BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Information retrieval system atau sistem temu kembali informasi adalah sebuah kesatuan sistem yang memiliki fungsi utama untuk menemukan kembali (*retrieve*) sebuah informasi yang relevan dengan *query* atau *keyword* masukan dari pengguna. Informasi yang diambil melalui *information retrieval* ini dapat berupa teks, citra, dan sebagainya.

Pada penerapannya terdapat berbagai metode yang dapat digunakan dalam *information retrieval*, metode yang sering digunakan adalah metode berbasis pembobotan frekuensi kemunculan kata (*term*) yaitu *term frequency inverse document frequency* (TF-IDF). Pencarian informasi menggunakan metode ini masih memiliki kelemahan, diantaranya tidak dapat mengetahui makna tersembunyi dari sebuah kata, tidak dapat mengetahui kata yang memiliki makna lebih dari satu (polisemi) dan kata yang memiliki makna sama dengan kata lainnya (sinonim).

Maka diperlukan sebuah sistem temu kembali informasi yang dapat menjawab kelemahan dari metode-metode berbasis pembobotan frekuensi kemunculan kata seperti TF-IDF dan *Vector Space Model* (VSM). Sistem temu kembali informasi yang dibuat menggunakan metode pemodelan topik *Probabilistic Latent Semantic Analysis* (PLSA), sehingga bukan hanya menyocokkan *query* yang diinputkan oleh pengguna dengan dokumen yang ada, namun juga dapat mengenali *query* dengan topik yang sama dengan dokumen-dokumen yang tersimpan dalam *database*.

3.2 Hasil Analisis

Hasil analisis yang dapat dilakukan dari sistem temu kembali informasi yang dibangun dapat membantu mahasiswa untuk mencari dokumen skripsi terdahulu dengan mudah, cepat, dan relevan. Pembuatan sistem temu kembali informasi ini menggunakan metode pemodelan topik *Probabilistic Latent Semantic Analysis (PLSA)*. Penggunaan metode ini dianggap mampu menjawab

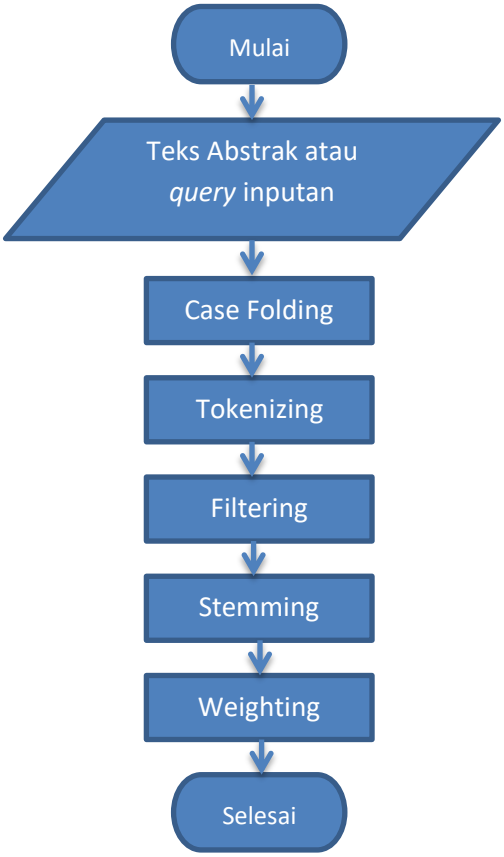
kelemahan dari beberapa metode sebelumnya seperti metode *Vector Space Model* (VSM). Dengan menggunakan metode pemodelan topik, bukan hanya *term frequency* yang menjadi acuan sebuah bobot dokumen terhadap *query*, melainkan terdapat satu buah variabel baru yaitu topik. Topik dalam hal ini bertindak sebagai penghubung antara *term* dan dokumen, masing-masing *term* dan dokumen akan memiliki bobot terhadap suatu topik. Dengan demikian metode ini dapat menggali sebuah makna yang terkandung dalam term maupun dokumen, sehingga akan lebih baik dalam mencari kemiripan *query* dengan dokumen yang ada dalam sistem.

3.2.1 Deskripsi Sistem

Sistem yang dibangun adalah aplikasi dengan konsep temu kembali informasi yang dapat melakukan pencarian tehadap dokumen skripsi menggunakan metode pemodelan topik *Probabilistic Latent Semantic Analysis (PLSA)*. Tujuan dari sistem ini adalah untuk mengukur kemiripan dan relevansi antara *query* yang dimasukkan pengguna dan dokumen yang terambil.

3.2.1.1 Preprocessing

Secara umum sistem ini memiliki beberapa tahapan proses, yang pertama adalah *preprocessing.* Tahap *preprocessing* mencakup *case folding*, *tokenizing*, *filtering*, *stemming* dan *weighting*. Gambar 3.1 akan menjelaskan alur proses *preprocessing*.



Gambar 3.1 Diagram Alir Proses *Preprocessing*

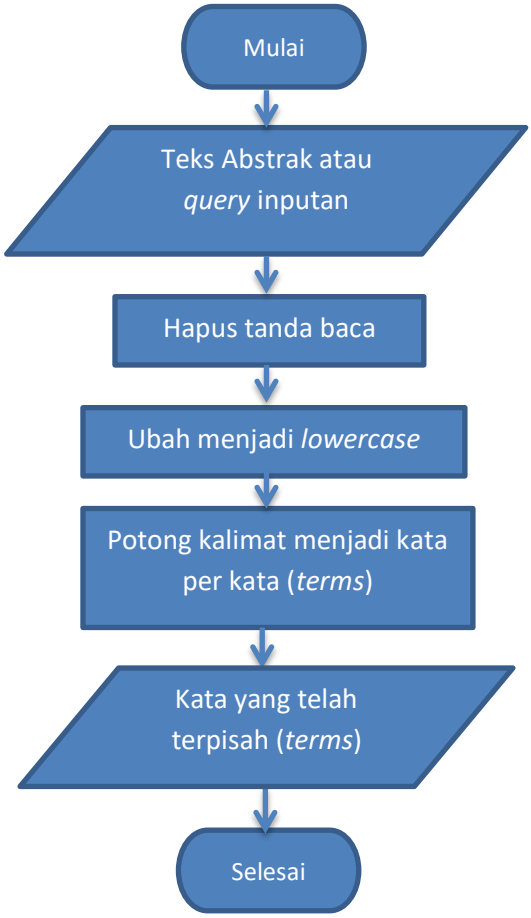
Penjelasan Gambar 3.1 :

1. Teks abstrak dan *query* yang dimasukkan oleh pengguna digunakan sebagai input yang akan dilakukan *preprocessing*.
2. Tahapan *preprocessing* yang pertama adalah *Case Folding* yang memiiliki fungsi untuk mengubah *case* dari karakter teks menjadi sama (*lowercase* atau *uppercase*).
3. Setelah dilakukan *case folding*, teks tersebut akan dilakukan *tokenizing*. Tahap ini bertujuan untuk memisahkan antar kata yang didalam teks inputan.
4. Tahap selanjutnya adalah *filtering*, yang dimaksud *filtering* adalah membuang kata-kata yang sering muncul dalam dokumen *(stopwords)*, penghilangan kata-kata ini bertujuan untuk menormalisasi data sehingga tidak mempengaruhi bobot kata (*term*) yang dianggap penting dalam suatu dokumen.

5. *Stemming* adalah proses pemisahan kata-kata yang berimbuhan sehingga berubah menjadi kata dasar.
6. Tahap terakhir dari *preprocessing* adalah *weighting,* yang berfungsi untuk memberi bobot pada tiap kata (*term*). Bobot ini selanjutnya akan digunakan sebagai nilai atribut pada metode selanjutnya.

3.2.1.1.1 *Tokenizing*

Salah satu tahapan penting dalam *preprocessing* adalah *tokenizing.* Tahap ini bertujuan untuk memisahkan antar kata yang didalam teks (dokumen). Proses ini merupakan proses pemotongan *string* masukan berdasarkan tiap kata yang menyusunnya. Pada prinsipnya proses ini adalah memisahkan setiap kata yang menyusun suatu dokumen. Pada umumnya setiap kata teridentifikasi atau terpisahkan dengan kata yang lain oleh karakter spasi, sehingga proses *tokenizing* mengandalkan karakter spasi pada dokumen untuk melakukan pemisahan kata. Berikut adalah diagram alir *tokenizing* :



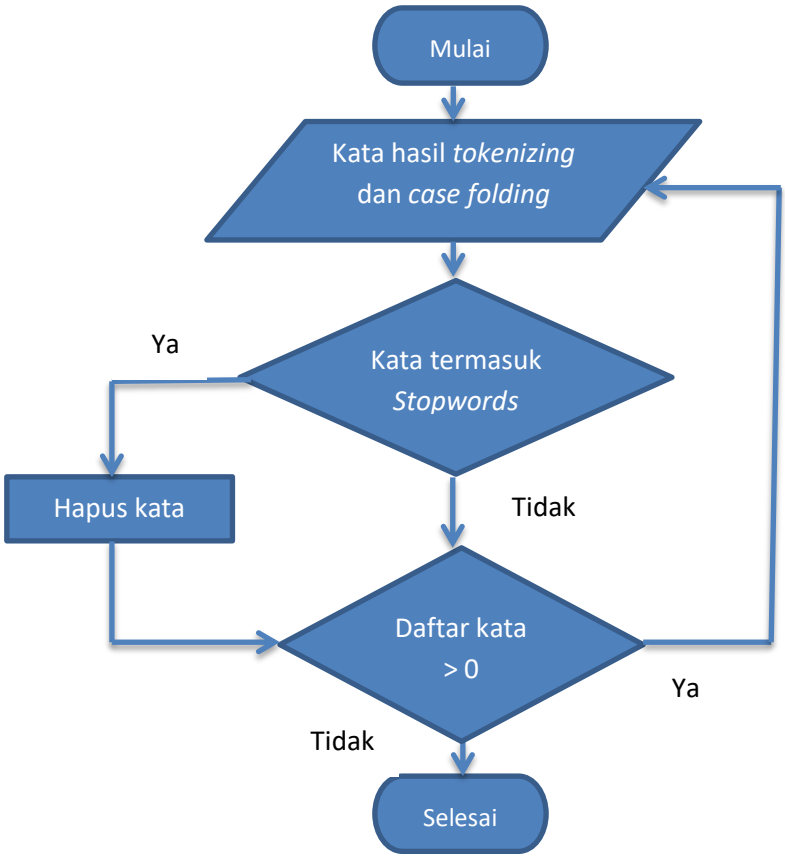
Gambar 3.2 Diagram Alir *Tokenizing*

Proses yang terjadi pada gambar 3.2 adalah sebagai berikut :

1. Sistem membaca teks abstrak atau *query* yang diinputkan pengguna, dan akan disimpan didalam *database*.
2. Sistem menghilangkan tanda baca atau karakter yang tidak digunakan dalam *preprocessing*.
3. Sistem mengubah *case* dari teks menjadi huruf kecil (*lowercase*).
4. Sistem memisah kalimat menjadi kata per kata menggunakan tanda spasi sebagai indicator pemisah kata.
5. Sistem menghasilkan kumpulan kata yang telah terpisah dari teks awal dan akan disimpan didalam *database* untuk diproses pada tahap selanjutnya.

3.2.1.1.2 *Filtering*

Proses *filtering* merupakan proses penting dalam tahapan *preprocessing*. Proses Filtering adalah proses pengambilan kata-kata yang dianggap penting atau mempunyai makna saja. Pada proses ini kata-kata yang dianggap tidak mempunyai makna seperti kata sambung akan dihilangkan. Pada proses *filtering* biasanya digunakan daftar stopword yang tersimpan dalam suatu tabel basis data, yang nantinya digunakan sebagai acuan penghilangan kata. Daftar *stopword*s berbeda untuk setiap bahasanya. Berikut diagram alir dari proses *filtering* :



Gambar 3.3 Diagram Alir Proses *Filtering*.

Proses yang terjadi pada gambar 3.3 adalah sebagai berikut :

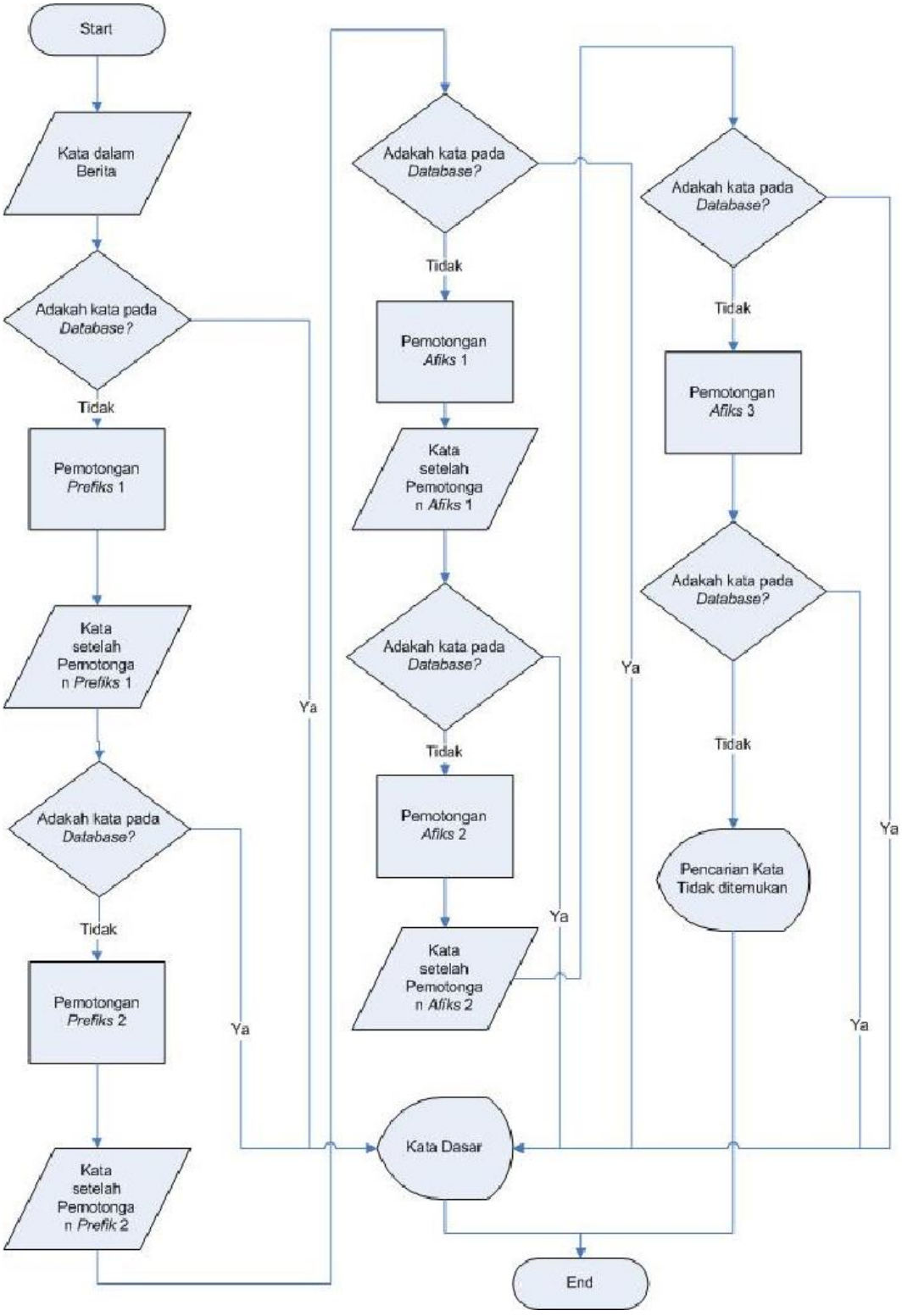
1. Sistem mengambil kata dari hasil *tokenizing* dan *case folding*.
2. Sistem melakukan pengecekan kata yang ada dalam *stoplist* yang tersimpan di *database* apakah ada kata yang sama apa tidak, jika ada

yang sama maka sistem akan melakukan penghapusan pada kata tersebut, jika tidak ada maka kata tersebut akan tersimpan untuk diproses pada tahapan selanjutnya.

3. Sistem akan melakukan pengecekan secara berulang-ulang (*iterative*) sampai tidak ada kata yang sama.

3.2.1.1.3 *Stemming*

Tahap selanjutnya dari *preprocessing* adalah proses *stemming*. Proses stemming adalah proses untuk mencari *root* dari kata hasil dari proses filtering. Pencarian *root* sebuah kata atau biasa disebut dengan kata dasar dapat memperkecil hasil indeks tanpa harus menghilangkan makna. Berikut adalah diagram alir algoritma *stemming* bahasa indonesia Nazief-Adriani : (Liyantanto, 2010)



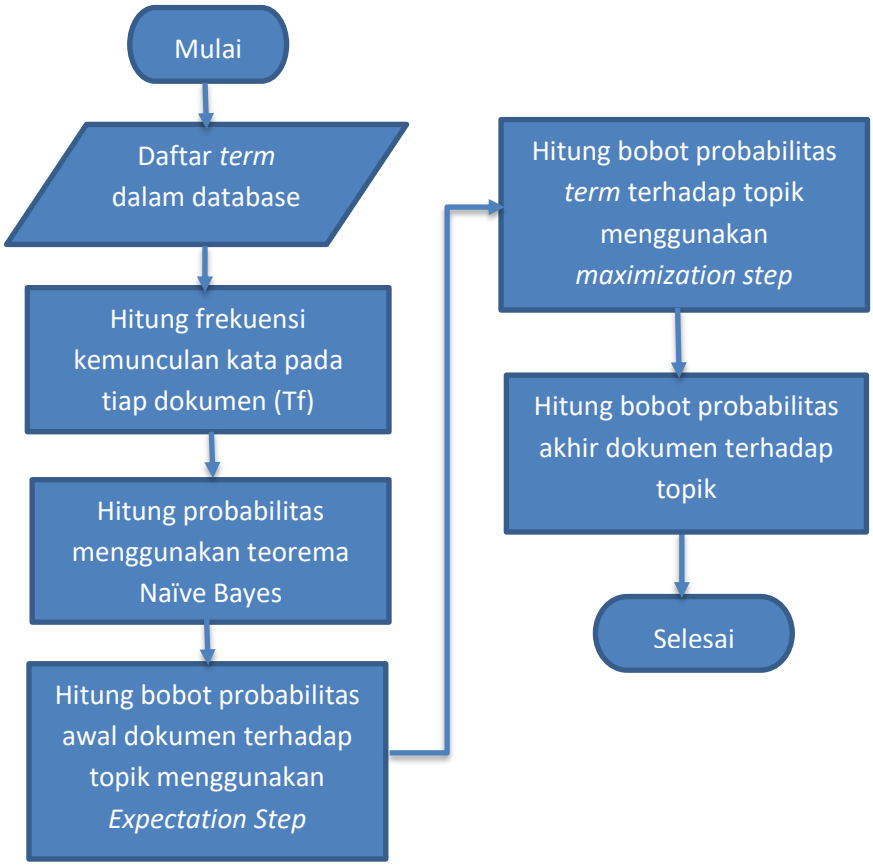
Gambar 3.4 Diagram Alir Algoritma *Stemming* Bahasa Indonesia Nazief-Adriani

Proses tahapan *stemming* Nazief dan Adriani yang terjadi pada gambar 3.4 :

1. Pertama cari kata yang akan diistem dalam kamus kata dasar. Jika ditemukan maka diasumsikan kata 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 di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
    a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
    b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
    a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
    b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika root word belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
5. Melakukan Recoding.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

3.2.1.2 Weighting (Pembobotan) Pemodelan Topik

Pembobotan pada pemodelan topik yang digunakan pada penilitian ini meliputi perhitungan *term frequency* (TF), probabilitas Naïve Bayes, dan perhitungan *Probabilistic Latent Semantic Analysis* (PLSA) yang terbagi menjadi dua tahap yaitu *Expectation* dan *Maximization*. Bobot yang dihasilkan dari proses pembobotan diatas adalah berupa bobot probabilitas tiap *term* terhadap topik dan bobot probabilitas dokumen terhadap topik. Berikut adalah flowchart pembobotan pemodelan topik yang digunakan dalam penelitian ini :



Gambar 3.5 Diagram Alir Pembobotan Metode Pemodelan Topik

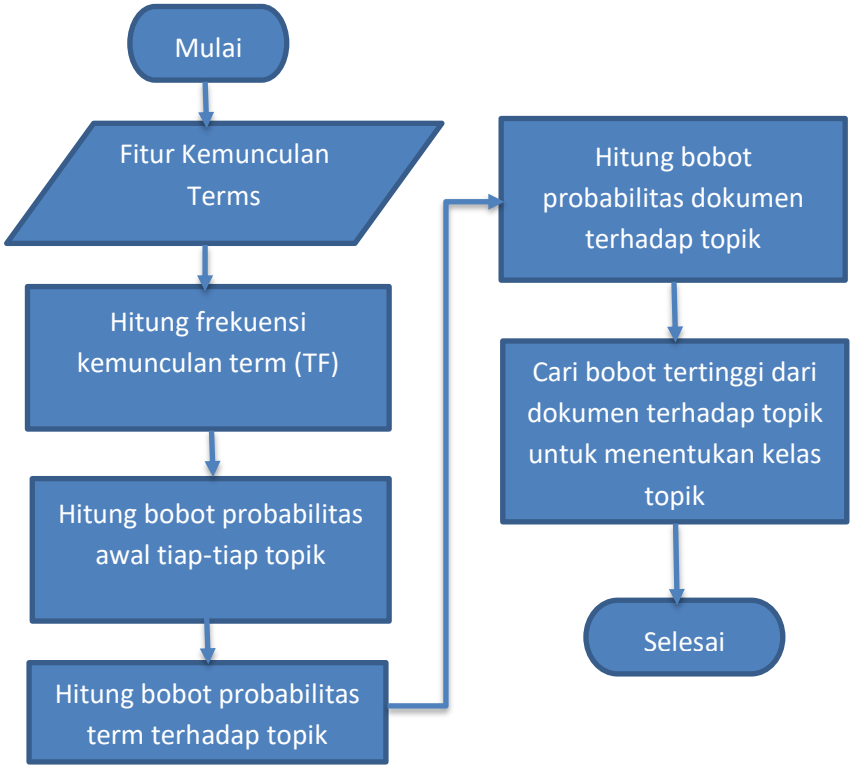
Proses algoritma pembobotan metode pemodelan topik yang terjadi pada gambar 3.5 adalah sebagai berikut :

1. Mengambil data *terms* yang telah tersimpan dalam *database*.

2. Menghitung jumlah kemunculan suatu kata atau *term* pada tiap dokumen yang ada dalam *database*.
3. Menghitung frekuensi dokumen pada tiap *term* yang muncul.
4. Menghitung bobot probabilitas dokumen terhadap topik menggunakan teorema Naïve Bayes, sehingga dapat menentukan sementara topik yang terdapat dalam dokumen.
5. Menghitung bobot probabilitas awal dokumen terhadap topik menggunakan *Expectation Step*. Tahap ini akan memperbaiki bobot yang dihasilkan dari perhitungan Naïve Bayes.
6. Menghitung bobot probabilitas *term* terhadap topik menggunakan *maximization step*. Bobot probabilitas ini yang akan digunakan dalam perhitungan kemiripan *query* dan dokumen.
7. Menghitung bobot probabilitas akhir dokumen terhadap topik dengan cara mengalikan bobot probabilitas *term* yang muncul dalam dokumen tersebut.

3.2.1.2.1 Teorema Naïve Bayes

Teorema Naïve Bayes digunakan dalam penelitian ini untuk menentukan bobot probabilitas pada tahap awal *Probabilistic Latent Semantic Analysis* yakni tahap *Expectation*. Pada penelitian ini untuk mengetahui suatu dokumen bagian dari suatu kategori maka dilakukan proses klasifikasi. Berikut adalah diagram alir proses perhitungan Naïve Bayes :



Gambar 3.6 Diagram Alir Proses Perhitungan Naïve Bayes

Contoh kasus terdapat empat dokumen yang telah dilakukan *preprocessing* dan masing-masing memiliki 3 (tiga) buah *terms* yang masing memiliki frekuensi kemunculan sebagai berikut :

D1 : identifikasi (2), warna(3), olah(2)

D2 : klasifikasi(3), cluster(2), indeks(2)

D3 : klasifikasi(2), identifikasi(1), tekstur(2)

D4 : identifikasi (1), warna (2), tekstur (3)

Tabel 3.1 Fitur Naïve Bayes

Dokumen	Kategori	Fitur (Kemunculan)
Dokumen 1	Pengolahan Citra	identifikasi(2), warna(3), olah(2)
Dokumen 2	Data Mining	klasifikasi(3), cluster(2), indeks(4)
Dokumen 3	?	klasifikasi(2), identifikasi(1), tekstur(2)

Dokumen 4	?	Identifikasi(1), warna(2), tekstur(3)

Tabel 3.2 Frekuensi Kemuculan Term Pada Dokumen

Dokumen	warna	indeks	olah	identifikasi	klasifikasi	cluster	tekstur
Dokumen 1	3	0	2	2	0	0	0
Dokumen 2	0	4	0	0	3	2	0
Dokumen 3	0	0	0	1	2	0	2
Dokumen 4	2	0	0	1	0	0	3

Tabel 3.3 Bobot Probabilitas Naïve Bayes

Kategori	*p(ci)*	*p(wkj.ci)*						
		warna	**indeks**	**olah**	**identifikasi**	**klasifikasi**	**cluster**	**tekstur**
Pengolahan Citra	1/2	4/14	1/14	3/14	3/14	1/14	1/14	1/14
Data Mining	1/2	1/16	5/16	1/16	1/16	4/16	3/16	1/16

p(“Pengolahan Citra”|”dokumen3”) = p(“Pengolahan Citra”) x p(“klasifikasi”|”Pengolahan Citra”) x p(“identifikasi”|”Pengolahan Citra”) x p(“tekstur”|”Pengolahan Citra”)
= 1/2 x 1/14 x 3/14 x 1/14 = **0,000546647**

p(“Data Mining”|”dokumen3”) = p(“Data Mining”) x p(“klasifikasi”|” Data Mining”) x p(“identifikasi”|” Data Mining”) x p(“tekstur”|” Data Mining”) = 1/2 x 4/16 x 1/16 x 1/16 = **0,000488281**

Dengan menggunakan teorema Naïve Bayes dapat diketahui bahwa dokumen 3 termasuk dalam kategori Pengolahan Citra karena bobot probabilitas dokumen 3 terhadap kategori Pengolahan Citra lebih besar daripada bobot probabilitas dokumen 3 terhadap kategori Data Mining yakni **0,000546647** > **0,000488281**.

3.2.1.2.2 *Probabilistic Latent Semantic Analysis*

Setelah diperoleh matriks awal probabilitas dari algoritma Naïve Bayes, matriks tersebut akan diproses ke dalam training untuk memperoleh probabilitas yang terbaik dengan menggunakan algoritma Expectation Maximization (EM).

Tabel 3.4 Fitur Naïve Bayes

Dokumen	Kategori	Fitur (Kemunculan)
Dokumen 1	Pengolahan Citra	identifikasi(2), warna(3), olah(2)
Dokumen 2	Data Mining	klasifikasi(3), cluster(2), indeks(4)
Dokumen 3	?	klasifikasi(2), identifikasi(1), tekstur(2)
Dokumen 4	?	identifikasi (1), warna (2), tekstur (3)

Dengan mengacu pada probabilitas Naïve Bayes sebelumnya, dapat dilakukan perhitungan *Expectation* sebagai berikut :

p(“Pengolahan Citra”|”dokumen3”) = (p(“Pengolahan Citra”) x p(“identifikasi”|”Pengolahan Citra”) x p(“klasifikasi”|”Pengolahan Citra”) x p(“tekstur”|”Pengolahan Citra”)) : ((p(“Pengolahan Citra”) x p(“identifikasi”|”Pengolahan Citra”) x p(“klasifikasi”|”Pengolahan Citra”) x p(“tekstur”|”Pengolahan Citra”)) + (p(“Data Mining”) x p(“identifikasi”|”Data Mining”) x p(“klasifikasi”|”Data Mining”) x p(“tekstur”|”Data Mining”))) = (1/2 x 3/14 x 1/14 x 1/14) : ((1/2 x 3/14 x 1/14 x 1/14)+ (1/2 x 1/16 x 4/16 x 1/16)) = ***0,528198074***

p(“Data Mining”|”dokumen3”) = (p(“Data Mining”) x p(“identifikasi”|”Pengolahan Citra”) x p(“klasifikasi”|”Pengolahan Citra”) x p(“tekstur”|”Pengolahan Citra”)) : ((p(“Pengolahan Citra”) x p(“identifikasi”|”Pengolahan Citra”) x p(“klasifikasi”|”Pengolahan Citra”) x p(“tekstur”|”Pengolahan Citra”)) + (p(“Data Mining”) x p(“identifikasi”|”Data Mining”) x p(“klasifikasi”|”Data Mining”) x p(“tekstur”|”Data Mining”))) =

(1/2 x 1/16 x 4/16 x 1/16) : ((1/2 x 3/14 x 1/14 x 1/14)+ (1/2 x 1/16 x 4/16 x 1/16)) = ***0,471801926***

p("Pengolahan Citra"|"dokumen4") = (1/2 x 3/14 x 4/14 x 1/14) : ((1/2 x 3/14 x 4/14 x 1/14)+ (1/2 x 1/16 x 1/16 x 1/16)) = ***0,947125019***

p("Data Mining"|"dokumen4") = (1/2 x 1/16 x 1/16 x 1/16) : ((1/2 x 3/14 x 4/14 x 1/14)+ (1/2 x 1/16 x 1/16 x 1/16)) = ***0,52874981***

Setelah dilakukan perhitungan pada tahap *Expectation*, langkah selanjutnya adalah proses *Maximization*, berikut adalah perhitungannya :

f(p) *= N + N("warna", "dokumen1") p("Pengolahan Citra"|"dokumen1") + N("warna", "dokumen4") p("Pengolahan Citra"|"dokumen4") + N("olah", "dokumen1") p("Pengolahan Citra"|"dokumen1") + N("identifikasi", "dokumen1") p("Pengolahan Citra"|"dokumen1") + N("identifikasi", "dokumen3") p("Pengolahan Citra"|"dokumen3") + N("identifikasi", "dokumen4") p("Pengolahan Citra"|"dokumen4") + N("klasifikasi", "dokumen3") p("Pengolahan Citra"|"dokumen3") + N("tekstur", "dokumen3") p("Pengolahan Citra"|"dokumen3") + N("tekstur", "dokumen4") p("Pengolahan Citra"|"dokumen4") + N("warna", "dokumen4") p("Data Mining"|"dokumen4") + N("indeks", "dokumen2") p("Data Mining"|"dokumen2") + N("identifikasi", "dokumen3") p("Data Mining"|"dokumen3") + N("identifikasi", "dokumen4") p("Data Mining"|"dokumen4") + N("klasifikasi", "dokumen2") p("Data Mining"|"dokumen2") + N("klasifikasi", "dokumen3") p("Data Mining"|"dokumen3") + N("cluster", "dokumen2") p("Data Mining"|"dokumen2") + N("tekstur", "dokumen3") p("Data Mining"|"dokumen3") + N("tekstur", "dokumen4") p("Data Mining"|"dokumen4")*

f(p) *= 7 + (3×1) + (2×0.9471) + (2×1) + (2×1) + (1×0.5282) + (1×0.9471) + (2×0.4718) + (2×0.5282) + (3×0.9472) + (2×0.0528) + (4×1) +*

(1×0.4718) + (1×0.0528) + (3×0.0528) + (2×0.4718) + (2×0.0528) + (2×0.4718) + (3×0.9471) = ***31.834***

p("warna"\|"Pengolahan Citra") *= (1 + N("warna", "dokumen1") p("Pengolahan Citra"\|"dokumen1") + N("warna", "dokumen2") p("Pengolahan Citra"\|"dokumen2") + N("warna", "dokumen3") p("Pengolahan Citra"\|"dokumen3") + N("warna", "dokumen4") p("Pengolahan Citra"\|"dokumen4")) : f(p)*

p("warna"\|"Pengolahan Citra") = (1 + (3×1) + (0×0) + (0×0.5282) + (2×0.9471) : 31.834 = ***0.185***

p("warna"\|"Data Mining") *= (1 + N("warna", "dokumen1") p("Data Mining"\|"dokumen1") + N("warna", "dokumen2") p("Data Mining"\|"dokumen2") + N("warna", "dokumen3") p("Data Mining"\|"dokumen3") + N("warna", "dokumen4") p("Data Mining"\|"dokumen4")) : f(p)*

p("warna"\|"Data Mining") = (1 + (3×0) + (0×1) + (0×0.4718) + (2×0.05287) : 31.834 = ***0.0347***

p("klasifikasi"\|"Pengolahan Citra") *= (1 + N("klasifikasi", "dokumen1") p("Pengolahan Citra"\|"dokumen1") + N("klasifikasi", "dokumen2") p("Pengolahan Citra"\|"dokumen2") + N("klasifikasi", "dokumen3") p("Pengolahan Citra"\|"dokumen3") + N("klasifikasi", "dokumen4") p("Pengolahan Citra"\|"dokumen4")) : f(p)*

p("klasifikasi"\|"Pengolahan Citra") = (1 + (0×1) + (3×0) + (2×0.5282) + (2×0.9471) : 31.834 = ***0.0645***

p("klasifikasi"\|"Data Mining") *= (1 + N("klasifikasi", "dokumen1") p("Data Mining"\|"dokumen1") + N("klasifikasi", "dokumen2") p("Data Mining"\|"dokumen2") + N("klasifikasi", "dokumen3") p("Data Mining"\|"dokumen3") + N("klasifikasi", "dokumen4") p("Data Mining"\|"dokumen4")) : f(p)*

p(“klasifikasi”|”Data Mining”) = (1 + (0×0) + (3×1) + (2×0.4718) + (0×0.05287) : 31.834 = ***0.1553***

p(“identifikasi”|”Pengolahan Citra”) *= (1 + N(“identifikasi”, “dokumen1”) p(“Pengolahan Citra”|”dokumen1”) + N(“identifikasi”, “dokumen2”) p(“Pengolahan Citra”|”dokumen2”) + N(“identifikasi”, “dokumen3”) p(“Pengolahan Citra”|”dokumen3”) + N(“identifikasi”, “dokumen4”) p(“Pengolahan Citra”|”dokumen4”)) : f(p)*

p(“identifikasi”|”Pengolahan Citra”) = (1 + (2×1) + (0×0) + (1×0.5282) + (1×0.9471) : 31.834 = ***0.1405***

p(“identifikasi”|”Data Mining”) *= (1 + N(“identifikasi”, “dokumen1”) p(“Data Mining”|”dokumen1”) + N(“identifikasi”, “dokumen2”) p(“Data Mining”|”dokumen2”) + N(“identifikasi”, “dokumen3”) p(“Data Mining”|”dokumen3”) + N(“identifikasi”, “dokumen4”) p(“Data Mining”|”dokumen4”)) : f(p)*

p(“identifikasi”|”Data Mining”) = (1 + (2×0) + (0×1) + (1×0.4718) + (1×0.05287) : 31.834 = ***0.0478***

p(“indeks”|”Pengolahan Citra”) *= (1 + N(“indeks”, “dokumen1”) p(“Pengolahan Citra”|”dokumen1”) + N(“indeks”, “dokumen2”) p(“Pengolahan Citra”|”dokumen2”) + N(“indeks”, “dokumen3”) p(“Pengolahan Citra”|”dokumen3”) + N(“indeks”, “dokumen4”) p(“Pengolahan Citra”|”dokumen4”)) : f(p)*

p(“indeks”|”Pengolahan Citra”) = (1 + (0×1) + (4×0) + (0×0.5282) + (0×0.9471) : 31.834 = ***0.0314***

p(“indeks”|”Data Mining”) *= (1 + N(“indeks”, “dokumen1”) p(“Data Mining”|”dokumen1”) + N(“indeks”, “dokumen2”) p(“Data Mining”|”dokumen2”) + N(“indeks”, “dokumen3”) p(“Data Mining”|”dokumen3”) + N(“indeks”, “dokumen4”) p(“Data Mining”|”dokumen4”)) : f(p)*

p(“indeks”|”Data Mining”) = (1 + (0×0) + (4×1) + (0×0.4718) + (0×0.05287) : 31.834 = ***0.157***

p(“olah”|”Pengolahan Citra”) *= (1 + N(“olah”, “dokumen1”) p(“Pengolahan Citra”|”dokumen1”) + N(“olah”, “dokumen2”) p(“Pengolahan Citra”|”dokumen2”) + N(“olah”, “dokumen3”) p(“Pengolahan Citra”|”dokumen3”) + N(“olah”, “dokumen4”) p(“Pengolahan Citra”|”dokumen4”)) : f(p)*

p(“olah”|”Pengolahan Citra”) = (1 + (2×1) + (0×0) + (0×0.5282) + (0×0.9471) : 31.834 = ***0.0942***

p(“olah”|”Data Mining”) *= (1 + N(“olah”, “dokumen1”) p(“Data Mining”|”dokumen1”) + N(“olah”, “dokumen2”) p(“Data Mining”|”dokumen2”) + N(“olah”, “dokumen3”) p(“Data Mining”|”dokumen3”) + N(“olah”, “dokumen4”) p(“Data Mining”|”dokumen4”)) : f(p)*

p(“olah”|”Data Mining”) = (1 + (2×0) + (0×1) + (0×0.4718) + (0×0.05287) : 31.834 = ***0.0314***

p(“cluster”|”Pengolahan Citra”) *= (1 + N(“cluster”, “dokumen1”) p(“Pengolahan Citra”|”dokumen1”) + N(“cluster”, “dokumen2”) p(“Pengolahan Citra”|”dokumen2”) + N(“cluster”, “dokumen3”) p(“Pengolahan Citra”|”dokumen3”) + N(“cluster”, “dokumen4”) p(“Pengolahan Citra”|”dokumen4”)) : f(p)*

p(“cluster”|”Pengolahan Citra”) = (1 + (0×1) + (2×0) + (0×0.5282) + (0×0.9471) : 31.834 = ***0.0314***

p(“cluster”|”Data Mining”) *= (1 + N(“cluster”, “dokumen1”) p(“Data Mining”|”dokumen1”) + N(“cluster”, “dokumen2”) p(“Data Mining”|”dokumen2”) + N(“cluster”, “dokumen3”) p(“Data Mining”|”dokumen3”) + N(“cluster”, “dokumen4”) p(“Data Mining”|”dokumen4”)) : f(p)*

p(“cluster”|”Data Mining”) = (1 + (0×0) + (2×1) + (0×0.4718) + (0×0.05287) : 31.834 = ***0.0942***

p(“tekstur”|”Pengolahan Citra”) *= (1 + N(“tekstur”, “dokumen1”) p(“Pengolahan Citra”|”dokumen1”) + N(“tekstur”, “dokumen2”) p(“Pengolahan Citra”|”dokumen2”) + N(“tekstur”, “dokumen3”) p(“Pengolahan Citra”|”dokumen3”) + N(“tekstur”, “dokumen4”) p(“Pengolahan Citra”|”dokumen4”)) : f(p)*

p(“tekstur”|”Pengolahan Citra”) = (1 + (0×1) + (0×0) + (2×0.5282) + (3×0.9471) : 31.834 = ***0.1538***

p(“tekstur”|”Data Mining”) *= (1 + N(“tekstur”, “dokumen1”) p(“Data Mining”|”dokumen1”) + N(“tekstur”, “dokumen2”) p(“Data Mining”|”dokumen2”) + N(“tekstur”, “dokumen3”) p(“Data Mining”|”dokumen3”) + N(“tekstur”, “dokumen4”) p(“Data Mining”|”dokumen4”)) : f(p)*

p(“tekstur”|”Data Mining”) = (1 + (0×0) + (0×1) + (2×0.4718) + (3×0.05287) : 31.834 = ***0.066***

p(“Pengolahan Citra”) *= (1 + p(“Pengolahan Citra”|”dokumen1”) + p(“Pengolahan Citra”|”dokumen2”) + p(“Pengolahan Citra”|”dokumen3”) + p(“Pengolahan Citra”|”dokumen4”)) : (jumlah kategori + jumlah dokumen)*

p(“Pengolahan Citra”) = 1 + 0 + 0.5282 + 0.9471 : (2+4) = ***0.4125***

p(“Data Mining”) *= (1 + p(“Data Mining”|”dokumen1”) + p(“Data Mining”|”dokumen2”) + p(“Data Mining”|”dokumen3”) + p(“Data Mining”|”dokumen4”)) : (jumlah kategori + jumlah dokumen)*

p(“Data Mining”) = 1 + 1 + 0.4718 + 0.0528 : (2+4) = ***0.2541***

Tabel 3.5 Bobot Probabilitas Akhir PLSA

Kategori	*p(ci)*	*p(wkj.ci)*						
		warna	**indeks**	**olah**	**identifikasi**	**klasifikasi**	**cluster**	**tekstur**
Pengolahan Citra	0,41255	0,18516	0,03141	0,09424	0,14058	0,06460	0,03141	0,15385
Data Mining	0,25411	0,03473	0,15706	0,03141	0,04789	0,15529	0,09424	0,06604

p("Pengolahan Citra"|"dokumen1") *= p("Pengolahan Citra") x p("warna"|"Pengolahan Citra") x p("olah"|"Pengolahan Citra") x p("identifikasi"|"Pengolahan Citra") = 0.41255 × 0.18516 × 0.09424 × 0.14058 =* ***0.00101***

p("Data Mining"|"dokumen1") *= p("Data Mining") x p("warna"|" Data Mining") x p("olah"|" Data Mining") x p("identifikasi"|" Data Mining") = 0.25411 × 0.03473 × 0.03141 × 0.04789 =* ***0.00001***

p("Pengolahan Citra"|"dokumen1") > p("Data Mining"|"dokumen1")

0.00101 > 0.00001 (Dokumen 1 merupakan kategori Pengolahan Citra)

p("Pengolahan Citra"|"dokumen2") *= p("Pengolahan Citra") x p("indeks"|"Pengolahan Citra") x p("klasifikasi"|"Pengolahan Citra") x p("cluster"|"Pengolahan Citra") = 0.41255 × 0.03141 × 0.06460 × 0.03141 =* ***0.00003***

p("Data Mining"|"dokumen2") *= p("Data Mining") x p("indeks"|"Data Mining") x p("klasifikasi"|"Data Mining") x p("cluster"|"Data Mining") = 0.25411 × 0.15706 × 0.15529 × 0.09424 =* ***0.00058***

p("Data Mining"|"dokumen2") > p("Pengolahan Citra"|"dokumen2")

0.00058 > 0.00003 (Dokumen 2 merupakan kategori Data Mining)

p("Pengolahan Citra"|"dokumen3") *= p("Pengolahan Citra") x p("klasifikasi"|"Pengolahan Citra") x p("identifikasi"|"Pengolahan*

Citra") x p("tekstur"|"Pengolahan Citra") = 0.41255 × 0.06460× 0.14058 × 0.15385 = ***0.00058***

p("Data Mining"|"dokumen3") *= p("Data Mining") x p("klasifikasi"|" Data Mining") x p("identifikasi"|" Data Mining") x p("tekstur"|" Data Mining") = 0.25411 × 0.15529 × 0.04789 × 0.06604 =* ***0.00012***

p("Pengolahan Citra"|"dokumen1") > p("Data Mining"|"dokumen1")

0.00058 > 0.00012 (Dokumen 3 merupakan kategori Pengolahan Citra)

p("Pengolahan Citra"|"dokumen4") *= p("Pengolahan Citra") x p("warna"|"Pengolahan Citra") x p("identifikasi"|"Pengolahan Citra") x p("tekstur"|"Pengolahan Citra") = 0.41255 × 0.18516 × 0.14058 × 0.15385 =* ***0.00165***

p("Data Mining"|"dokumen4") *= p("Data Mining") x p("warna"|" Data Mining") x p("identifikasi"|" Data Mining") x p("tekstur"|" Data Mining") = 0.25411 × 0.03473 × 0.04789 × 0.06604 =* ***0.00003***

p("Pengolahan Citra"|"dokumen4") > p("Data Mining"|"dokumen4")

0.00165 > 0.00003 (Dokumen 4 merupakan kategori Pengolahan Citra)

Dari perhitungan *Probabilistic Latent Semantic Analysis* diatas didapat bobot probabilitas dari tiap-tiap dokumen terhadap kategori/topik.

Tabel 3.6 Bobot Probabilitas Dokumen Terhadap Topik

Dokumen	Pengolahan Citra	Data Mining
d1	0,00101	0,00001
d2	0,00003	0,00058
d3	0,00058	0,00012
d4	0,00165	0,00003

3.2.1.3 Perangkingan

Perangkingan dilakukan untuk menentukan dokumen mana yang memiliki kemiripan paling tinggi dengan *query* yang dimasukkan oleh pengguna. Pada penelitian ini digunakan perhitungan *cosine similarity* untuk menghitung kemiripan dokumen dengan *query*. Sebelum menghitung kemiripan dengan *cosine similarity*, terlebih dahulu dihitung vektor dokumen dan vektor *query*. Vektor dokumen didapat dari matriks bobot probabilitas dokumen terhapap topik yang dikalikan dengan matriks diagonal bobot probabilitas topik. Perhitungan vektor dokumen dapat dilihat seperti dibawah ini :

$$dokumen \overbrace{\begin{bmatrix} 0{,}00101 & 0{,}00101 \\ 0{,}00003 & 0{,}00058 \\ 0{,}00058 & 0{,}00012 \\ 0{,}00165 & 0{,}00003 \end{bmatrix}}^{topik} \times \overbrace{\begin{bmatrix} 0{,}41255 & 0{,}00000 \\ 0{,}00000 & 0{,}25411 \end{bmatrix}}^{p(ci)}$$

$$= \overbrace{\begin{bmatrix} 0{,}00042 & 3{,}37 \times 10^{-6} \\ 0{,}00001 & 0{,}00015 \\ 0{,}00024 & 0{,}00003 \\ 0{,}00068 & 0{,}00001 \end{bmatrix}}$$

Gambar 3.7 Matriks Bobot Dokumen Terhadap Topik

Tabel 3.7 Vektor Dokumen

Dokumen	Penolahahan Citra	Data Mining
d1	0,00042	$3{,}37 \times 10^{-6}$
d2	0,00001	0,00015
d3	0,00024	0,00003
d4	0,00068	0,00001

Setelah menghitung vektor dokumen, langkah selanjutnya adalah menghitung vektor *query*. Vektor *query* didapat dari rata-rata bobot probabilitas terms terhadap topik. Berikut adalah contoh perhitungan vektor *query* :

Q = Klasifikasi Warna

- $\frac{p("klasifikasi"|"Pengolahan\ Citra")+ p("warna"|"Pengolahan\ Citra")}{2}$
- $\frac{p("klasifikasi"|"Data\ Mining")+ p("warna"|"Data\ Mining")}{2}$
- $\frac{0.06460+0.18516}{2} = \mathbf{0.124876644}$
- $\frac{0.15529+0.03473}{2} = \mathbf{0.095014039}$

$$\boldsymbol{similarity} = \mathbf{cos}(\boldsymbol{\theta}) = \frac{\boldsymbol{A.B}}{\left|\left|\boldsymbol{A}\right|\right|\left|\left|\boldsymbol{B}\right|\right|}$$

- $sim(d1) = \frac{(0.12488 \times 0{,}00042)+(0.09501\times 3{,}37\times10^{-6})}{\sqrt{0.12488^2+ 0.09501^2} \times \sqrt{0{,}00042^2+ 3{,}37\times10^{-6^2}}} = \frac{0.00005}{0.00007} = \mathbf{0.8007}$
- $sim(d2) = \frac{(0.12488 \times 0{,}00001)+(0.09501\times0{,}00015)}{\sqrt{0.12488^2+ 0.09501^2} \times \sqrt{0{,}00001^2+ 0{,}00015}} = \frac{0.000015}{0.000023} = \mathbf{0.66192}$
- $sim(d3) = \frac{(0.12488 \times 0{,}00024)+(0.09501\times0{,}00003)}{\sqrt{0.12488^2+ 0.09501^2} \times \sqrt{0{,}00024^2+ 0{,}00003}} = \frac{0.0000327}{0.0000376} = \mathbf{0.86890}$
- $sim(d4) = \frac{(0.12488 \times 0{,}00068)+(0.09501\times0{,}00001)}{\sqrt{0.12488^2+ 0.09501^2} \times \sqrt{0{,}00068^2+ 0{,}00001}} = \frac{0.0000857}{0.0001069} = \mathbf{0.80209}$

Berdasarkan urutan rangking dokumen terhadap *query* adalah : d3 > d4 > d1 > d2. Dokumen 3 yang merupakan kategori Pengolahan Citra memiliki bobot paling tinggi diantara keempat dokumen dengan bobot *similarity* 0.86890. Dari perhitungan *cosine similarity* didapatkan hasil bobot kemiripan dokumen terhadap *query "klasifikasi warna"* sebagai berikut :

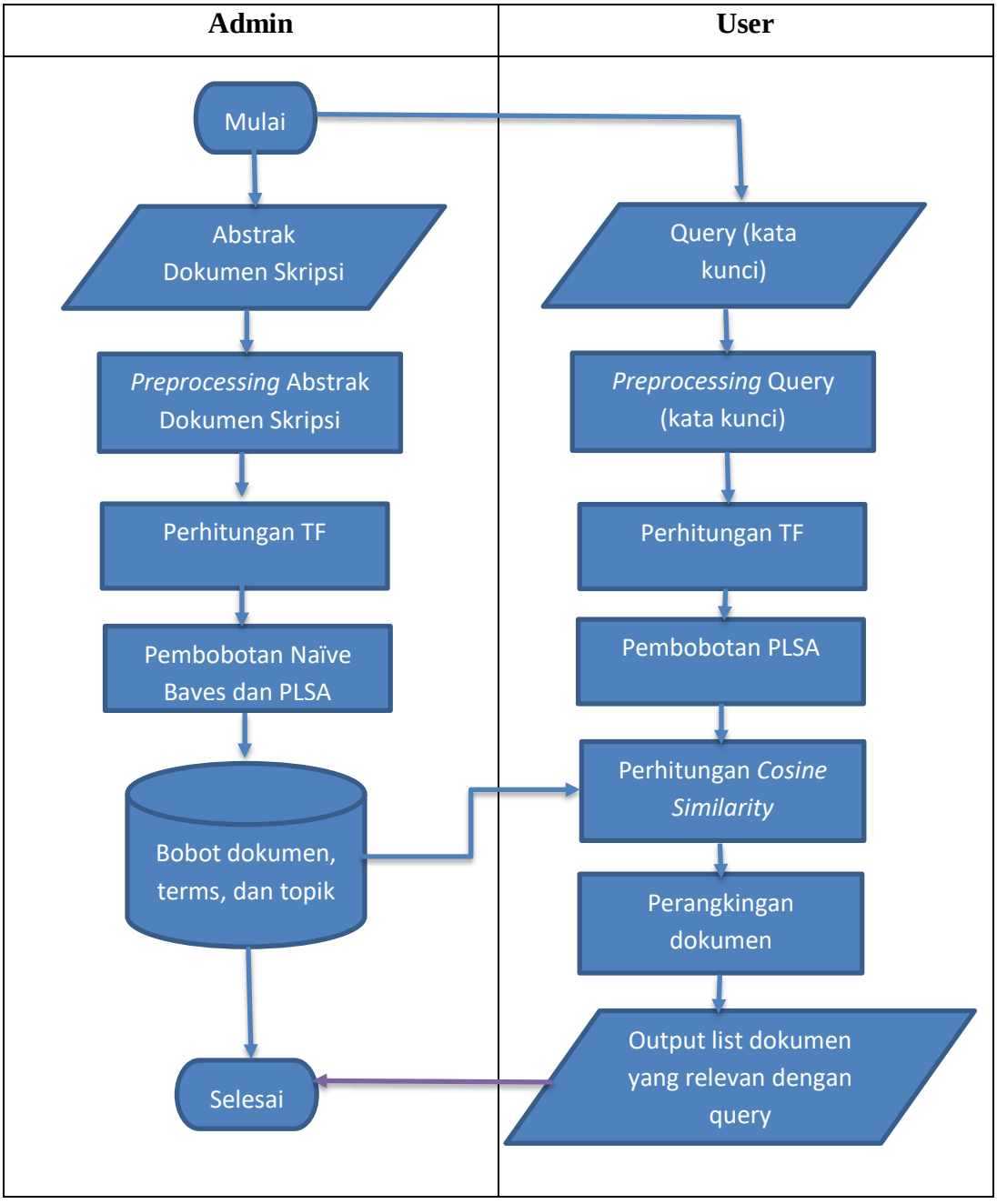
Tabel 3.8 Hasil Perangkingan

Dokumen	**Terms**	**Bobot Kemiripan**	**Kategori**
d1	identifikasi(2), warna(3), olah(2)	**0.80070**	Pengolahan Citra
d2	klasifikasi(3), cluster(2), indeks(4)	**0.66192**	Data Mining
d3	klasifikasi(2), identifikasi(1), tekstur(2)	**0.86890**	Pengolahan Citra
d4	identifikasi (1), warna (2), tekstur (3)	**0.80209**	Pengolahan Citra

Hasil perangkingan ini berdasarkan bobot similaritas yang dihitung menggunakan *cosine similarity*. Sebagai hipotesis awal jumlah kata dalam query yang dapat dimasukkan dalam sistem sama halnya dengan *search engine* pada umumnya, semakin banyak kata yang relevan dengan sebuah dokumen maka akan semakin tinggi pula bobot similaritasnya, dan berlaku pula sebaliknya, semakin banyak kata yang tidak relevan dengan suatu dokumen maka semakin rendah pula bobot kemiripannya. Metode *cosine similarity* ini menghitung jarak kedekatan antara *query* dan dokumen berdasarkan sudut vektor dari masing-masing *query* dan dokumen tersebut. Vektor tersebut terbentuk dari bobot-bobot probabilitas antara *term*, dokumen, dan topik tanpa memperhitungkan urutan kata dalam dokumen maupun *query*, sehingga *query* dengan urutan kata yang berbeda namun memiliki frekuensi term yang sama tidak akan mempengaruhi hasil perangkingan *cosine similarity*.

3.3 Perancangan Sistem

3.3.1 *Flowchart* Sistem



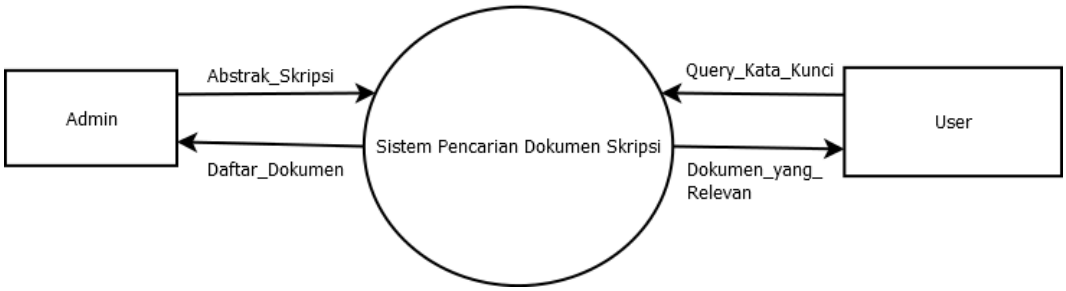
Gambar 3.8 Flowchart Sistem Pencarian Dokumen Skripsi

Berikut penjelasan *flowchart* sistem pencarian dokumen skripsi Gambar 3.7 :

1. Sistem menerima input berupa teks abstrak dokumen skripsi yang diinputkan oleh admin dan *query* (kata kunci) yang dimasukkan pengguna.
2. Inputan teks abstrak dan *query* kemudian dilakukan *preprocessing* meliputi *case folding*, *tokenizing*, *filtering* dan *stemming*, sehingga didapatkan *terms* untuk dilakukan pengindeksan.
3. Setelah dilakukan *preprocessing* pada abstrak dan *query*, langkah selanjutnya adalah pembobotan TF-IDF, Naïve Bayes, dan PLSA pada *terms* hasil *preprocessing* sebelumnya.
4. Sistem melakukan perhitungan kemiripan (similarity) antara bobot probabilitas dokumen dan bobot probabilitas *query*.
5. Sistem melakukan perangkingan dokumen terhadap *query* yang didapat dari bobot kemiripan *cosine similarity*.
6. Sistem menghasilkan *output* berupa daftar dokumen yang relevan dengan *query*.

3.3.2 Diagram Konteks

Berikut adalah diagram konteks sistem temu kembali dokumen skripsi Teknik Informatika Universitas Muhammadiyah Gresik menggunakan pemodelan topik.

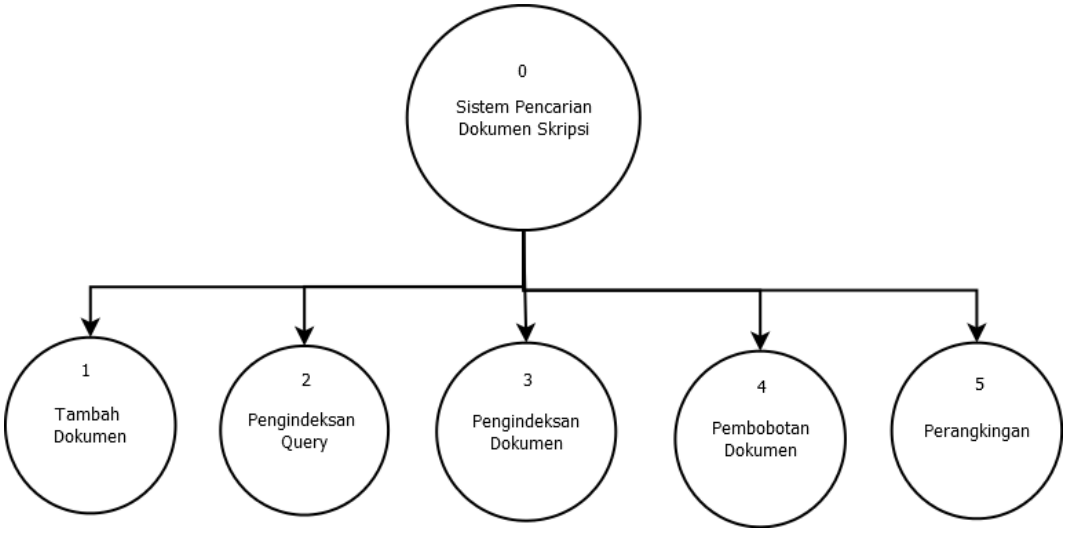


Gambar 3.9 Diagram Konteks Sistem Pencarian Dokumen Skripsi

Context Diagram yang ditunjukkan pada Gambar 3.8 diatas menggambarkan *input* dan *output* antar sistem dengan entitas luar. Sistem menerima *input* dari Admin berupa abstrak dari dokumen skripsi yang akan dijadikan master data dokumen, serta menerima *input* dari pengguna berupa kata kunci atau *query* dan sistem akan mengembalikan daftar dokumen yang relevan dengan *query*.

3.3.3 Diagram Berjenjang

Pembuatan sistem pencarian dokumen skripsi ini diperlukan bagan berjenjang, dimana merupakan awal dari penggambaran *Data Flow Diagram* (DFD) ke level-level lebih bawah lagi. Sistem pencarian dokumen skripsi ini mempunyai 2 (dua) level seperti yang terlihat pada gambar 3.9.



Gambar 3.10 Diagram Berjenjang Sistem Pencarian Dokumen Skripsi

Keterangan :

1. Top Level : Aplikasi Sistem Temu Kembali Dokumen Skripsi Teknik Informatika Universitas Muhammadiyah Gresik menggunakan Pemodelan Topik PLSA.
2. Level 0 : Merupakan hasil *break down* dari proses aplikasi

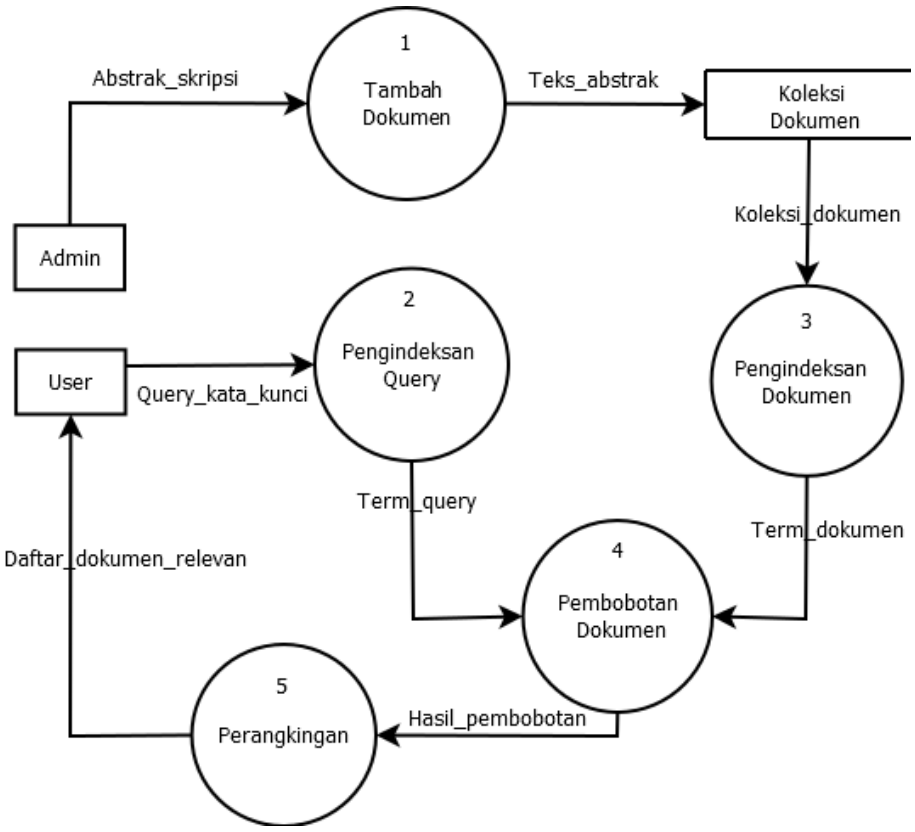
Sistem temu kembali dokumen skripsi dengan Pemodelan topik menjadi beberapa sub sistem seperti berikut :

a. Tambah Data
b. Pengindeksan Query
c. Pengindeksan Dokumen
d. Pembobotan Dokumen
e. Perangkingan

3.3.4 *Data Flow Diagram*

3.3.4.1 DFD Level 0

Pada Gambar 3.10 dapat dilihat DFD level 0 dari SIstem Pencarian Dokumen Skripsi Teknik Informatika Universitas Muhammadiyah Gresik sebagai berikut :



Gambar 3.11 DFD Level 0 Sistem Pencarian Dokumen Skripsi

3.4 Perancangan Basis Data

Database (Basis Data) adalah kumpulan dari data yang berhubungan antara satu dengan yang lainya, tersimpan di perangkat keras komputer dan menggunakan perangkat lunak untuk memanipulasinya. *Database* merupakan salah satu Komponen yang penting dalam sistem komputerisasi, karena *database* merupakan data dalam menyediakan informasi bagi para pengguna.

3.4.1 Desain Tabel

Desain tabel pada sistem temu kembali dokumen skripsi ini adalah sebagai berikut :

1. **Tabel *User***

   Tabel 3.9 dibawah ini digunakan untuk memberikan hak akses dari pengguna sistem.

Tabel 3.9 Tabel *user*

Field	Type	Key	Extra
Id_user	Int(11)	Primary_key	autoincrement
Nama	Varchar(100)		
Username	Varchar(50)		
Password	Varchar(50)		
Level	Varchar(50)		

2. **Tabel Dokumen**

   Tabel 3.10 dibawah ini digunakan untuk menyimpan semua dokumen yang dimasukkan oleh admin sebagai *corpus* yang nantinya akan di-*retrieve* oleh sistem.

Tabel 3.10 Tabel Dokumen

Field	Type	Key	Extra
Id	Int(11)	Primary_key	autoincrement
Nim	Varchar(10)		
Author	Varchar(50)		

Nama_dokumen	Varchar(100)		
Judul	Varchar(1000)		
Isi_abstrak	Text		
Tahun	Varchar(5)		

3. Tabel *Stopword*

Tabel 3.11 dibawah ini digunakan untuk menyimpan *stoplist* atau kata-kata yang sering muncul dalam teks bahasa Indonesia. Tabel ini yang akan menjadi acuan dalam proses penghapusan *stopword* atau *stopword removal*.

Tabel 3.11 Tabel *Stopword*

Field	Type	Key	Extra
Id_stopword	Int(11)	Primary_key	autoincrement
Kata_stopword	Varchar(50)		

4. Tabel Kata Dasar

Tabel 3.12 dibawah ini digunakan untuk menyimpan daftar kata dasar dalam bahasa Indonesia. Tabel ini digunakan sebagai acuan dalam menentukan kata dasar bahasa Indonesia pada proses *stemming*.

Tabel 3.12 Tabel Kata Dasar

Field	Type	Key	Extra
Id_katadasar	Int(11)	Primary_key	autoincrement
Katadasar	Varchar(70)		
Tipe_katadasar	Varchar(25)		

5. Tabel *Terms*

Tabel 3.13 dibawah ini digunakan unruk menyimpan kumpulan kata (*terms*) pada tiap-tiap dokumen. *Terms* inilah yang akan digunakan pada proses pembobotan.

Tabel 3.13 Tabel *Terms*

Field	Type	Key	Extra
Id_dokumen	Int(11)	Foreign_key	
Filtered_token	Mediumtext		
Term	Mediumtext		

6. Tabel Riwayat Hasil

Tabel 3.14 dibawah ini digunakan untuk menyimpan riwayat hasil pencarian yang telah dilakukan oleh pengguna.

Tabel 3.14 Tabel Riwayat Hasil

Field	Type	Key	Extra
Id_result	Int(11)	Primary_key	autoincrement
Keyword	Varchar(500)		
Tanggal	Datetime		

7. Tabel Riwayat Hasil Dokumen

Tabel 3.15 dibawah ini digunakan untuk menyimpan dokumen-dokumen yang berhasil di-*retrieve* pada pencarian-pencarian sebelumnya.

Tabel 3.15 Tabel Riwayat Hasil Dokumen

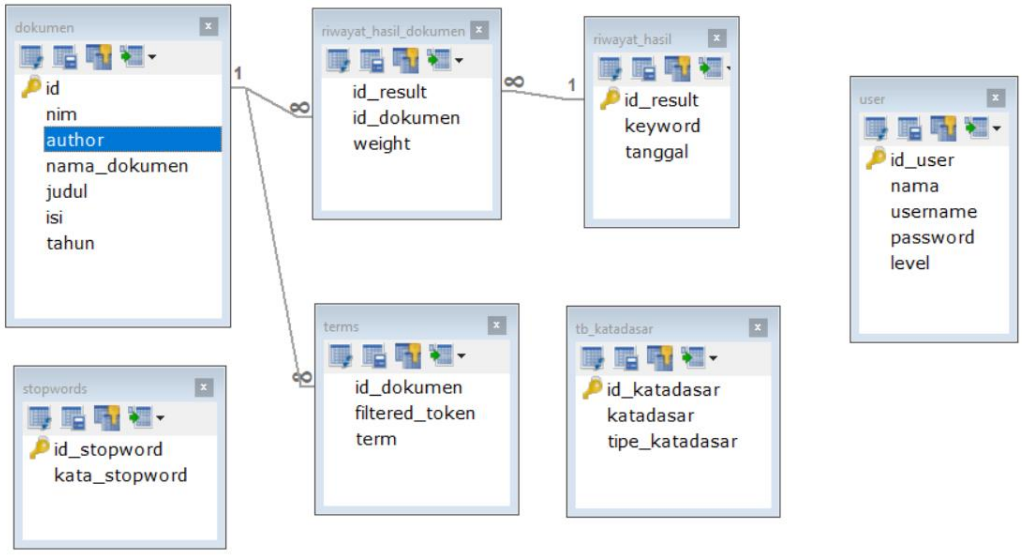
Field	Type	Key	Extra
Id_result	Int(11)	Foreign_key	
Id_dokumen	Int(11)	Foreign_key	
Weight	Double		

3.4.2 Entity Relationship Diagram

Entity Relationship Diagram **(**ERD) adalah model konseptual yang mendeskripsikan hubungan antar penyimpanan (dalam DFD). Karena itu, ERD berbeda dengan DFD (DFD memodelkan fungsi sistem), atau dengan STD (*State Transition Diagram*, yang memodelkan sistem dari segi ketergantungan terhadap

waktu). ERD digunakan untuk memodelkan struktur data dan hubungan antar data, karena hal ini relatif kompleks.

Berikut adalah gambaran dari ERD pada sistem temu kembali dokumen skripsi di jelaskan pada Gambar 3.12 :



Gambar 3.12 ERD Sistem Pencarian Dokumen Skripsi

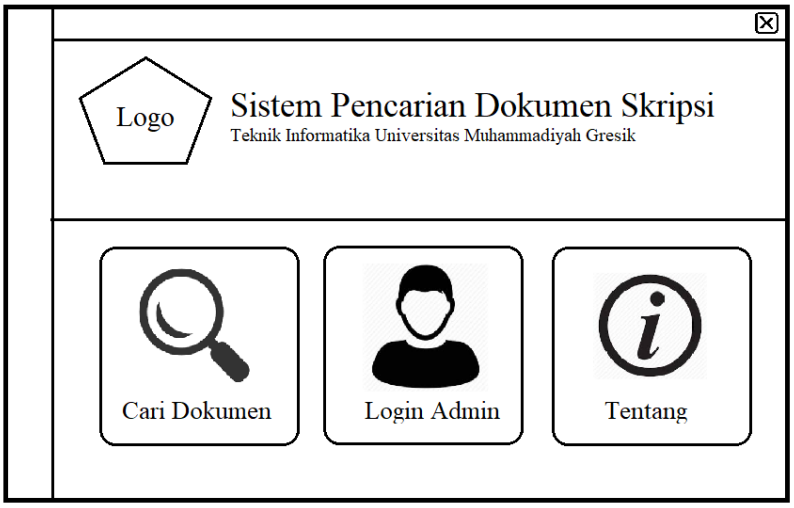
3.5 Perancangan Antarmuka

Antarmuka pemakai (*User Interface*) merupakan mekanisme komunikasi antara pengguna dengan sistem. Antarmuka pemakai dapat menerima informasi dari pengguna dan memberikan informasi kepada pengguna untuk membantu mengarahkan alur penelusuran masalah sampai ditemukan suatu solusi. Dalam sistem temu kembali dokumen skripsi Teknik Informatika Universitas Muhammadiyah Gresik ialah sebagai media pengguna untuk mencari dokumen skripsi yang relevan dengan *query* yang dimasukkan, berikut adalah desain *interface* dari sistem temu kembali dokumen skripsi.

3.5.1 Halaman Awal

Gambar dibawah ini adalah halaman awal pada sistem pencarian dokumen skripsi teknik informatika Universitas Muhammadiyah Gresik. Halaman awal ini terdapat 3 (tiga) buah menu utama yaitu menu

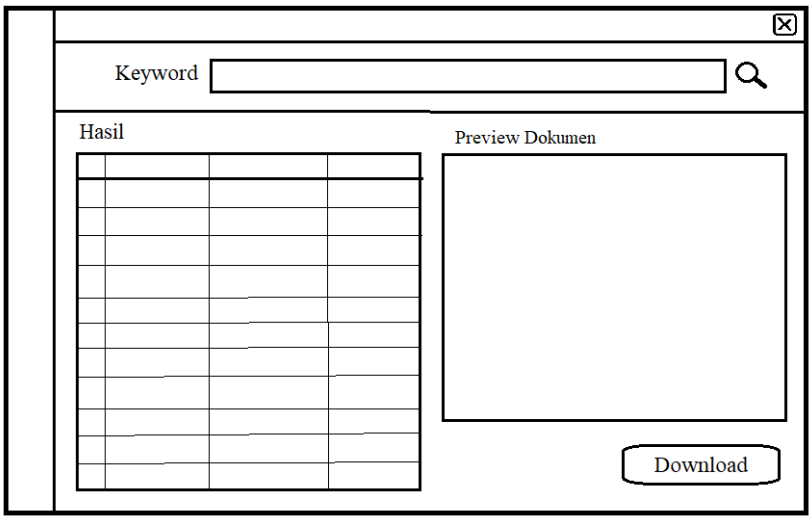
pencarian, menu admin, dan tentang. Berikut gambar 3.13 *interface* halaman awal :



Gambar 3.13 Halaman Awal Sistem

3.5.2 Antarmuka Menu Pencarian

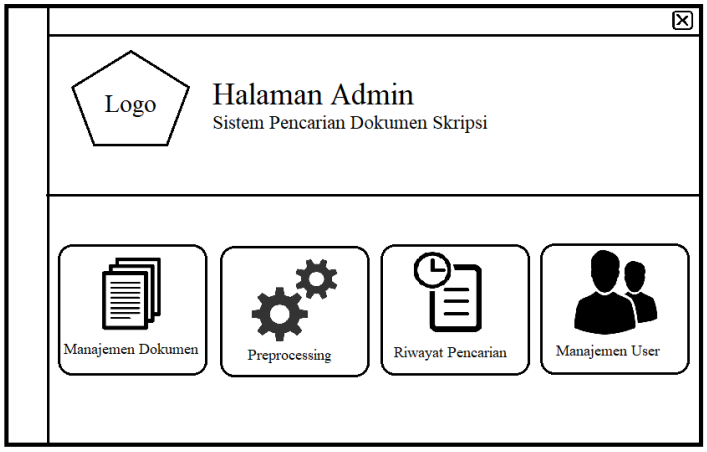
Gambar dibawah ini adalah halaman menu pencarian dokumen skripsi. Menu pencarian ini berfungsi untuk mencari dokumen skripsi yang telah tersimpan dalam database, dengan memasukkan *query/keyword* sebagai *input* sistem. Kemudian hasil dokumen akan terlihat pada tabel hasil dibawahnya. Berikut adalah tampilan menu pencarian :



Gambar 3.14 Halaman Menu Pencarian

3.5.3 Halaman Admin

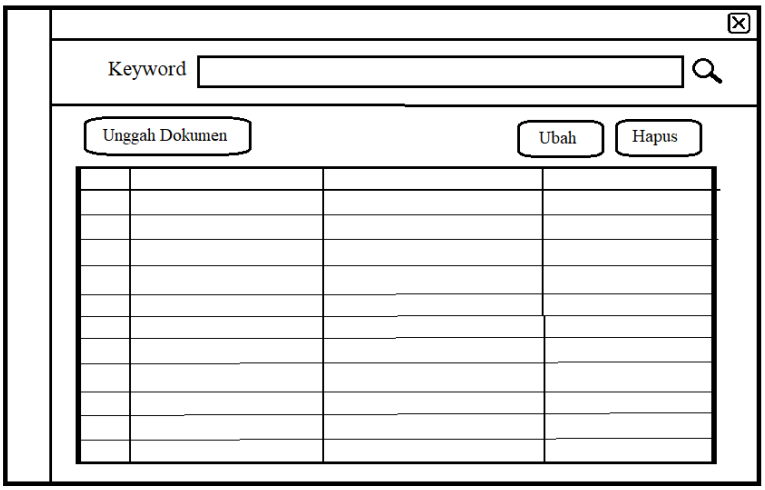
Gambar 3.15 dibawah ini adalah tampilan halaman admin. Pada halaman admin terdapat 4 (empat) buah menu, yaitu menu manajemen dokumen, menu *preprocessing*, menu riwayat pencarian, dan menu manajemen user. Berikut adalah tampilan halaman admin :



Gambar 3.15 Halaman Admin

3.5.4 Halaman Manajemen Dokumen

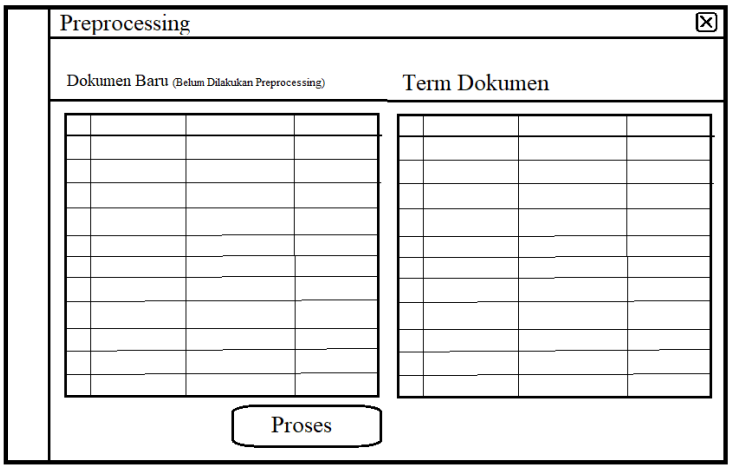
Gambar 3.16 dibawah ini adalah tampilan dari menu manajemen dokumen. Menu manajemen dokumen ini berfungsi untuk mengatur dokumen-dokumen skripsi yang terdapat dalam database. Pada menu ini admin dapat menambah, mengubah, dan menghapus dokumen. Berikut adalah tampilan dari halaman manajemen dokumen :



Gambar 3.16 Halaman Manajemen Dokumen

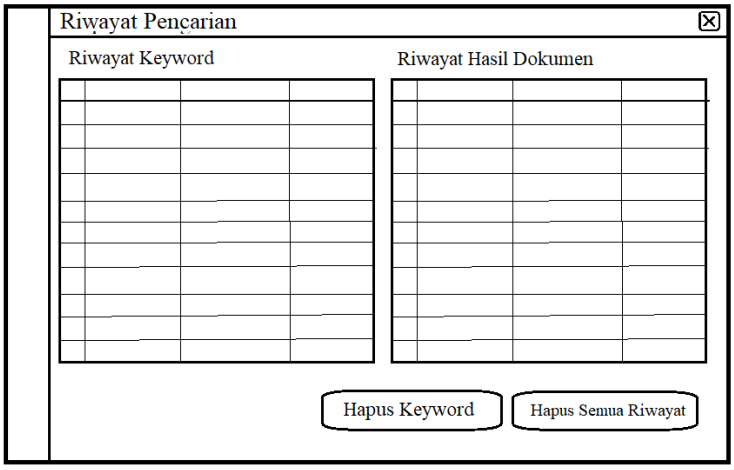
3.5.5 Halaman Menu *Preprocessing*

Gambar 3.17 dibawah ini adalah tampilan dari menu *preprocessing*. Menu *preprocessing* ini berfungsi untuk melakukan prapemrosesan pada dokumen yang baru ditambahkan pada menu sebelumnya. Hasil dari *preprocessing* ini akan digunakan pada proses pembobotan. Berikut adalah tampilan dari halaman menu *preprocessing* :



Gambar 3.17 Halaman Menu *Preprocesssing*

3.5.6 Halaman Riwayat Pencarian



Gambar 3.18 Halaman Riwayat Pencarian

Gambar 3.18 diatas adalah tampilan dari halaman riwayat pencarian. Pada halaman ini terdapat riwayat *query* yang telah diinputkan oleh pengguna pada pencarian-pencarian sebelumnya.

3.6 Skenario Pengujian

Pada penelitian ini, untuk mengukur evaluasi kinerja sistem temu kembali informasi digunakan pengujian *recall*, *precision*, *accuracy*, dan *F-Measure*. *Recall* adalah rasio antara dokumen relevan yang berhasil ditemukembalikan (*retrieved*) dari seluruh dokumen relevan yang terdapat dalam sistem. *Precision* adalah rasio dokumen relevan yang berhasil ditemukembalikan dari seluruh dokumen yang berhasil ditemukembalikan.

Tabel 3.16 Parameter menghitung precision dan recall

Keterangan	**Relevan**	**Tidak Relevan**
Terambil	True positive (tp)	False positive (fp)
Tidak terambil	False negative (fn)	True negative (tn)

Rumus untuk menghitung Precision :

$$Precision = \frac{tp}{tp+fp} \quad (3.1)$$

Rumus untuk menghitung Recall :

$$Recall = \frac{tp}{tp+ fn} \quad (3.2)$$

Rumus untuk menghitung Accuracy :

$$Accuracy = \frac{tp+tn}{tp+fp+tn+fn} \quad (3.3)$$

Rumus untuk menghitung F-Measure :

$$F - measure = \frac{2 \times precision \times recall}{precision+recall} \quad (3.4)$$

Nilai *precision*, *recall*, dan *accuracy* dinyatakan dalam persen. Semakin tinggi ketiga nilai tersebut menunjukkan semakin baiknya kinerja aplikasi. Evaluasi yang akan dilakukan dalam penelitian ini adalah menghitung nilai dari *precision*, *recall*, *accuracy* dan *f-measure* berdasarkan dokumen yang berhasil ditemukan kembali oleh sistem aplikasi yang dibuat. Sedangkan untuk menentukan nilai dari *precision*, *recall*, *accuracy*, dan *f-measure* harus didapatkan jumlah dokumen yang relevan terhadap suatu topik dokumen abstrak.

Menurut Rijsbergen (1979) relevansi merupakan sesuatu yang bersifat subyektif. Setiap orang mempunyai perbedaan dalam mengartikan sesuatu dokumen yang relevan terhadap sebuah topik informasi. Sehingga dalam pelaksanaan pengujian sistem ini dibutuhkan seorang pakar yang dianggap mampu menilai apakah sebuah dokumen dikatakan relevan dengan *query* atau tidak relevan.

3.7 Spesifikasi Pembuatan Sistem

Kebutuhan perangkat lunak serta perangkat keras dari sistem sebagai berikut :

a. Kebutuhan Perangkat Lunak
   1. *Windows 10* sebagai sistem operasi yang digunakan.
   2. *Java versi 8* dan *NetBeans IDE 8.0.2* sebagai bahasa pemrograman berbasis desktop dan sekaligus *compiler*nya.
   3. *SQLyog Enterprise 8.18.0.0* sebagai database server.
   4. *XAMPP Control Panel 3.2.1*

b. Kebutuhan Perangkat Keras
   1. Komputer Intel pentium 2,13 GHz sekelas atau lebih tinggi
   2. RAM 2 GB atau lebih
   3. Hardisk dengan kapasitas 500 gigabyte atau lebih
   4. Monitor, mouse, keyboard standard