BAB 3
ANALISIS MASALAH DAN PERANCANGAN

3.1 *State of the Art*

Pada penelitian sebelumnya sudah ada yang menggunakan metode *Stemming* untuk *preprocessing text* dalam mengolah data pelatihan dan data uji untuk kasus kata yang baku seperti pada tabel 3.1

Tabel 3.1 Analisis penelitian sebelumnya

Penelitian / Fitur	*Stemming*	Bahasa Indonesia	Dokumen
Klasifikasi Dokumen Teks berbahasa Indonesia dengan menggunakan Naive Bayes		✓	✓
Spam Filtering with Naive Bayesian Classication	✓		✓
Bayesian Learning for Automatic Arabic Text Categorization	✓		✓

Dari penelitian yang sudah simpulkan pada tabel 3.1 maka akan dibuat penelitian optimasi akurasi untuk metode Naive Bayes dengan penambahan tahapan Nazief dan Adriani *stemmer* pada *preprocessing text* untuk menangani keberagaman bahasa sehari-hari(kata tidak baku) pada bahasa indonesia.

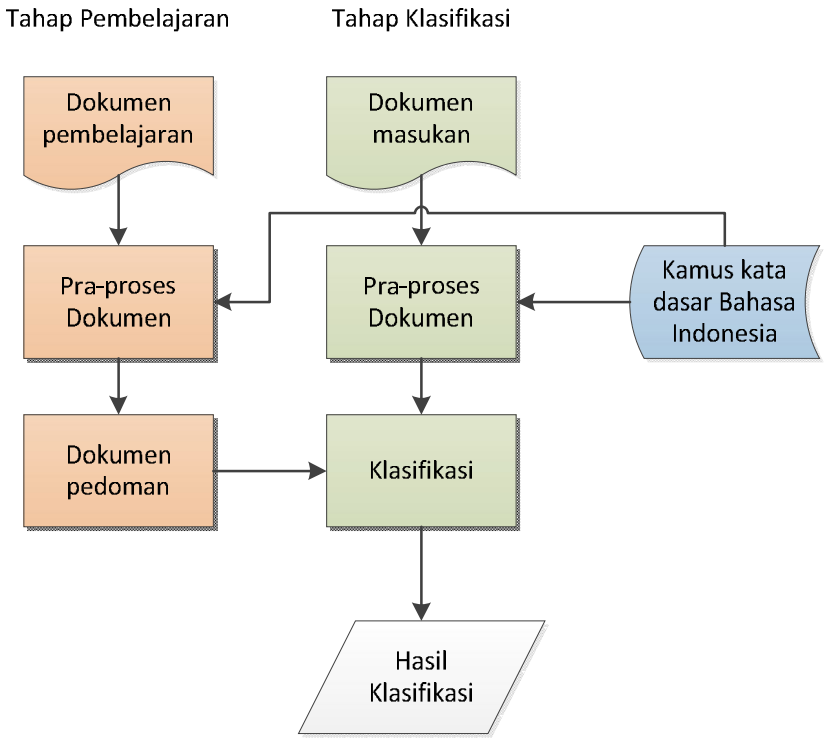
3.2 Analisis Masalah

Beragamnya penggunaan kata dalam bahasa Indonesia seperti kata baku yang berimbuhan (peninggalan, mendukung) dan kata yang tidak baku (tdk, ga, ngga, brp, knp) pada posting twitter menjadi faktor yang mempengaruhi akurasi dalam klasifikasi dokumen. Maka diperlukan optimasi *stemming* untuk menangani kata-kata yang tidak baku

3.3 Arsitektur Sistem

Gambar 3.1 memperlihatkan arsitektur sistem yang akan dikembangkan. Sistem ini menerima masukan berupa data teks dari *posting twitter* untuk

dilakukan tahap pembelajaran dan tahap klasifikasi. Data teks dari *posting twitter* ini akan di praproses terlebih dulu sebelum diproses lebih lanjut.



Gambar 3.1 Arsitektur Sistem Klasifikasi *Posting Twitter*

Text mining mempunyai definisi sebagai menambang data berupa teks dimana sumber data biasanya didapat dari suatu dokumen dengan tujuan mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen. Secara umum sistem ini dibagi menjadi beberapa tahapan proses, pada gambar 3.1 blok diagram proses kerja pada text mining klasifikasi objek wisata sebagai berikut.

1. Optimasi analisis sentimen ini memiliki dua tahap yaitu tahap pembelajaran dan tahap klasifikasi.
2. Tahap pembelajaran harus dijalankan terlebih dahulu dengan melalui pra-proses yaitu *case folding*, *tokenizing*, *stemming* dan *stopwords* untuk menghasilkan dokumen pembelajaran sebagai pedoman untuk setiap kategori yaitu positif negatif dan netral.

3. Tahap pembelajaran memiliki modul yang hampir sama dengan tahap klasifikasi. Bedanya hanya pada tahap pembelajaran tidak menjalankan modul klasifikasi, tetapi hanya menghasilkan dokumen yang mengandung kata-kata untuk mengkarakteristik suatu kategori
4. Setelah memiliki data pembelajaran, maka tahap klasifikasi baru dapat dijalankan.
5. Tahap klasifikasi dapat dispesifikasi mulai dari sistem menerima *input* berupa teks dokumen *posting twitter* sampai menampilkan hasil klasifikasi yaitu positif, negatif dan netral.
6. Modul praproses dokumen dimulai dengan melakukan pembacaan kalimat dari teks dokumen *posting twitter*.
7. Kalimat yang sudah dibaca akan menjalani proses *case folding* dan *tokenizing* sehingga berbentuk kata yang independent.
8. Kata yang mempunyai imbuhan, variasi dan kata yang tidak baku akan masuk ke tahapan *stemming* dengan bantuan kamus kata dasar bahasa Indonesia. Kata yang cocok dengan daftar *stopwords* dihilangkan.
9. Selanjutnya modul Klasifikasi melakukan penghitungan berdasarkan metode *Naive Bayes* untuk menentukan kategori yang akan dipetakan ke dokumen *input* dan menampilkan hasil klasifikasi, kecepatan dan akurasinya.

3.4 Analisis Data Masukan

Data masukan berupa *tweets* diambil dari seluruh *timeline* pengguna *twitter*. Ada beberapa *tweets* mengenai salah satu kandidat capres 2014 yang ada pada *timeline*, untuk penelitian ini data diambil secara random sehingga komposisi *tweet* yang mengandung selain teks dan angka dihapus dengan tahapan *preprocessing teks*. Contoh *tweet* dari pengguna *twitter* mengenai salah satu kandidat capres 2014 :

"Bagaimanapun jg dukung ==> jokowi jd presiden agar INDONESIA hebat \(´▽`)/"

3.5 Praproses dokumen

Tahap praproses dokumen merupakan tahap yang seharusnya dilakukan sebelum metode klasifikasi. Tujuan dari tahap ini adalah menghilangkan *noise*, menyeragamkan bentuk kata dan mengurangi volume kosakata. Tahap ini meliputi *case folding, tokenizing,* penghapusan *stopword* dan *stemming*. Berikut penerapan pada tahap praproses dokumen pada sistem klasifikasi :

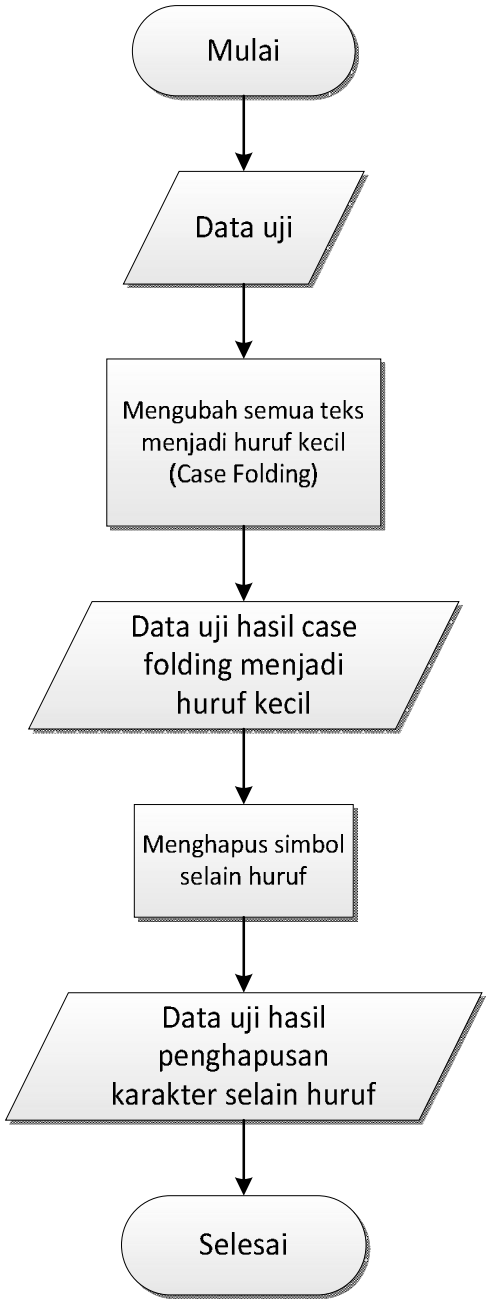
3.5.1 Dokumen *input*

Contoh *tweet* yang akan memasuki tahap *document preprocessing* dapat dilihat di bawah ini

"Bagaimanapun jg dukung ==> jokowi jd presiden agar INDONESIA hebat \(´▽`)/"

3.5.2 *Case folding*

Pada tahap ini dilakukan pengubah huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai dengan 'z' yang diterima. Karakter selain huruf dihilangkan dan dianggap sebagai *delimiter*. Contoh hasil *case folding* terlihat pada gambar 3.2 :



Gambar 3.2 *Flowchart* tahapan *case folding*

Pada tahapan ini, ada beberapa aturan proses agar hasil *case folding* dapat sesuai dengan yang diharapkan. Adapun aturan-aturan tersebut dapat dilihat pada tabel 3.2 :

Tabel 3.2 Aturan tahapan *case folding*

Kondisi	Aksi
Inputan data latih memiliki huruf kapital [A…..Z].	Maka akan mengubah semua inputan tersebut menjadi huruf kecil [a……z] semua.
Inputan data latih memiliki karakter simbol	Maka akan menghapus karakter simbol tersebut dari inputan
Inputan data latih memiliki huruf kecil	Tidak ada aksi
Inputan data latih memiliki spasi	Tidak ada aksi

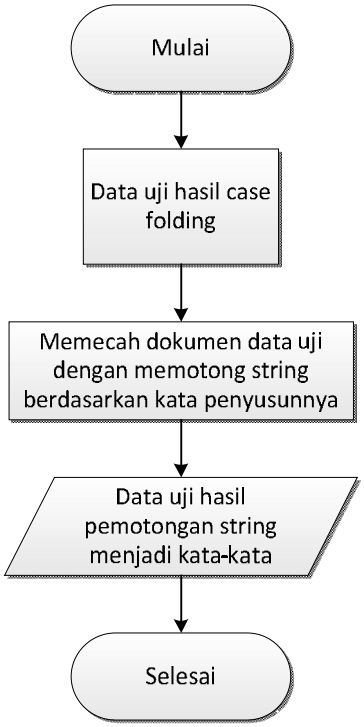
Berikut ini adalah contoh tahapan *case folding* yang akan di ilustrasikan pada tabel 3.3 :

Tabel 3.3 Ilustrasi tahapan *Case folding*

Contoh Data			
Data Uji		**Tahapan *Case Folding***	
Input	*Bagaimanapun jg dukung ➔ jokowi jd presiden agar INDONESIA hebat \(´▽`)/*	*Output*	*bagaimanapun jg dukung jokowi jd presiden agar indonesia hebat*

3.5.3 *Tokenizing*

Tokenizing merupakan proses pemotongan string *input* berdasarkan tiap kata yang menyusunya serta membedakan karakter-karakter tertentu yang dapat diperlakukan sebagai pemisah kata atau bukan. Tahapan ini dilakukan setelah inputan data uji melewati tahap *Case Folding*. Proses *tokenizing* ini mempunyai alur yang digambarkan pada gambar 3.3 sebagai berikut



Gambar 3.3 *Flowchart* Tahapan *Parsing*

Pada tahapan ini dilakukan pemecahan deskripsi dari data latih menjadi bab-bab, paragrap, kalimat, dan menjadi kata-kata dengan memotong string dari penyusunnya. Ada beberapa aturan proses agar hasilnya sesuai dengan yang diinginkan. Adapun aturan-aturan tersebut pada tabel 3.3 :

Tabel 3.4 Aturan tahapan *Parsing*

Kondisi	Aksi
Jika inputan data uji bertemu spasi	Maka akan memecah dari deskripsi data latih menjadi bab-bab per bagian kata atau *string*.
Jika Inputan data latih memiliki huruf.	Tidak ada aksi

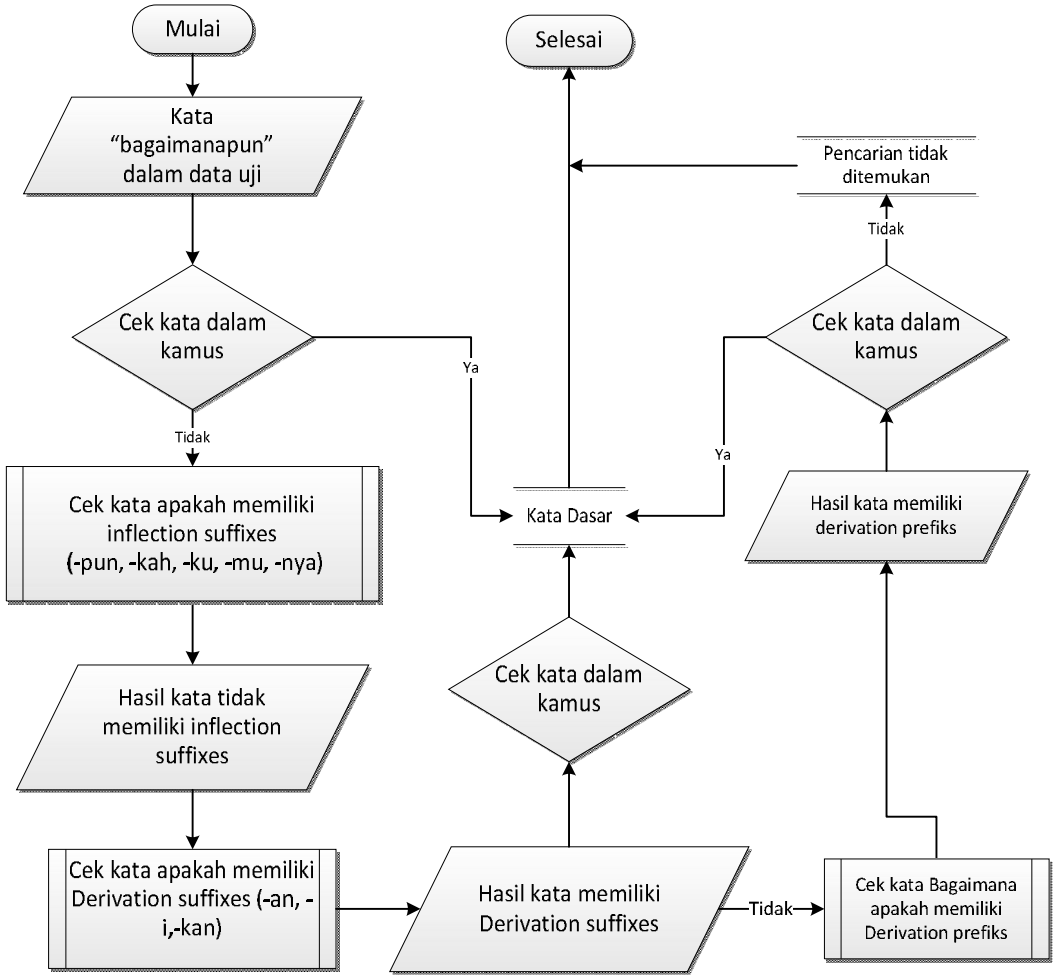
Tabel 3.5 dibawah ini merupakan contoh tahapan *Tokenizing* sebagai berikut :

Tabel 3.5 Ilustrasi *Tokenizing*

Contoh Data			
Data Latih hasil *Case Folding*		**Tahapan *Tokenizing***	
Input	bagaimanapun jg dukung jokowi jd presiden agar indonesia hebat	*Output*	bagaimanapun jg dukung jokowi jd presiden agar indonesia hebat

3.5.4 *Stemming*

Pada tahap ini dilakukan pembuangan imbuhan kata. Misalnya kata “bagaimanapun” maka langkah yang dilakukan stemming dijalankan selanjutnya mengecek kata "bagaimanapun" pada kamus kata dasar, jika tidak ada maka akan masuk ke tahap penghapusan partikel yaitu "-pun" pada akhiran kata dasar selanjutnya pengecekan awalan jika tidak ada maka algoritma mengembalikan kata menjadi "bagaimana" dan stemming berhenti. Untuk stemming kata "jg" dan "jd" maka sistem akan mengecek pada kamus yang berisi singkatan dari kata-kata dasar anomali yang sudah didaftarkan contoh yang terdapat pada kamus kata "jg"="juga", "jd"="jadi". Sehingga semua kata tidak yang terdaftar pada kamus yang berisi singkatan dari kata-kata dasar akan diganti ke kata dasar yang baku. Proses stemming ini mempunyai alur yang digambarkan pada gambar 3.4 sebagai berikut :



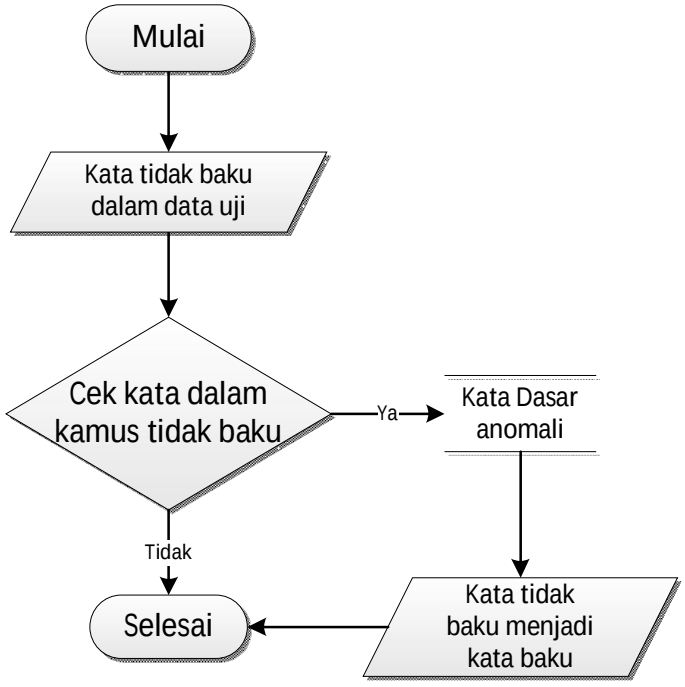
Gambar 3.4 *Flowchart Stemming* kata baku

pada *flowchart stemming* di atas tahapan yang dilakukan adalah sebagai berikut :

1. Kata yang hendak di *stemming* dicari terlebih dahulu pada kamus. Jika ditemukan, maka kata tersebut adalah kata dasar, jika tidak maka langkah 2 yang dilakukan.
2. Pada kata dalam data uji akan dicek apakah memiliki *inflectional suffixes*, yaitu akhiran(“-lah”,“-kah”,“-tah”,“-pun”) dan kata ganti kepunyaan atau *possessive pronoun* PP (“-ku”, “-mu”, “-nya”) ternyata pada kata dalam data uji tidak terdapat *inflectional suffixes* dan *possessive pronoun* kemudian sistem melakukan proses selanjutnya *Derivation prefiks.*

3. Pada kata dalam data uji akan dicek apakah memiliki *Derivation suffixes*, yaitu akhiran(“-an”, ”-i”, “-kan”) ternyata pada kata dalam data uji terdapat *Derivation suffixes* maka sistem menghapus akhiran *derivation suffixes* lalu sistem mengecek ke kamus kata dasar jika kata hasil *derivation suffixes* ada di dalam kamus kata dasar maka sistem berhenti jika tidak ditemukan akan dilakukan proses selanjutnya.
4. Kata dalam data uji akan dicek, apakah memiliki *derivation prefixes*, ternyata kata dalam data uji mengandung *derivation prefiixes* kemudian sistem mencari kata ke kamus kata dasar jika ditemukan maka kata dalam data uji adalah kata dasar dan proses berhenti, jika tidak maka kata dikembalikan dan proses berhenti.

Pada tweet uji terdapat dua kata yang tidak baku yaitu jg dan jd. Maka kedua kata tersebut harus diubah ke dalam bentuk kata yang baku melalui proses *stemming* yang menggunakan kamus yang berisi daftar kata-kata anomali dari kata dasarnya begitu juga untuk singkatan-singkatan dari kata dasar. Berikut contoh proses *stemming* untuk salah satu kata yang tidak baku :



Gambar 3.5 *Flowchart Stemming* kata tidak baku

1. Kata tidak baku yang akan di-*stemming* langsung dicek dalam kamus apakah termasuk ke dalam kata anomali atau singkatan dari kata dasar.
2. Ternyata kata tidak baku termasuk dalam kata anomali dari kata dasarnya maka hasilnya diganti menjadi kata baku dan proses berhenti.
3. Jika kata tidak baku atau kata singkatan dari kata dasar tidak terdapat dalam kamus kata tidak baku maka proses akan berhenti.

Adapun isi dari sebagian daftar kata anomali pada kamus kata dasar anomali yang disajikan tabel 3.6 :

Tabel 3.6 Kamus kata dasar anomali

Kata anomali	Kata dasarnya
jg	juga
jd	jadi
wkt	waktu
brp	berapa
jml	jumlah

Berikut ini merupakan contoh tahapan *stemming* pada tabel 3.7 :

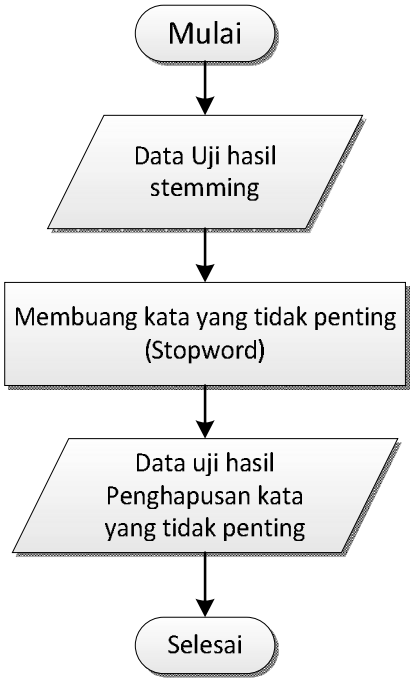
Tabel 3.7 Ilustrasi *Stemming*

Contoh Data			
Data Latih hasil *Tokenizing*		**Tahapan *Stemming***	
Input	bagaimanapun jg dukung jokowi jd presiden agar indonesia hebat	*Output*	bagaimana juga dukung jokowi jadi presiden agar indonesia hebat

3.5.5 Penghapusan *Stopwords*

Kata-kata yang terkandung pada daftar stopwords yang terdapat pada daftar kata khusus stopword bahasa indonesia terdiri berisi 810 kata-kata yang sering muncul namun tidak memiliki arti. Pada contoh kata “bagaimana”, “juga”, “agar”

dan “jadi” terdapat di bank kata stopwords sehingga kata tersebut harus dihilangkan. Kata yang tertinggal dapat dilihat pada Gambar 3.6 berikut :



Gambar 3.6 *Flowchart Stopword*

Pada tahapan ini, ada beberapa aturan proses agar hasil *stopword* sesuai apa yang diharapkan. Adapun aturan-aturan tersebut disajikan pada tabel 3.8.

Tabel 3.8 Aturan tahapan *Stopword*

Kondisi	Aksi
Jika Inputan data latih mengandung kata pada *database stopword*	Maka akan menghapus kata atau *string* dalam data latih.
Jika Inputan data latih tidak mengandung kata pada kamus *stopword*	Maka tidak akan dihapus kata atau *string* data latih

Adapun isi dari sebagian daftar kata *stopword* yang disajikan pada tabel 3.9 :

Tabel 3.9 Daftar sebagian Kata-kata *Stopword*

Data *Stopword*			
telah	punya	mendapatkan	dari
untuk	setiap	pernah	lain
agar	memang	lakukan	melakukannya
jadi	seperti	ada	antara
juga	hampir	semua	setelah

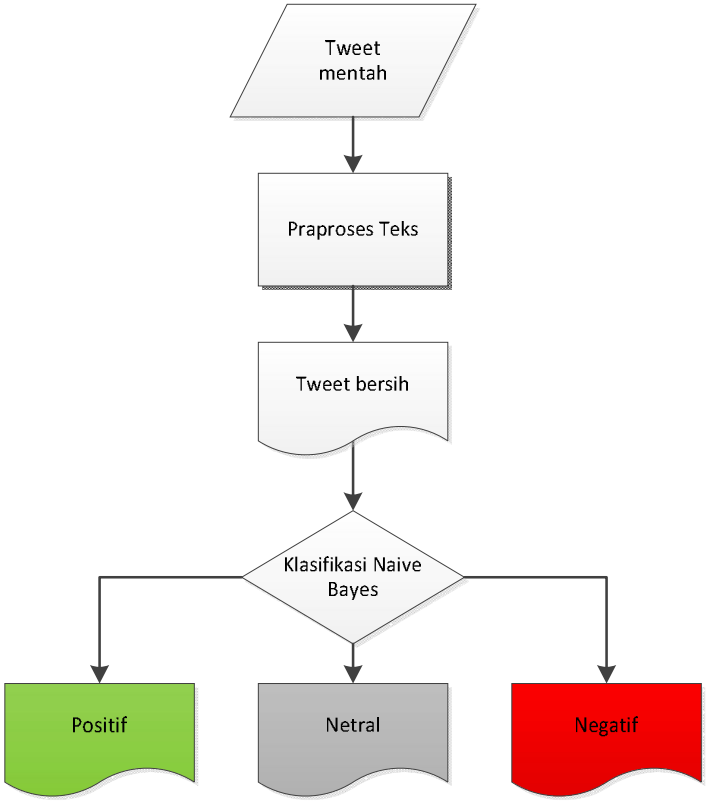
Tabel 3.10 merupakan contoh tahapan *stopword* sebagai berikut.

Tabel 3.10 Ilustrasi *Stopword*

Contoh Data			
Data Latih hasil *stemming*		**Tahapan *Stopword***	
Input	bagaimanapun juga dukung jokowi jadi presiden agar indonesia hebat	*Output*	dukung jokowi presiden indonesia hebat

Dari *tweet "Bagaimanapun jg dukung ==> jokowi jd presiden agar indonesia hebat \(´▽`)/"* setelah melalui proses *preprocessing text* hasilnya menjadi *"dukung, jokowi, presiden, indonesia, hebat*" dan ini belum diketahui kategori kelasnya sehingga harus melalui proses perhitungan dengan metode *Naive Bayes.*

3.6 Analisis Algoritma



Gambar 3.7 *Flowchart Naive bayes*

Gambar 3.7 memperlihatkan arsitektur sistem yang akan dikembangkan. Sistem ini menerima masukan berupa data *tweet* mentah. Data ini akan di praproses terlebih dulu sebelum diproses lebih lanjut.

Dari gambar diagram alur Algoritma *Naive Bayes* di atas dapat jelaskan sebagai berikut :

1. *Tweet* mentah yaitu kumpulan *tweet* yang belum dilakukan *preprocessing text* yaitu dengan *parsing*, *case folding*, penghapusan *stopword* dan *stemming* contoh `"dukung jokowi jd presiden wujud indonesia maju jg hebat \(´▽`)/"`
2. Koleksi *tweet* diproses terlebih dahulu sebelum digunakan dalam program. Proses ini disebut *preprocessing text*. *Preprocessing text* bertujuan untuk mengurangi volume kosakata, menyeragamkan kata dan menghilangkan *noise*.

3. *Tweet* bersih yaitu kumpulan *tweet* yang sudah dihilang *noise*, variasi kata dari kata contoh "`dukung jokowi jadi presiden wujud indonesia maju hebat`"
4. Klasifikasi *naive Bayes* yaitu mengklasifikasikan apakah tweet yang dijadikan data uji hasilnya sentimen positif atau negatif.
5. Total masing-masing *tweet* positif, *tweet* negatif dan *tweet* netral hasil akhir dari klasifikasi

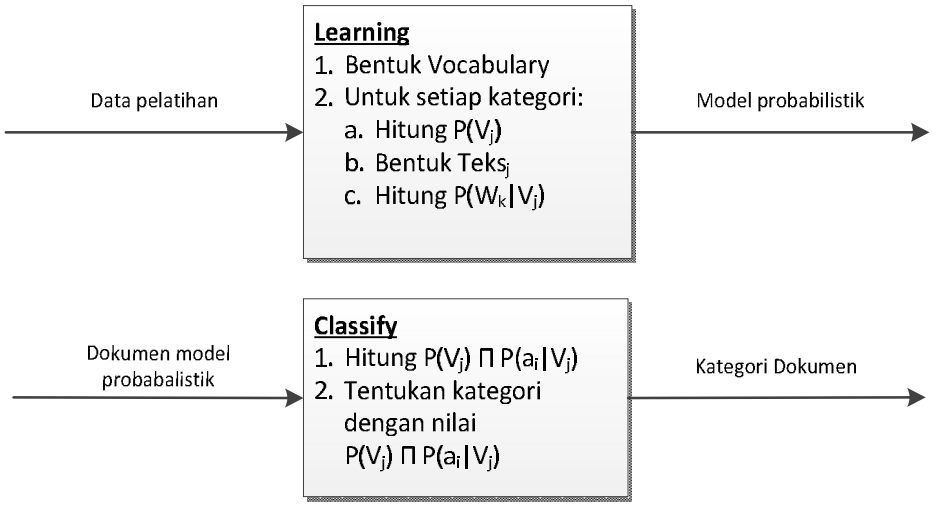
Algoritma *Naive Bayes* merupakan pengklasifikasian dengan metode probabilitas, yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai teorema *Bayes*. Dari contoh *tweet* yang telah dibersihkan melalui tahapan *preprocessing text* maka hasil kata disajikan pada tabel 3.11 :

Tabel 3.11 Indeks kata data uji

Kata pada data uji
dukung
presiden
indonesia
hebat

dari data uji diatas maka didapat fitur kemunculan yang penting diantaranya dukung, presiden, maju, hebat. Setelah mendapat fitur yang diperlukan maka dilakukan proses klasifikasi dengan algoritma *Naive Bayes.*

Algoritma untuk *Naive Bayes Classifier* yang ada pada gambar 3.8:



Gambar 3.8 Algoritma Naive Bayes

Dari Algoritma Naive Bayes pada gambar 3.7 tahapan yang dilakukan adalah sebagai berikut :

1. Data pelatihan : *Tweet* yang akan dijadikan sebagai data pelatihan untuk data testing contoh dari kumpulan training data yang sudah melalui tahapan *preprocessing text* dari masing-masing kategori
2. Bentuk *Vocabulary* : Kumpulkan semua kata yang unik dari *tweet* pada tabel *training data*, contoh : tolak, jangan, salah, dukungan, maju, hebat, presiden, cawapres, jakarta.
3. Hitung $P(V_j)$: Total kelas ada tiga yaitu positif, negatif dan non opini maka kemungkinannya adalah :

Tabel 3.12 Nilai $P(V_j)$ untuk setiap kategori

Kategori	$P(V_j)$
Positif	$\frac{1}{3}$
Negatif	$\frac{1}{3}$
Non opini	$\frac{1}{3}$

4. Bentuk $Teks_j$: untuk kelas positif contoh kata uniknya yaitu “dukung”, “maju”, “hebat”. Untuk kelas negatif contoh yaitu “tolak”, “jangan”, “salah”. Untuk kelas non opini contohnya “presiden”, “cawapres”, “jakarta”

Tabel 3.13 Data pelatihan

Kategori	**Hasil *keyword* data latih**
Positif	dukung(8), maju(2), hebat(2)
Negatif	tolak(3), jangan(4), salah(4)
Netral	presiden(10), cawapres(3), jakarta(5)

5. Setelah dilakukan pembuangan kata yang tidak relevan pada setiap karakter, maka proses selanjutnya pembelajaran data latih akan dimasukakan ke dalam model probabilitas *naïve bayes* (*learning*), ditentukan dengan persamaan 2.10.

Tabel 3.14 Model probabilistik

Kategori	**Dokumen** $P(w_k\|v_j) = \frac{n_k + 1}{n + \|kosakata\|}$								
	cawapres	dukung	hebat	jangan	maju	presiden	salah	tolak	jakarta
Positif	$\frac{1}{21}$	$\frac{9}{21}$	$\frac{3}{21}$	$\frac{1}{21}$	$\frac{3}{21}$	$\frac{1}{21}$	$\frac{1}{21}$	$\frac{1}{21}$	$\frac{1}{21}$
Negatif	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{5}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{5}{20}$	$\frac{4}{20}$	$\frac{1}{20}$
Netral	$\frac{4}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{11}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$\frac{6}{27}$

6. Klasifikasi *Naïve Bayes* Untuk Data Uji

   Selanjutnya, setelah didapatkan data latih, maka tahapan dilanjutkan ke dalam proses pengklasifikasian untuk menguji model yang telah dibangun kepada data uji untuk mengukur ketepatan atau performa model probabilitas dari data latih dalam mengklasifikasikan data uji.

Kasus : Berapa nilai klasifikasi jika terdapat data uji yang belum diketahui kategori jika memiliki kemunculan kosakata pada tabel 3.14.

Tabel 3.15 Data uji

Doc ke-n	Hasil Kemunculan Kosakata	Kategori
Baru	"dukung"(3), "presiden"(1), "maju"(2), "hebat"(3)	?

Selanjutnya, Pada tahap ini kemunculan kata yang pada data uji akan dilihat pada model probabilitas pada tabel 3.14 untuk dicari ***Vmap*** pada setiap kategori berikut ini :

Tabel 3.16 Nilai V_{map}

Kategori	$P(V_j)$	Dokumen $P(w_k\|v_j) = \frac{n_k + 1}{n + \|kosakata\|}$				Nilai V_{map}
		dukung	presiden	maju	hebat	
Positif	$\frac{1}{3}$	$\frac{9}{21}$	$\frac{1}{21}$	$\frac{3}{21}$	$\frac{3}{21}$	$1{,}39x10^{-4}$
Negatif	$\frac{1}{3}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$2{,}08x10^{-6}$
Non opini	$\frac{1}{3}$	$\frac{1}{27}$	$\frac{11}{27}$	$\frac{1}{27}$	$\frac{1}{27}$	$6{,}90x10^{-6}$

Berdasarkan tabel 3.15 dapat dilihat peluang kemunculan kata yag nilainya telah diperoleh dari model probabilitas kemunculan kata pada data latih di tabel 3.13. Peluang kemunculan kata yang besar akan menghasilkan Vmap yang tinggi, sehingga dokumen data uji akan terklasifikasi ke dalam karakter dengan Vmap yang paling tinggi. Pada kasus data uji diatas, dapat disimpulkan bahwa dokumen terklasifikaasi ke dalam kategori positif**.** Pada penelitian jumlah data yang digunakan adalah 300 *tweets*. Pada hasil perhitungan didapat jumlah *tweets* yang

benar adalah dan yang salah. Dari hasil klasifikasi didapat jumlah *True* Positif 50, *True* Negatif 90 dan *True* Netral 63.

3.7 Evaluasi Akurasi

Untuk menghitung akurasi dan *error* dari hasil klasifikasi maka diperlukan data hasil klasifikasi dengan keseluruhan data yang sebenarnya atau aktual berikut perhitungan akurasi *Naive Bayes* :

True Positif = 50

True Negatif = 90

True Netral = 63

False Positif = 17

False Negatif = 68

False Netral = 12

Jumlah data uji yang benar dari hasil perhitungan = 203

Jumlah data uji yang salah dari hasil perhitungan = 97

Total data uji keseluruhan : 300

$$Accuracy = \frac{TP+TN+TNet}{TP+TN+FN+FP+FNet+TNet} = \frac{50+90+63}{50+90+68+17+12+63} = \frac{203}{300} \times 100\% = 67{,}67\%$$

$$Error\ rate = \frac{97}{300} \times 100\% = 32{,}32\%$$

Hasil uji coba sistem *stopword* dengan *stemming* menggunakan proporsi dokumen uji disajikan pada tabel 3.17 :

Tabel 3.17 Tabel hasil klasifikasi

Proporsi Dokumen Uji	Penggunaan *stemming* dengan *Stopwords*
300 Data	67,67%

3.8 Analisa kebutuhan non fungsionalitas

Analisis kebutuhan non fungsional adalah langkah dimana seorang pembangun perangkat lunak menganalisis sumber daya yang akan digunakan dan menggunakan perangkat lunak yang dibangun. Perangkat keras dan perangkat

lunak yang dimiliki harus sesuai dengan kebutuhan, sehingga dapat ditentukan kompabilitas aplikasi yang dibangun terhadap sumber daya yang ada.

3.8.1 Analisa Kebutuhan Perangkat Lunak

Adapun perangkat lunak yang digunakan untuk optimasi analisis sentimen pada *posting twitter* menggunakan metode *Naive Bayes* :

1. Netbeans
2. Visual paradigm
3. Ms Visio

3.8.2 Analisa kebutuhan perangkat keras

Adapun perangkat keras yang digunakan untuk optimasi analisis sentimen pada *posting twitter* menggunakan metode *Naive Bayes :*

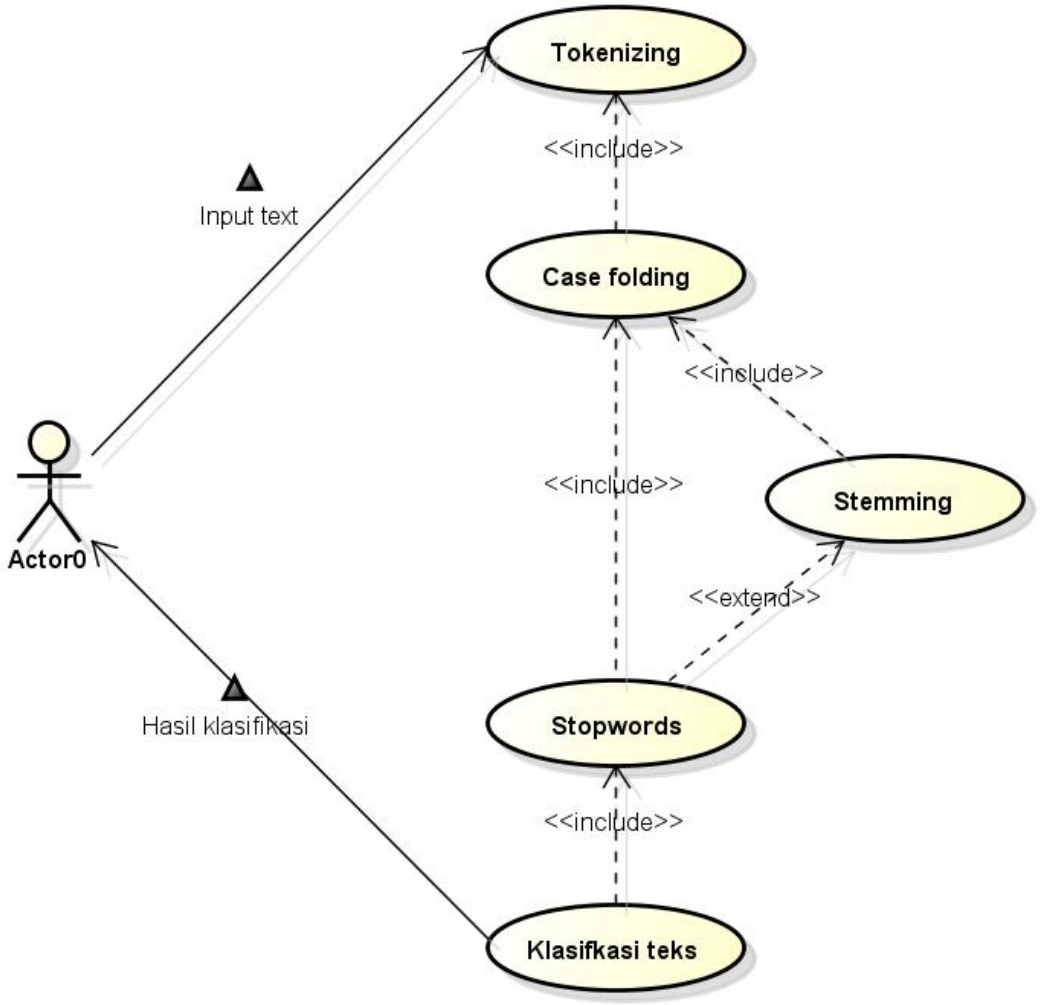
1. Prosesor Intel Core i3 (2,2 GHz).
2. Memori 2.00 GB DDR2.
3. Harddisk 500 GB
4. Monitor 14.1 inch

3.9 Analisis kebutuhan fungsional

Analsis kebutuhan fungsional mendefinisikan aksi dasar yang ada dalam perangkat lunak yang dibangun untuk menerima dan memproses masukan dan menghasilkan keluaran.

3.9.1 *Use Case*

Diagram *use case* adalah diagram yang menggambarkan secara umum yang menjadi masukkan, proses, dam keluaran yang terjadi pada sebuah sistem. Diagram *use case* untuk perangkat lunak yang akan dibangun sebagai berikut :



Gambar 3.9 *Use Case Diagram*

Definisi Aktor berfungsi untuk menjelaskan *actor* yang terlibat pada *use case diagram.* Berikut ini adalah tabel 3.18 yang menerangkan definisi *actor*.

Tabel 3.18 Definisi Aktor

No	Aktor	Deskripsi
1	Pengguna	Merupakan orang yang akan menggunakan aplikasi

3.9.1.1 Definisi *Use Case*

Tabel 3.19 Deskripsi *Use Case*

No	*Use Case*	Deskripsi
1	*Tokenizing*	Melakukan pemotongan kalimat menjadi kata-kata
2	*Case Folding*	Mengubah karakter menjadi huruf kecil dan menghapus delimiter pada kata-kata
3	*Stemming*	Mengubah kata yang berimbuhan dan kata tidak baku menjadi kata dasarnya
4	*Stopwords*	Menghilangkan kata yang tidak memiliki arti
5	Klasifikasi teks	Menghitung seluruh data uji ke dalam masing-masing kelas kategori

3.9.1.2 Skenario *Use Case*

Skenario *use case* menggambarkan alur penggunaan sistem dimana setiap skenario digambarkan dari sudut pandang aktor, seseorang atau peranti yang berinteraksi dengan perangkat lunak dalam berbagai cara.

Tabel 3.20 Tabel Skenario *Use Case Tokenizing*

Identifikasi		
Nomor	1.	
Nama	*Tokenizing*	
Tujuan	Melakukan pemotongan kalimat menjadi per kata	
Aktor		
Skenario Utama		
Kondisi Awal	Pengguna menginput data *testing*	
Aksi Aktor		**Reaksi Sistem**
Menginput teks		Melakukan pemotongan seluruh kalimat
Kondisi Akhir	Seluruh kalimat telah dipotong menjadi kata per kata	

Tabel 3.21 Skenario *Use Case Case Folding*

Identifikasi		
Nomor	2.	
Nama	*Case folding.*	
Tujuan	Mengubah karakter menjadi huruf kecil dan menghapus delimiter pada kata-kata	
Aktor		
Skenario Utama		
Kondisi Awal	Seluruh kalimat sudah menjadi potongan kata per kata	
Aksi Aktor		**Reaksi Sistem**
		Sistem melakukan pembersihan setiap karakter dari delimiter dan mengubah setiap karakter menjadi huruf kecil
Kondisi Akhir	Seluruh kata sudah menjadi karakter yang bersih dari delimiter menjadi huruf kecil.	

Tabel 3.22 Skenario *Use Case Stemming*

Identifikasi		
Nomor	3.	
Nama	*Stemming*.	
Tujuan	Mengubah kata berimbuhan menjadi kata dasarnya	
Aktor		
Skenario Utama		
Kondisi Awal	Seluruh kata sudah bersih dari delimiter dan menjadi huruf kecil	
Aksi Aktor		**Reaksi Sistem**
		Sistem menghilangkanimbuhan dan mengubah menjadi kata dasarnya
Kondisi Akhir	Seluruh kata sudah menjadi kata dasarnya.	

Tabel 3.23 Skenario *Use Case Stopwords*

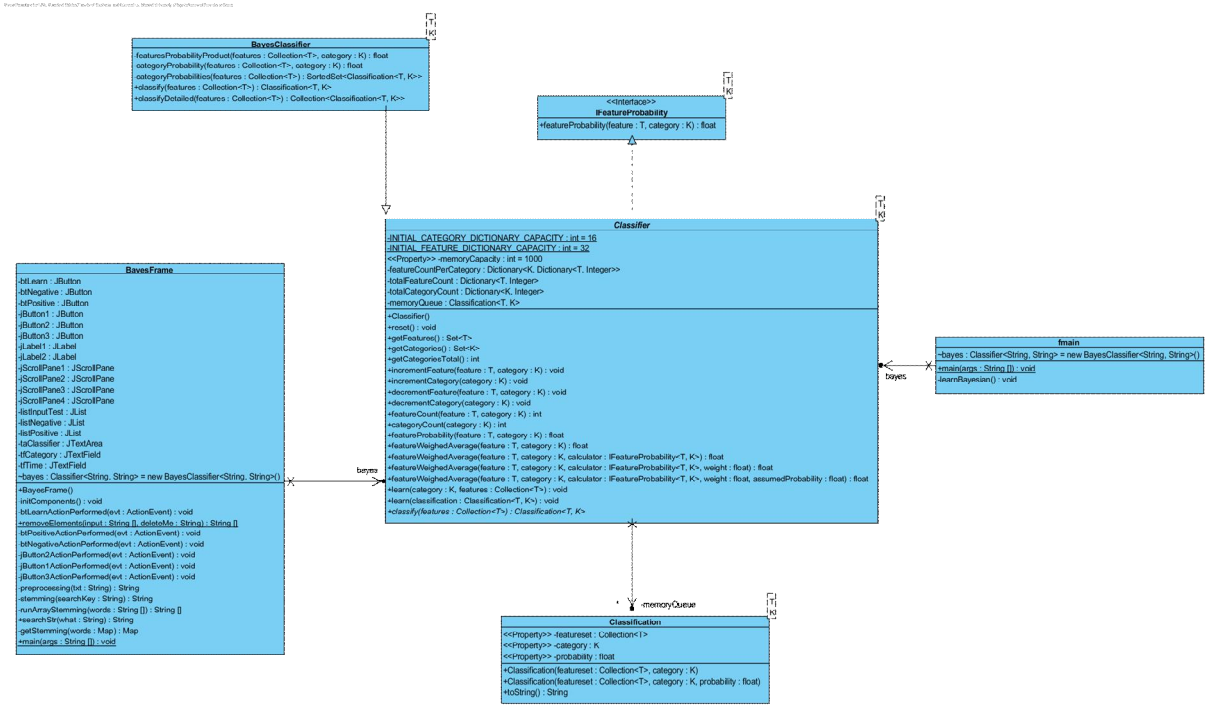
Identifikasi		
Nomor	4.	
Nama	*Stopwords*.	
Tujuan	Menghilangkan kata yang tidak memiliki arti	
Aktor		
Skenario Utama		
Kondisi Awal	Seluruh kata sudah menjadi kata dasarnya	
Aksi Aktor		**Reaksi Sistem**
		Menghilangkan setiap kata dasar yang tidak memiliki arti
Kondisi Akhir	Terkumpul kata dasar yang memiliki arti.	

Tabel 3.24 Klasfikasi Teks

Identifikasi		
Nomor	5.	
Nama	Klasifikasi teks.	
Tujuan	Menghitung seluruh data uji ke dalam masing-masing kelas kategori	
Aktor		
Skenario Utama		
Kondisi Awal	Terkumpul kata dasar yang memiliki arti.	
Aksi Aktor		**Reaksi Sistem**
		Menghitung seluruh data uji
Kondisi Akhir	Menampilkan total seluruh data uji pada masing-masing kelas kategori dan akurasinya	

3.9.2 *Class diagram*

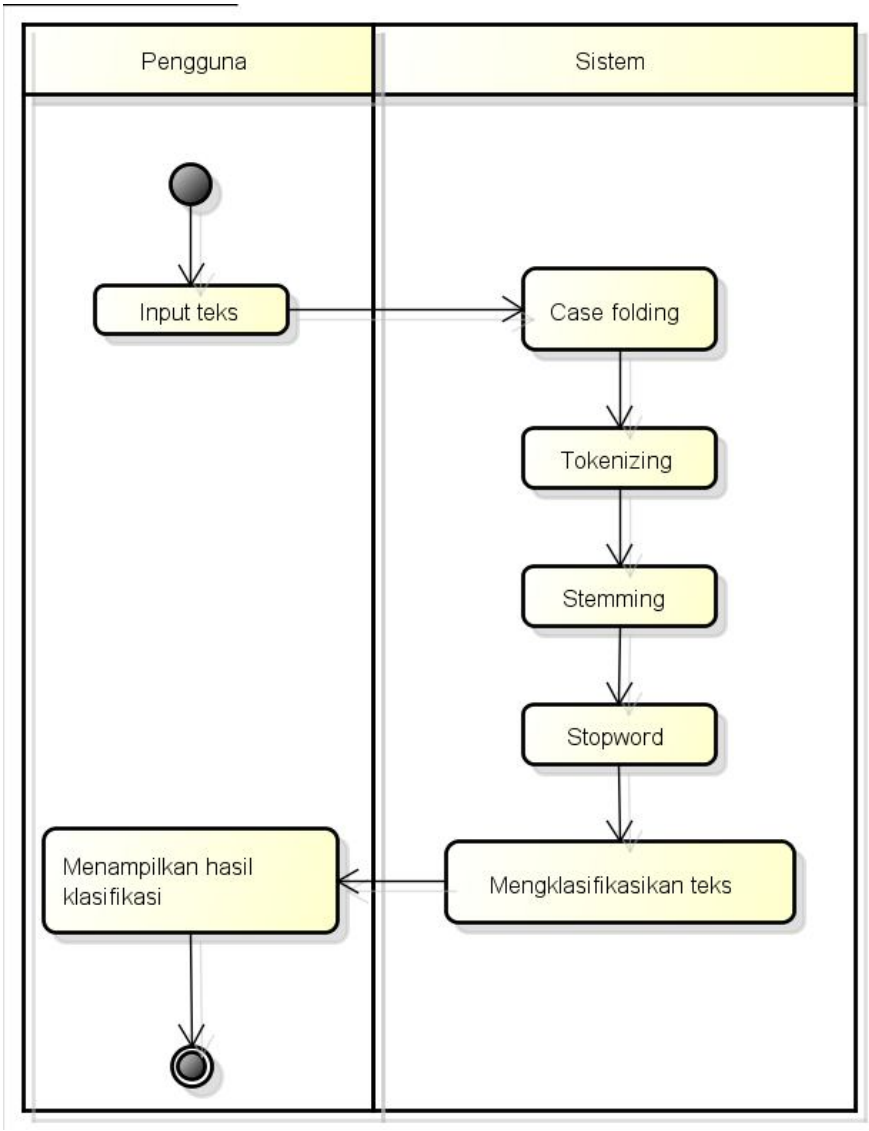
Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas-kelas yang terdapat pada klasifikasi teks adalah kelas BayesClassifier, Classification, Classifier, IfeatureProbability, BayesFrame, fmain, BayesFrame



Gambar 3.10 *Class Diagram*

3.9.3 *Activity Diagram*

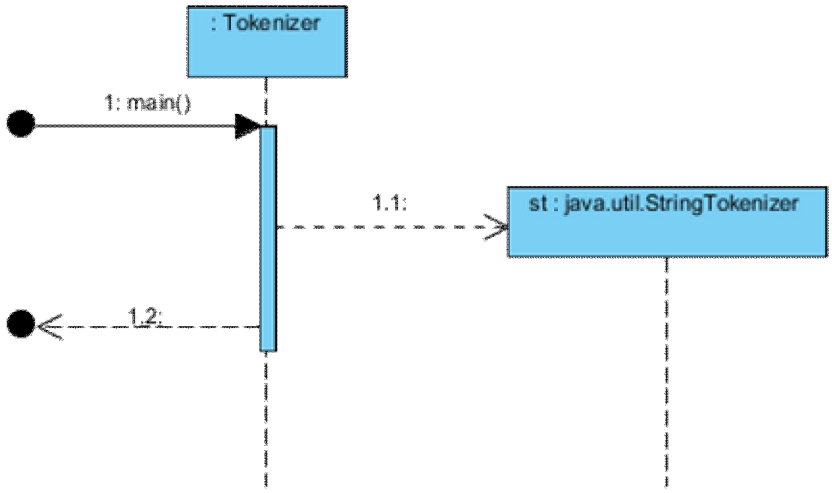
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Dalam diagram aktivitas yang terlibat adalah antara *user* dan sistem. Terdapat tujuh aktivitas yang tergambar sesuai prosesnya masing-masing yaitu *Activity diagra*m *classifying text*.



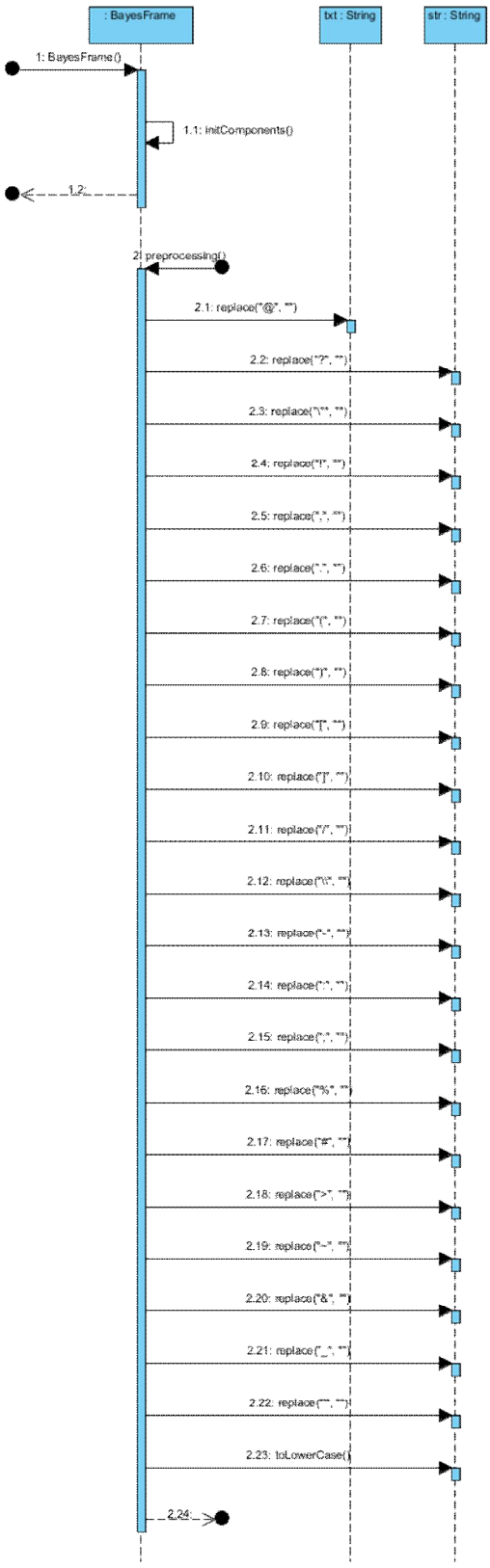
Gambar 3.11 *Activity Diagram*

3.9.4 *Sequence Diagram*

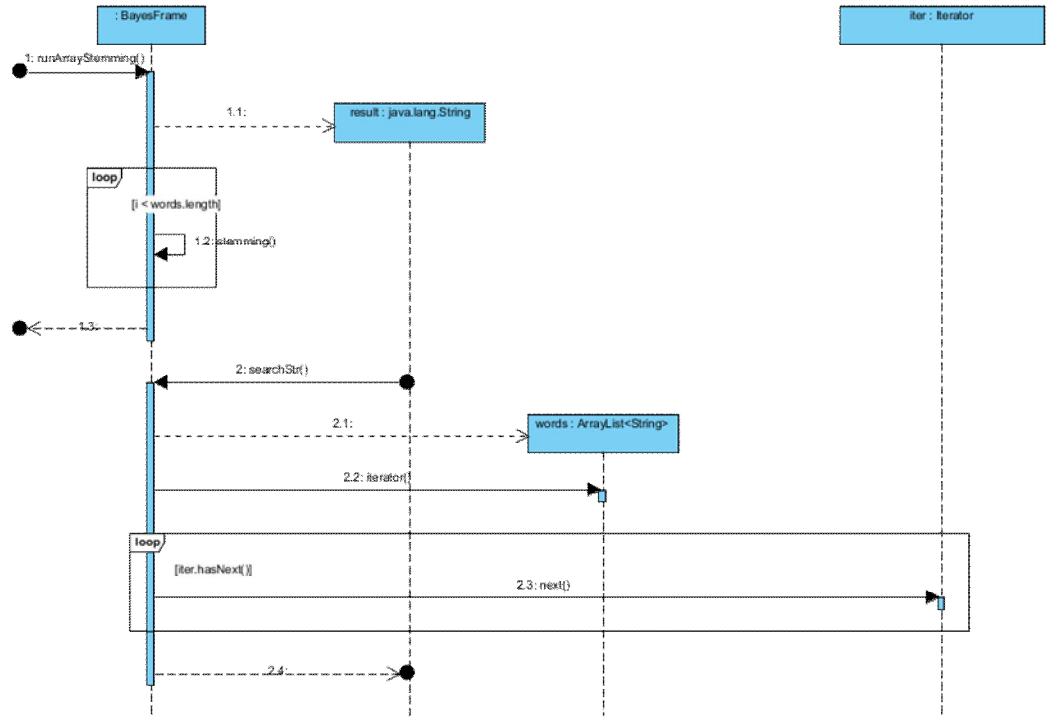
Diagram sekuen atau *sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.



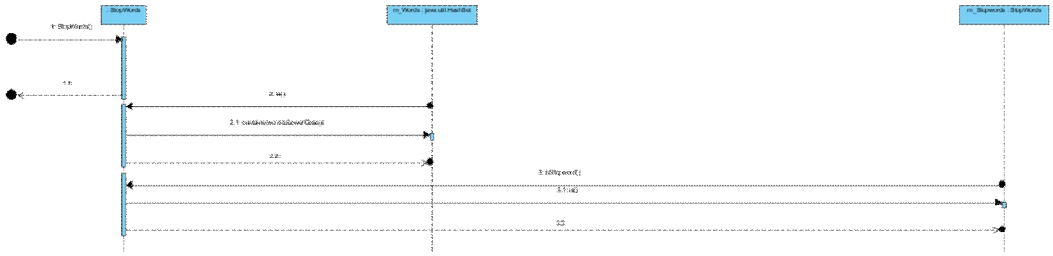
Gambar 3.12 *Sequence Diagram Tokenizing*



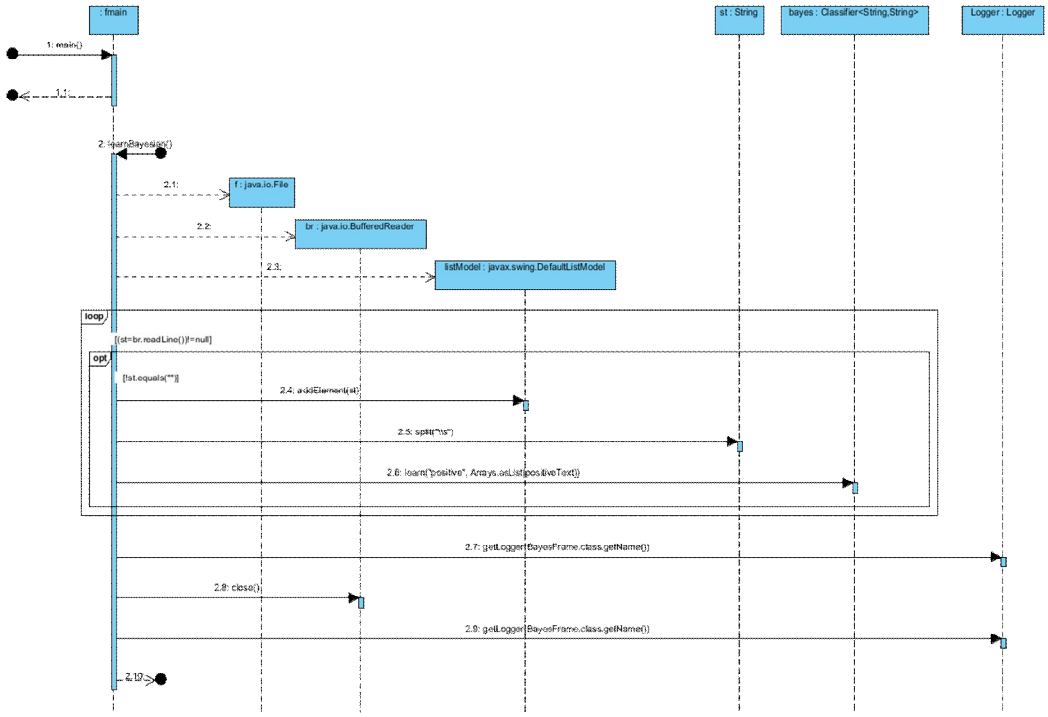
Gambar 3.13 ***Sequence Diagram Case Folding***



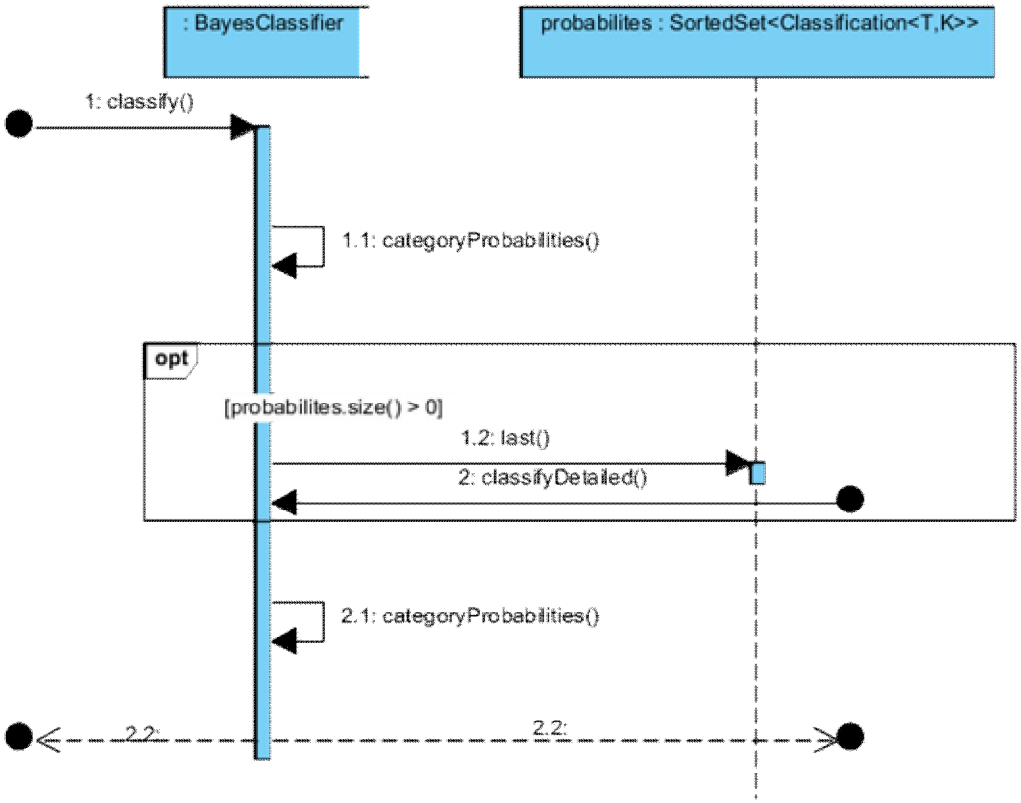
Gambar 3.14 ***Sequence Diagram Stemming***



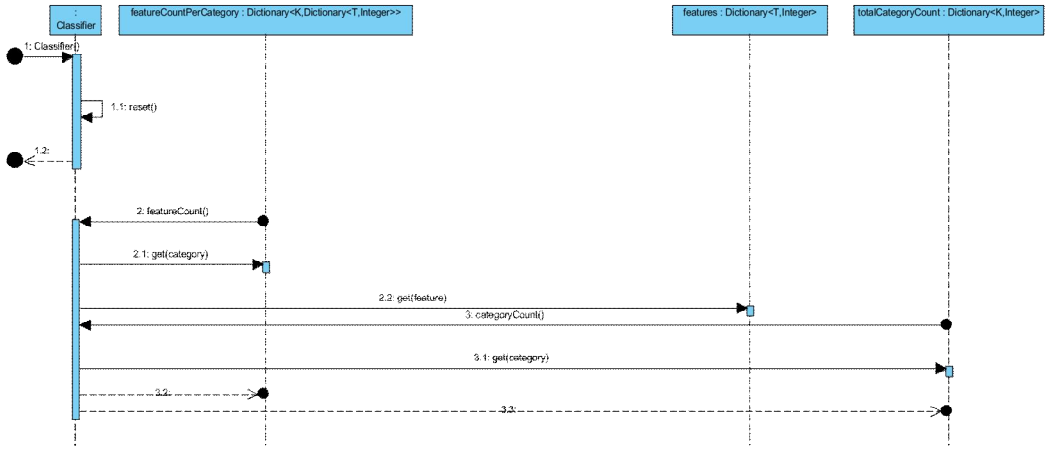
Gambar 3.15 ***Sequence diagram Stopwords***



Gambar 3.16 ***Sequence Diagram fmain***



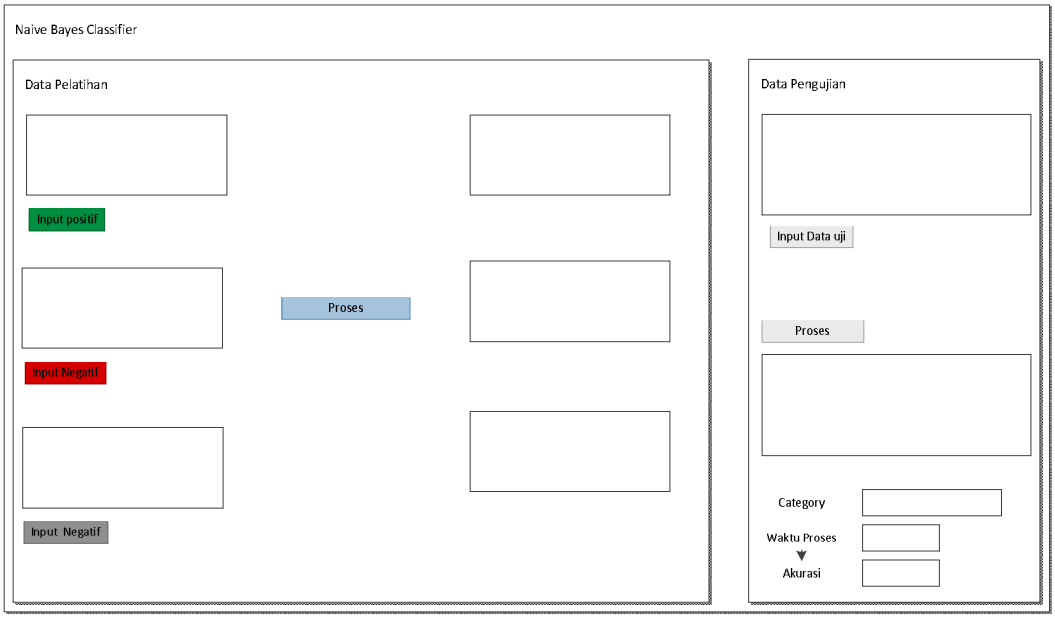
Gambar 3.17 ***Sequence Diagram BayesClassifier***



Gambar 3.18 ***Sequence Diagram Classifier***

3.9.5 Perancangan Antarmuka Perangkat Lunak

Perancangan antarmuka merupakan salah satu bagian penting dalam perancangan sistem karena nantinya antarmuka tersebut akan menjadi fasilitas yang menjembatani interaksi manusia dengan sistem.



Gambar 3.19 Antarmuka Klasifikasi teks