BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Stemming merupakan sebuah cara untuk menghilangkan imbuhan seperti berupa awalan dan akhiran agar didapatkan kata dasar (*root word*) (Novitasari, 2016). Proses *stemming* memiliki aturan atau *algoritma* yang berbeda dalam menghilangkan imbuhan disetiap bahasanya, seperti bahasa inggris memiliki perbedaan aturan penggunaan tata bahasa dengan bahasa indonesia.

. Pada *algoritma stemming* Nazief & Adriani tidak semua kata yang memiliki imbuhan bisa diselesaikan, ada beberapa kata yang mengalami pemenggalan imbuhan yang lebih (*overstemming*) atau pemenggalan imbuhan yang terlalu sedikit (*understemming*). Maka diperlukan sebuah sistem dengan penambahan *algoritma* untuk mengatasi kelemahan dari *algoritma stemming* Nazief & Adriani. *Algoritma* yang ditambahkan adalah metode *N-Gram & SDC* (*Sorensen Dice Coefficient*), jadi hasil kata yang mengalami pemenggalan imbuhan yang lebih (*overstemming*) atau pemenggalan imbuhan yang terlalu sedikit (*understemming*) setelah proses *prepocessing* akan diproses kembali dengan *N-Gram & SDC* (*Sorensen Dice Coefficient*) untuk menemukan relevansi kata dasar.

3.2 Hasil Analisis

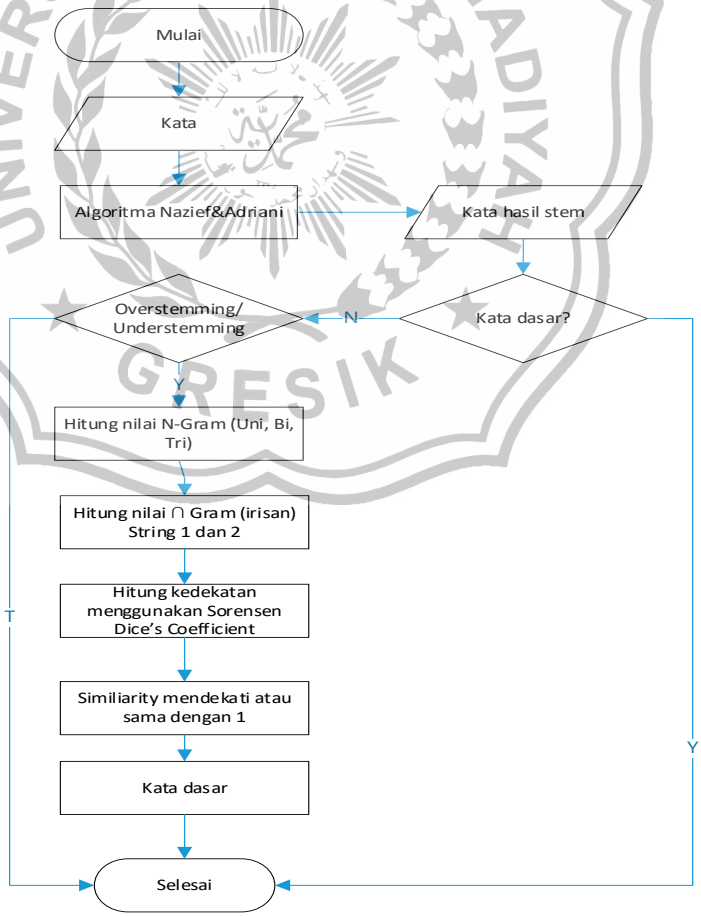
Hasil analisis yang dapat dilakukan dari modifikasi Nazief & Adriani *stemmer* untuk bahasa indonesia yaitu didapatkannya perubahan kata yang mengalami *overstemming* dan *understemming* pada proses *stemming* Nazief & Adriani sehingga didapatkan sebuah kata dasar menurut kamus yang ada di *database*. Modifikasi ini menggunakan metode *N-Gram & SDC*. Penggunaan metode ini dianggap mampu mengatasi kelemahan dari *algoritma stemming* Nazief & Adriani. Dengan menggunakan metode ini, *string* yang akan diukur yaitu *string* kata yang dihasilkan oleh *algoritma*

stemming Nazief & Adriani dengan kamus kata dasar yang tersimpan dalam *database*. Setelah diukur relevansi kata dasar yang memiliki nilai mendekati 1 dianggap memiliki kesamaan kata dengan kata yang sebelumnya di *stemming* dengan *algoritma stemming* Nazief & Adriani.

3.3 Perancangan sistem

3.3.1 *Flowchart* Sistem

Sistem yang akan dibangun adalah sistem yang dapat merubah kata yang telah mengalami *overstemming* dan *understemming* pada proses *stemming* Nazief & Adriani sehingga didapatkan relevansi kata dasar menurut kamus yang ada di *database*. Modifikasi ini menggunakan metode *N-Gram & SDC*. Gambaran umum sistem yang akan dibangun seperti pada gambar 3.1.



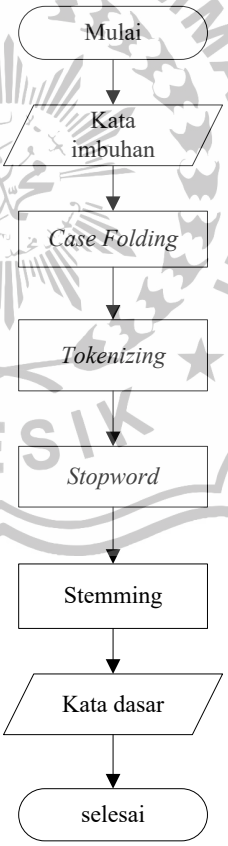
Gambar 3. 1 Gambaran umum sistem yang akan dibangun

3.3.1.1 Kata Imbuhan

Kata yang diinputkan adalah kata yang memiliki imbuhan seperti awalan (*prefix*), sisipan (*infix*), akhiran (*suffix*), dan gabungan awalan akhiran (*confixes*). Seperti contoh kata Keliaran, Membeku, Memberikan, dll

3.3.1.2 Prepocessing

Secara umum sistem ini memiliki beberapa tahapan proses, yang pertama adalah *preprocessing.* Tahap *preprocessing* mencakup berbagai proses seperti *case folding, tokenizing, stopword* dan *stemming*. Alur proses dalam tahap *preprocessing* seperti pada gambar 3.2



Gambar 3. 2 *Flowchart* tahap *preprocessing*

Pada tahap *preprocessing* ada beberapa proses yang akan dilakukan. Pertama yaitu memasukan kata yang memiliki imbuhan.

Setelah kata masukkan maka kata tersebut akan dilakukan proses *case folding* yakni kata yang dimasukkan akan dirubah menjadi huruf kecil (*lowercase*). Selanjutnya proses *tokenizing* tahap pemotongan *string* inputan berdasarkan kata yang menyusunnya. Lalu proses *stopword* yaitu pembuangan kata yang tidak memiliki makna dan terakhir *stemming* yaitu penghilangan imbuhan pada kata sehingga menjadi kata dasar.

3.3.1.2.1 *Case folding*

Tahap ini merupakan tahap dimana kata yang dimasukkan pertama diproses. *Case folding* merupakan proses penyamaan huruf dengan mengubahnya menjadi huruf kecil (*lowercase*). Pada kata yang dimasukkan terdapat huruf besar (kapital) dan kecil. Perbedaan bentuk huruf pada kata tersebut akan menganggu proses selanjutnya maka dari itu diperlukan proses *case folding* untuk merubah kata menjadi huruf kecil semua.

Tabel 3.1 Contoh proses *case folding*

Input	**Output**
Berlari	berlari
Meraba	meraba
Membeku	membeku

3.3.1.2.2 *Tokenizing*

Proses *tokenizing* adalah tahap pemotongan *string* inputan berdasarkan kata yang menyusunnya. Pada prinsipnya proses ini adalah memisahkan setiap kata pada suatu kalimat. Setiap kata terindentifikasi mengandalkan spasi pada setiap kata akan dilakukan pemisahan kata.

Tabel 3.2 Contoh Proses *tokenization*

Input	**Output**
bertanggung jawab	bertanggung, jawab
air mata	air , mata
Membeku	membeku

3.3.1.2.3 *Stopword*

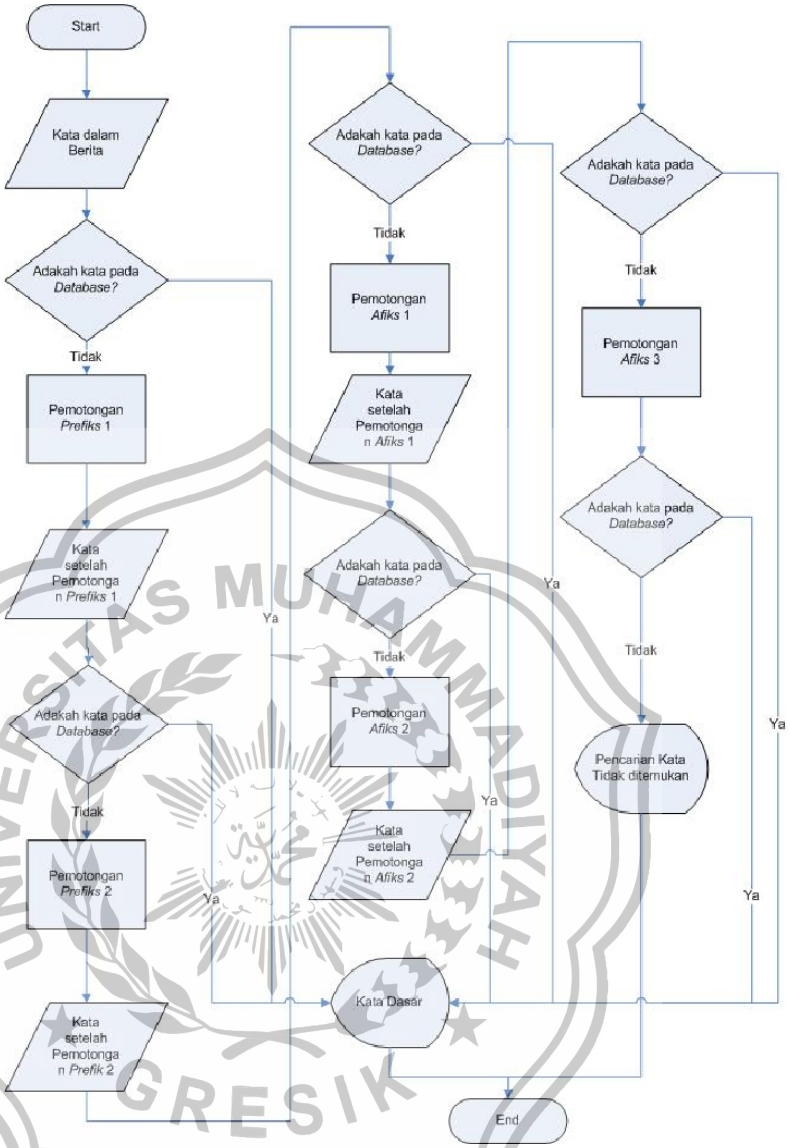
Stopword removal adalah proses menghilangkan kata-kata yang tidak memiliki arti seperti kata "yang", "pada", "itu" dan lain sebagainya.

Tabel 3.3 Contoh Proses *stopword*

Input	**Output**
Yang	(dihapus)
Pada	(dihapus)
Membeku	membeku

3.3.1.2.4 *Stemming*

Proses *stemming* merupakan proses pencarian kata dasar terhadap sebuah kata yang telah dilakukan proses *case folding*. Pada proses ini kata akan dirubah menjadi kata dasar sehingga kata yang memiliki imbuhan seperti awalan (*prefix*), sisipan (*infix*), akhiran (*suffix*), dan gabungan awalan akhiran (*confixes*) akan dihapus dan disesuaikan kata dasarnya. Proses *stemming* menggunakan algoritma Nazief & Adriani. Alur proses dalam *stemming* seperti pada gambar 2.1.



Gambar 3. 3 Alur proses *stemming*

Proses tahapan *stemming* Nazief & Adriani yang terjadi pada gambar 3.3 :

1. Cari kata yang akan di*stemming* dalam kamus kata dasar. Jika ditemukan maka diasumsikan kata adalah kata dasar. Maka algoritma berhenti.
2. Hilangkan *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). Jika berupa *particles* (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.

3. Hilangkan *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. Hilangkan *Derivation Prefix* (“di-”, “ke-”, “se-”, “me-”, “be”, “pe-”, “te-”) dengan iterasi maksimum sebanyak 3 kali .
    a. Langkah 4 berhenti jika :
        1. Terjadi kombinasi awalan dan akhiran.
        2. Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
        3. Tiga awalan telah dihilangkan.

        Tabel 3.4 Kombinasi Awalan-Akhiran yang tidak diizinkan Awalan Akhiran yang tidak diizinkan

Awalan	**Akhiran yang tidak diizinkan**
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

    b. Identifikasi tipe awalan dan hilangkan. Tipe awalan ada 2 yaitu:
        1. Standart : “di-”, “ke-”, “se-” dapat langsung dihilangkan dari kata

2. Kompleks : “me-”, “be”, “pe-”, “te-” adalah tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 3.4 untuk mendapatkan pemenggalan yang tepat.

c. Cari kata yang telah dihilangkan awalannya ini di dalam kamus. Apabila tidak ditemukan, maka langkah 4 diulangi kembali. Apabila ditemukan maka keseluruhan proses selesai.

5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses *recoding* dilakukan dengan mengacu pada aturan pada Tabel 2.3. *Recoding* dilakukan dengan menambahkan karakter *recoding* di awal kata yang dipenggal. Karakter *recoding* adalah huruf kecil setelah tanda hubung (‘-’) dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata “menangkap” (aturan 15), setelah dipenggal menjadi “nangkap”. Karena tidak *valid*, maka *recoding* dilakukan dan menghasilkan kata “tangkap”.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai kata dasar. Proses selesai.

Contoh hasil proses *stemming* yang dilakukan dengan menggunakan *algoritma stemming* Nazief & Adriani dengan kata “mengecek” dilihatkan pada tabel 3.5

Tabel 3.5 Contoh *stemming* kata “membeku”

Algoritma *stemming* Nazief & Adriani	Hasil *Stemming*	Keterangan
Cari kata dikamus kata dasar	membeku	Tidak ditemukan kata pada kamus kata dasar
Hilangkan partikel (“-lah”,	membeku	Tidak terdapat

"-kah", "-tah" atau "-pun")		Partikel
Hilangkan kata ganti ("-ku", "-mu", atau "-nya"),	membe	Terdapat akhiran "-ku"
Hilangkan akhiran ("-i", "-an" atau "-kan")	membe	Tidak Terdapat akhiran
Hilangkan awalan ("di-", "ke-", "se-", "me-", "be", "pe-", "te-")	mbe	Terdapat awalan "-me"
Recoding	mbe	Pada aturan 17 tabel 2.3 dilakukan recoding pada kata membeku, setelah dilakukan proses, hasil masih "mbe" dan dicek pada kamus tidak valid sehingga tetap menghasilkan kata "membeku"
Hasil *stemming*	membeku	

3.3.1.3 *Overstemming* atau *Understemming*

Overstemming dan *Understemming* adalah sebuah kesalahan dan kekurangan dari proses *stemming*. Kata yang mengalami *Overstemming* atau *Understemming* tidak dapat ditemukan pada kamus kata dasar. Kata yang didapatkan bisa mengalami pemenggalan imbuhan yang lebih (*overstemming*) atau pemenggalan imbuhan yang terlalu sedikit (*understemming*). Seperti pada kata "membeku" hasil

dari *stemming*nya adalah "membeku". Jika kita lihat, kata " membeku" bukanlah sebuah kata dasar. Kata "membeku" mengalami *understemming* yaitu pemenggalan imbuhan yang terlalu sedikit, dimana pengguna bisa menganalisis kata dasar dari kata "membeku" adalah "beku" dari kata imbuhan perulangan "membeku". Namun, apabila kata imbuhan yang melalui proses *stemming* ditemukan kata dasarnya maka, tidak diperlukan metode *N-Gram Technique* dan *Sorensen-Dice Coefficient* untuk mendapatkan relevansi kata dasar.

3.3.1.4 Hitung Jarak Kedekatan Kata Dengan Metode *Sorensen-Dice Coefficient*

Penelitian ini menggunakan metode *Sorensen-Dice Coefficient* ditemukan oleh Throvald Sorensen dan Lee Raymond Dice dalam proses mengukur kedekatan antar kata. Alur proses dalam perhitungan kedekatan dengan *Sorensen-Dice* seperti pada gambar 2.1.

Proses tahapan metode *Sorensen-Dice Coefficient* yang terjadi pada gambar 2.1:

1. Dimulai dengan 2 *string* sebagai inputan *string* 1 didapatkan dari hasil *prepocessing* dan *string* 2 didapatkan dari kamus kata dasar. Contoh kasus permasalahan dari hasil *stemming algoritma* Nazief & Adriani yaitu kata "membeku". Penganalisisan pengguna pada kata "membeku" adalah bukan sebuah kata dasar, melainkan kata yang mengalami *understemming* yaitu pemenggalan imbuhan yang terlalu sedikit. Dilihat dari kata imbuhan awal yaitu kata "membeku", kata hasil *stemming* yang dihasilkan "membeku seharusnya adalah "beku". Karena kesalahan dari *algoritma* Nazief & Adriani kata tersebut masih memiliki imbuhan. Maka dari itu "membeku" dianggap sebagai *string* 1 dan untuk *string* 2 adalah kata yang berada pada kamus kata dasar. Sebagai contoh kata yang diambil dari kamus kata dasar sebanyak 30 kata ditampilkan pada tabel 3.6 sebagai berikut:

Tabel 3.6 Tabel kamus kata dasar

No	Kamus Kata
1	Lembek
2	Bekuk
3	Embel
4	Membal
5	Tembekar
6	Beku
7	Memble
8	Membran
9	Bebek
10	Bekal
11	Bekam
12	Elok
13	Foli
14	Folio
15	Gol

No	Kamus Kata
16	Golok
17	Golong
18	Holofit
19	Holofrasis
20	Holosen
21	Jolor
22	Jolok
23	Kologen
24	Kolofon
25	Kolom
26	Kolong
27	Toleh
28	Tolol
29	Toleransi
30	Tolong

2. Setelah ditentukan *N-Gram* proses selanjutnya yaitu menghitung *N-Gram* meliputi (*Tri-Gram, Four-Gram, Five-Gram*) atau huruf ejaan yang ada pada kata yang akan diproses. Pada proses ini juga sudah dimulai perbandingan antara *string* 1 dan *string* 2

String 1 => membeku: Tri-Gram = 5
Four-Gram = 4
Five-Gram = 3
String 2 => beku: Tri-Gram = 2
Four-Gram = 1
Five-Gram = 0

Tabel 3.7 Nilai Gram Pada *String* 1 dan *String* 2

Perbandingan *N-Gram*	*Tri-Gram*		*Four-Gram*		*Five-Gram*	
membeku : lembek	5	4	4	3	3	2
membeku : bebek	5	3	4	2	3	1
membeku : bekal	5	3	4	2	3	1
membeku : bekam	5	1	4	0	3	0
membeku : beku	5	2	4	1	3	0
membeku : bekuk	5	3	4	2	3	1
membeku : elok	5	2	4	1	3	0
membeku : embel	5	3	4	2	3	1
membeku : foli	5	2	4	1	3	0
membeku : folio	5	3	4	2	3	1
membeku : gol	5	1	4	0	3	0
membeku : holofit	5	5	4	4	3	3
membeku : holofrasis	5	8	4	7	3	6
membeku : holosen	5	5	4	4	3	3
membeku : jolok	5	3	4	2	3	1
membeku : jolor	5	3	4	2	3	1
membeku : kolofon	5	5	4	4	3	3
membeku : kologen	5	5	4	4	3	3
membeku : kolom	5	3	4	2	3	1

membeku : kolong	5	4	4	3	3	2
membeku : membal	5	4	4	3	3	2
membeku : membeku	5	7	4	6	3	5
membeku : memble	5	3	4	2	3	1
membeku : membran	5	4	4	3	3	2
membeku : tembekar	5	6	4	5	3	4
membeku : toleh	5	3	4	2	3	1
membeku : tolol	5	3	4	2	3	1
membeku : tolong	5	4	4	3	3	2
membeku :golok	5	3	4	2	3	1
membeku :golong	5	4	4	3	3	2

3. Setelah panjang *N-Gram* ditemukan pada rumus 2.1, maka selanjutnya adalah mencari nilai ∩ yaitu mencari irisan atau kata unik yang sama antara *string* 1 dengan *string* 2 berdasarkan nilai Gram (*Tri-Gram, Four-Gram, Five-Gram*).

   string 1 => membeku : ∩ *Tri-Gram*= 2

   ∩ *Four-Gram* = 1

   string 2 => beku ∩ *Five-Gram* = 0

   Pada perbandingan dua *string* tersebut diketahui masing-masing memiliki irisan ∩ atau karakter yang mewakili di salah satu kata tersebut.

Tabel 3.8 Jumlah *Gram* sama (∩) irisan

Perbandingan String	∩ Tri-Gram	∩ Four-Gram	∩ Five-Gram
membeku : lembek	3	2	1
membeku : bebek	1	0	0
membeku : bekal	1	0	0

membeku : bekam	1	0	0
membeku : beku	2	1	0
membeku : bekuk	2	1	0
membeku : elok	0	0	0
membeku : embel	1	1	0
membeku : foli	0	0	0
membeku : folio	0	0	0
membeku : gol	0	0	0
membeku : holofit	0	0	0
membeku : holofrasis	0	0	0
membeku : holosen	0	0	0
membeku : jolok	0	0	0
membeku : jolor	0	0	0
membeku : kolofon	0	0	0
membeku : kologen	0	0	0
membeku : kolom	0	0	0
membeku : kolong	0	0	0
membeku : membal	2	1	0
membeku : membeku	0	0	0
membeku : memble	2	1	0
membeku : membran	2	1	0
membeku : tembekar	3	2	1
membeku : toleh	0	0	0
membeku : tolol	0	0	0
membeku : tolong	0	0	0
membeku :golok	0	0	0
membeku :golong	0	0	0

4. Proses selanjutnya yaitu menghitung nilai *Similiarity* menggunakan *SDC* atau nilai *Sorensen-Dice Coefficient,* yaitu hitung jumlah *N-gram* unik dan memiliki struktur yang sama dari masing-masing teks yang dibandingkan. Menggunakan rumus pada rumus 2.2.

   S membeku/beku: *SDC* *Tri-Gram* = 0.5714

   SDCFour-Gram = 0.4

   SDC Five-Gram = 0

   Kata membeku dan beku memperoleh hasil tersebut diatas, dan setelah dilakukan perbandingan secara manual melalui kamus masih belum menemukan hasil yang valid, maka dilakukan lagi proses pencarian S *similiarity* terhadap kata-kata yang terdapat dalam kamus program.

   Tabel 3.9 Nilai *Similiarity* (S)

Perbandingan *Similiarity String* 1 : *string* 2	*SDC N3-Gram*	*SDC N4-Gram*	*SDC N5-Gram*
membeku : lembek	0.6666	0.5714	0.4
membeku : bebek	0.25	0	0
membeku : bekal	0.25	0	0
membeku : bekam	0.3333	0	0
membeku : beku	0.5714	0.4	0
membeku : bekuk	0.5	0.3333	0
membeku : elok	0	0	0
membeku : embel	0.25	0.3333	0
membeku : foli	0	0	0
membeku : folio	0	0	0
membeku : gol	0	0	0
membeku : holofit	0	0	0
membeku : holofrasis	0	0	0

membeku : holosen	0	0	0
membeku : jolok	0	0	0
membeku : jolor	0	0	0
membeku : kolofon	0	0	0
membeku : kologen	0	0	0
membeku : kolom	0	0	0
membeku : kolong	0	0	0
membeku : membal	0.4444	0.2857	0
membeku : membeku	0	0	0
membeku : memble	0.5	0.3333	0
membeku : membran	0.4444	0.2857	0
membeku : tembekar	0.5454	0.4444	0.2857
membeku : toleh	0	0	0
membeku : tolol	0	0	0
membeku : tolong	0	0	0
membeku :golok	0	0	0
membeku :golong	0	0	0

3.3.1.5 *Similarity* mendekati 1 atau *similarity* sama dengan 1

Setelah proses mencari nilai *SDC* setiap kata. Maka proses selanjutnya adalah mengurutkan nilai dari yang terbesar sampai yang terkecil untuk dicari kata yang paling mendekati *string* 1 berdasarkan nilai *N-Gram* yang telah diketahui. Kata yang mendekati *string* 1 yaitu kata yang mendekati nilai 1 atau sama dengan 1 berdasarkan hasil.

Tabel 3.10 Pengurutan kata dari yang nilai terbesar

Pengurutan String 1 : String 2	*SDC N3-Gram*	*SDC N4-Gram*	*SDC N5-Gram*
membeku : lembek	0.6667	0.5714	0.4
membeku : beku	0.5714	0.4	0
membeku : tembekar	0.5454	0.4444	0.2857
membeku : bekuk	0.5	0.3333	0
membeku : memble	0.5	0.3333	0
membeku : membal	0.4444	0.2857	0
membeku : membran	0.4444	0.2857	0
membeku : bekam	0.3333	0	0
membeku : bebek	0.25	0	0
membeku : bekal	0.25	0	0
membeku : embel	0.25	0.3333	0
membeku : elok	0	0	0
membeku : foli	0	0	0
membeku : folio	0	0	0
membeku : gol	0	0	0
membeku : holofit	0	0	0
membeku : holofrasis	0	0	0
membeku : holosen	0	0	0
membeku : jolok	0	0	0
membeku : jolor	0	0	0
membeku : kolofon	0	0	0
membeku : kologen	0	0	0
membeku : kolom	0	0	0
membeku : kolong	0	0	0

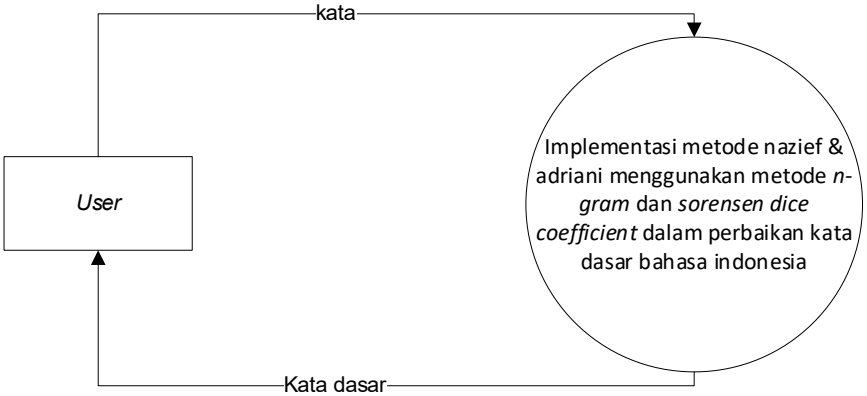
membeku : membeku	0	0	0
membeku : toleh	0	0	0
membeku : tolol	0	0	0
membeku : tolong	0	0	
membeku :golok	0	0	0
membeku :golong	0	0	0

3.3.1.6 Kata dasar

Proses terakhir yaitu didapatkan relevansi kata dasar. Pada kata imbuhan "Membeku" yang diproses *stemming* menjadi "membeku" selanjutnya dibandingkan dengan kamus kata dasar dengan menggunakan metode *N-Gram* dan *Sorensen Dice's Similiarity* didapatkan relevansi kata dasar pada N-ke3 dan N-ke4 yaitu kata "beku", dapat dilihat pada tabel 3.13 pada pengurutan nilai *Sorensen Dice's Similiarity* kata "beku" mempunyai nilai relevansi terhadap N-ke3 dan N-ke4 sebesar 0.5714 dan 0.4.

3.3.2 Diagram Konteks

Diagram konteks merupakan diagram yang menunjukan sebuah proses tunggal dalam sistem yang berhubungan dengan bagian yang terkait. Rangkaian diagram konteks yang digunakan pada penelitian ini seperti pada gambar 3.4

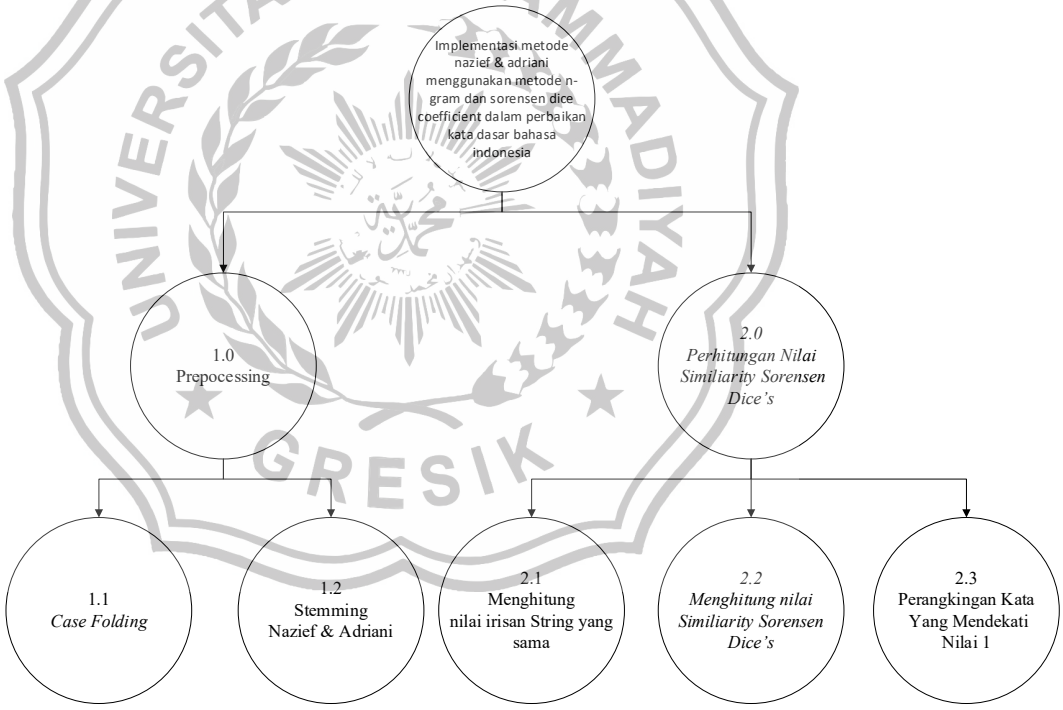


Gambar 3.4 Diagram Konteks Modifikasi Nazief & Adriani *Stemmer*

Dari gambar 3.5 tersebut menggambarkan bahwa melibatkan satu pihak. User mengirimkan *input* berupa kata yang memiliki imbuhan seperti awalan (*prefix*), sisipan (*infix*), akhiran (*suffix*), dan gabungan awalan akhiran (*confixes*) yang digunakan sebagai data yang akan diproses. Setelah didapatkan hasil nilai kesamaan kata *(string)* maka *output* atau keluaran dari sistem berupa relevansi kata dasar yang paling mendekati.

3.3.3 Diagram Berjenjang

Diagram berjenjang sangat diperlukan dalam perancangan semua proses yang ada. Diagram berjenjang merupakan pengambaran proses dari awal sampai ke level-level berikutnya. Dalam penelitian Modifikasi Nazief & Adriani *Stemmer* ini mempunyai tiga level seperti pada gambar 3.5



Gambar 3.5 Diagram berjenjang di Modifikasi Nazief & Adriani *Stemmer*

Berikut penjelasan gambar 3.6 berdasarkan kerangka diagram berjenjang diatas terlihat bahwa sistem yang dibuat terdiri dari dua level

1. Top level : perbaikan metode *nazief&adriani* menggunakan kombinasi metode *N-Gram* dan *Sorensen-Dice Coefficient*
2. Level 0 : merupakan hasil *break down* dari proses keseluruhan dari perbaikan metode *Nazief&Adriani* menggunakan kombinasi metode

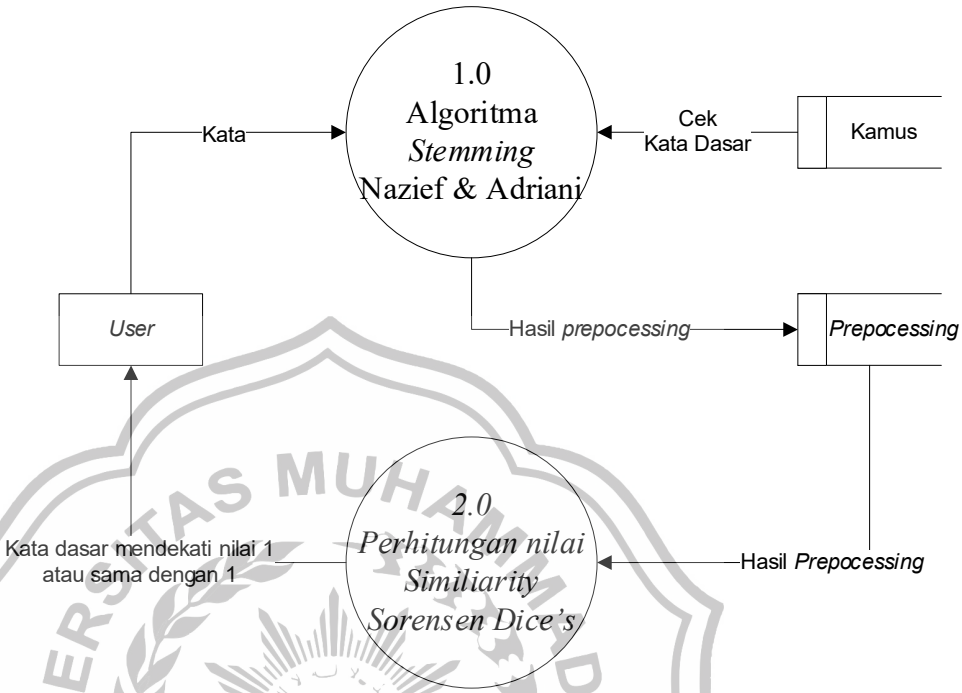
N-gram dan *Sorensen-Dice Coefficient* menjadi beberapa sub proses yaitu:

a. *Prepocessing*
b. Perhitungan nilai *Sorensen-Dice Coefficient*

3. Level 1 : merupakan sub proses dari beberapa proses pada level 0 dalam perbaikan metode *Nazief&Adriani* menggunakan kombinasi metode *N-gram* dan *Sorensen-Dice Coefficient* yang menggambarkan beberapa proses detail yaitu :
    1. Hasil dari sub proses *Algoritma Stemming* Nazief & Adriani :
        a. *Case folding*
        b. *Stemming* Nazief & Adriani
    2. Hasil dari sub proses Perhitungan nilai *Sorensen-Dice Coefficient*:
        a. Menghitung nilai irisan *String* yang sama
        b. Menghitung nilai *Sorensen-Dice Coefficient*
        c. Perangkingan kata yang mendekati nilai 1

3.3.4 Data Flow Diagram

3.3.4.1 Data Flow Diagram Level 0



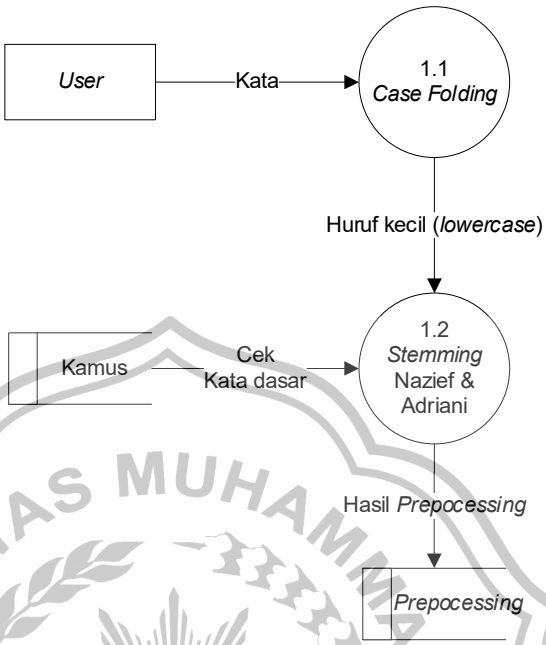
Gambar 3.6 Data Flow Diagram level 0 Modifikasi Nazief & Adriani *Stemmer*

Berdasarkan pada gambar 3.6, dapat dijelaskan bahwa DFD level 0 menjelaskan beberapa proses yang terjadi. Beberapa proses yang ada pada DFD level 0 antara lain :

a. *Prepocessing*
b. Perhitungan nilai *Sorensen-Dice Coefficient*

3.3.4.2 Data Flow Diagram Level 1

1) DFD Level 1 Proses 1

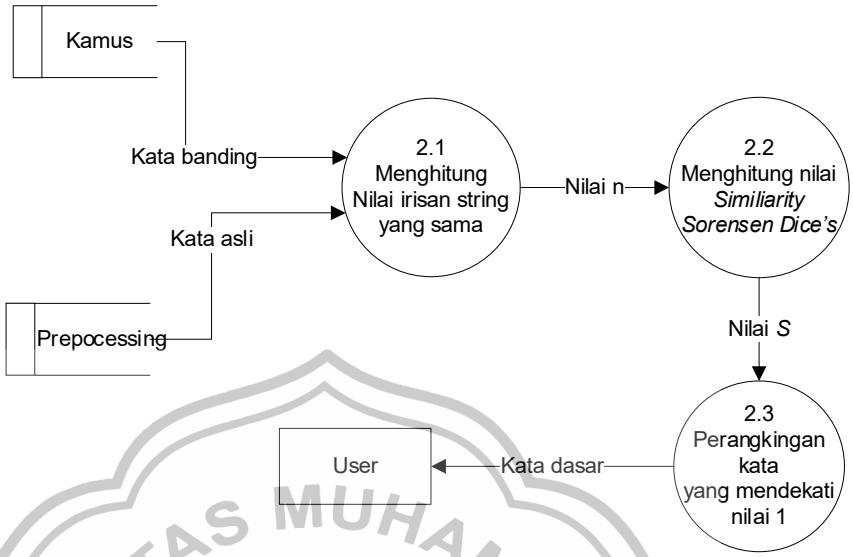


Gambar 3.7 Data Flow Diagram level 1 Proses 1 Modifikasi Nazief & Adriani *Stemmer*

Berdasarkan pada gambar 3.7 dapat dijelaskan bahwa DFD Level 1 proses 1 menjelasakan dua proses detail dari proses *prepocessing* yakni :

a. *Case Folding*
b. *Stemming* Nazief & Adriani

2) DFD Level 1 Proses 2



Gambar 3.8 Data Flow Diagram level 1 Proses 2 Modifikasi Nazief & Adriani *Stemmer*

Berdasarkan pada gambar 3.8 dapat dijelaskan bahwa DFD Level 1 proses 2 menjelasakan tiga proses detail dari proses perhitungan *Sorensen-Dice Coefficient* yakni :

a. Menghitung nilai irisan string
b. Menghitung nilai *Sorensen-Dice Coefficient*
c. Perangkingan kata yang mendekati nilai 1

3.4 Perancangan Basis Data

Basis data adalah kumpulan file-file yang mempunyai kaitan antara satu file dengan file lain sehingga membentuk suatu bangunan data untuk menginformasikan suatu hasil dari sebuah proses. Berikut untuk struktur dan desain tabel dari *database* yang digunakan dalam proses pembuatan sistem perbaikan metode *Nazief&Adriani* menggunakan kombinasi metode *N-gram* dan *Sorensen-Dice Coefficient.*

3.4.1 Desain Tabel

Desain tabel merupakan susunan dari tabel yang akan digunakan atau diimplementasikan kedalam database, dimana desain tabel memuat detail tipe data dan *primary key* serta *foreign key* dari tabel tersebut.

1. Tabel Kamus Kata Dasar

Tabel Kata dasar berisi kumpulan kata dasar dari sebuah kata berbahasa indonesia yang nantinya akan digunakan pada proses *stemming* dan juga sebagai kata sumber yang nanti akan menjadi pembanding untuk kata pembanding pada proses metode *Sorensen Dice's Similiarity.* Struktur tabel kamus kata dasar seperti pada tabel 3.11

Tabel 3.11 Tabel Kata Dasar

Nama Kolom	Tipe	Ukuran	Keterangan
Id_katadasar	int	15	*Primary key*
Kata_dasar	varchar	50	
Tipe_kata	varchar	30	

2. Tabel kata

Tabel kata merupakan tempat untuk menyimpan kata sebelum dilakukan proses *prepocessing*. Struktur tabel kata seperti pada tabel 3.12

Tabel 3.12 Tabel Kata

Nama Kolom	Tipe	Ukuran	Keterangan
Id_kata	int	15	*Primary key*
Kata_imbuhan	varchar	50	

3. Tabel *prepocessing*

Tabel *prepocessing* merupakan tempat kata yang telah melalui tahapan proses *prepocessing* dan menjadi kata pembanding untuk

dicari nilai *N-Gram* dan *SDC*. Struktur tabel *preppocessing* seperti pada tabel 3.13

Tabel 3.13 Tabel *Prepocessing*

Nama Kolom	Tipe	Ukuran	Keterangan
Id_proses	int	15	*Primary key*
Id_kata	int	15	*Foreign key*
Jumlah_token	int	20	
Case_fold	Var	50	

4. Tabel *Stemming*

Tabel *Stemming* merupakan tempat penyimpanan kata hasil *Tokenizing* dan *Case folding* yang telah dilakukan proses menggunakan metode *Nazief&Adriani*. Struktur tabel *stemming* seperti pada tabel 3.14

Tabel 3.14 Tabel *Stemming*

Nama Kolom	Tipe	Ukuran	Keterangan
Id_stemming	Int	20	*Primary Key*
Id_proses	Int	20	*Primary Key*
Hasil_stemming	var	50	

5. Tabel *N-Gram*

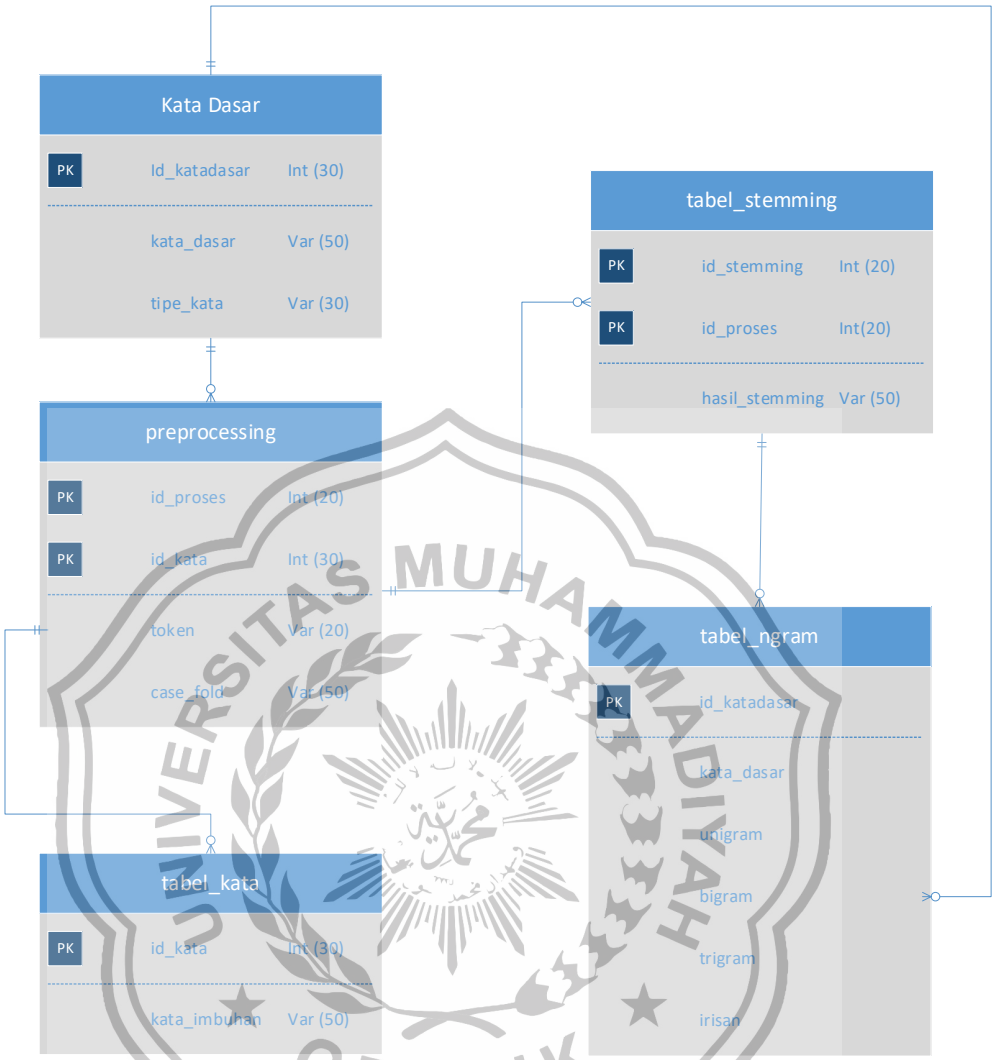
Tabel *N-gram* merupakan tempat penyimpanan kata yang telah melalui tahapan metode *Nazief&Adriani*. Struktur tabel *N-Gram* seperti pada tabel 3.15

Tabel 3.15 Tabel *N-Gram*

Nama Kolom	Tipe	Ukuran	Keterangan
Id_katadasar	int	15	*Primary key*
Katadasar	varchar	50	*Foreign key*
Id_katadasar	int	10	
Trigram	Int	20	
Fourgram	Int	20	
Fivegram	Int	20	
Irisan	int	20	

3.4.2 *Entity Relationship Diagram*

Entity relationship diagram (ERD) merupakan model konseptual yang mengambarkan hubungan antar tabel yang ada. ERD digunakan untuk memodelkan struktur data dan hubungan antar data. Desain *entity relationship diagram* pada pembuatan implementasi modifikasi nazief & Adriani seperti pada gambar 3.9



Gambar 3.9 *Entity relationship diagram* dalam Perbaikan Nazief & Adriani *Stemmer*

3.5 Perancangan Antar Muka

Rancangan antarmuka (*interface*) berfungsi sebagai alat komunikasi antara sistem dengan pengguna. Antarmuka akan memberikan informasi berupa tampilan disertai dengan data-data yang diminta oleh pengguna. Dalam penelitian ini desain antarmuka dapat digunakan sebagai media pemasukan kata yang akan diproses dan menampilkan hasil kata yang telah diproses.

3.5.1 Halaman Beranda

Halaman beranda merupakan halaman pertama yang muncul ketika user membuka sistem. Halaman ini berisikan informasi mengenai nama sistem dan kegunaannya. Tampilan rancangan halaman beranda seperti pada gambar 3.10

Gambar 3.10 Tampilan rancangan halaman beranda

3.5.2 Halaman Kamus

Halaman kamus merupakan halaman yang berfungsi untuk menampilkan kumpulan kamus kata dasar. Pada halaman ini terdapat tombol tambah yang berfungsi untuk menambahkan kata dasar. Pada kolom aksi terdapat tombol hapus yang berfungsi untuk menghapus setiap kata. Tampilan rancangan halaman kamus seperti pada gambar 3.11

Gambar 3.11 Tampilan rancangan halaman kamus

3.5.3 Halaman Tambah Kamus

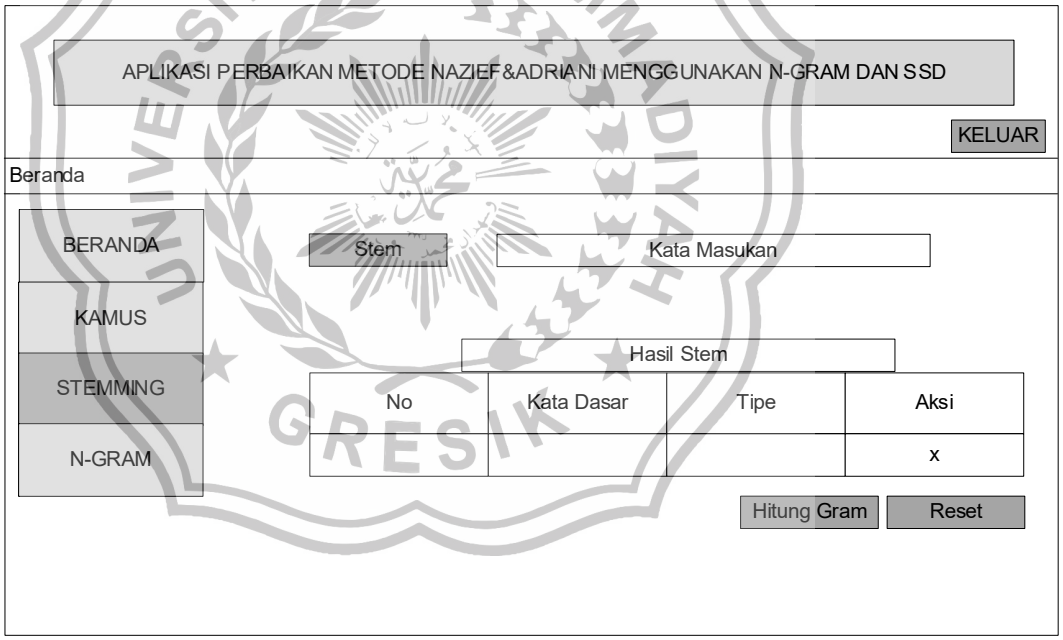
Halaman tambah kamus merupakan halaman yang berfungsi untuk menambahkan kata baru pada kamus. Untuk mengakses halaman ini, pengguna harus menekan tombol tambah pada halaman kamus. Tampilan rancangan halaman tambah kamus seperti pada gambar 3.12

Gambar 3.12 Tampilan rancangan halaman Tambah kamus

Pengguna dapat menginputkan sebuah kata dasar pada kolom masukan kata baru dan dilanjutkan dengan menekan tombol tambah, maka pada kolom jumlah huruf akan secara otomatis menghitung jumlah huruf yang diinputkan pada kolom masukan kata baru. Selanjutnya tekan tombol simpan untuk menyimpan kata pada kamus.

3.5.4 Halaman *stemming*

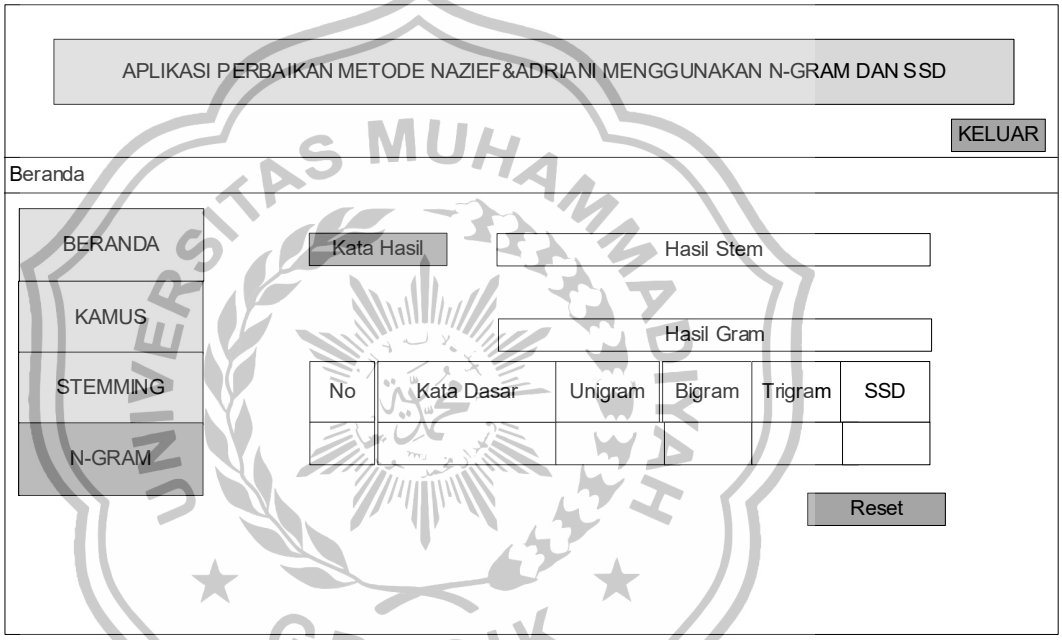
Halaman *stemming* merupakan halaman yang berfungsi untuk meng*input*kan kata yang akan dilakukan proses *stemming* dengan menggunakan metode *Nazief&Adriani*. Pada kolom masukkan kata berfungsi untuk tempat penginputan kata berimbuhan. Jika pengguna menekan tombol proses maka hasil *output* akan terlihat pada kolom hasil *prepocessing*. Jika *output* yang ditampilkan pada kolom hasil *prepocessing* belum mendapatkan seperti yang diinginkan, pengguna bisa memproses hasil stem tersebut dengan menekan tombol "hitung *gram*", maka pada kolom hasil modifikasi akan menampilkan hasil dari metode *N-Gram* dan *Sorensen-Dice Coefficient*. Tampilan rancangan halaman *stemming* seperti pada gambar 3.13



Gambar 3. 13 Tampilan rancangan halaman *stemming*

3.5.5 Halaman *N-Gram*

Halaman *N-Gram* merupakan halaman yang berfungsi untuk melihat hasil dari proses *Sorensen-Dice Coefficient* dan *N-Gram.* Setelah dilakukan *Preprocessing* pada metode *Nazief&Adriani,* maka kata hasil stem tersebut akan dilakukan proses selanjutnya yaitu mencari nilai *N-Gram* pada kata masukan dengan kata dasar yang ada dikamus. Isi dari tabel berisikan nilai yang paling mendekati atau sama dengan 1, kata dikamus yang mendekati ataun sama dengan 1 hasilnya akan ditampilkan pada tabel ini.



Gambar 3. 14 Tampilan rancangan halaman *N-Gram*

3.6 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. *PHP* dan *Sublime* sebagai bahasa pemrograman berbasis *desktop* dan sekaligus *compiler*nya.
3. *SQLyog* sebagai *database server*.
4. *XAMPP Control Panel*

b. Kebutuhan Perangkat Keras

1. Komputer *Intel pentium* 2,0 GHz sekelas atau lebih tinggi
2. RAM 2 GB atau lebih
3. *Hardisk* dengan kapasitas 500 *gigabyte* atau lebih
4. Monitor, *mouse*, *keyboard* standard

3.7 Skenario Pengujian

Pada penelitian ini, untuk mengukur evaluasi kinerja sistem temu kembali informasi digunakan pengujian akurasi.

Tabel 3.16 Parameter menghitung akurasi

Keterangan	**Relevan**	**Tidak Relevan**
Terambil	True (T)	False (F)

Rumus untuk menghitung Akurasi:

$$Akurasi = \frac{RW}{W}\ x\ 100\ \% \qquad (3.1)$$

Nilai akurasi dinyatakan dalam persen. Semakin tinggi nilai tersebut menunjukkan semakin baiknya kinerja aplikasi. Evaluasi yang akan dilakukan dalam penelitian ini adalah menghitung nilai dari akurasi berdasarkan relevansi kata dasar yang ditemukan pada tiap *N-Gram*. Sedangkan untuk menentukan nilai dari akurasi harus didapatkan jumlah kata yang relevan terhadap suatu kata dasar pada kamus

Data yang akan di uji adalah 20 kata berimbuhan pada kata daar bahasa indonesia, dapat di lihat pada tabel berikut ini.

Tabel 3.17 Tabel kata berimbuhan

No.	**Kata Imbuhan**	**No.**	**Kata Imbuhan**
1	memadamkan	11	mengobarkan
2	mengunjungi	12	menerapkan
3	pengawal	13	mengasihani
4	menandakan	14	memobilisasi

5	perubahan	15	memuaskan
6	merubah	16	meragukan
7	penyaringan	17	memesan
8	memberikan	18	mengadakan
9	pemajakan	19	pengertian
10	berisikan	20	dikatakan