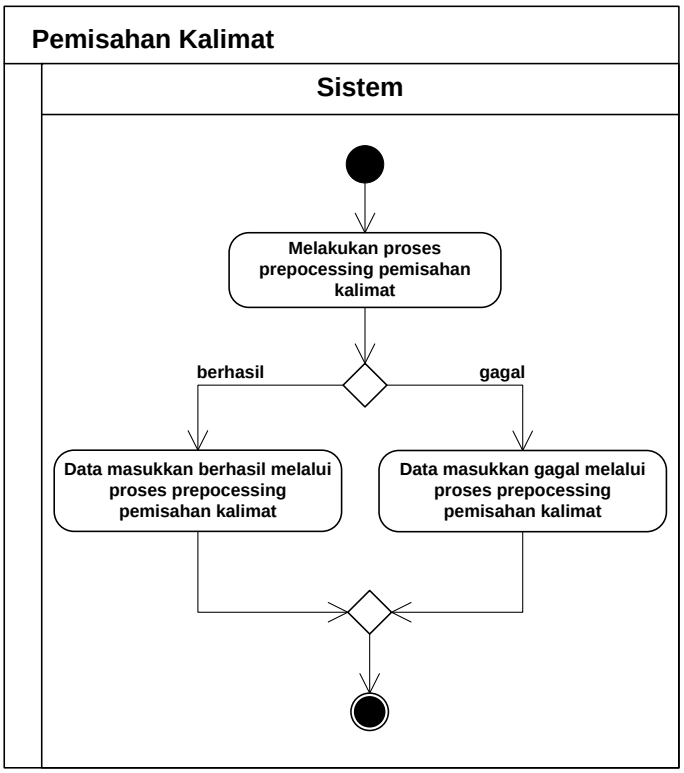
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang dirancang, menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum, menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana alir berakhir. Adapun *activity diagram* dari masing-masing *use case scenario*.



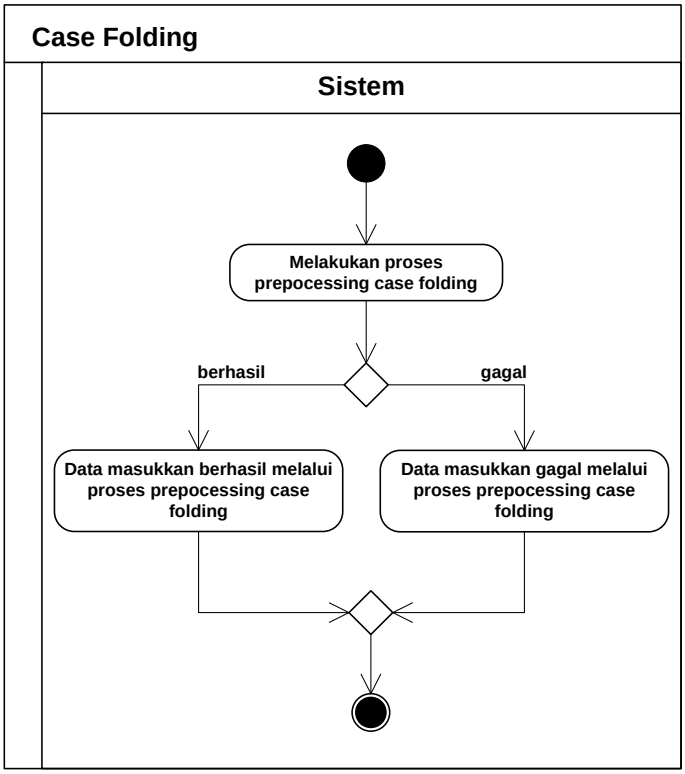
Gambar 3.17 *Activity Diagram* Pemilihan Data Masukan



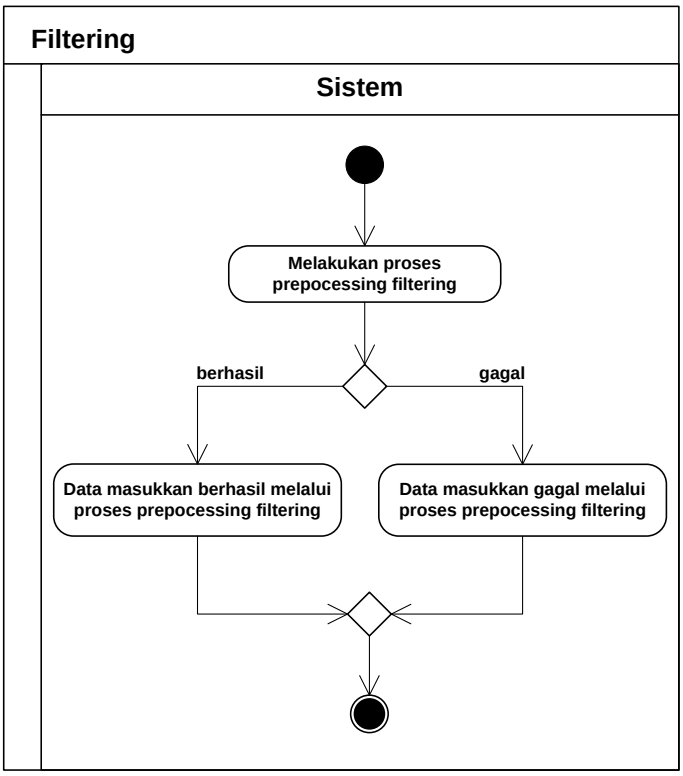
Gambar 3.18 *Activity Diagram Preprocessing*



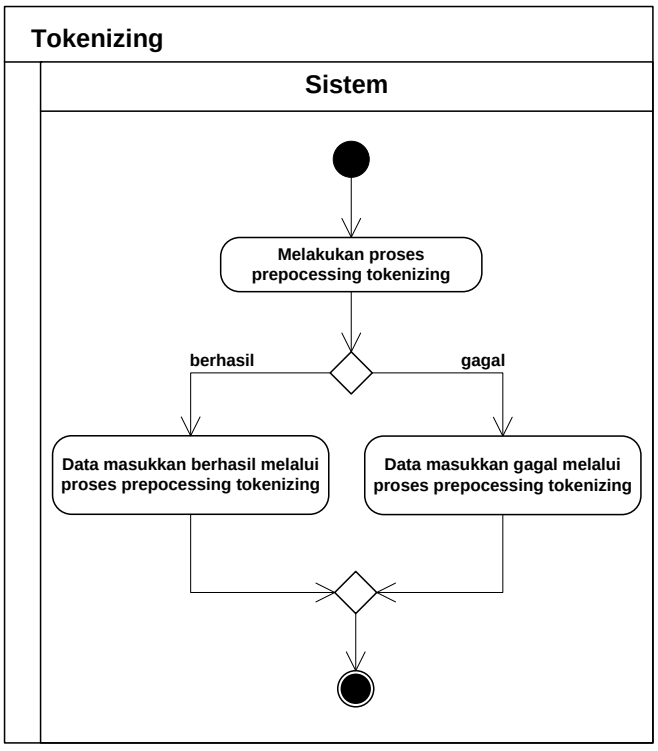
Gambar 3.19 *Activity Diagram* Pemisahan Kalimat



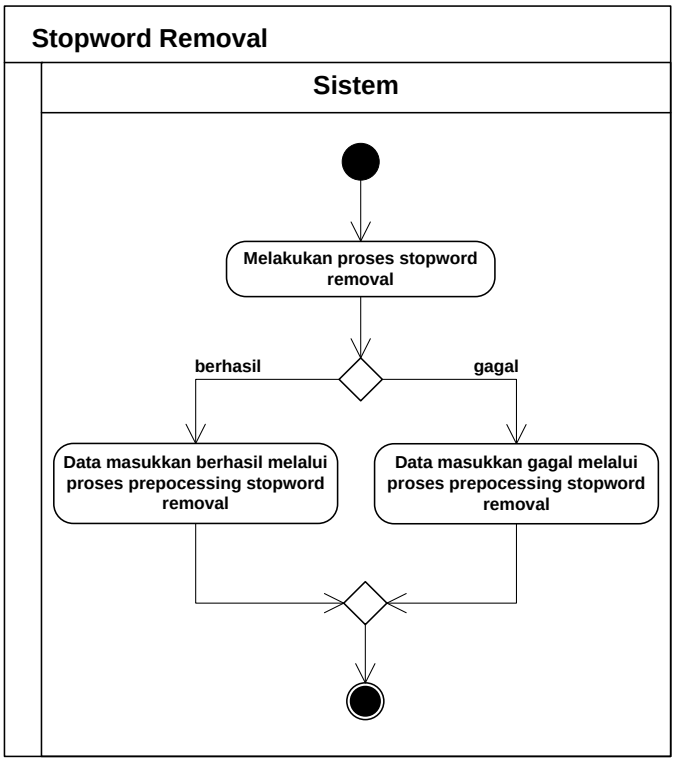
Gambar 3.20 *Activity Diagram Case Folding*



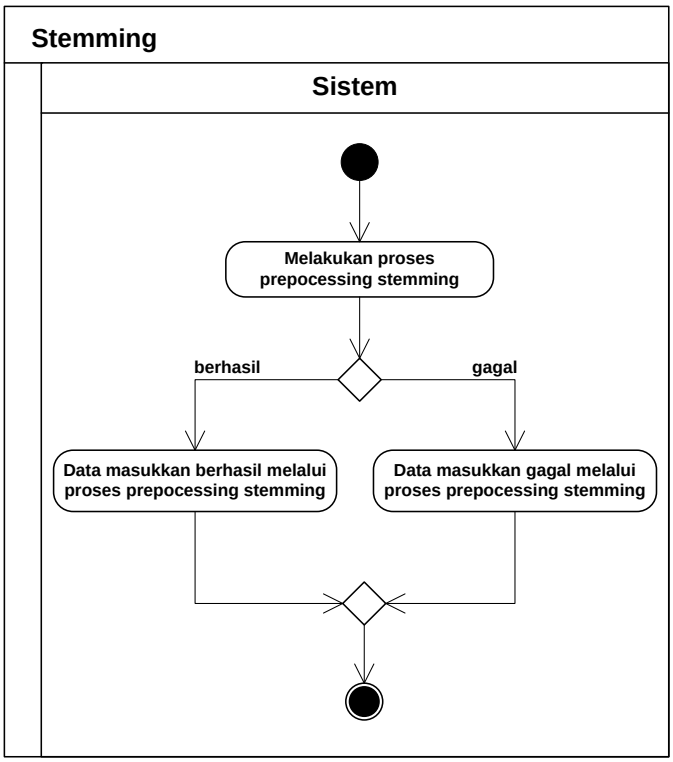
Gambar 3.21 ***Activity Diagram Filtering***



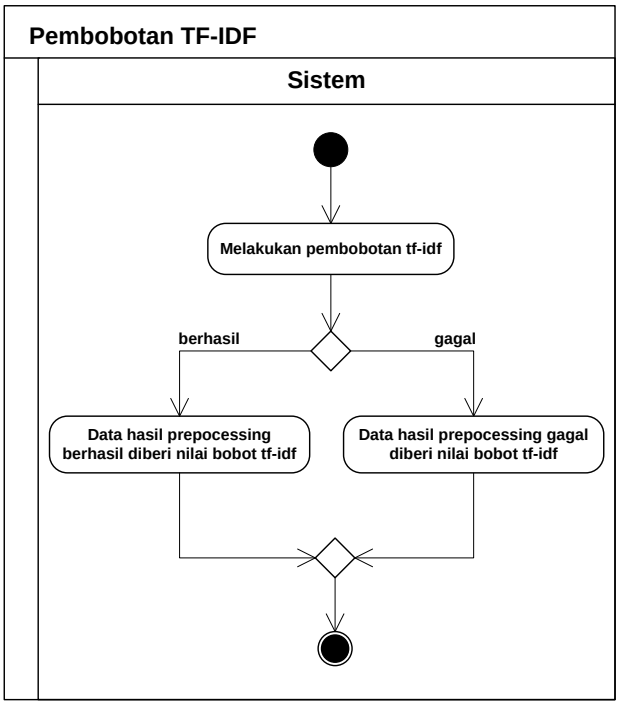
Gambar 3.22 ***Activity Diagram Tokenizing***



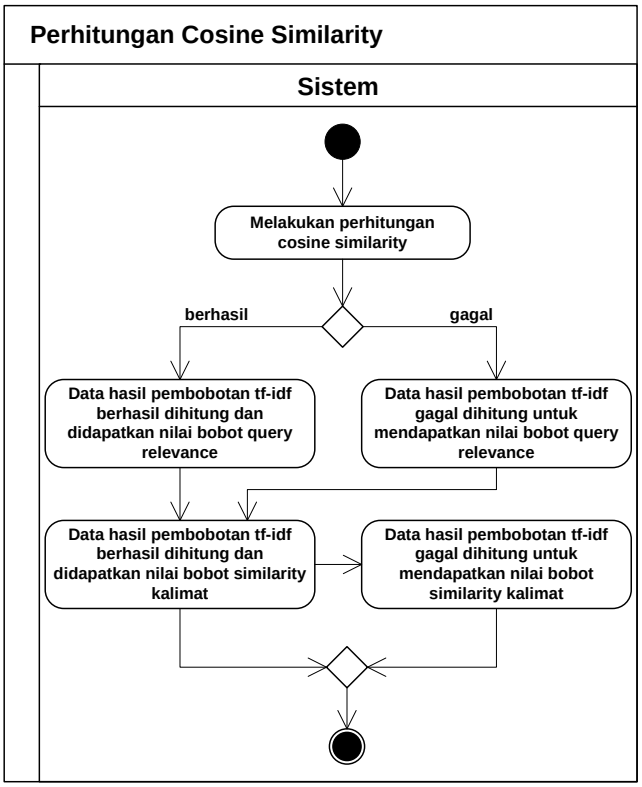
Gambar 3.23 ***Activity Diagram Stopword Removal***



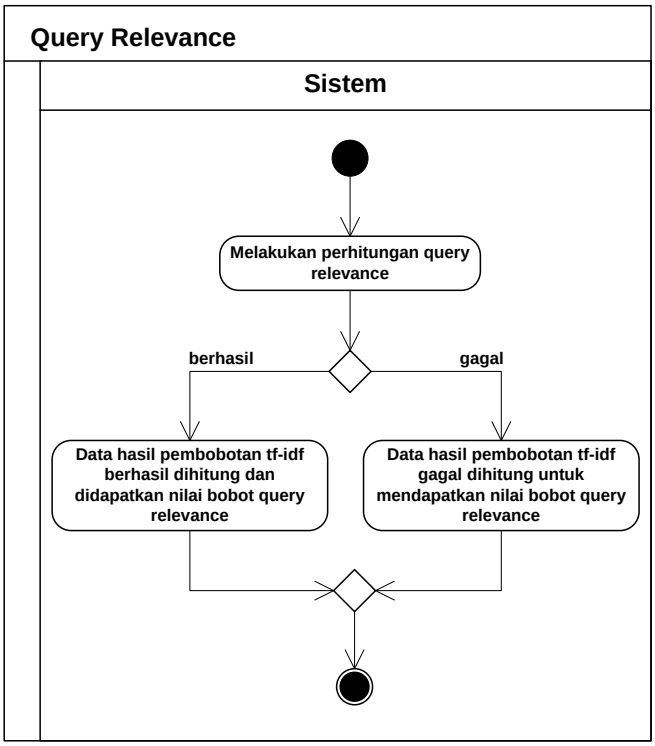
Gambar 3.24 ***Activity Diagram Stemming***



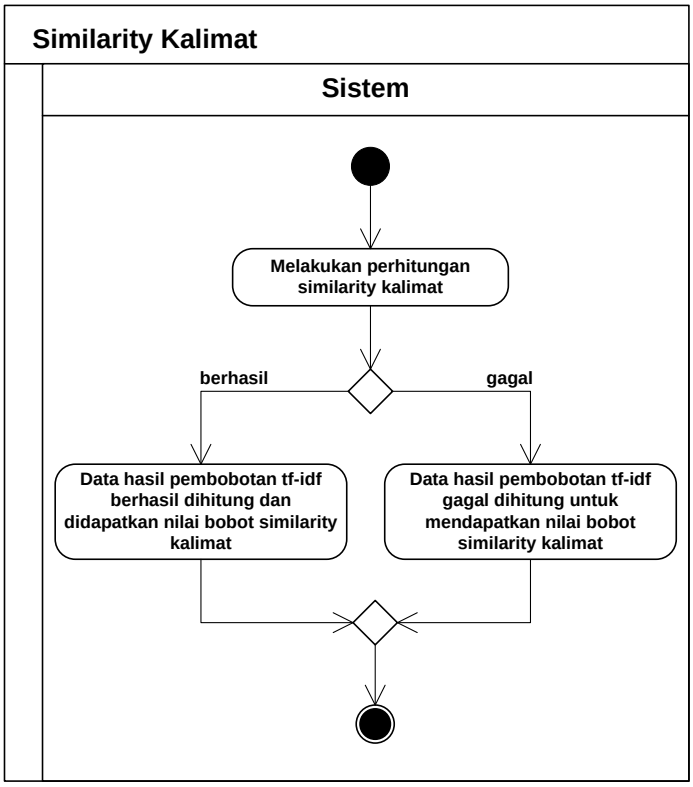
Gambar 3.25 *Activity Diagram* Pembobotan TF-IDF



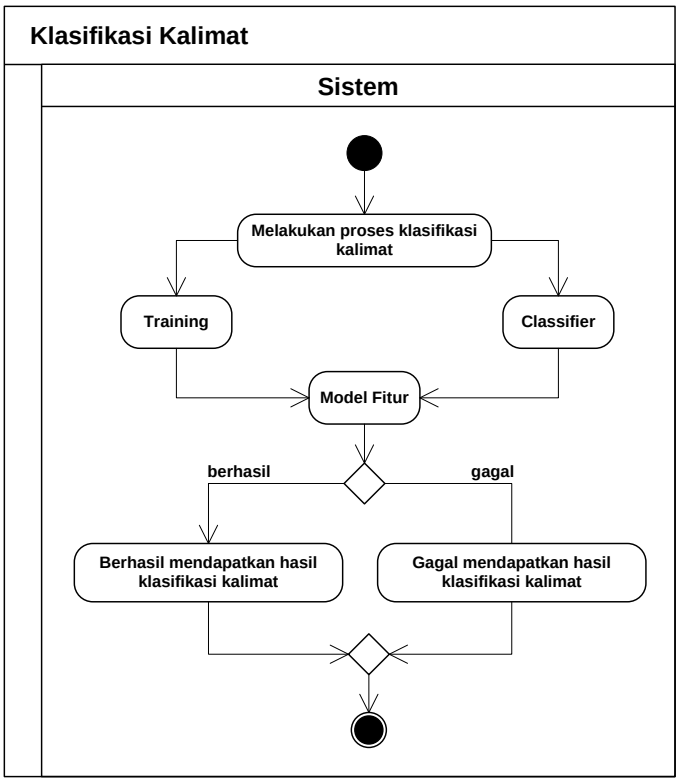
Gambar 3.26 *Activity Diagram* Perhitungan *Cosine Similarity*



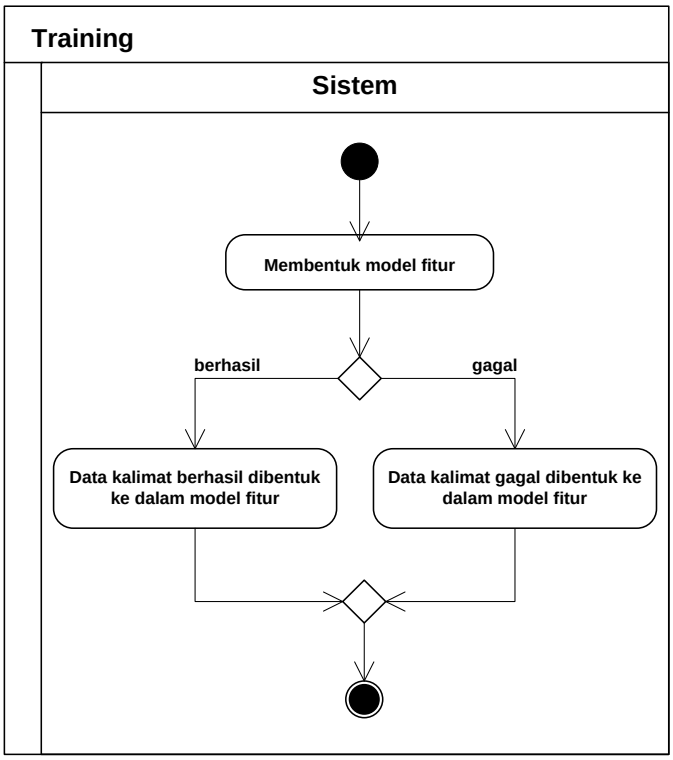
Gambar 3.27 ***Activity Diagram Query Relevance***



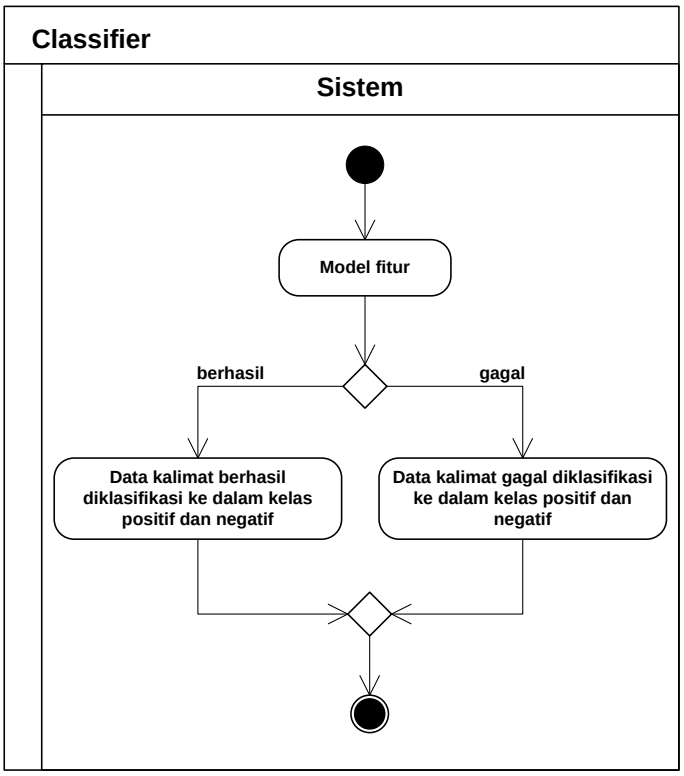
Gambar 3.28 ***Activity Diagram Similarity*** **Kalimat**



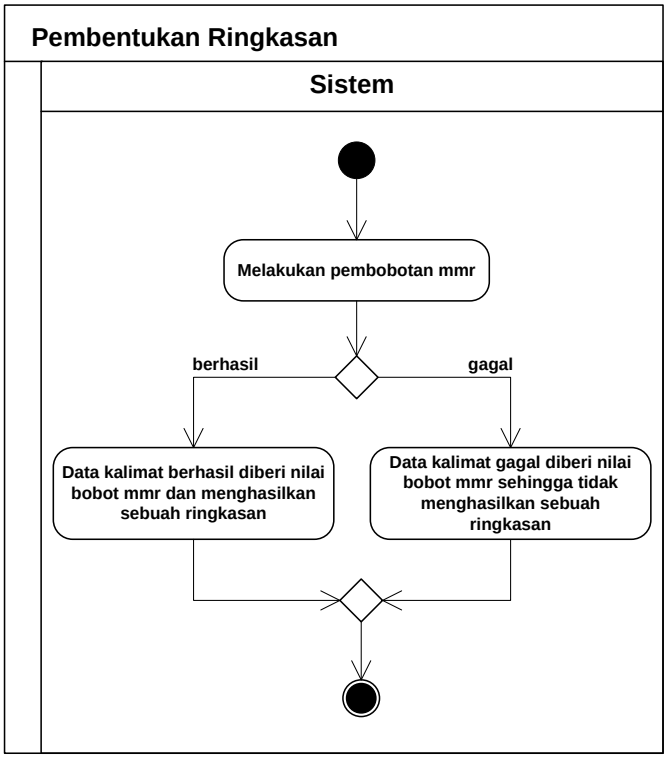
Gambar 3.29 *Activity Diagram* Klasifikasi Kalimat



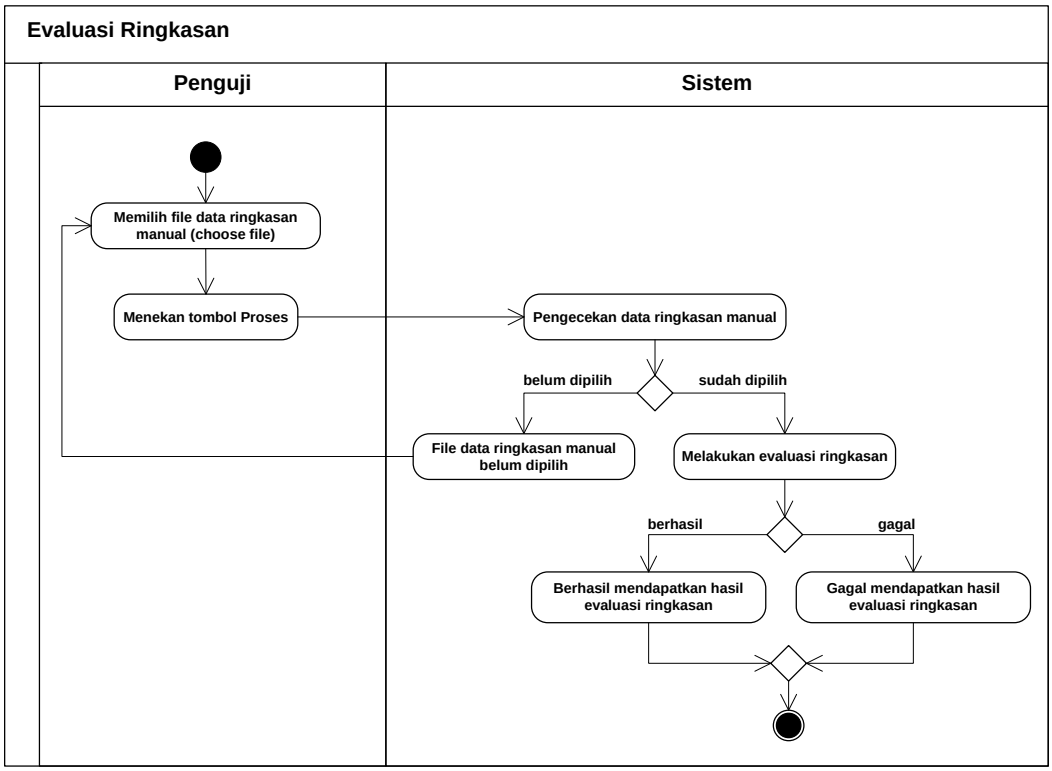
Gambar 3.30 *Activity Diagram Training*



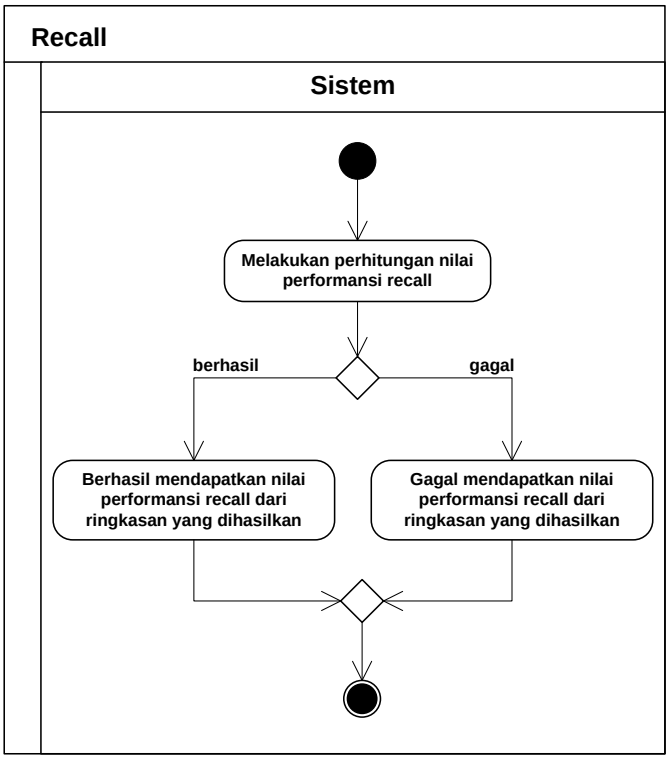
Gambar 3.31 ***Activity Diagram Classifier***



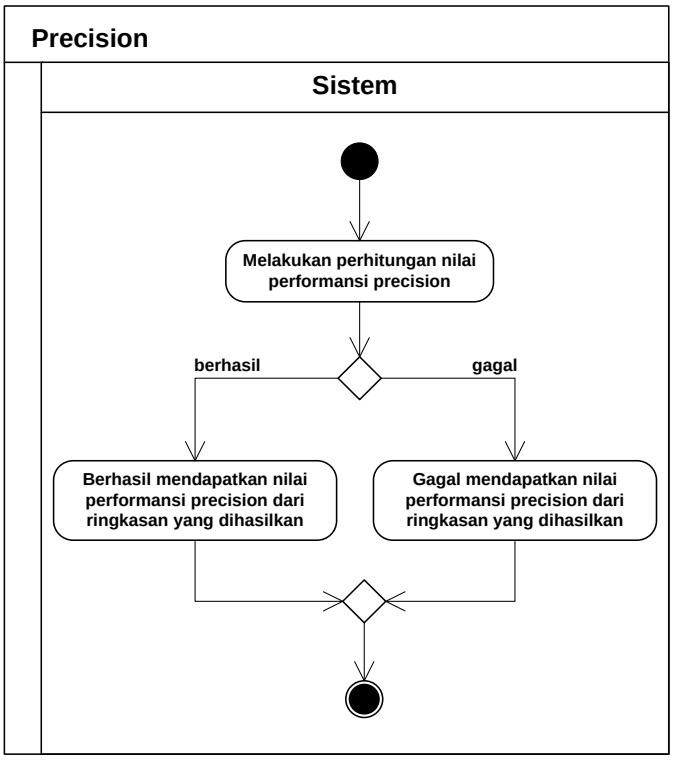
Gambar 3.32 ***Activity Diagram*** **Pembentukan Ringkasan**



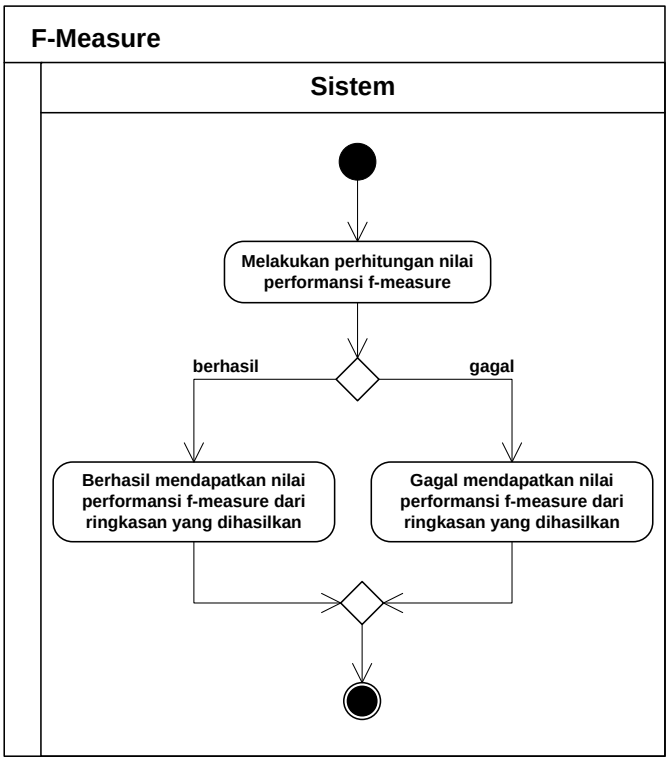
Gambar 3.33 *Activity Diagram* Evaluasi Ringkasan



Gambar 3.34 *Activity Diagram Recall*



Gambar 3.35 ***Activity Diagram Precision***



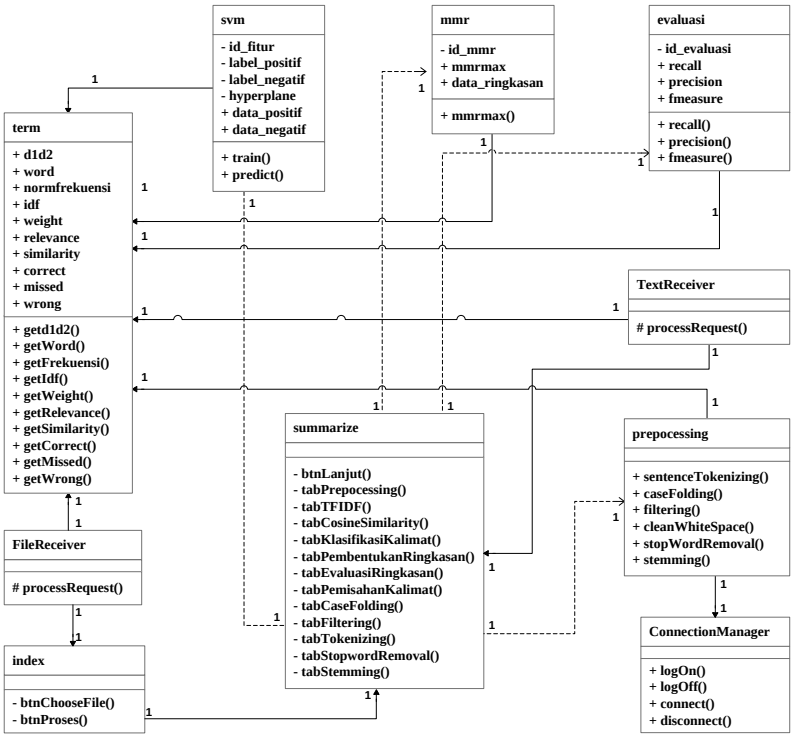
Gambar 3.36 ***Activity Diagram F-Measure***

3.12.4 *Class Diagram*

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari desain berorientasi objek. Berikut adalah penjelasan berdasarkan jenis pada *class diagram* pada Tabel 3.47 dan bentuk *class diagram* pada Gambar 3.37 berikut:

Tabel 3.47 Jenis *Class*

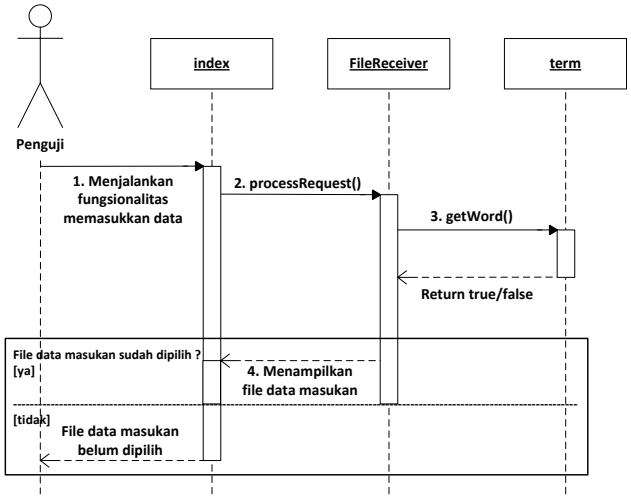
Jenis	Nama *Class*
Model	term
View	index
	summarize
Controller	ConnectionManager
	FileReceiver
	TextReceiver
	Preprocessing
	svm
	mmr
	evaluasi



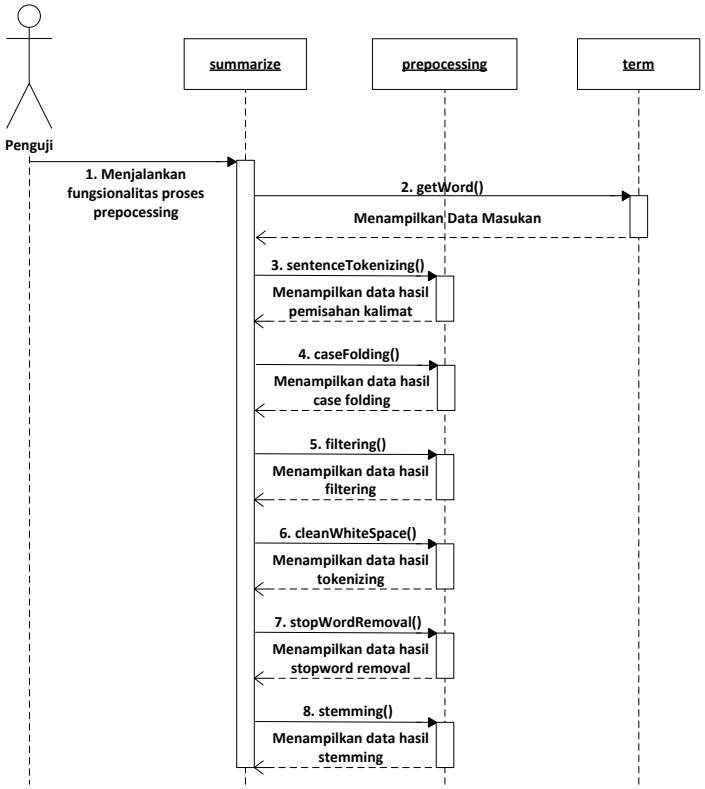
Gambar 3.37 *Class Diagram*

3.12.5 *Sequence Diagram*

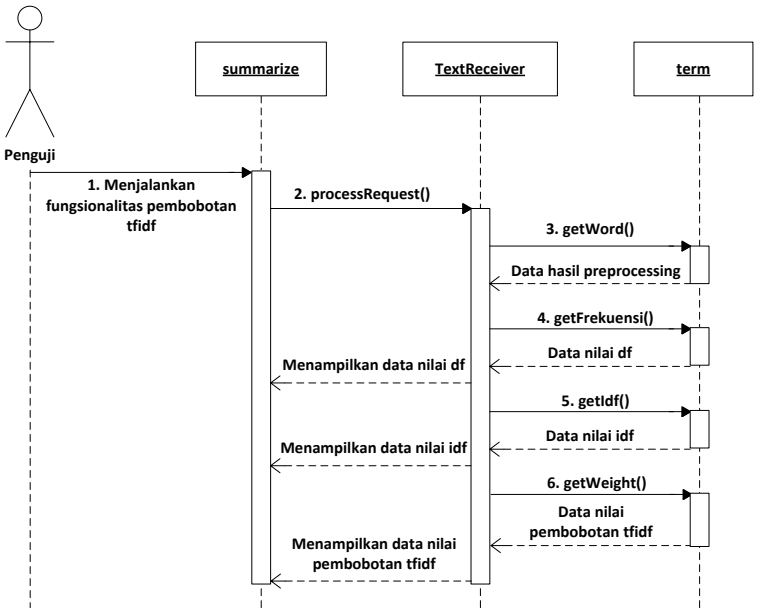
Berikut ini adalah *sequence diagram* yang terdapat pada pembangunan aplikasi yang dibangun.



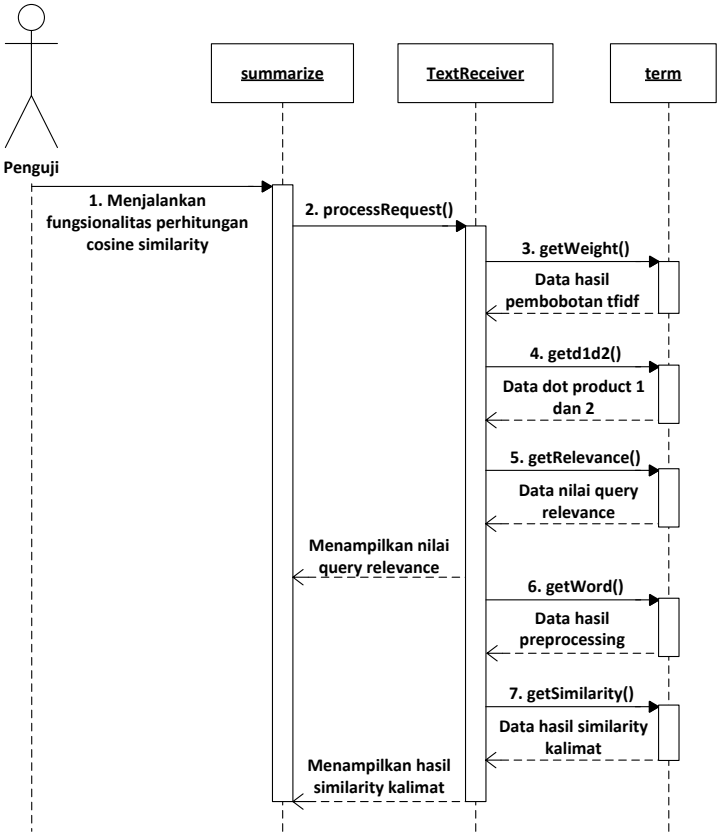
Gambar 3.38 *Sequence Diagram* Pemilihan Data Masukan



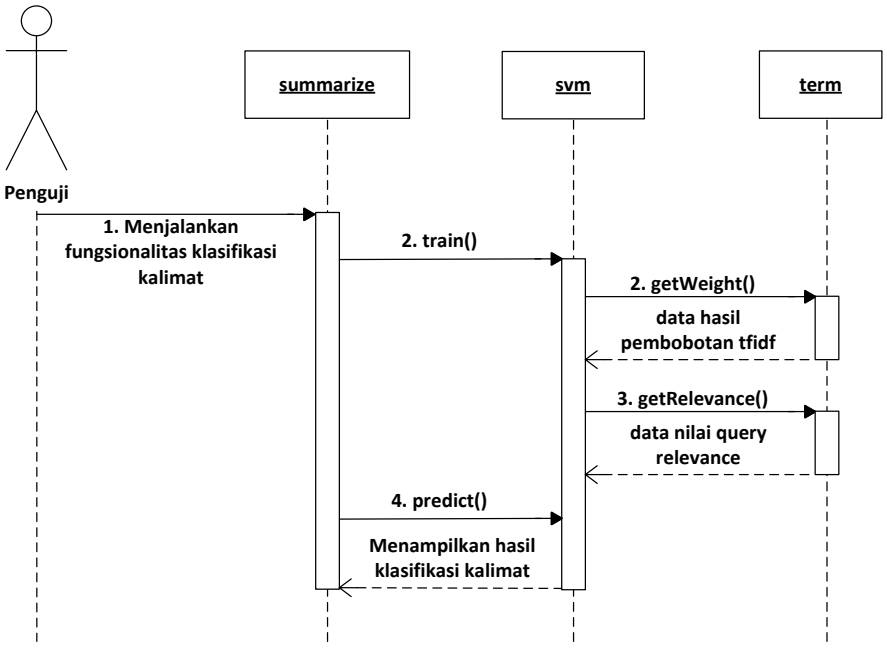
Gambar 3.39 *Sequence Diagram* Proses *Preprocessing*



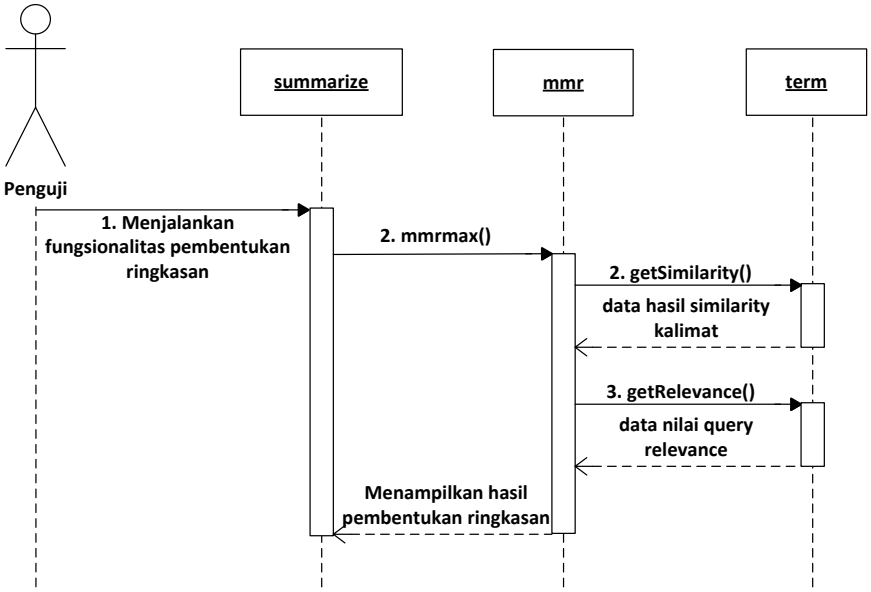
Gambar 3.40 *Sequence Diagram* Pembobotan TF-IDF



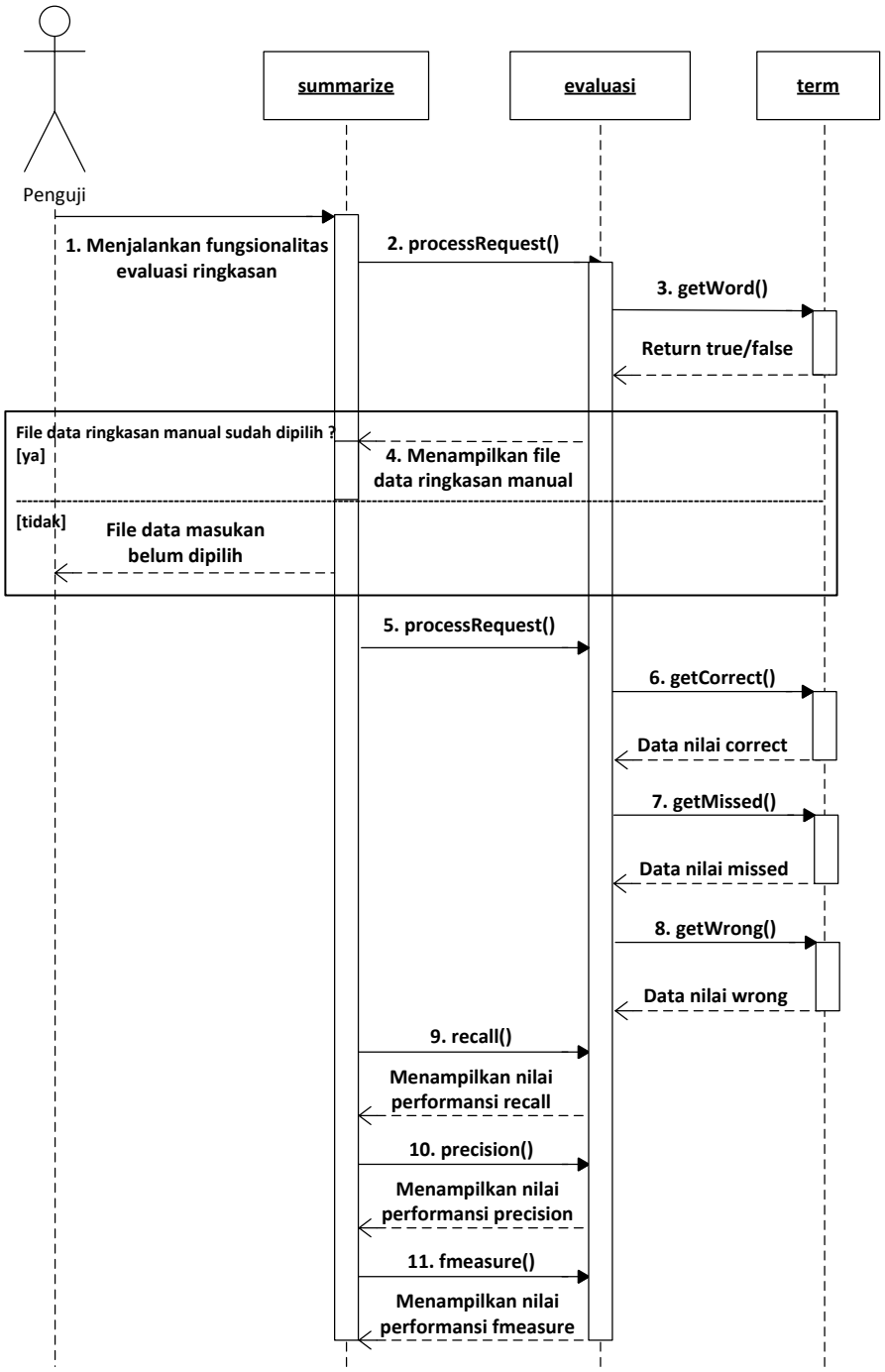
Gambar 3.41 *Sequence Diagram* Perhitungan *Cosine Similarity*



Gambar 3.42 *Sequence Diagram* Klasifikasi Kalimat



Gambar 3.43 *Sequence Diagram* Pembentukan Ringkasan



Gambar 3.44 *Sequence Diagram* Evaluasi Ringkasan

3.13 Perancangan Sistem

Perancangan sistem merupakan penggambaran dan perencanaan sistem untuk mengimplementasikan hasil analisis yang dilakukan sebelumnya. Perancangan sistem yang dibuat terdiri dari perancangan basis data serta perancangan antarmuka.

3.13.1 Struktur Tabel

Struktur tabel menggambarkan detail tabel yang berisi *field*, tipe data, dan keterangan lainnya.

1. Tabel tb_stopword

Tabel stopword merupakan tabel yang akan digunakan untuk menyimpan data kamus atau *list stopword*. Untuk lebih jelasnya ditampilkan pada Tabel 3.48 berikut:

Tabel 3.48 Struktur Tabel tb_stopword

Nama *Field*	Tipe Data	Ukuran	Keterangan
id_kata	integer	10	*PRIMARY KEY, AUTO INCREMENT*
kata	varchar	20	*NOT NULL*

2. Tabel tb_katadasar

Tabel tb_katadasar merupakan tabel yang akan digunakan untuk menyimpan list kata dasar. Untuk lebih jelasnya ditampilkan pada Tabel 3.49 berikut:

Tabel 3.49 Struktur Tabel tb_katadasar

Nama *Field*	Tipe Data	Ukuran	Keterangan
id_katadasar	integer	10	*PRIMARY KEY, AUTO INCREMENT*
katadasar	varchar	20	*NOT NULL*
tipe_katadasar	varchar	25	*NOT NULL*

3. Tabel tb_svm

Tabel svmtraining adalah tabel yang akan digunakan untuk menyimpan data *training* dan *classifier*. Untuk lebih jelasnya ditampilkan pada Tabel 3.50 berikut:

Tabel 3.50 Struktur Tabel tb_svm

Nama *Field*	Tipe Data	Ukuran	Keterangan
id_fitur	integer	10	*PRIMARY KEY, AUTO INCREMENT*

Nama *Field*	Tipe Data	Ukuran	Keterangan
label_positif	integer	1	*NOT NULL*
label_negatif	integer	1	*NOT NULL*
hyperplane	double	-	*NOT NULL*
data_positif	double	-	*NOT NULL*
data_negatif	double	-	*NOT NULL*

4. Tabel tb_mmr

Tabel mmr merupakan tabel yang akan digunakan untuk menyimpan data hasil pembobotan mmr dan pembentukan ringkasan. Untuk lebih jelasnya ditampilkan pada Tabel 3.51 berikut:

Tabel 3.51 Struktur Tabel tb_mmr

Nama *Field*	Tipe Data	Ukuran	Keterangan
id_mmr	integer	10	*PRIMARY KEY*
mmrmax	double	-	*NOT NULL*
data_ringkasan	text	-	*NOT NULL*

5. Tabel tb_evaluasi

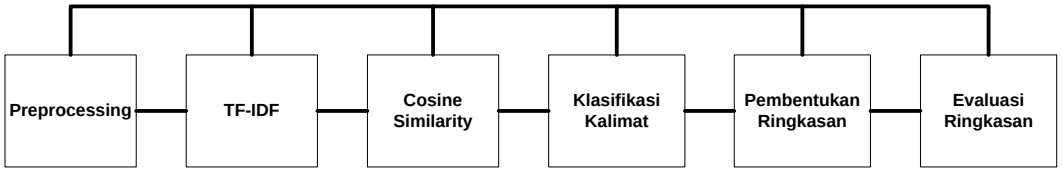
Tabel evaluasi merupakan tabel yang akan digunakan untuk menyimpan data hasil evaluasi ringkasan. Untuk lebih jelasnya ditampilkan pada Tabel 3.52 berikut:

Tabel 3.52 Struktur Tabel tb_evaluasi

Nama *Field*	Tipe Data	Ukuran	Keterangan
id_evaluasi	integer	10	*PRIMARY KEY*
recall	double	-	*NOT NULL*
precision	double	-	*NOT NULL*
fmeasure	double	-	*NOT NULL*

3.13.2 Struktur Menu

Perancangan struktur menu merupakan gambar alur pemakaian sistem, sehingga sistem yang dibangun mudah dipahami dan mudah digunakan. Berikut ini adalah perancangan struktur menu yang akan diterapkan pada aplikasi. Adapun perancangan struktur menu dapat dilihat pada Gambar 3.45 berikut:



Gambar 3.45 Struktur Menu

3.14 Perancangan Antarmuka

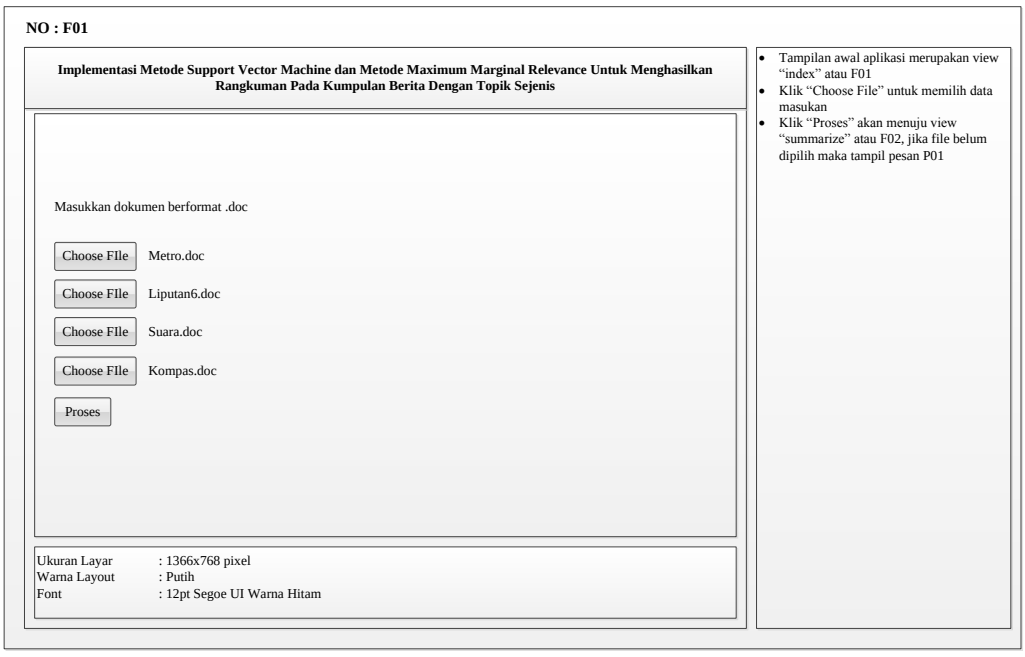
Perancangan antarmuka mendeskripsikan rencana tampilan yang akan digunakan pada aplikasi yang akan dibangun. Perancangan antarmuka yang ada terdiri dari perancangan *form*, perancangan pesan dan jaringan semantik.

3.14.1 Perancangan *Form*

Perancangan antarmuka *form* merupakan rancangan dalam menentukan tata letak dari tampilan sistem yang akan dibuat. Rancangan tersebut dibuat sebagai perancangan *form* pada sistem. Berikut adalah perancangan *form* pada aplikasi yang akan dibangun.

1. F01 – *Form* Pemilihan Data Masukan

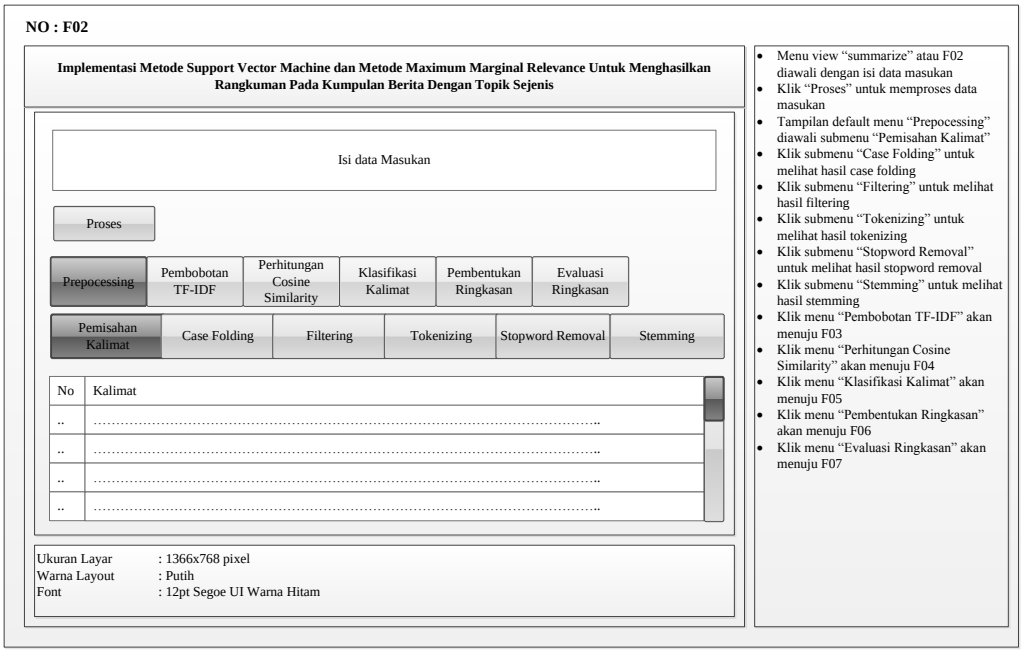
Perancangan F01 merupakan halaman utama pada saat pertama kali menjalankan program. Tampilan *default* awal program diawali tampilan Pemilihan Data Masukkan. Perancangan F01 dapat dilihat pada Gambar 3.46 berikut:



Gambar 3.46 Perancangan *Form* Pemilihan Data Masukan

2. F02 – *Form* Proses *Preprocessing*

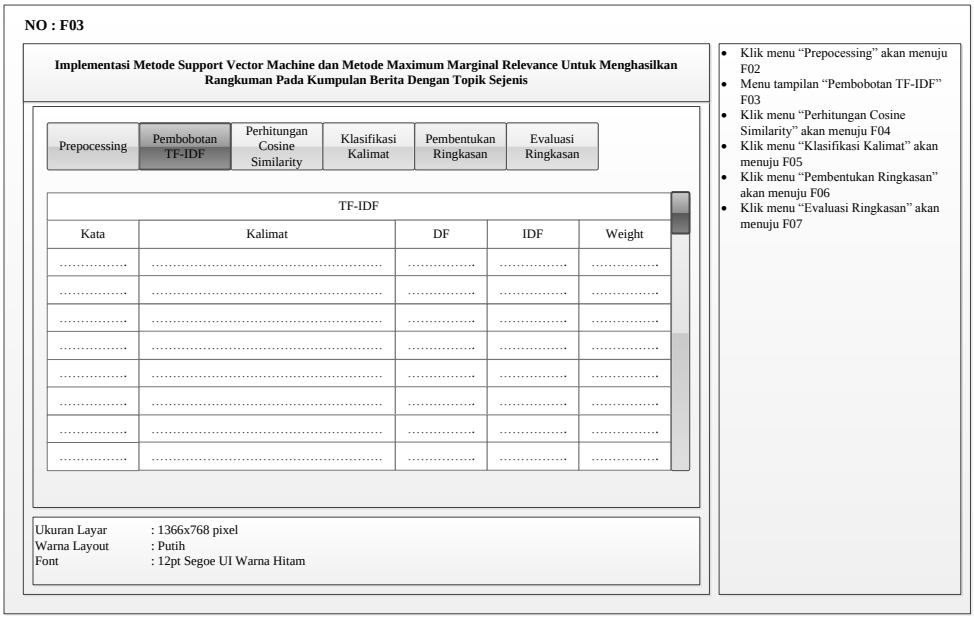
Perancangan F02 proses *preprocessing* merupakan halaman proses selanjutnya setelah data masukan berhasil valid. Tampilan F02 menampilkan tahap-tahap proses *preprocessing* yang dilakukan, dengan tampilan *default* diawali pemisahan kalimat. Perancangan F02 dapat dilihat pada Gambar 3.47 berikut:



Gambar 3.47 Perancangan *Form* Proses *Preprocessing*

3. F03 – *Form* Pembobotan TF-IDF

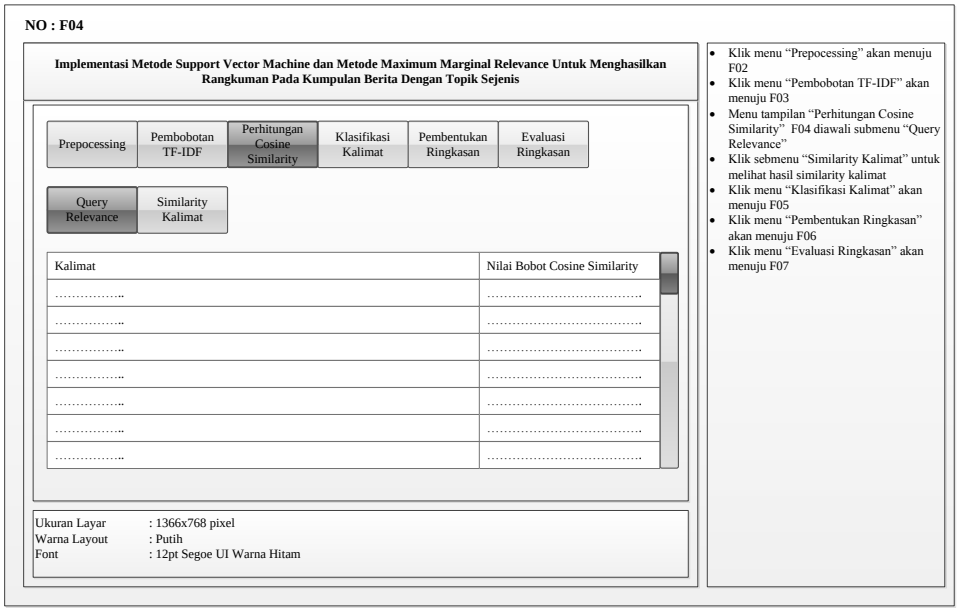
Perancangan F03 pembobotan TF-IDF merupakan halaman proses selanjutnya setelah proses *preprocessing* berhasil dilakukan. Tampilan F03 menampilkan nilai hasil pembobotan tf-idf yang dilakukan. Perancangan F03 dapat dilihat pada Gambar 3.48 berikut:



Gambar 3.48 Perancangan *Form* Pembobotan TF-IDF

4. F04 - *Form* Perhitungan *Cosine Similarity*

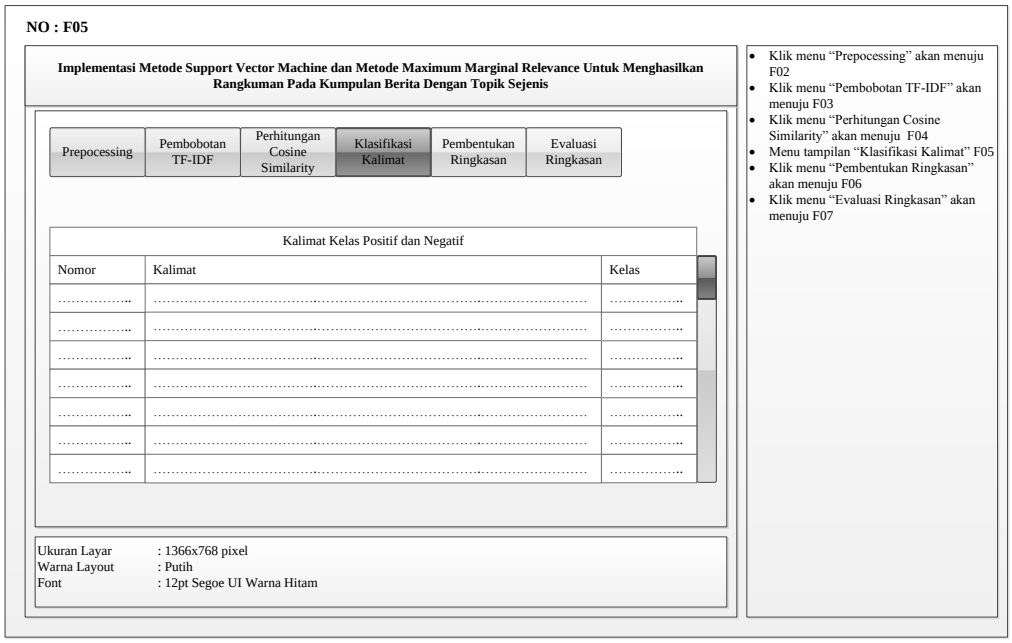
Perancangan F04 perhitungan *cosine similarity* merupakan halaman proses selanjutnya setelah pembobotan tf-idf berhasil dilakukan. Tampilan F04 menampilkan nilai hasil pembobotan *query relevance* dan *similarity* kalimat yang merupakan bagian dari perhitungan *cosine similarity*. Perancangan F04 dapat dilihat pada Gambar 3.49 berikut:



Gambar 3.49 Perancangan *Form* Perhitungan *Cosine Similarity*

5. F05 – *Form* Klasifikasi Kalimat

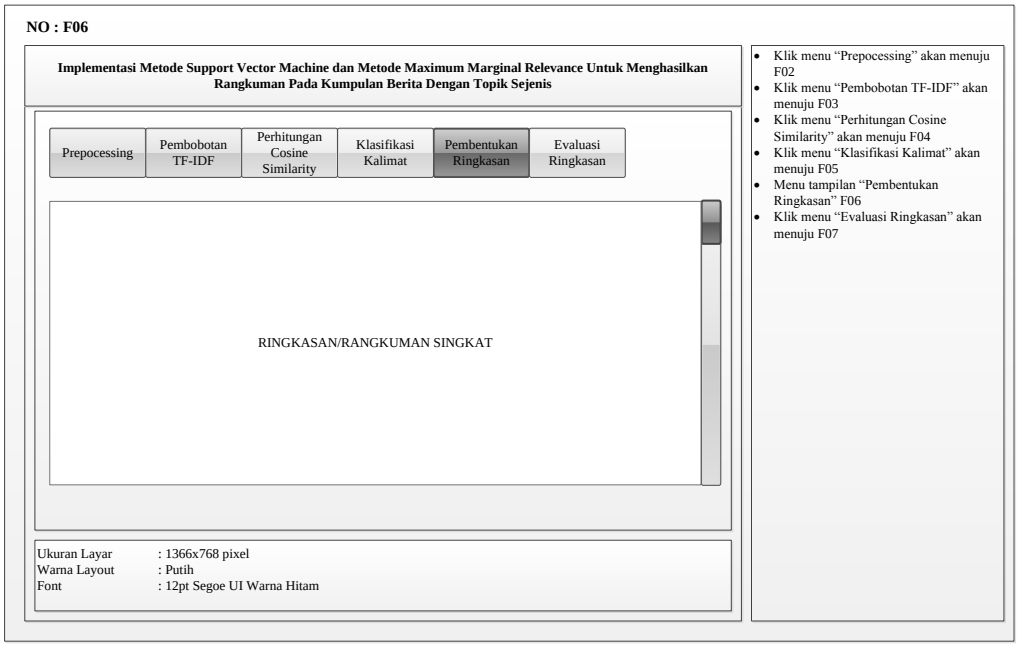
Perancangan F05 klasifikasi kalimat merupakan halaman proses selanjutnya setelah perhitungan *cosine similarity* berhasil dilakukan. Tampilan F05 menampilkan hasil klasifikasi kalimat kedalam kelas positif dan kelas negatif. Perancangan F05 dapat dilihat pada Gambar 3.50 berikut:



Gambar 3.50 Perancangan *Form* Klasifikasi Kalimat

6. F06 – *Form* Pembentukan Ringkasan

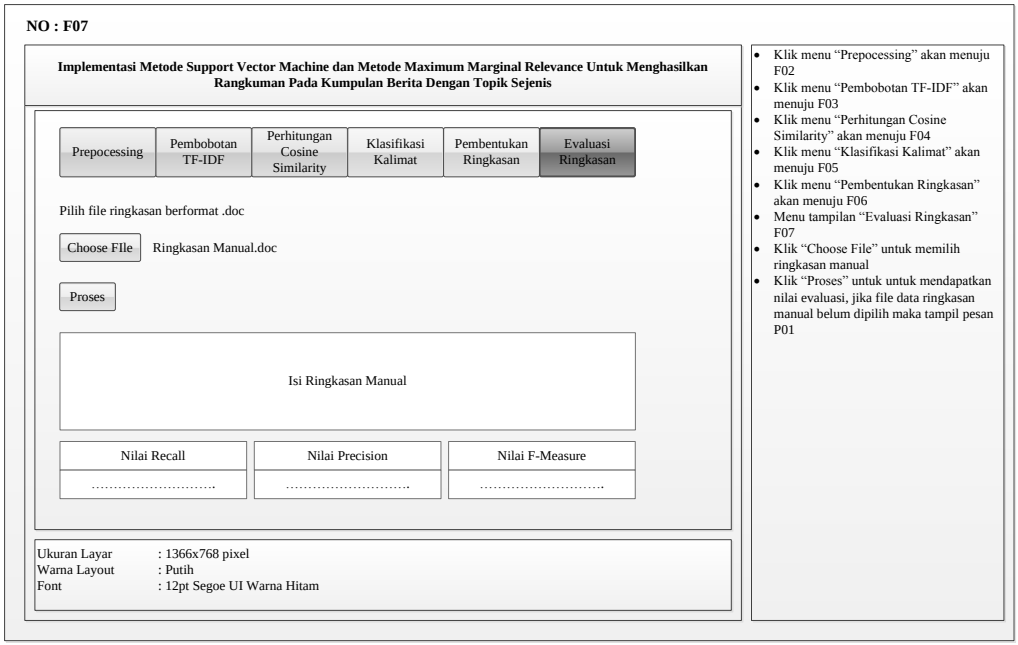
Perancangan F06 pembentukan ringkasan merupakan halaman proses selanjutnya setelah proses klasifikasi kalimat berhasil dilakukan. Tampilan F06 menampilkan sebuah ringkasan yang dihasilkan dengan melakukan pembobotan mmr. Perancangan F06 dapat dilihat pada Gambar 3.51 berikut:



Gambar 3.51 Perancangan *Form* Pembentukan Ringkasan

7. F07 – Evaluasi Ringkasan

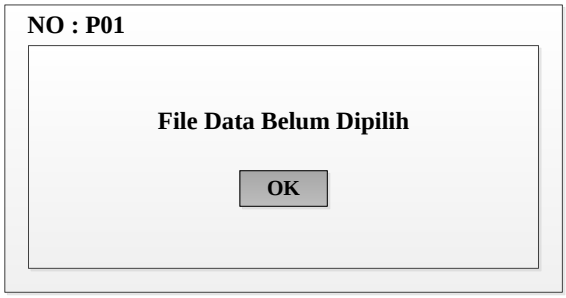
Perancangan F07 evaluasi ringkasan merupakan halaman proses selanjutnya setelah proses pembentukan ringkasan berhasil dilakukan. Tampilan F07 menampilkan nilai hasil evaluasi ringkasan berupa nilai performansi (*recall, precision* dan *f-measure*). Perancangan F07 dapat dilihat pada Gambar 3.52 berikut:



Gambar 3.52 Perancangan *Form* Evaluasi Ringkasan

3.14.2 Perancangan Pesan

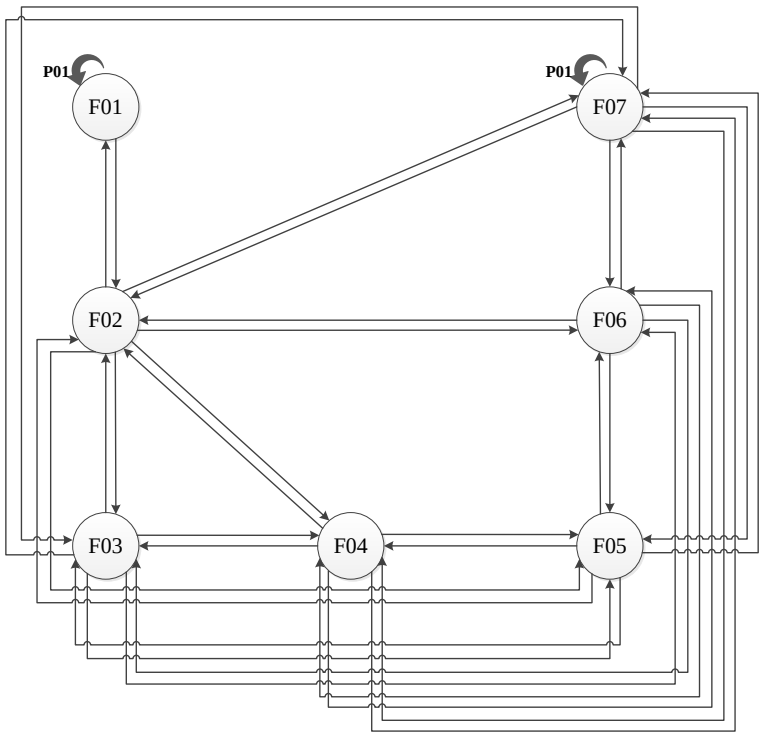
Perancangan pesan merupakan validasi dan pesan yang akan disampaikan oleh sistem dalam keadaan-keadaan tertentu. Adapun perancangan pesan dapat dilihat pada Gambar 3.53 berikut:



Gambar 3.53 Perancangan Pesan

3.14.3 Jaringan Semantik

Jaringan semantik menggambarkan keterhubungan navigasi menu dari satu antarmuka ke antarmuka lain. Jaringan semantik yang terbentuk pada sistem ini dapat dilihat pada Gambar 3.54 berikut:



Gambar 3.54 Jaringan Semantik