BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisa sistem dilakukan untuk mempelajari dan menganalisa sistem sebelumnya yang diteliti oleh Azhar Firdaus, Ernawati dan Arie Vatresia (2014) berdasarkan Jurnal Teknologi Informasi. Aplikasi deteksi kemiripan dokumen yang diteliti sebelumnya menggunakan metode *cosine similarity* untuk perhitungan nilai kemiripan dokumen. Sedangkan untuk tahapan *text preprocessing* dokumen menggunakan *text mining* dengan beberapa tahapan seperti *case folding*, *stopword*, *tagging*, *stemming*, *filtering* dan *analyzing.* Algoritma nazief adriani digunakan dalam proses *stemming* untuk mengembalikan kata berimbuhan menjadi kata dasar. Dalam melakukan perhitungan nilai kemiripan dokumen kemudian dilakukan suatu pembobotan kata pada dua dokumen dalam hal ini peneliti sebelumnya menggunakan pembobotan *binary term* untuk setiap kata pada dokumen. Pembobotan kata merupakan tahapan awa dalam menentukan nilai kemiripan dari sebuah dokumen dengan melakukan *skoring* dan sangat berpengaruh terhadap nilai *similarity* yang didapatkan. Peneliti sebelumnya menggunakan *binary term* atau disebut tipe *booelan* yang bernilai 0 dan 1. Setiap kata direpresentasikan dengan nilai 0 jika terdapat kata yang tidak ada pada dokumen tersebut dan bernilai 1 jika terdapat kata yang sama pada dokumen tersebut. Dari *skoring* kata pada masing dokumen akan di akumulasikan untuk dilakukan perhitungan *similarity* dengan metode *cosine similarity*

3.1.1 Hasil Analisis Sistem

Berdasarkan Analisa Sistem yang telah dilakukan penelitian selanjutnya ingin menggunakan pembobotan yang berbeda dari peneliti sebelumnya, jika penelitian yang sebelumnya menggunakan pembobotan *binary term* pada penelitian selanjutnya akan digunakan pembobotan kata dengan TF-IDF (*Term Frequency and Invers Documenct Frequency).* Pembobotan TF-IDF digunakan pada penelitian selanjutnya dikarenakan terdapat peningkatan ketelitian yang

mempengaruhi nilai perhitungan kemiripan dokumen seperti yang terlihat pada bab 2.6 analisa optimasi dan bab 2.7 penelitian sebelumnya terdapat selisih nilai 0,1761. Hasil selisih nilai tersebut menjadi acuan untuk menggunakan pembobotan TF-IDF pada penelitian selanjutnya.

Aplikasi deteksi kemiripan dokumen adalah suatu aplikasi yang nantinya akan mencari nilai *similarity* dari beberapa dokumen atau lebih dokumen dan juga mencari letak kemiripan dokumen berdasarkan sebuah kata. Proses dalam menentukan kemiripan dokumen dilakukan dengan menggunakan sebuah dokumen yang berisi teks dengan beberapa ketentuan meliputi :

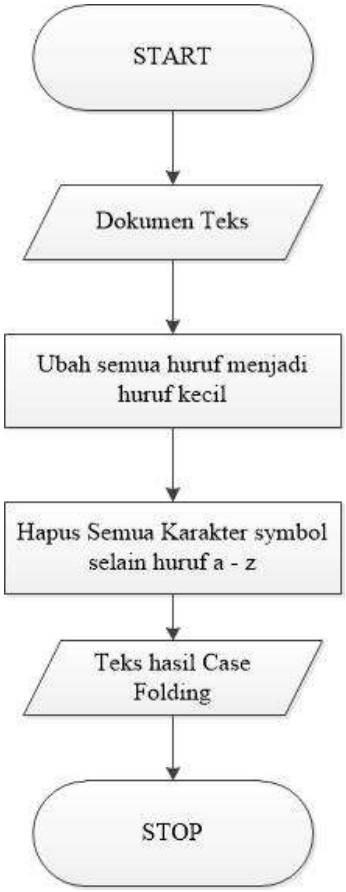
1. Teks berbahasa Indonesia.
2. Teks memiliki aturan kata sesuai EYD.
3. Dokumen yang akan diolah berupa ekstensi type .pdf , .doc.

Dokumen yang akan dibandingkan akan dibagi menjadi dua type dokumen yaitu : dokumen sumber dan dokumen pembanding. Dokumen sumber merupakan sebuah dokumen atau kumpulan dokumen yang menjadi sumber dokumen sedangkan dokumen pembanding adalah dokumen yang akan dibandingkan dengan dokumen sumber untuk mencari *similarity* dari dokumen pembanding. Dokumen sumber dan dokumen pembanding nantinya akan melalui sebuah proses pemilahan kata atau biasa disebut dengan (*Text Preprocessing)* dengan menggunakan tahapan *text mining* yang lebih jelasnya akan dibahas dalam Ektrasi dokumen.

Ekstrasi dokumen adalah salah satu bentuk atau teknik untuk mengekstrak sebuah data untuk mengetahui isi dari suatu dokumen. Dalam hal ini dokumen yang akan di ekstrak adalah sebuah teks, dimana teks pada umumnya memiliki tingkat penulisan yang berbeda satu dengan yang lain. Diperulkan suatu ilmu untuk mengektrak dokumen yang berisi sebuah kata dengan suatu proses yaitu *text mining* dengan beberapa tahapan sebagai berikut :

1. Case Folding dan Tokenizing.

Pengertian *Case Folding* dan *Tokenizing* sudah dijelaksan pada bab sebelumnya dapat dilihat pada bab 2 landasan teori. Sedangkan pada bab ini akan dijelaskan alur flow dari *case folding* dan tokenizing pada gambar 3.1



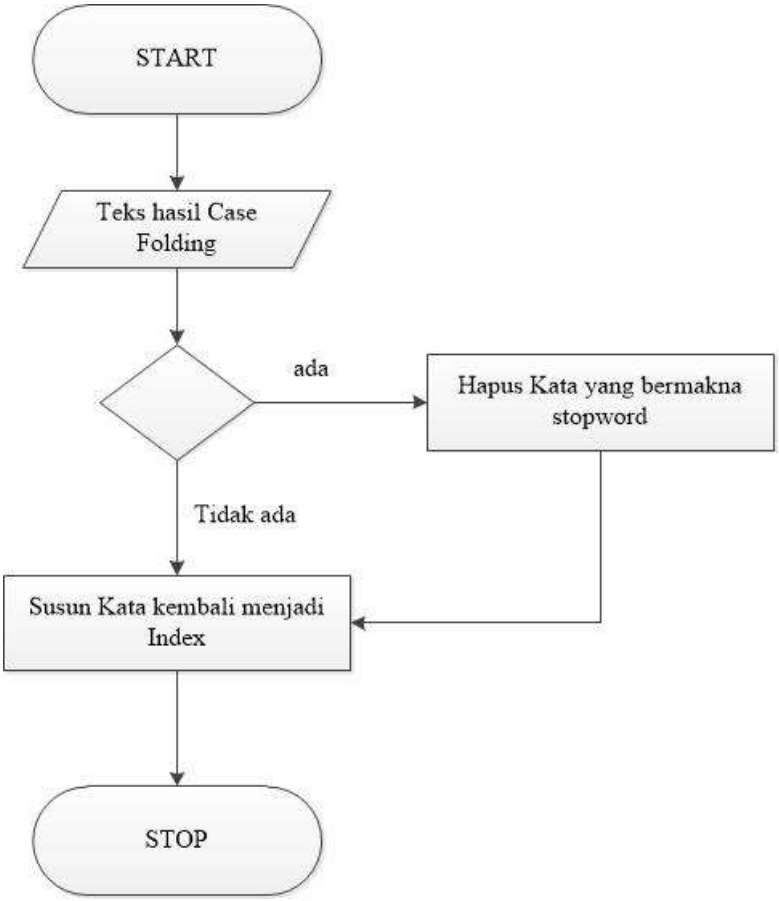
Gambar 3.1 Alur Flow Case Folding dan Tokenizing

Keterangan :

1. Memulai program.
2. Memasukan dokumen teks
3. Melakukan penyeragaman kata dengan mengubah huruf menjadi huruf kecil.
4. Menghapus semua karakter symbol dan lain sebagainya kecuali huruf a-z.
5. Output dari *case folding* dan *tokeninzing.*
6. Selesai

2. Filtering

Pengertian *Filtering* sudah dijelaksan pada bab sebelumnya bisa dilihat pada bab 2 landasan teori. Sedangkan pada bab ini akan dijelaskan alur flow dari *filtering* dapat dilihat pada Gambar 3.2



Gambar 3.2 Alur Flow Filtering

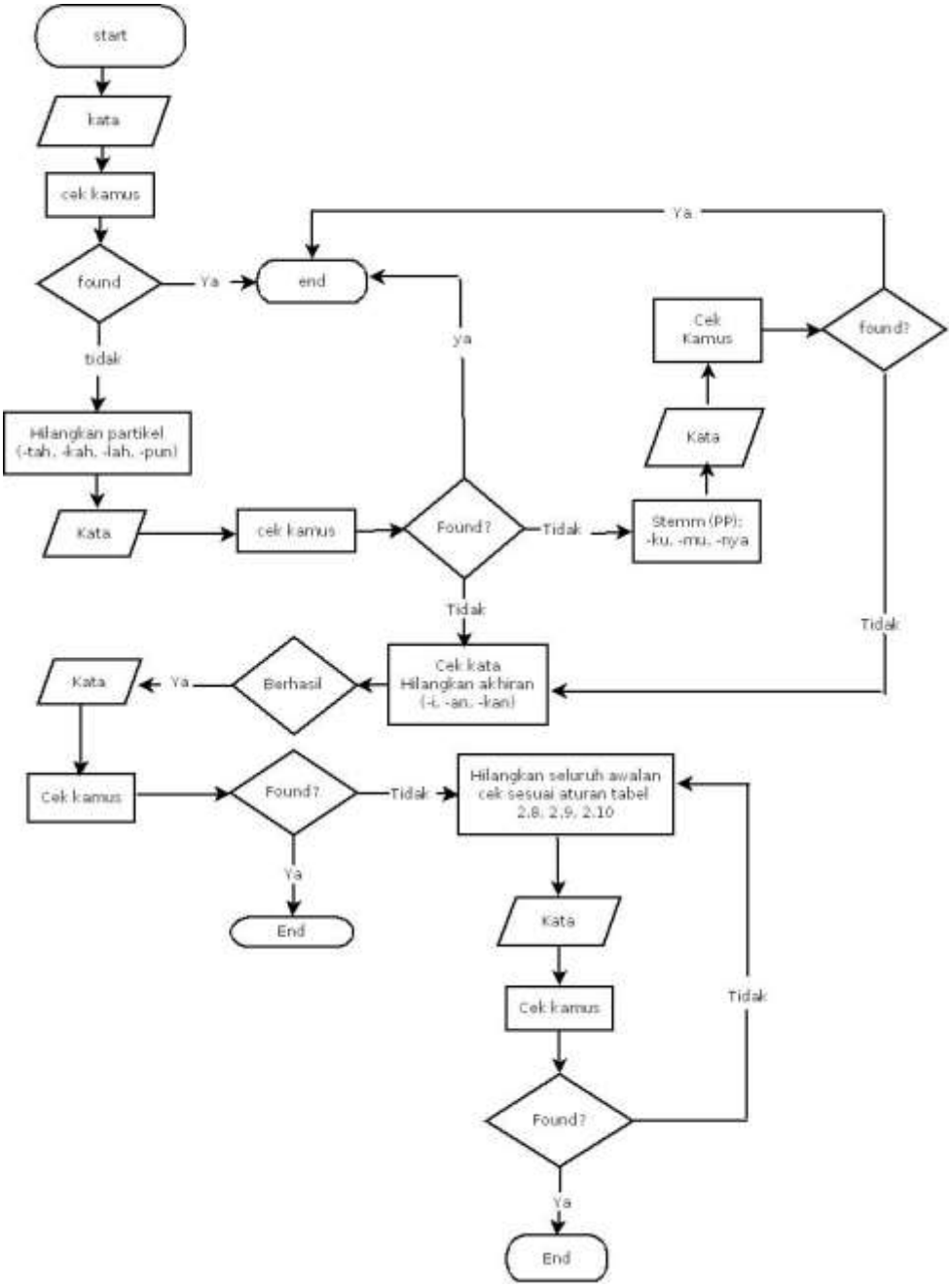
Keterangan :

1. Memulai program.
2. Output dari *case folding* akan diolah kembali untuk dicek kata-kata yang terdapat makna *stopword.*
3. Cek kata stopword jika ditemukan hapus kata tersebut jika tidak susun kembali kata menjadi sebuah index.
4. Selesai.

3. Stemming Algoritma Nazief & Adriani

Stemming Nazief dan Adriani dikembangkan berdasarkan aturan morfologi Bahasa Indonesia yang mengelompokkan imbuhan menjadi awalan (prefix), sisipan (infix), akhiran (suffix) dan gabungan awalan akhiran (confixes). Algoritma ini menggunakan kamus kata dasar dan mendukung recoding, yakni

penyusunan kembali kata-kata yang mengalami proses stemming berlebih**.** Untuk lebih jelasnya mengenai *stemming* sudah dijelaksan pada bab sebelumnya dapat dilihat pada bab 2 landasan teori. Sedangkan pada bab ini akan dijelaskan alur flow dari *stemming* dengan algoritma nazief dan adriani dapat dilihat pada Gambar 3.3



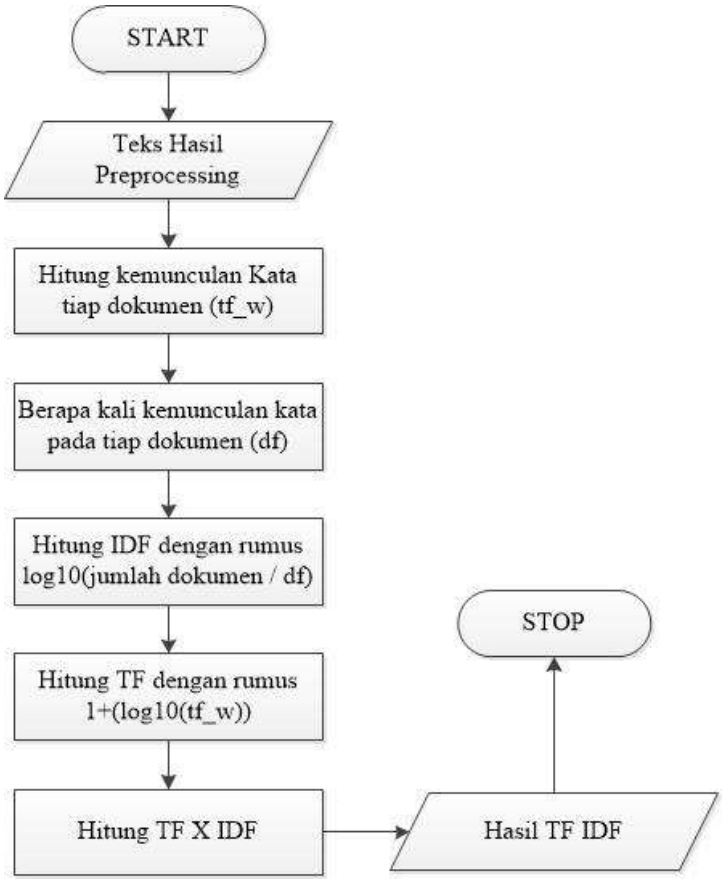
Gambar 3.3 Stemming dengan Algortitma Nazief dan Adriani

Keterangan :

1. Cari kata dalam kamus jika ditemukan maka diasumsikan bahwa kata tersebut adalah kata dasar. Algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 2.
2. Hilangkan *inflectional suffixes* bila ada. Dimulai dari *inflectional particle* (“-lah”, “-kah”, “-tah” dan “-pun”), kemudian *possessive pronoun* (“-ku”, “-mu” dan “-nya”). Cari kata pada kamus jika ditemukan algoritma berhenti, jika kata tidak tidak ditemukan dalam kamus lakukan langkah 3.
3. Hilangkan *derivation suffixes* (“-an”, “-i” dan “-kan”). Jika akhiran “-an” dihapus dan ditemukan akhiran “-k”, maka akhiran “-k” dihapus. Cari kata pada kamus jika ditemukan algoritma berhenti, jika kata tidak tidak ditemukan maka lakukan langkah 4.
4. Pada langkah 4 terdapat tiga iterasi.
   1) Iterasi berhenti jika :
      a Ditemukannya kombinasi awalan yang tidak diizikan berdasarkan awalan
      b Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
      c Tiga awalan telah dihilangkan.
   2) Identifikasikasikan tipe awalan dan hilangkan. Awalan terdiri dari dua tipe:
5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses *recording* dilakukan dengan mengacu pada aturan tabel 2.4. *Recording* dilakukan dengan menambahkan karakter *recording* di awal kata yang dipenggal. Pada tabel 2.4, karakter *recording* adalah huruf kecil setelah tanda hubung (‘-‘) dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata “menangkap” (aturan 15) pada tabel 2.4, setelah dipenggal menjadi “nangkap”. Karena tidak valid, maka *recording* dilakukan dan menghasilkan kata “tangkap”.
6. Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini di anggap sebagai kata dasar.

4. Pembobotan TF-IDF

TF-IDF (Term Frequency-Inversed Document Frequency) merupakan salah satu algoritma pembobotan sebuah kata seperti yang dijelaskan pada bab sebelumya bisa dilihat pada bab 2 landasan teori mengenai pengertian serta rumus dalam pembobotan TF-IDF. Dalam hal ini pembobotan TF-IDF memiliki alur pembobotan dapat dilihat pada Gambar 3.4



Gambar 3.4 Alur Flow Pembobotan TF-IDF

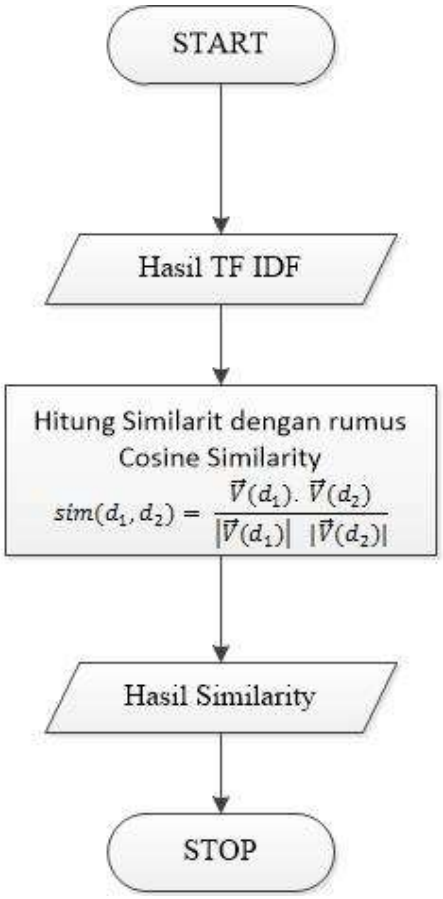
Keterangan :

1. Mulai.
2. Ambil kata hasil *text preprocessing* dalam *database.*
3. Hitung jumlah kemunculan kata tiap dokumen dengan disimbolkan sebagai (*tf_w*).
4. Hitung berapa kali kemunculan kata yang sama pada tiap dokumen dengan disimbolkan sebagai (*df).*
5. Hitung *idf* dengan rumus *log10(* jumlah dokumen / *df).*

6. Hitung *tf* dengan rumus 1+*log10(tf_w).*
7. Hitung *tf x idf.*
8. Hasil nilai
9. Berhenti

5. Metode Cosine similarity

Cosine Similarity merupakan salah satu metode untuk mencari suatu nilai *similarity* pada suatu dokumen dengan menggunakan pemodelan perhitungan vektor yang pada bab sebelumnya sudah dijelaskan bisa dilihat pada bab 2 landasan teori mengenasi pengertian metode *cosine similarity* beserta rumusnya dapat dilihat pada bab 2 sub bab 2.5 *Cosine Similarity* sedangkan untuk bab ini akan dijelaskan alur flow Metode *cosine similarity* seperti terlihat pada Gambar 3.5



Gambar 3.5 Alur Flow Metode Cosine Similarity

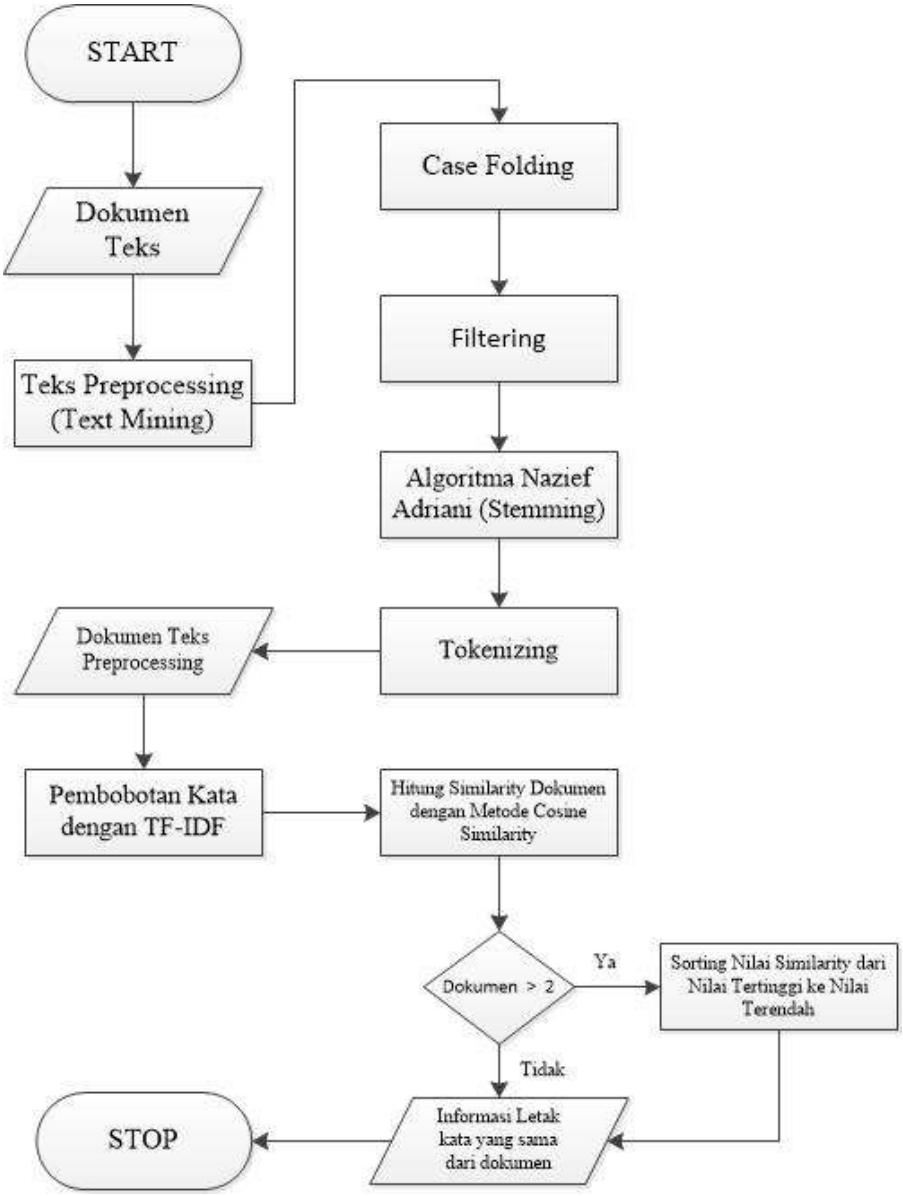
Keterangan Gambar 3.5 :

1. Memulai program.
2. Mengambil hasil nilai pembobotan *tf-idf.*
3. Lakukan perhitugan sesuai dengan rumus.
4. Hasil perhitungan.
5. Selesai.

Sistem ini dapat digunakan untuk semua pihak yang membutuhkan informasi letak kemiripan kata dari dokumen. Berdasarkan hasil analisa sistem dapat diketahui kebutuhan fungsional dari aplikasi deteksi kemiripan dokumen yang akan dibuat nantinya antara lain sebagai berikut :

1. Melakukan penyeragaman kata dengan mengubah seluruh kata menjadi huruf kecil biasa disebut *case-folding.*
2. Melakukan *filtering* terhadap kata-kata yang tidak penting dalam hal ini biasa disebut *stopword*
3. Mengubah kata yang memiliki imbuhan dalam Bahasa Indonesia menjadi sebuah kata dasar istilahnya adalah *stemming* menggunakan Algoritma Nazief dan Adriani.
4. Melakukan pembobotan kata dari dokumen dengan TF-IDF.
5. Melakukan sebuah penilaian similarity dokumen dengan *cosine similarity.*
6. Mencari letak kemiripan kata dari dokumen yang sedang dibandigkan.

Sistem yang dibangun merupakan sebuah aplikasi atau tool untuk mendeteksi kemiripan dokumen dengan memanfaatkan suatu ilmu yang mengacu pada *text preprocessing* atau biasa disebut dengan *text mining.* Dalam aplikasi ini data yang akan diolah adalah sebuah text dengan alir data seperti terlihat pada Gambar 3.6



Gambar 3.6 Diagram Alir Data Aplikasi Deteksi Kemirpan Dokumen

Berdasarkan Gambar 3.6 Diagram Alir Data Aplikasi Deteksi kemiripan Dokumen didapatkan alir data sebagai berikut :

1. Start atau memulai program

   Pada tahap ini merupakan permulaan dengan membuka aplikasi atau jika belum mempunyai aplikasi ini bisa melakukan instalasi aplikasi terlebih dahulu.

2. Input Dokumen Teks

   Pada tahap ini pengguna melakukan input dokumen teks bisa berupa format teks langsung atau format dokumen berekstensi .docx, .pdf dan dalam tahap ini pengguna juga diharuskan memilih dokumen yang di input termasuk dokumen sumber atau dokumen pembanding.

3. *Text preprocessing* atau *text mining*

   Text preprocessing atau *text mining* merupakan tahapan awal dalam mengolah teks untuk dapat dilakukan deteksi kemiripan dokumen.

4. *Case Folding*

   Case Folding merupakan salah satu tahapan dalam *text mining* yang berfungsi melakukan penyeragaman kata dengan mengubah semua huruf menjadi huruf kecil.

5. *Filtering*

   Filtering merupakan salah satu tahapan dalam *text mining* yang berfungsi melakukan *filter* kata yang bermakna *stopword. Stopword* merupakan kata hanya sebagai penghubung antar kata seperti halnya : dan, ke, atau, yang, atau, sebagai, dari, adalah, kemana, dimana, tetapi, tersebut, walupun, dan lain sebagainya dan kata yang bermkana *stopword* akan dihapus. Untuk lebih jelasnya mengenai *stopword* dapat dilihat pada bab 2 landasan teori sub bab 2.2 stop word.

6. *Stemming* dengan Algoritma Nazief dan Adriani

   Stemming merupakan salah satu tahapan dalam *text mining* yang berfungsi merubah kata menjadi kata dasar tanpa imbuhan untuk lebih jelasnya mengenai *stemming* dan Algoritma Nazief dan Adriani dapat dilihat pada bab 2 landasan teori sub bab 2.3 Algoritma Nazief dan Adriani

7. *Tokenizing*

   Tokenizing merupakan suatu tahapan pemotongan *string* kata berdasrakan penyusunan kata tersebut.

8. Dokumen *text preprocessing*

   Pada tahap ini akan didapatkan keluaran kata hasil dari *text preprocessing* setelah melalui beberapa tahapan dalam *text mining* seperti *case folding*, *filtering*, *stemming* dan *tokenizing.*

9. Pembobotan kata dengan TF-IDF

   Pembobotan kata adalah tahapan yang dilakukan selanjutnya dalam mendetksi kemiripan dokumen dengan memberi bobot nilai pada kata hasil dari *text preprocessing* dengan rumus sebagai berikut :

$$tf\_wt_{t,d} = \begin{cases} 1 + \log_{10}(tf_{t,d}) \;\; Jika \;\; tf_{t,d} > 0 \\ 0 \qquad\qquad\qquad jika\; tf_{t,d} \leq 0 \end{cases}$$

   Untuk lebih jelasnya mengenai perhitungan pembobotan tf-idf dapat dilihat pada bab 2 landasan teori sub bab 2.4 TF-IDF.

10. Hitung *similarity* dokumen dengan Metode *cosine similarity*

    Pada tahap ini dilakukan suatu perhitungan *similarity* menggunakan metode *cosine similarity* dengan menggunakan bobot nilai yang sudah diberikan sebelumnya menggunakan pembobotan TF-IDF. Perhitungan *cosine similarity* dapat dilakukan dengan rumus sebagai berikut :

$$sim(d_1, d_2) = \frac{\vec{V}(d_1).\ \vec{V}(d_2)}{|\vec{V}(d_1)|\ \ |\vec{V}(d_2)|}$$

    Untuk lebih jelasnya mengenai *cosine similarity* dapat dilihat pada bab 2 landasan teori sub bab 2.5 *cosine similarity.*

11. *Sorting* nilai similarity dokumen dari nilai tertinggi hingga terendah.

    Pada tahap ini dilakukan sebuah sorting jika dokumen yang dibandingkan lebih dari dua dokumen dan jika tidak langsung tampilkan nilai similarity dokumen. Nilai similarity ini menjadi acuan untuk melihat data memiliki kemiripan atau tidak yang nantinya bisa dibuktikan dengan melihat informasi letak kemiripan kata yang sama dari dokumen tersebut.

12. Informasi Letak kata yang sama.

    Pada tahap ini merupakan tahap dimana pengguna dapat mengetahui informasi letak kata yang sama dari dokumen yang sedang dibandingkan.

13. Stop.

Berdasarkan uraian diatas dapat dilakukan hipotesa dengan menggunakan representasi model data yang sudah ada dengan menggunakan data dari penelitian sebelumnya yang dilakukan oleh Azhar Firdaus, Ernawati dan Arie Vatresia (2014). Jika pada bab 2 landasan teori sub bab 2.7 penelitian sebelumnya menggunakan data uji pertama untuk representasi model akan digunakan data uji yang kedua.

3.1.2 Representasi Model

Table 3.1 data uji kedua

No	Ko	Dokumen
1	D1	Manusia memanfaatkan komputer untuk menyelesaikan pekerjaan. Komputer mempermudah pekerjaan sehingga lebih efektif dan efesien. Manusia menggunakan komputer untuk membantu pekerjaan mereka pada hampir seluruh bidang pekerjaan. Peranan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. Komputer memiliki peranan yang sangat penting bagi kehidupan manusia.
2	D2	Komputer dimanfaatkan manusia dalam penyelesaian pekerjaan. Pekerjaan dipermudah sehingga lebih efektif dan efesien. Komputer digunakan hampir disetiap bidang pekerjaan. Kegunaan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. Komputer memiliki peranan yang sangat penting bagi kehidupan manusia.

Berdasarkan Table 3.1 dilakukan sebuah representasi model dengan data uji kedua yang sama dengan penelitian sebelumnya data tersebut dilakukan sebuah pembelajaran menggunakan sistem dan pembobotan yang berbeda sesuai dengan yang digambarkan pada Gambar 3.1 Diagram Alir Data Aplikasi Deteksi Kemiripan.

Case folding

Table 3.2 Case folding Dokumen Pembanding atau (D1)

Teks Asli	Teks Hasil Case Folding
Manusia memanfaatkan komputer untuk menyelesaikan pekerjaan. Komputer mempermudah pekerjaan sehingga lebih efektif dan efesien. Manusia menggunakan komputer untuk membantu pekerjaan mereka pada hampir seluruh bidang pekerjaan. Peranan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. Komputer memiliki peranan yang sangat penting bagi kehidupan manusia.	manusia memanfaatkan komputer untuk menyelesaikan pekerjaan. komputer mempermudah pekerjaan sehingga lebih efektif dan efesien. manusia menggunakan komputer untuk membantu pekerjaan mereka pada hampir seluruh bidang pekerjaan. peranan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. komputer memiliki peranan yang sangat penting bagi kehidupan manusia.

Table 3.3 Case Folding Dokumen sumber atau (D2)

Teks Asli	Teks Hasil Case Folding
Komputer dimanfaatkan manusia dalam penyelesaian pekerjaan. Pekerjaan dipermudah sehingga lebih efektif dan efesien. Komputer digunakan hampir disetiap bidang pekerjaan. Kegunaan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. Komputer memiliki peranan yang sangat penting bagi kehidupan manusia.	komputer dimanfaatkan manusia dalam penyelesaian pekerjaan. Pekerjaan dipermudah sehingga lebih efektif dan efesien. komputer digunakan hampir disetiap bidang pekerjaan. kegunaan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. komputer memiliki peranan yang sangat penting bagi kehidupan manusia.

Berdasarkan Table 3.2 dan 3.3 dokumen data teks asli dilakukan penyeragaman kata menjadi huruf kecil semua hasilnya perbedaannya dapat dilihat pada masing-masing table 3.2 dan 3.3 kolom teks asli dan teks hasil case folding

Filtering

Table 3.4 Filtering Dokumen pembanding atau (D1)

Teks Hasil Case Folding	Teks Hasil Filtering
manusia memanfaatkan komputer untuk menyelesaikan pekerjaan. komputer mempermudah pekerjaan sehingga lebih efektif dan efesien. manusia menggunakan komputer untuk membantu pekerjaan mereka pada hampir seluruh	memanfaatkan menyelesaikan pekerjaan mempermudah pekerjaan efektif efisien membantu pekerjaan bidang pekerjaan peranan ditemukan bidang kesehaatan keamanan

bidang pekerjaan. peranan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. komputer memiliki peranan yang sangat penting bagi kehidupan manusia.	perkantoran pendidikan memiliki peranan kehidupan

Table 3.5 Filtering Dokumen sumber atau (D2)

Teks Hasil Case Folding	Teks Hasil Filtering
komputer dimanfaatkan manusia dalam penyelesaian pekerjaan. Pekerjaan dipermudah sehingga lebih efektif dan efesien. komputer digunakan hampir disetiap bidang pekerjaan. kegunaan komputer dapat ditemukan pada bidang kesehatan, keamanan, perkantoran dan pendidikan. komputer memiliki peranan yang sangat penting bagi kehidupan manusia	dimanfaatkan penyelesaian pekerjaan pekerjaan dipermudah efektif efisien disetiap bidang pekerjaan kegunaan ditemukan bidang kesehatan keamanan perkantoran pendidikan memiliki peranan kehidupan

Berdasarkan table 3.4 dan 3.5 teks hasil case folding kemudian diteruskan ke tahapan proses *text mining* berikutnya yaitu *filtering* dengan dilakukan menghapus kata yang bermakna stopword atau kata penghubung seperti kata : yang, dalam, manusia, dan sehingga, bagi, pada, hamper, dapat. Kata yang mengandung stopword akan dihapus dan hasilnya perbedaannya dapat dilihat pada table 3.4 dan table 3.5 kolom teks hasil case folding dan kolom teks hasil filtering.

Stemming

Table 3.6 Stemming Dokumen pembanding atau (D1)

Teks Hasil Filtering	Teks Hasil Stemming
memanfaatkan menyelesaikan pekerjaan mempermudah pekerjaan efektif efisien membantu pekerjaan bidang pekerjaan peranan ditemukan bidang kesehaatan keamanan perkantoran pendidikan memiliki peranan kehidupan	manfaat selesai kerja mudah kerja efektif efisien bantu kerja bidang kerja peran temu bidang sehat aman kantor didik milik peran hidup

Table 3.7 Stemming Dokumen Sumber atau (D2)

Teks Hasil Filtering	Teks Hasil Stemming
dimanfaatkan penyelesaian pekerjaan pekerjaan dipermudah efektif efisien disetiap bidang pekerjaan kegunaan ditemukan bidang kesehatan keamanan perkantoran pendidikan memiliki peranan kehidupan	Manfaat selesai kerja kerja mudah efektif disetiap bidang kerja guna temu bidang sehat aman kantor didik milikperan hidup

Berdasarkan table 3.6 dan 3.7 teks hasil filtering akan diproses ke tahapan berikutnya yang terdapat dalam *text mining* yaitu *stemming* dengan cara mengubah kata menjadi kata dasar tanpa imbuhan seperti kata imbuhan : di, pen, ke, per, mem dan lain sebagianya sesuai dengan aturan Algoritma Nazief dan Adriani sehingga didapatkan hasil perbedaan teks yang dapat dilihat pada masing table 3.6 dan 3.7 kolom teks hasil *filtering* dan teks hasil *stemming.* Dari hasil teks *stemming* kemudian dilakukan *tokenizing* pengurutan kata yang membentuknya sesuai teks hasilnya dapat dilihat pada table 3.8 *tokenizing.*

Tokenizing

Table 3.8 Tokenizing

Kata hasil Stemming	Type Dokumen
manfaat	1
selesai	1
kerja	1
mudah	1
kerja	1
efektif	1
efesien	1
bantu	1
kerja	1
bidang	1
kerja	1
peran	1

Lanjutan dari Tabel 3.8

Kata Hasil Stemming	**Type Dokumen**
temu	1
bidang	1
sehat	1
aman	1
kantor	1
didik	1
milik	1
peran	1
hidup	1
manfaat	2
selesai	2
kerja	2
kerja	2
mudah	2
efektif	2
efesien	2
disetiap	2
bidang	2
kerja	2
guna	2
temu	2
bidang	2
sehat	2
aman	2
kantor	2
didik	2
milik	2
peran	2
hidup	2

Table 3.9 Pembobotan TF-IDF

NO	Kata	Dokumen		Bobot		df	idf	Nilai TF-IDF	
		D1	D2	tf_wt1	tf_wt2			tf_idf 1	tf_idf 2
1	aman	1	1	1	1	2	1	**1**	**1**
2	bantu	1	0	1	0	1	0.301	**0.301**	**0**
3	bidang	2	2	1.301	1.301	2	1	**1.301**	**1.301**
4	didik	1	1	1	1	2	1	**1**	**1**
5	disetiap	0	1	0	1	1	0.301	**0**	**0.301**
6	efektif	1	1	1	1	2	1	**1**	**1**
7	efisien	1	1	1	1	2	1	**1**	**1**
8	guna	0	1	0	1	1	1	**0**	**0.301**
9	hidup	1	1	1	1	2	1	**1**	**1**
10	kantor	1	1	1	1	2	1	**1**	**1**
11	kerja	4	3	1.602	1.477	2	1	**1.602**	**1.477**
12	manfaat	1	1	1	1	2	1	**1**	**1**
13	milik	1	1	1	1	2	1	**1**	**1**
14	mudah	1	1	1	1	2	1	**1**	**1**
15	peran	2	1	1.301	1	2	1	**1.301**	**1**
16	sehat	1	1	1	1	2	1	**1**	**1**
17	selesai	1	1	1	1	2	1	**1**	**1**
18	temu	1	1	1	1	2	1	**1**	**1**

$$sim(d_1, d_2) = \frac{\vec{V}(d_1).\ \vec{V}(d_2)}{|\vec{V}(d_1)|\ \ |\vec{V}(d_2)|}$$

$$\vec{V}(D_1) = \frac{1\ 0{,}301\ 0{,}301\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1{,}602\ 1\ 1\ 1\ 1{,}301\ 1\ 1\ 1}{\sqrt{1^2 + 0{,}301^2 + 0{,}301^2 + 1^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 1^2 + 1^2 + 1{,}602^2} + 1^2 + 1^2 + 1^2 + 1{,}301^2 + 1^2 + 1^2 + 1^2}$$

$$\vec{V}(D_1) = \frac{1\ 0{,}301\ 0{,}301\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1{,}602\ 1\ 1\ 1\ 1{,}301\ 1\ 1\ 1}{\sqrt{18{,}042207}}$$

$$\vec{V}(D_1) = \frac{1\ 0{,}301\ 0{,}301\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 1{,}602\ 1\ 1\ 1\ 1{,}301\ 1\ 1\ 1}{4{,}24761}$$

D1 = {0,23543} {0,07086 }{0,30629}{0,23543}{0}{0,23543} {0,23543} {0} {0,23543}

= {0,23543} {0,23543} {0,37715} {0,23543} {0,23543} {0,23543} {0,30629} {0,23543}

= {0,23543} {0,23543}

$$\vec{V}(D_2) = \frac{1\ 0\ 1{,}301\ 1\ 0{,}301\ 1\ 1\ 0{,}301\ 1\ 1\ 1{,}477\ 1\ 1\ 1\ 1\ 1\ 1\ 1}{\sqrt{\begin{matrix}1^2 + 0^2 + 1{,}301^2 + 1^2 + 0{,}301^2 + 1^2 + 1^2 + 0{,}301^2 + 1^2 + 1^2 \\ + 1{,}477^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2\end{matrix}}}$$

$$\vec{V}(D_2) = \frac{1\ 0\ 1{,}301\ 1\ 0{,}301\ 1\ 1\ 0{,}301\ 1\ 1\ 1{,}477\ 1\ 1\ 1\ 1\ 1\ 1\ 1}{\sqrt{17{,}055333}}$$

$$\vec{V}(D_2) = \frac{1\ 0\ 1{,}301\ 1\ 0{,}301\ 1\ 1\ 0{,}301\ 1\ 1\ 1{,}477\ 1\ 1\ 1\ 1\ 1\ 1\ 1}{4{,}12981}$$

D2 = {0,24214} {0 }{0,31503}{0,24214}{0,07288}{0,24214} {0,24214} {0,07288} {0,24214}

{0,24214}{0,35764}{0,24214}{0,24214}{0,24214}{0,24214}{0,24214}{0,24214} {0,24214}

Sim(D1,D2) = $(D1_1 * D2_1) + (D1_2 * D2_2) + \cdots + (D1_9 * D2_9)$

Sim (D1,D2) = **0,9896**

Table 3.10 Hasil Nilai perhitungan dengan Data uji kedua

Dokumen Sumber	Dokumen Pembanding	Hasil Perhitungan dengan algoritma Nazief & Adriani (Penelitian Sebelumnya)	Hasil Perhitungan dengan Text Mining pembobotan TF IDF (Penelitian Selanjutnya	Batasan Threshold	Status Dokumen
D1	D2	0,9381	0,9896	< 0.50 = tidak mirip > 0.50 = mirip	Mirip

Berdasarkan Table 3.9 Pembobotan TF-IDF didapatkan hasil pembobotan kata dari dua dokumen dapat dilihat pada table 3.9 kolom nilai TF-IDF. Dari hasil nilai pembobotan kata kemudian akan diakumulasikan dengan metode *cosine similarity* untuk mendapatkan hasil nilai similarity dua dokumen sehingga didapatkan hasil keluaran seperti terlihat pada table 3.10 terdapat peningkatan ketelitian dari penelitian sebelumnya dan penelitian selanjutnya dengan nilai perbedaan ketelitian 0,0515 dengan batasan threshold tertentu.

3.2 Perancangan system

Berdasarkan dari alur flow pada deskripsi sistem dapat dimodelkan sebuah desain sistem yang sesuai dengan urutan proses yang telah ditetapkan, yaitu *Context Diagram*, Diagram Jenjang, Diagram Alir Data (*Data Flow Diagram*), Desain Basis Data (*Database*), Ektrasi Dokumen, Desain Antarmuka (*Interface*).

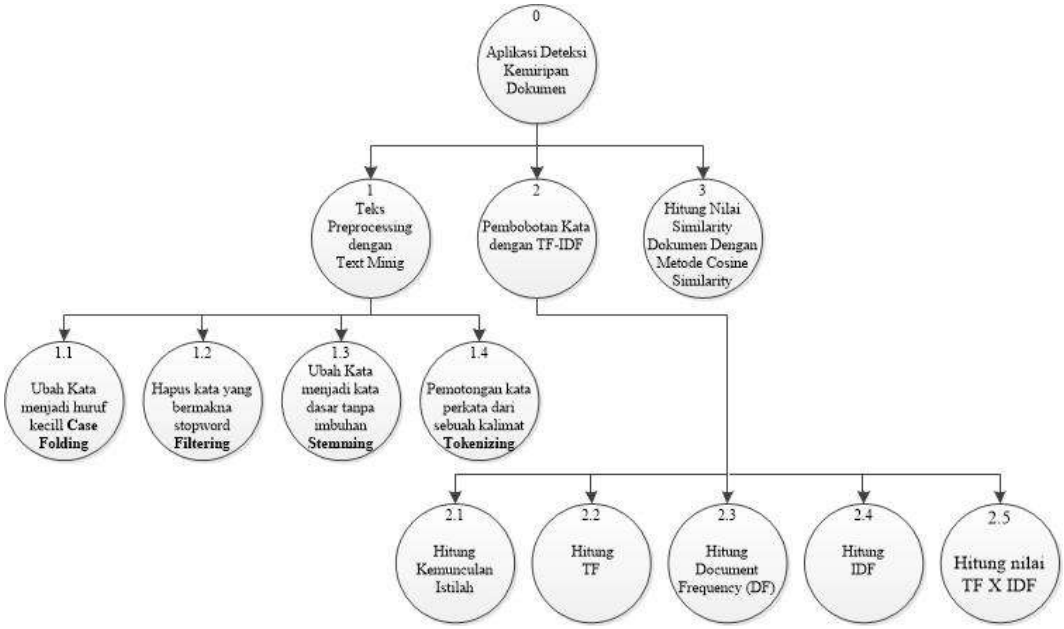
3.2.1 Diagram konteks

Gambar 3.7 Diagram Konteks

Berdasarkan Gambar 3.7 pengguna memasukan dokumen teks kedalam aplikasi deteksi kemiripan dokumen dan pengguna akan mendapat keluaran dari sistem yaitu sebuah informasi mengetahui letak kemiripan kata yang sedang dicari *similaritynya.*

3.2.2 Diagram Berjenjang

Diagram berjenjang merupakan pemecahan proses dari sebuah sistem dimana pada aplikasi deteksi kemiripan dibagi beberapa proses anatar lain : simpan dokumen, text preprocessing pembobotan TF-IDF, perhitungan nilai *similarity* dengan metode cosine similarity sedangkan untuk pemecahan proses



dari text preprocessing terdapat *case folding, filtering, stemming, tokenizing.*

Gambar 3.8 Diagram Berjenjang

Berdasarkan Gambar 3.3 didapatkan beberapa pemecahan proses yang terbagi dalam beberapa level sebagai berikut :

Level 0 Aplikasi Detksi Kemiripan dokumen

Level 1.

1. Teks Preprocessing dengan *Text Mining*

Teks Preprocessing merupakan tahapan awal dalam mengolah dokumen teks yang diunggah oleh pengguna kedalam sistem dan diproses menggunakan teknik *text mining.*

2. Pembobotan kata dengan TF-IDF

   Pembobotan Kata dengan TF-IDF dilakukan untuk memberikan nilai bobot kata dari masing dokumen.

3. Hitung Nilai Similarity dengan Metode *Cosine Similarity*

   Hitung Nilai similarity dilakukan untuk mencari nilai similarity dokumen dengan menggunakan metode cosine similarity dari bobot kata yang telah diperoleh dengan TF-IDF. Kemudian dari nilai bobot tersebut kita akumulasikan terhadap rumus *cosine similarity* seperti terlihat pada persamaan ke 4

Level 2 proses 1

1.0 *Case Folding*

Case Foldng merupakan salah satu sub proses yang terdapat dalam tahapn *text mining* digunakan untuk penyeragaman kata dari huruf besar menjadi huruf kecil.

1.1 *Filtering*

Filtering merupakan salah satu sub proses yang terdapat dalam tahapan *text mining* digunakan untuk menghapus kata yang bermakna *stopword*

1.2 Algortitma Nazief dan Adriani *Stemming*

Stemming merupakan salah satu sub proses yang terdapat dalam tahapan *text mining* digunakan untuk mengubah kata menjadi kata dasar tanpa imbuhan

1.3 *Tokenizing*

Tokenizing merupakan salah satu sub proses yang terdapat dalam tahapan *text mining* sebagai pengurutan kata dari kata yang membentuk pada suatu dokumen

Level 2 proses 2

2.0 Hitung kemunculan istilah

Proses ini dilakukan suatu perhitungan kemunculan istilah kata pada masing masing dokumen dari index kata.

2.1 Hitung TF (Term Frequency)

Proses ini dilakukan suatu perhitungan TF dari tiap kata dari masing masing dokumen dengan rumus seperti terlihat pada persamaan ke 1

2.2 Hitung DF (Document Frequency)

Proses ini dilakukan suatu perhitungan kemunculuan jumlah kata pada tiap dokumen.

2.3 Hitung IDF (Inverse Document Frequency)

Proses ini dilakukan suatu perhitungan inverse dokumen frequency pada masing kata pada tiap dokumen dengan rumus seperti terlihat pada persamaan ke 2.

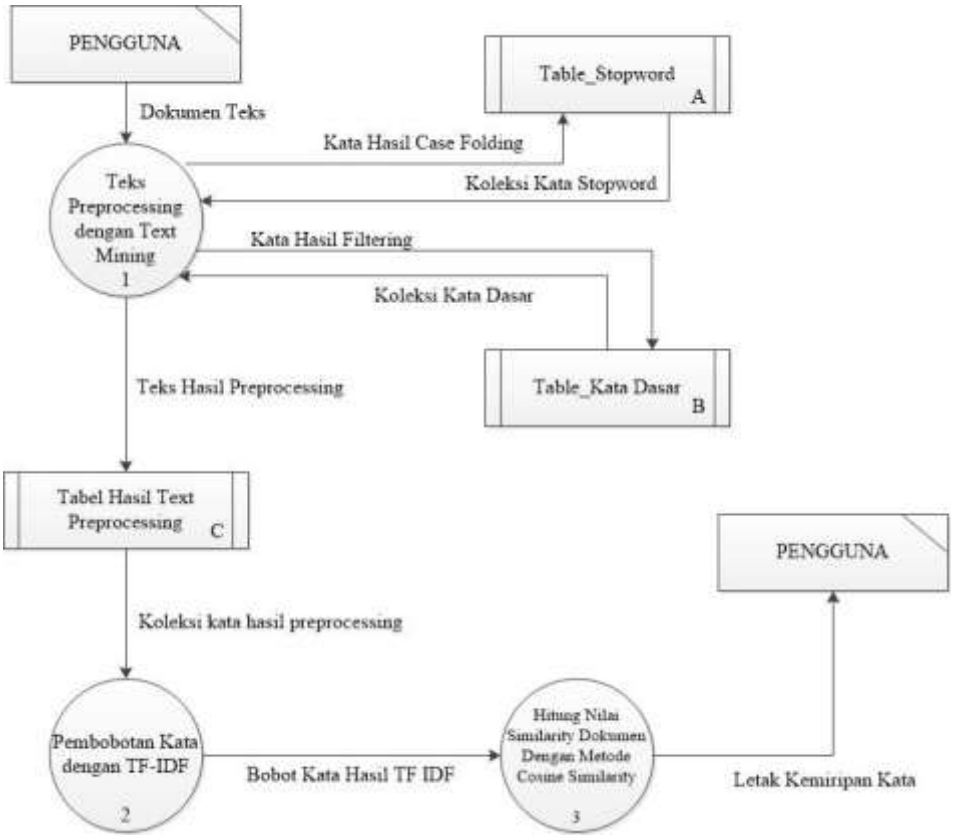
2.4 Hitung TF X IDF

Proses ini dilakukan akumulasi terkahir dengan menghitung nilai TF X IDF seperti terlihat pada persamaan ke 3 untuk rumus keseluruhan dari pembobotan kata dengan menggunakan TF IDF untuk melihat hasil perhitungan dapat dilihat pada table 3.9

3.2.3 Data Flow Diagram

Data Flow Diagram (DFD) adalah suatu diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem, yang penggunaannya sangat membantu untuk memahami sistem secara logika, tersruktur dan jelas. DFD merupakan alat bantu dalam menggambarkan atau menjelaskan sistem yang sedang berjalan logis sedangkan untuk aplikasi deteksi kemiripan dokumen memiliki arus dari data sistem yang digambarkan sebagai berikut:

DFD Level 1



Gambar 3.9 DFD Level 1

Berdasarkan Gambar 3.4 didapatkan data flow diagram sebagai berikut :

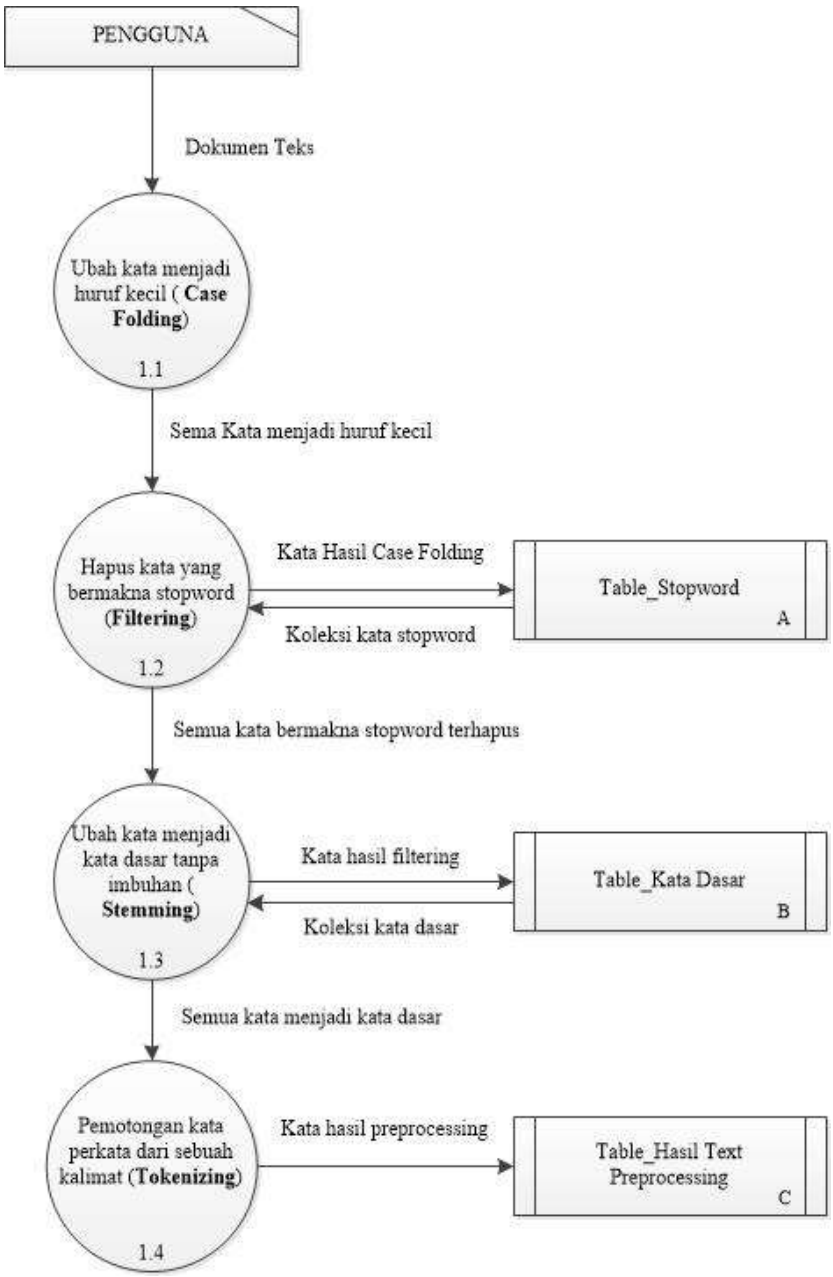
1. Pada diagram tersebut dimulai dari pengguna dengna memasukkan sebuah dokumen kedalam aplikasi.
2. Kemudian dari data yang masuk akan disimpan data asli ke dalam *database* kedalam *table* dokumen.asli.
3. Dari dokumen asli tersebut akan diolah kembali dengan *text mining* dan hasil dari text mining akan dimasukan ke dalam *database* dengan *table* hasil *text preprocessing.*
4. Dalam tahapan text mining terdapat tahapan *case folding* atau penyeragaman kata menjadi huruf kecil.
5. Dari hasil *case folding* akan dilanjutkan ke tahapan *text mining* yaitu *filtering* dengan mengahpus kata yang bermakna *stopword* atau biasa disebut kata penghubung.

6. Hasil dari *filtering* kemudian dilanjutkan ke tahapan berikutnya yaitu *stemming* dengan mengubah kata menjadi kata dasar tanpa imbuhan dalam kosakata Bahasa Indonesia dan hasilnya ditampilkan dalam bentuk *Tokenizing* atau dipotong sesuai kata perkata dari sebuah kalimat dan disimpan dalam *storage text* hasil *preprocessing*.
7. Setelah itu data hasil *text preprocessing* akan diolah kembali untuk diberikan nilai bobot dari kata-kata pada tiap dokumen dengan TF-IDF.
8. Selanjutnya dari pembobotan yang telah dilakukan data akan diambil untuk dilakukan perhitungan nilai *similarity* dengan metode *cosine similarity* dengan rumus dan dapat dilihat seperti pada persamaan ke 4
9. Hasil nilai similarity akan ditampilkan pada pengguna aplikasi.
10. Hasil nilai similarity akan menjadi acuan dari awal untuk mengetahui kemiripan dokumen dan nantinya di buktikan dengan melihat letak dari kemiripan kata dari dokumen tersebut.

DFD Level 2 Proses 1

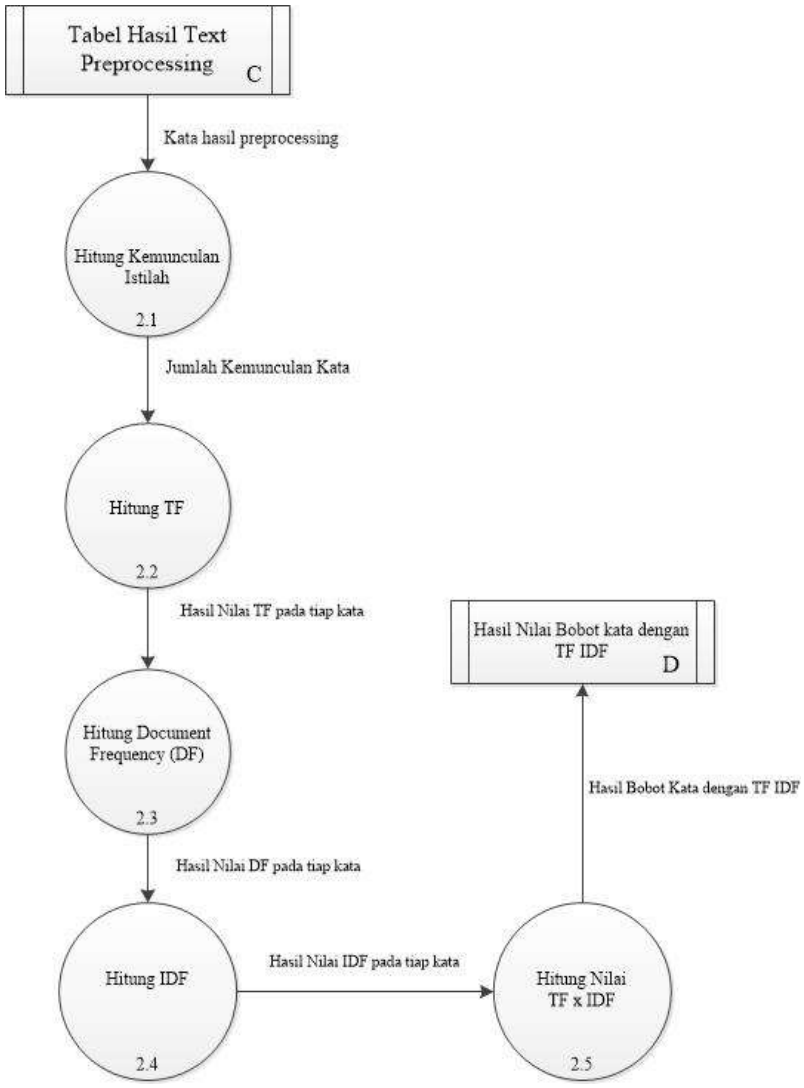
Berdasarkan Gambar 3.5 didapatkan data flow sebagai berikut :

1. Pada diagram tersebut dimulai dari pengguna dengan memasukan sebuah dokumen berisi sebuah teks berbahasa Indonesia.
2. Kemudian dokumen tersebut akan masuk tahap awal *text preprocessing* atau *text mining* yaitu *case folding* dengan melakukan penyeragaman kata menjadi huruf kecil
3. Setelah itu hasil dari *case folding* akan diteruskan ke proses *filtering* dengan menghapus kata-kata yang bermakna *stopword*.
4. Selanjutnya kata-kata tersebut akan diubah ke bentuk kata dasar sebelum mendapatkan kata-kata imbuhan proses ini biasa disebut *stemming*. Dengan mengguakan algoritma nazief & adriani.
5. Hasil dari keseluruhan proses akan dilakukan sebuah pemotongan kata perkata (*tokenizing)* yang nantinya akan dimasukkan ke dalam table hasil text preprocessing



Gambar 3.10 DFD Level 2 Proses 1

DFD Level 2 Proses 2



Gambar 3.11 DFD Level 2 Proses 2

Berdasarkan gambar 3.11 didapatkan sebuah alur flow data dari proses pembobotan dengan TF IDF sebagai berikut :

1. Pengguna akan memasukkan kata hasil text preprocessing untuk diolah oleh sistem untuk dilakukan suatu pembobotan kata pada masing dokumen.
2. Kata hasil text preprocessing akan dilakukan perhitungan jumlah istilah kemunculan kata pada tiap dokumen sebagai acuan awal dari pembobotan TF IDF.

3. Kata hasil text preprocessing kemudian dilakukan perhitungan nilai TF dari tiap kata dari masing dokumen dengan rumus dan dapat dilihat seperti persamaan ke 1
4. Hitung Jumlah kata yang sama dan muncul di masing dokumen perhitungan ini digunakan untuk menghitung DF (Document Frequency).
5. Kata hasil preprocessing kemudian dilakukan perhitungan niali IDF dari tiap kata dari masing dokumen dengan rumus dan dapat dilihat seperti persamaan ke 2
6. Hasil TF dan IDF diakumulasikan dengan perkalian untuk mendapatkan hasil nilai bobot kata dengan TF IDF
7. Selesai dan di simpan ke dalam table hasil nilai bobot kata dengan TF IDF.

3.3 Desain database

1. Tabel kata dasar

Tabel kata dasar berisi kumpulan kata dasar darii sebuah kata dengan berbahasa Indonesia yang nantiny akan digunakan pada proses *stemming* dengan struktur table seperti terlihat pada table 3.11

Tabel 3.11 Kata dasar

Nama Field	Type	Keterangan
Id_katadasar	Int Auto Increment	Primary Key
Kata_dasar	Varchar (255)	
Tipe_keterangan	Varchar (255)	

2. Tabel stopword

Tabel *stopword* adalah kumpulan kata yang bermakna *stopword* yang nantinya akan digunakan pada proses *filtering* dan kata yang sama dengan kata *stopword* akan dihilangkan dengan struktur table seperti terlihat pada table 3.12

Tabel 3.12 Stopword

Nama Field	Type	Keterangan
Id_stopword	Int Auto Increment	Primary Key
Stopword	Varchar (255)	

3. Tabel dokumen asli

Tabel dokumen asli merupakan tempat untuk menyimpan kata asli dari sebuah dokumen sebelum dilakukan proses *text preprocessing* dengan struktur table sebagai seperti terlihat pada table 3.13

Tabel 3.13 Dokumen Asli

Nama Field	Type	Keterangan
Id_dokumen	Int Auto Increment	Primary Key
Judul_dokumen	Varchar (255)	
Isi_dokumen	Text	

4. Tabel dokumen sumber

Tabel dokumen sumber merupakan kumpulan kata yang telah melalui tahapan *text preprocessing* dan menjadi dokumen sumber yang nanti akan menjadi pembanding untuk dokumen pembanding dengan struktur table seperti terlihat pada table 3.14

Tabel 3.14 Dokumen sumber

Nama Field	Type	Keterangan
Id_sumber	Int Auto Increment	Primary Key
Kata	Varchar (255)	
Dok_sumber	Int (20)	
Id_banding	Int	Foreign Key

5. Tabel dokumen pembanding

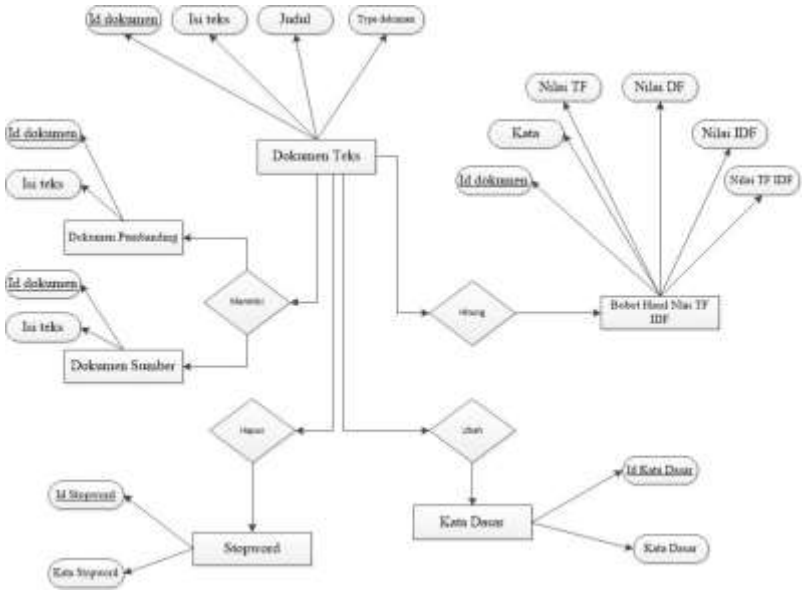
Tabel dokumen pembanding merupakan kumpulan kata yang telah melalui tahapan *text preprocessing* dan menjadi dokumen pembanding untuk dicari nilai similarity dengan dokumen sumber dengan struktur table terlihat pada table 3.15

Tabel 3.15 Dokumen pembanding

Nama Field	Type	Keterangan
Id_banding	Int Auto Increment	Primary Key
Kata	Varchar (255)	
Dok_banding	Int (20)	

3.3.1 Entity Relation Diagram (ERD)

Konsep data model merupakan bentuk data yang masih di konsep untuk direalisasikan dengan tabel-tabel yang lain dan data ini bukan merupakan tabel pada keadaan yang sebenarnya karena masih perlu dilakukan proses *generic* untuk menjadi tabel yang sesuai dengan sebenarnya. Karena masih konsep maka kunci-kunci relasi dari tabel yang lain belum di masukkan diagram ERD *database* yang dirancang dapat dilihat pada Gambar 3.12



Gambar 3.12 Entity Relation Diagram

Berdasarkan gambar 3.12 didapatkan entity relation diagram sebagai berikut:

1. Dokumen teks yang akan diupload memiliki dua tipe dokumen yaitu dokumen pembanding dan dokumen sumber yang akan menjadi sumber dokumen yang akan dibandingkan dengan dokumen pembanding.
2. Dokumen teks sudah diupload akan diproses untuk dicari kata yang bermakna stopword dan kemudian kata tersebut akan dihapus secara otomatis.
3. Dokumen teks sudah diupload akan diubah menjadi kata dasar tanpa imbuhan sesuai EYD Bahasa Indonesia.
4. Dokumen teks yang sudah melalui tahap *text preprocessing* akan dilakukan perhitungan bobot kata TF IDF.

3.4 Desain Interface

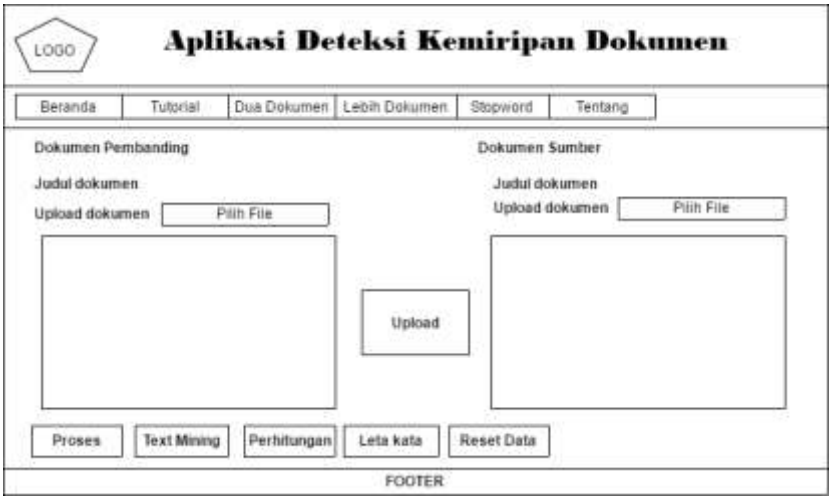
3.4.1 Desain Tampilan Antar Muka (Beranda)

Halaman Antar Muka merupakan tampilan awal dari aplikasi deteksi kemiripan dokumen yang menampilkan deskripsi dari aplikasi tersebut dengan beberapa fitur menu seperti : tutorial yang berisi tentang cara pemakaian aplikasi, menu dua dokumen khusus untuk membandingkan dua dokumen, lebih dokumen merupakan fitur menu untuk membandingkan lebih dokumen, stopword merupakan menu tampilan kata-kata yang bermakna stopword dalam Bahasa Indonesia dan menu tentang merupakan tentang pembuatan aplikasi dan lain sebagainya yang berhubungan dengan pembuatan aplikasi. Berikut merupakan tampilan dari beberapa menu aplikasi tampilan antar muka dapat dilihat pada Gambar 3.13

Gambar 3.13 Desain Tampilan Antar Muka

3.4.2 Desain Tampilan Dua Dokumen

Desain Tampilan dua dokumen memiliki fitur seperti upload file jika dokumen bersumber dari file .pdf atau doc dan terdapat sebuah kolom teks jika dokumen bersumber dari teks langsung seperti terlihat pada Gambar 3.14

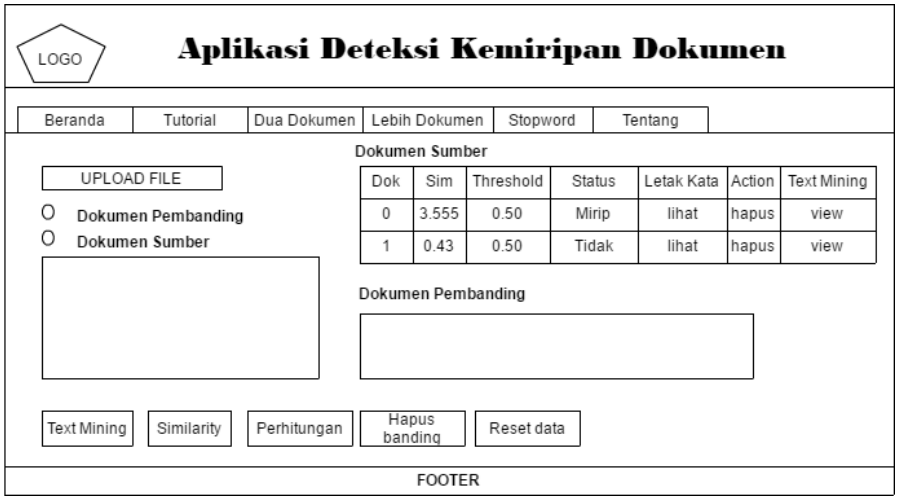


Gambar 3.14 Desain Tampilan Dua Dokumen

3.4.3 Desain Tampilan Lebih Dokumen

Desain Tampilan lebih dokumen memiliki beberapa fitur yang sama seperti fitur dua dokumen yang membedakan terdapat satu table untuk

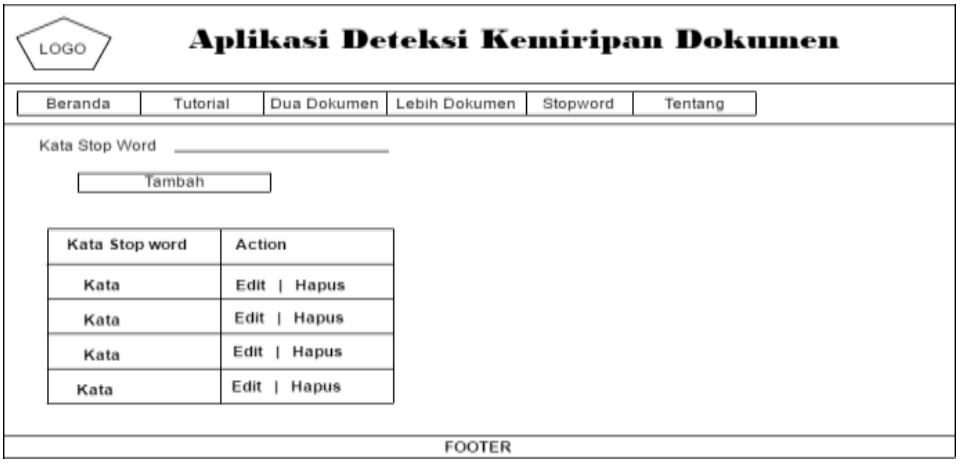
menampung hasil similarity lebih dari dua dokumen seperti terlihat pada Gambar 3.15



Gambar 3.15 Desain Tampilan Lebih Dokumen

3.4.4 Desain Tampilan Stopword

Desain tampilan stopword terdapat fitur untuk menambah, mengubah, atau mengahpus kata-kata yang bermakna stopword seperti terlihat pada Gambar 3.16



Gambar 3.16 Desain Tampilan Stopword

3.5 Kebutuhan Pembuatan Sistem

3.5.1 Perangkat Lunak

1. Sistem Operasi windows 8

   Sistem operasi digunakan untuk mengimplementasi Aplikasi deteksi kemiripan dokumen
2. PHP 5.6.3 dan Apache 2.4.3

   PHP adalah salah satu Bahasa pemrograman yang digunakan untuk membangun aplikasi deteksi kemripan dokumen dan apache sebagai server untuk menjalankan program yang dibuat dengan Bahasa pemrograman PHP
3. MySQL Server 5.5.27 sebagai *database* server

   Mysql adalah salah satu databse server yang digunakan untuk menampung dokumen dari Aplikasi deteksi yang akan dibangun nantinya.
4. Mozilla Firefox 38.0.1 (*browser*)

   Mozila firefox adalah salah satu browser untuk menampilkan output atau keluaran dari program yang telah dibuat
5. Edit Plus dan Microsoft Visio

   Edit plus adalah *software* text editor untuk menulisakn barisan perintah-perintah code sedangkan Microsoft Visio adalah *software* untuk membuat *paper flow diagram, DFD.*

3.5.2 Kebutuhan perangkat keras

Sistem perangkat keras (*Hardware*) adalah komponen-komponen pendukung kinerja dari sistem komputer. Komponen-komponen yang dapat dipakai untuk menjalankan aplikasi deteksi kemiripan dokumen adalah sebagai berikut:

1. Prosesor Intel Core^{tm} i3-380M
2. Memory RAM 512 MB atau lebih

3. Monitor VGA atau SVGA dengan resolusi 800x 600 atau lebih
4. Hardisk minimal 40 GB atau lebih
5. Mouse
6. Keyboard
7. Printer

3.6 Skenario Pengujian

Skenario pengujian aplikasi deteksi kemiripan dokumen menggunakan text mining dengan metode cosine similarity difokuskan kepada hasil dari perhitungan similarity dengan batasan threshold tertentu yang nantinya dari hasil nilai similarity dokumen dibuktikan dengan letak kemiripan kata yang sama dari dokumen yang sedang dibandingkan tersebut. Scenario pengujian dilakukan dengan langkah-langkah sebagai berikut :

1. Menginputkan dokumen yang telah dipilih baik sebagai dokumen sumber ataupun dokumen pembanding.
2. Dokumen teks yang digunakan untuk melakukan skenario pengujian merupakan abstrak skripsi teknik informatika periode 2012 - 2015
3. Dokumen yang dinputkan bisa berupa teks bersumber dari internet ataupun dokumen lainnya dalam bentuk format ekstensi .docx dan .pdf.
4. Pada dokumen yang telah dipilih kemudian dilakukan tahap preprocessing data menggunakan text mining.dan melakukan perhitungan nilai similarity dengan metode cosine similarity.
5. Hasil nilai similarity menjadi acuan dengan dibuktikan dengan letak kemiripan kata dengan cara sebagai berikut :
    1. *Include* seluruh kata dokumen pembanding dari database
    2. Jadikan sebagai *array*
    3. Cari kata kedalam dokumen sumber dengan melakukan pencocokan kata perkata atau biasa disebut fungsi *regex.*
6. Melakukan analisis terhadap hasil uji coba.

Analisa dilakukan dengan mencatat hasil nilai similarity dokumen dan kemudian dilakukan perhitungan persantase nilai similarity dokumen.