

**ANALISIS SENTIMEN TENTANG OPINI FILM PADA  
DOKUMEN TWITTER BERBAHASA INDONESIA  
MENGGUNAKAN NAIVE BAYES DENGAN  
PERBAIKAN KATA TIDAK BAKU**

**SKRIPSI**

Untuk memenuhi sebagian persyaratan  
memperoleh gelar Sarjana Komputer

Disusun oleh:

Prananda Antinasari

NIM: 135150218113019



**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BRAWIJAYA  
MALANG  
2017**

**PENGESAHAN**

ANALISIS SENTIMEN TENTANG OPINI FILM PADA DOKUMEN TWITTER

BERBAHASA INDONESIA MENGGUNAKAN NAIVE BAYES DENGAN PERBAIKAN

KATA TIDAK BAKU

Universitas Brawijaya

Diajukan untuk memenuhi sebagian persyaratan

memperoleh gelar Sarjana Komputer

Disusun Oleh:

Prananda Antinasari

NIM: 135150218113019

Skrripsi ini telah diuji dan dinyatakan lulus pada

24 Juli 2017

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

Dosen Pembimbing II

Rizal Setya Perdana, S.Kom, M.Kom

NIK: 201603 910118 1 001

M. Ali Fauzi, S.Kom, M.Kom

NIK: 201502 890101 1 001

Mengetahui

Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D

NIP: 19710518 200312 1 001





## **PERNYATAAN ORISINALITAS**

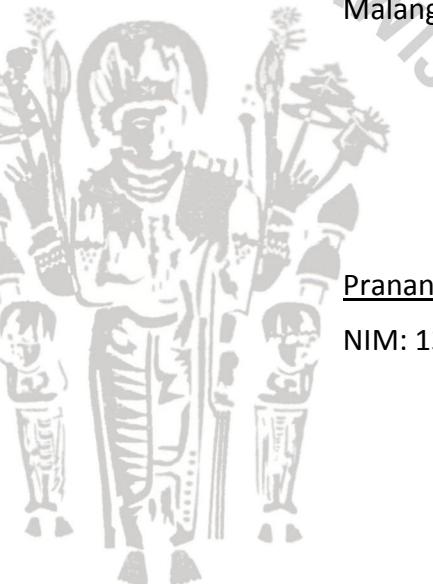
Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (<sup>vi</sup>sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 24 Juli 2017

Prananda Antinasari

NIM: 135150218113019





## **KATA PENGANTAR**

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat hidayah serta karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul *“Analisis Sentimen Tentang Opini Film pada Dokumen Twitter Berbahasa Indonesia Menggunakan Naive Bayes dengan Perbaikan Kata Tidak Baku”*.

Penulis menyadari bahwa masih banyak kekurangan dalam penyusunan skripsi ini. Dan juga skripsi dapat selesai atas dukungan dan bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis mengucapkan banyak terimakasih kepada:

1. Bapak Rizal Setya Perdana, S.Kom, M.Kom sebagai dosen pembimbing I dan Bapak M. Ali Fauzi, S.Kom, M.Kom sebagai dosen pembimbing II yang telah meluangkan waktu untuk memberikan pengarahan dan bimbingan kepada penulis.
2. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
4. Bapak M. Tanzil Furqon, S.Kom, M.CompSc selaku Sekretaris Jurusan Teknik Informatika.
5. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
6. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
7. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan dukungan selama penyelesaian skripsi ini.
8. Kedua orang tua penulis, adik tersayang penulis, serta keluarga penulis yang selalu memberikan motivasi dan doa kepada penulis hingga skripsi ini dapat terselesaikan.
9. Sahabat-sahabat penulis yang selalu meneman, memberikan dukungan, bantuan kepada penulis, Maharani Tri A, Dessy Amry, Siti Robbana, Ainul Furkan, Ferly Gunawan, Umi Roffiqoh, Yusna-Tarita, Normi Prima, hingga skripsi ini dapat terselesaikan.
10. Seluruh teman-teman penulis yang telah membantu kelancaran penulisan skripsi yang tidak dapat penulis sebutkan satu per satu.

UNIVERSITAS BRAWIJAYA



Malang, 24 Juli 2017

Penulis

nandaopra@gmail.com

Penulis menyadari bahwa skripsi ini masih jauh dari kata sempurna. Untuk itu dengan segala kerendahan hati, penulis mohon maaf yang sebesar-besarnya.

Kritik dan saran yang membangun sangat penulis harapkan demi kesempurnaan skripsi ini. Semoga skripsi ini dapat memberikan informasi dan manfaat bagi banyak pihak.



## ABSTRAK

Pertumbuhan media sosial yang sangat pesat tidak membuat Twitter ditinggalkan oleh penggunanya. Twitter merupakan salah satu media sosial yang memungkinkan penggunanya untuk melakukan interaksi, berbagi informasi, atau bahkan untuk mengutarakan perasaan dan opini, termasuk juga dalam mengutarakan opini film. Komentar atau Tweet mengenai film yang ada pada Twitter dapat dijadikan sebagai evaluasi dalam menonton film dan meningkatkan produksi film. Untuk mengetahui hal tersebut, analisis sentimen dapat digunakan untuk mengklasifikasikan kedalam sentimen negatif atau positif. Didalam Tweet terkandung banyak ragam bahasa yang digunakan, yaitu diantaranya bahasa dalam bentuk tidak baku seperti bahasa *slang*, penyingkatan kata, dan salah eja. Oleh sebab itu dibutuhkan penanganan khusus pada Tweet. Pada penelitian ini digunakan kamus kata tidak baku dan normalisasi Levenshtein Distance untuk memperbaiki kata yang tidak baku menjadi kata baku dengan pengklasifikasian Naive Bayes. Berdasarkan hasil pengujian yang telah dilakukan didapatkan akurasi tertinggi dengan nilai *accuracy*, *precision*, *recall*, dan *f-measure* sebesar 98.33%, 96.77%, 100%, dan 98.36%.

**Kata kunci:** *twitter, tweet, analisis sentimen, perbaikan kata tidak baku, normalisasi levenshtein distance, naive bayes*



## ABSTRACT

*The rapid growth of social media does not make Twitter left by its users.*

*Twitter is one of the social media that allows user to interact each other, share information, or even to express feelings and opinions, including in expressing film opinions. Comments or Tweets about movies that exist on Twitter can be used as an evaluation in watching movies and increasing film production. To figure it out, sentiment analysis can be used to classify into negative or positive sentiments. In Tweets contain many languages used in the form of non-standard languages such as slang, word-outs, and misspellings. Therefore it takes special handling on Twitter comments. In this research used non-standard word dictionary and Levenshtein Distance normalization to improve non-standard word to standard word by classification Naive Bayes. Based on the result of the test, the highest accuracy, precision, recall, and f-measure value are 98.33%, 96.77%, 100%, and 98.36%.*

**Keywords:** twitter, tweet, sentiment analysis, non-standard word improvement, levenshtein distance normalization, naive bayes



<b>DAFTAR ISI</b>	
PENGESAHAN .....	
PERNYATAAN ORISINALITAS .....	iii
KATA PENGANTAR .....	iv
ABSTRAK .....	vii
ABSTRACT .....	viii
DAFTAR ISI .....	xii
DAFTAR TABEL .....	xiii
DAFTAR GAMBAR .....	xvii
DAFTAR KODE PROGRAM .....	xiv
DAFTAR LAMPIRAN .....	xv
BAB 1 PENDAHULUAN .....	1
1.1 Latar belakang .....	1
1.2 Rumusan masalah .....	3
1.3 Tujuan .....	3
1.4 Manfaat .....	4
1.5 Batasan masalah .....	4
1.6 Sistematika pembahasan .....	5
BAB 2 LANDASAN KEPUSTAKAAN .....	7
2.1 Kajian Pustaka .....	7
2.2 Twitter .....	8
2.3 Opini Film .....	8
2.4 <i>Text Mining</i> .....	8
2.5 <i>Text Pre-processing</i> .....	9
2.5.1 <i>Tokenizing</i> .....	10
2.5.2 <i>Cleansing</i> .....	10
2.5.3 <i>Case folding</i> .....	11
2.5.4 <i>Filtering</i> .....	11
2.5.5 <i>Stemming</i> .....	12

2.6 Perbaikan Kata Tidak Baku.....	12
2.6.1 Levenshtein Distance .....	13
2.7 Pembobotan Kata .....	15
2.8 Sentiment Analysis.....	15
2.9 Naive Bayes Classifier .....	15
2.9.1 Gaussian Naive Bayes.....	16
2.9.2 Bernoulli Naive Bayes.....	17
2.9.3 Multinomial Naive Bayes .....	17
2.10 Evaluasi .....	17
<b>BAB 3 METODOLOGI PENELITIAN .....</b>	<b>20</b>
3.1 Studi Literatur .....	21
3.2 Analisis kebutuhan.....	21
3.2.1 Kebutuhan fungsional .....	21
3.2.2 Kebutuhan Perangkat.....	22
3.2.3 Kebutuhan Data .....	22
3.3 Pengumpulan Data .....	22
3.4 Perancangan Sistem.....	22
3.5 Implementasi .....	23
3.6 Pengujian dan Analisis .....	23
3.7 Kesimpulan dan Saran .....	23
<b>BAB 4 ANALISIS DAN PERANCANGAN .....</b>	<b>25</b>
4.1 Deskripsi permasalahan.....	25
4.2 Deskripsi umum sistem.....	25
4.3 Pre-processing.....	27
4.3.1 Tokenizing .....	28
4.3.2 Cleansing .....	29
4.3.3 Case folding .....	30
4.3.4 Perbaikan kata tidak baku.....	30
4.3.5 Filtering .....	32
4.3.6 Stemming .....	33
4.4 Normalisasi Levenshtein Distance .....	34

4.5 Penyelesaian metode <i>Naive Bayes Classifier</i> .....	35
4.6 Perhitungan Manual .....	38
4.7 Perancangan Database .....	45
4.7.1 Perancangan Tabel data_latih .....	45
4.7.2 Perancangan Tabel data_uji .....	46
4.8 Perancangan Pengujian .....	46
4.8.1 Skenario pengujian pengaruh penggunaan <i>pre-processing</i> .....	46
4.8.2 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku .....	47
4.8.3 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance .....	48
<b>BAB 5 IMPLEMENTASI .....</b>	<b>49</b>
5.1 Batasan implementasi .....	49
5.2 Implementasi aplikasi .....	49
5.3 <i>Pre-processing</i> .....	51
5.3.1 <i>Tokenizing</i> .....	51
5.3.2 <i>Cleansing</i> .....	52
5.3.3 <i>Case Folding</i> .....	52
5.3.4 Perbaikan kata tidak baku .....	52
5.3.5 <i>Filtering</i> .....	53
5.3.6 <i>Stemming</i> .....	53
5.4 Normalisasi Levenshtein Distance .....	53
5.4.1 Normalisasi .....	54
5.4.2 <i>Edit distance</i> .....	54
5.5 Proses Analisis sentimen.....	55
5.5.1 <i>Prior</i> .....	55
5.5.2 <i>Conditional Probability</i> .....	56
5.5.3 <i>Total Conditional Probability</i> .....	56
5.5.4 <i>Posterior</i> .....	57
5.5.5 Klasifikasi .....	57
5.6 Proses Evaluasi.....	58



5.6.1 Accuracy .....	58
5.6.2 Precision .....	58
5.6.3 Recall .....	59
5.6.4 F-Measure .....	59
<b>BAB 6 PENGUJIAN DAN ANALISIS .....</b>	<b>60</b>
6.1 Skenario Pengujian .....	60
6.1.1 Pengujian pengaruh penggunaan <i>pre-processing</i> .....	60
6.1.2 Pengujian pengaruh penggunaan perbaikan kata tidak baku .....	61
6.1.3 Pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein distance .....	61
6.2 Hasil Pengujian .....	62
6.2.1 Hasil pengujian pengaruh penggunaan <i>pre-processing</i> .....	62
6.2.2 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku .....	63
6.2.3 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein distance .....	63
6.2.4 Hasil dari seluruh pengujian .....	64
6.3 Analisis Hasil .....	67
<b>BAB 7 PENUTUP .....</b>	<b>69</b>
7.1 Kesimpulan .....	69
7.2 Saran .....	69
<b>DAFTAR PUSTAKA .....</b>	<b>71</b>
<b>LAMPIRAN .....</b>	<b>74</b>

<b>DAFTAR TABEL</b>	
Tabel 2.1 Tabel Kontingensi .....	18
Tabel 4.1 Manualisasi Dokumen Tweet Opini Film.....	38
Tabel 4.2 Manualisasi <i>Tokenizing</i> .....	39
Tabel 4.3 Manualisasi <i>Cleansing</i> .....	39
Tabel 4.4 Manualisasi <i>Case Folding</i> .....	40
Tabel 4.5 Contoh Kamus_katabaku .....	40
Tabel 4.6 Manualisasi Perbaikan kata tidak baku.....	41
Tabel 4.7 Manualisasi <i>Filtering</i> .....	41
Tabel 4.8 Manualisasi <i>Stemming</i> .....	42
Tabel 4.9 Manualisasi normalisasi Levenshtein Distance.....	42
Tabel 4.10 Perhitungan Levenshtein Distance.....	43
Tabel 4.11 Frekuensi kemunculan kata .....	43
Tabel 4.12 Perhitungan <i>Prior</i> kelas positif dan negatif.....	44
Tabel 4.13 Perhitungan <i>Conditional Probability</i> .....	44
Tabel 4.14 Perhitungan <i>Total conditional probability</i> .....	44
Tabel 4.15 Perhitungan <i>Posterior</i> kelas Positif dan Negatif.....	45
Tabel 4.16 Perancangan Tabel data_latih .....	45
Tabel 4.17 Perancangan Tabel data_uji .....	46
Tabel 4.18 Skenario pengujian pengaruh penggunaan <i>pre-processing</i> .....	47
Tabel 4.19 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku .....	47
Tabel 4.20 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance .....	48
Tabel 5.1 Fungsi analisis sentimen.....	50
Tabel 6.1 Hasil pengujian pengaruh penggunaan <i>pre-processing</i> .....	60
Tabel 6.2 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku.....	61
Tabel 6.3 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance .....	62

<b>DAFTAR GAMBAR</b>	
Gambar 2.1 Tahapan <i>Pre-processing</i> .....	10
Gambar 2.2 Tahap <i>Tokenizing</i> .....	10
Gambar 2.3 Tahap <i>Cleansing</i> .....	11
Gambar 2.4 Tahap <i>Case Folding</i> .....	11
Gambar 2.5 Tahap <i>Filtering</i> .....	12
Gambar 2.6 Tahap <i>Stemming</i> .....	12
Gambar 2.7 Matriks perhitungan <i>Edit distance</i> .....	14
Gambar 3.1 Tahapan Penelitian.....	20
Gambar 4.1 Diagram alir sistem.....	26
Gambar 4.2 Diagram alir <i>pre-processing</i> .....	27
Gambar 4.3 Diagram alir <i>tokenizing</i> .....	28
Gambar 4.4 Diagram alir <i>cleansing</i> .....	29
Gambar 4.5 Diagram alir <i>case folding</i> .....	30
Gambar 4.6 Diagram alir perbaikan kata tidak baku .....	31
Gambar 4.7 Diagram alir <i>filtering</i> .....	32
Gambar 4.8 Diagram alir <i>stemming</i> .....	33
Gambar 4.9 Diagram alir normalisasi Levenshtein Distance .....	35
Gambar 4.10 Diagram alir proses <i>Naive Bayes Classifier</i> .....	37
Gambar 4.11 Perancangan Database.....	45
Gambar 6.1 Nilai <i>accuracy</i> dari seluruh pengujian .....	64
Gambar 6.2 Nilai <i>precision</i> dari seluruh pengujian .....	65
Gambar 6.3 Nilai <i>recall</i> dari seluruh pengujian.....	66
Gambar 6.4 Nilai <i>f-measure</i> dari seluruh pengujian.....	66
Gambar 6.5 Ilustrasi proses pengujian .....	67

<b>DAFTAR KODE PROGRAM</b>	
Kode Program 5.1 <i>Tokenizing</i> .....	51
Kode Program 5.2 <i>Cleansing</i> .....	52
Kode Program 5.3 <i>Case Folding</i> .....	52
Kode Program 5.4 Perbaikan kata tidak baku.....	53
Kode Program 5.5 <i>Filtering</i> .....	53
Kode Program 5.6 <i>Stemming</i> .....	53
Kode Program 5.7 Normalisasi.....	54
Kode Program 5.8 <i>Edit Distance</i> .....	55
Kode Program 5.9 <i>Prior</i> .....	55
Kode Program 5.10 <i>Conditional probability</i> .....	56
Kode Program 5.11 <i>Total conditional probability</i> .....	57
Kode Program 5.12 <i>Posterior</i> .....	57
Kode Program 5.13 Klasifikasi .....	58
Kode Program 5.14 <i>Accuracy</i> .....	58
Kode Program 5.15 <i>Precision</i> .....	59
Kode Program 5.16 <i>Recall</i> .....	59
Kode Program 5.17 <i>F-measure</i> .....	59

## **DAFTAR LAMPIRAN**

Lampiran 1 data_latih .....	74
Lampiran 2 data_uji .....	83
Lampiran 3 kamus_katabaku .....	87



## **1.1 Latar belakang**

Minat masyarakat pada dunia perfilman saat ini cukup tinggi dengan banyaknya film yang menarik dan tersedia untuk segala usia. Film merupakan sebuah karya seni yang dapat dijadikan sebagai alternatif untuk hiburan, mendapatkan informasi, dan pendidikan. Film terdiri dari beberapa jenis, diantaranya adalah jenis aksi, petualangan, komedi, dokumenter, horor, dan lain sebagainya. Dalam menonton film tentunya para penonton akan memiliki opini mengenai film yang telah ditonton. Biasanya para penonton akan menceritakan kesan atau komentar mengenai film tersebut. Pertumbuhan teknologi informasi, terutama media sosial telah banyak mengubah cara manusia untuk melakukan komunikasi dengan orang lain. Penggunaan sosial media banyak dimanfaatkan oleh masyarakat umum untuk mengekspresikan opini, pengalaman, maupun hal yang menjadi perhatian mereka (Troussas, et al., 2013).

Twitter merupakan salah satu media sosial yang memungkinkan penggunanya untuk berbagi informasi dengan sesama secara *realtime*. Informasi yang dibagikan pada Twitter biasanya disebut dengan kicauan (Tweet) yang terdiri dari 140 karakter (Kwak, et al., 2010). Berdasarkan penelitian yang dilakukan oleh Semiocast, yaitu lembaga riset media sosial yang berada di Paris, Prancis, mengatakan bahwa jumlah pemilik akun Twitter di Indonesia merupakan yang terbesar kelima di dunia, dan berada pada posisi ketiga negara yang paling aktif mengirim Tweet perhari (Asih, 2012). Tingginya pengguna Twitter menjadi peluang untuk masyarakat dalam melakukan jual beli, menyampaikan informasi, promosi, atau bahkan untuk mengekspresikan diri dengan menyampaikan pendapat, pikiran atau pendirian yang biasa disebut dengan opini. Opini dapat digunakan sebagai sarana untuk memberikan penilaian terhadap film yang diproduksi untuk masyarakat. Hal tersebut dapat dimanfaatkan oleh produser film untuk mengetahui bagaimana tanggapan masyarakat mengenai film yang telah diproduksi, begitu juga dengan masyarakat dapat meninjau film yang akan ditonton. Disinilah analisis sentimen memiliki peran yang sangat penting. Analisis sentimen atau penggalian opini dapat digunakan untuk memperoleh gambaran umum persepsi masyarakat terhadap kualitas film, apakah cenderung positif atau negatif.

Penggunaan analisis sentimen pada umumnya digunakan untuk menganalisis tentang suatu produk dalam meningkatkan kualitas produk yang akan datang. Dalam hal ini analisis sentimen dapat diterapkan pada opini film pada dokumen Twitter berbahasa Indonesia. Pada penulisan opini film terkadang terdapat penulisan Tweet yang sulit dibaca. Hal ini disebabkan oleh beberapa faktor seperti penulisan kata yang disingkat, penggunaan bahasa modern atau *slang*, salah dalam mengetik huruf dan tidak baku dalam penulisan opini (Agarwal, et al., 2014). Beberapa hal tersebut menyebabkan perlu dilakukan representasi ulang

kata sebelum dilakukan penentuan sentimen opini. Kata tersebut nantinya diklasifikasikan agar dapat diketahui apakah masuk ke dalam sentimen negatif atau positif. Dalam hal ini maka dibutuhkan metode klasifikasi untuk menganalisis komentar atau opini.

Pada penelitian sebelumnya yang berkaitan dengan analisis sentimen sudah pernah dilakukan oleh Firmansyah (2016). Pada penelitian ini metode yang digunakan adalah Naive Bayes yang berfungsi untuk mengklasifikasikan sentimen dari dokumen *review* aplikasi *mobile*. Selain Naive Bayes pada penelitian ini juga digunakan metode Query Expansion untuk mengoptimasi komentar singkat. Hasil pengujian pada penelitian ini menunjukkan bahwa dengan menggunakan algoritme Naive Bayes didapatkan hasil akurasi yang cukup baik yaitu sebesar 95%.

Dan dengan penambahan penggunaan metode Query Expansion menambah nilai akurasi dari Naive Bayes yaitu menjadi 98% (Firmansyah, 2016). Namun pada penelitian ini masih memiliki kekurangan yaitu masih belum dilakukannya proses normalisasi bahasa atau perbaikan kata tidak baku, sehingga menyebabkan menurunnya tingkat akurasi.

Penelitian selanjutnya yang berkaitan dengan analisis sentimen yaitu penelitian yang dilakukan oleh Nugraha (2014). Pada penelitian ini dilakukan pengklasifikasian terhadap sentimen positif dan negatif mengenai *review* film berbahasa Inggris dengan menggunakan metode K-Nearest Neighbor (K-NN). Pada penelitian ini membandingkan metode K-NN dengan metode Naive Bayes. Hasil dari akurasi yang didapat menunjukkan bahwa metode Naive Bayes lebih baik dibandingkan dengan metode K-NN. Dihasilkan metode Naive Bayes memiliki nilai akurasi tertinggi sebesar 81% dibandingkan dengan menggunakan metode K-NN yang bernilai akurasi sebesar 71%. Dan pada penelitian ini juga masih belum dilakukannya perbaikan kata tidak baku, sehingga tingkat akurasi tertinggi yang dicapai untuk metode Naive Bayes hanya sebesar 81%.

Kemudian penelitian selanjutnya yang berkaitan dengan analisis sentimen dan penanganan singkatan kata dan ejaan kata yang salah, yaitu penelitian yang dilakukan oleh Manalu (2014). Pada penelitian ini dilakukan pengklasifikasian terhadap sentimen positif, sentimen negatif, dan sentimen netral mengenai analisis sentimen pada Twitter tentang provider telekomunikasi. Metode yang digunakan pada penelitian ini yaitu dengan menggunakan algoritme Naive Bayes. Selain itu pada penelitian ini juga dilakukan seleksi fitur dengan menggunakan proses *text mining* serta proses n-gram karakter untuk menangani singkatan kata dan ejaan kata yang salah. Hasil pengujian yang dihasilkan mencapai 93% dari jumlah 100 tweet yang diuji dengan 2700 *data training*. Namun pada penelitian ini masih belum dilakukan perbaikan kata tidak baku untuk bahasa modern (*slang*).

Seiring dengan perkembangan teknologi metode untuk melakukan analisis maupun pengklasifikasian dengan menggunakan komputer sangat beragam, dengan berbagai data yang dapat diambil untuk pengujian. Pada penelitian ini peneliti menggunakan data dokumentasi Twitter untuk melakukan klasifikasi. Banyaknya informasi yang terkandung dalam Twitter menjadikan data ini dipilih.

Namun tantangan klasifikasi sentimen dengan menggunakan Twitter sebagai sumber datanya adalah pada bahasa yang digunakan. Bahasa yang digunakan pada Twitter memiliki beberapa karakteristik, yaitu kata yang digunakan tidak mengikuti Kamus Besar Bahasa Indonesia (KBBI), pengulangan karakter yang tidak seharusnya, dan penulisan dengan kata singkat. Oleh karena itu perlu dilakukan normalisasi bahasa dengan cara memperbaiki kata-kata tidak baku menjadi kata baku.

Berdasarkan uraian di atas, peneliti mengusulkan penelitian yang berjudul “Analisis Sentimen Tentang Opini Film Pada Dokumen Twitter Berbahasa Indonesia Menggunakan Naive Bayes Dengan Perbaikan Kata Tidak Baku”.

Penelitian ini menggunakan metode Naive Bayes karena metode ini memiliki tingkat akurasi yang lebih baik dibandingkan dengan algoritme lain. Dan untuk menangani masalah kata yang tidak baku, akan dilakukan suatu proses perbaikan kata tidak baku dengan mencocokkan kamus\_katabaku dan dengan dilakukan proses normalisasi Levenshtein Distance. Tujuan dari analisis sentimen ini untuk menganalisis dan mengevaluasi tentang opini film pada dokumen Twitter berbahasa Indonesia, sehingga diketahui penilaian terhadap film dan dapat dijadikan evaluasi bagi produser film untuk memperbagus film kedepannya dan untuk masyarakat dapat memilih film berdasarkan respon yang bagus.

## **1.2 Rumusan masalah**

Dari latar belakang yang telah diuraikan, maka penulis merumuskan pokok permasalahan, yaitu:

1. Bagaimana mengimplementasikan metode Naive Bayes dengan perbaikan kata tidak baku ke dalam aplikasi analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia?
2. Bagaimana pengaruh penggunaan *Pre-processing* dan perbaikan kata tidak baku terhadap hasil klasifikasi analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia?
3. Bagaimana hasil akurasi analisis sentimen tentang opini film pada dokumen twitter berbahasa Indonesia menggunakan metode Naive Bayes dengan perbaikan kata tidak baku?

## **1.3 Tujuan**

Berikut ini adalah tujuan-tujuan yang ingin dicapai oleh penulis dalam penelitian analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia, yang dibagi menjadi dua bagian, yaitu tujuan umum dan tujuan khusus. Tujuan umum dari penelitian ini adalah untuk menerapkan metode Naive Bayes pada analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia. Sementara untuk tujuan khusus dari penelitian ini dijelaskan sebagai berikut:

1. Menerapkan metode Naive Bayes dengan menambahkan perbaikan kata tidak baku ke dalam aplikasi analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia.
2. Mengetahui pengaruh penggunaan *Pre-processing* dan perbaikan kata tidak baku terhadap hasil klasifikasi analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia.
3. Mengetahui hasil akurasi analisis sentimen tentang opini film pada dokumen twitter berbahasa Indonesia menggunakan Naive Bayes dengan perbaikan kata tidak baku.

## 1.4 Manfaat

Berikut adalah manfaat dari penelitian yang dilakukan, diharapkan dalam penelitian ini dapat memberikan manfaat bagi pihak-pihak yang terkait, antara lain yaitu:

### 1. Bagi Pengguna

Diharapkan pengguna dapat mengetahui respon masyarakat mengenai opini film pada Twitter dan dapat dengan mudah mengelompokkan opini ke dalam kategori positif dan negatif.

### 2. Bagi Pembaca

Diharapkan hasil dari penelitian ini dapat menambah wawasan mengenai klasifikasi *text* dengan menggunakan metode Naive Bayes.

### 3. Bagi Penulis

Mendapatkan wawasan dan pengetahuan mengenai pengklasifikasian *text*.

## 1.5 Batasan masalah

Berikut ini adalah batasan masalah dalam penelitian analisis sentimen tentang opini film berbahasa Indonesia pada dokumen Twitter yang digunakan untuk memfokuskan pada pokok permasalahan yang dihadapi, diantaranya:

1. Penelitian ini menggunakan metode *Naive Bayes Classifier* tanpa membandingkan dengan metode yang lain.
2. Penelitian ini melakukan perbaikan kata tidak baku menjadi baku dengan kamus\_katabaku dan normalisasi Levenshtein Distance, namun tidak semua kata tidak baku dapat dinormalisasi.
3. Dokumen yang diproses adalah komentar atau opini film berbahasa Indonesia pada dokumen Twitter.
4. Kategori yang digunakan adalah positif dan negatif untuk analisis sentimen.
5. Perancangan dan implementasi dengan menggunakan bahasa pemrograman Python.

## **1.6 Sistematika pembahasan**

Berikut ini adalah sistematika pembahasan dalam penelitian analisis sentimen tentang opini film pada dokumen Twitter yang telah direncanakan, diantaranya:

### **BAB 1 : PENDAHULUAN**

Pada bab ini, menjelaskan latar belakang terkait permasalahan pada analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia, selanjutnya membuat rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, dan sistematika pembahasan.

### **BAB 2 : LANDASAN KEPUSTAKAAN**

Pada bab ini, berisi pembahasan teori-teori dasar untuk mendukung penelitian. Teori – teori dasar yang dikaji didasarkan pada penelitian sebelumnya, seperti buku, jurnal, paper, research, dan sumber referensi lainnya. Dan untuk teori yang dikaji dalam penelitian ini meliputi teori penjabaran Twitter, *text mining*, *text pre-processing*, perbaikan kata tidak baku, normalisasi Levenshtein Distance, *sentiment analysis*, dan penjelasan mengenai metode Naive Bayes.

### **BAB 3 : METODOLOGI PENELITIAN**

Pada bab ini, menjelaskan metode penelitian yang digunakan dalam analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia yang meliputi studi literatur, analisis kebutuhan, pengumpulan data, perancangan sistem, implementasi, pengujian dan analisis serta kesimpulan dan saran.

### **BAB 4 : ANALISIS DAN PERANCANGAN**

Pada bab ini, menjelaskan tentang analisis dan perancangan sistem yang terdiri dari perancangan sistem untuk klasifikasi dengan menggunakan metode Naive Bayes, perancangan *database*, serta uji coba dan evaluasi.

### **BAB 5 : IMPLEMENTASI**

Pada bab ini, menjelaskan tentang implementasi sistem. Dalam implementasi ini berisi batasan implementasi, proses implementasi, penggunaan algoritme pada sistem, dan proses evaluasi.

### **BAB 6 : PENGUJIAN DAN ANALISIS**

Pada bab ini, berisi pembahasan ruang lingkup pengujian yang nantinya akan dianalisis. Konteks pengujian dan analisis diurutkan berdasarkan perancangan dan implementasi pada bab sebelumnya.

## **BAB 7**

### **: KESIMPULAN DAN SARAN**

Pada bab ini, berisi pemaparan kesimpulan umum dari laporan penelitian dengan saran yang dapat digunakan untuk memperbaiki dan menjadi pedoman dalam penelitian lebih lanjut.



## **BAB 2 LANDASAN KEPUSTAKAAN**

Bab ini berisi pembahasan mengenai kajian pustaka dan teori-teori dasar yang digunakan dalam penulisan skripsi. Kajian pustaka dalam penelitian ini adalah membahas penelitian yang sudah ada dan membandingkan dengan penelitian yang diusulkan. Dasar teori yang dipelajari disini terkait teori penjabaran *text mining*, *sentiment analysis*, *pre-processing*, perbaikan kata tidak baku, dan penjelasan megenai metode Naive Bayes.

### **2.1 Kajian Pustaka**

Pada pembahasan kajian pustaka ini akan dibahas mengenai penelitian terkait yang sudah pernah dilakukan sebelumnya, diantaranya penelitian analisis sentimen yang dilakukan oleh Firmansyah (2016). Pada penelitian ini metode yang digunakan adalah Naive Bayes yang digunakan untuk mengklasifikasikan sentimen dari dokumen *review* aplikasi *mobile*. Selain Naive Bayes pada penelitian ini juga digunakan metode Query Expansion untuk mengoptimasi komentar singkat. Hasil pengujian pada penelitian ini menunjukkan bahwa dengan menggunakan algoritme Naive Bayes didapatkan hasil akurasi yang baik yaitu sebesar 95%. Dan dengan penambahan penggunaan metode Query Expansion menambah nilai akurasi dari Naive Bayes yaitu menjadi 98% (Firmansyah, 2016). Namun pada penelitian ini masih memiliki kekurangan yaitu masih belum dilakukannya proses normalisasi bahasa atau perbaikan kata tidak baku, sehingga menyebabkan menurunnya tingkat akurasi.

Kemudian penelitian selanjutnya terkait dengan analisis sentimen dilakukan oleh Nugraha (2014). Pada penelitian ini dilakukan pengklasifikasian terhadap sentimen positif dan negatif dari *review* film berbahasa Inggris dengan menggunakan metode K-NN dengan jumlah data sebanyak 700 dokumen. Pada penelitian ini membandingkan metode K-NN dengan metode Naive Bayes. Hasil dari akurasi yang didapat menunjukkan bahwa metode Naive Bayes lebih baik dibandingkan dengan metode K-NN. Dihasilkan metode Naive Bayes memiliki nilai akurasi tertinggi sebesar 81% dibandingkan dengan menggunakan metode K-NN yang bernilai akurasi sebesar 71%. Dan pada penelitian ini juga masih belum dilakukannya perbaikan kata tidak baku, sehingga tingkat akurasi tertinggi yang dicapai untuk metode Naive Bayes hanya sebesar 81%.

Kemudian penelitian selanjutnya yaitu terkait analisis sentimen dengan penanganan singkatan kata dan ejaan kata yang salah, yaitu penelitian yang dilakukan oleh Manalu (2014). Pada penelitian ini dilakukan pengklasifikasian terhadap sentimen positif, sentimen negatif, dan sentimen netral mengenai analisis sentimen pada Twitter tentang provider telekomunikasi. Metode yang digunakan pada penelitian ini yaitu dengan menggunakan algoritme Naive Bayes. Selain itu pada penelitian ini juga dilakukan seleksi fitur dengan menggunakan proses *text mining* serta proses n-gram karakter untuk menangani singkatan kata

dan ejaan kata yang salah. Hasil pengujian yang dihasilkan mencapai 93% dari jumlah 100 tweet yang diuji dengan 2700 *data training*. Namun pada penelitian ini masih belum dilakukan perbaikan kata tidak baku untuk bahasa modern (*slang*).

## 2.2 Twitter

Twitter adalah media sosial dengan layanan *microblogging* yang memungkinkan penggunanya untuk berbagi informasi dengan sesama secara *realtime*. Informasi yang dibagikan pada Twitter biasanya disebut dengan kicauan (Tweet) yang dibatasi sampai 140 karakter (Kwak, et al., 2010). Karakter yang dibatasi tersebut menyebabkan sebuah tweet sering kali mengandung singkatan, bahasa *slang* maupun kesalahan pengejaan (Agarwal, et al., 2014). Sejak awal, Twitter memang diciptakan sebagai layanan berbasis *mobile* yang didesain sesuai dengan batasan karakter pada sebuah pesan teks (SMS), dan sampai hari ini, Twitter masih bisa digunakan pada sembarang telepon genggam yang memiliki kemampuan untuk mengirim dan menerima pesan teks (The Twitter Government and Election Team, 2014).

Twitter dibuat dengan tujuan dapat dijadikan tempat untuk saling berbagi pengalaman antar sesama penggunanya tanpa adanya penghalang. Dengan menggunakan, pengguna akan mudah untuk mengikuti tren, cerita, informasi dan berita dari seluruh penjuru dunia. Selain itu, Twitter juga membantu penggunanya untuk selalu terhubung dengan orang-orang terdekatnya. Ketika penggunanya mengirimkan Tweet, pesan tersebut bersifat publik dan bisa diakses oleh siapapun, dimanapun dan kapanpun. Bahkan, bagi orang-orang yang mengikuti (*follow*) akun Twitter tersebut, Tweet tersebut akan secara otomatis muncul di lini masa orang tersebut.

## 2.3 Opini Film

Pengertian opini menurut Kamus Besar Bahasa Indonesia (KBBI) adalah pendapat, pikiran atau pendirian. Pendapat sendiri merupakan pandangan seseorang mengenai suatu hal, yaitu bersifat subjektif dan bukan fakta. Opini juga mempunyai pengertian sebagai suatu pandangan keputusan atau kesimpulan yang ada dalam pikiran dan dapat dikeluarkan dengan pernyataan yang diucapkan atau ditulis (Moore, 1987). Opini dapat digunakan sebagai sarana untuk memberikan penilaian terhadap suatu film yaitu berupa opini film. Opini film digunakan untuk mengetahui pendapat seseorang mengenai film apakah cenderung bagus atau sebaliknya tergantung pada sudut pandang dan latar belakang yang dimiliki, karena pendapat seseorang mengenai suatu hal dapat berbeda-beda.

## 2.4 Text Mining

*Text mining* atau yang dapat disebut juga dengan penggalian data teks, teks cerdas, atau pencarian pengetahuan dalam bentuk teks merupakan ilmu untuk memproses dan menganalisa data dalam jumlah yang besar untuk mengekstrak

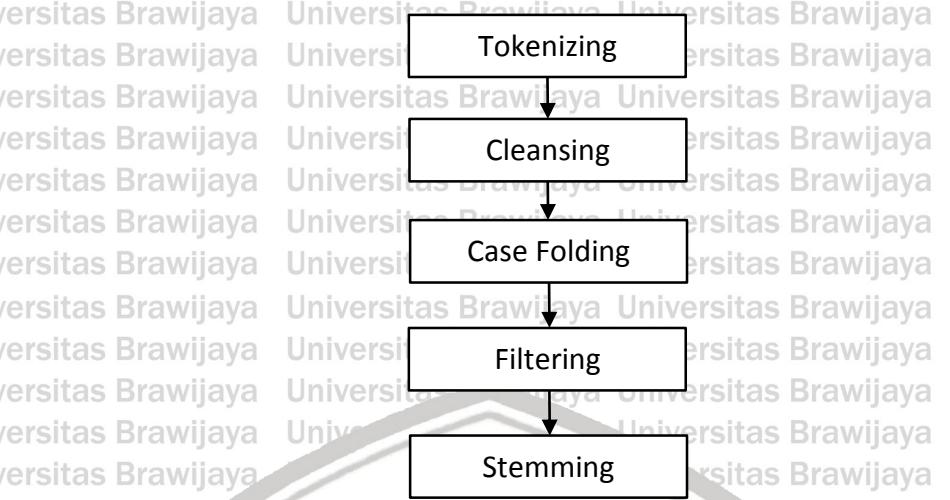
informasi yang sebelumnya tidak diketahui (Adiwijaya, 2006). *Text mining* atau penggalian teks adalah sebuah metode turunan dari *data mining* yang memiliki tujuan untuk mencari pola atau mencari informasi menarik dari kumpulan data yang berbentuk *natural language text*. *Text mining* memberi cara untuk melakukan jelajah data yang berukuran sangat besar, dan data tersebut dihasilkan dari masyarakat yang bergantung pada komputer. Dalam hal ini *text mining* dapat berupa analisis untuk mendapatkan informasi melalui kata-kata yang penting secara semantik dan kalimat-kalimat yang relevan (Olson & Shi, 2008). Jika dibandingkan dengan data yang tersimpan di dalam database, data berbentuk teks memiliki karakteristik yang tidak teratur, tidak terstruktur, dan sulit untuk diolah dengan cara biasa, padahal dalam perkembangan dunia modern, teks menjadi salah satu bentuk vital dalam pertukaran data dan informasi antar pengguna.

Seperti halnya *data mining* yang bertujuan untuk mencari pola unik di dalam data, *text mining* juga memiliki tujuan untuk mencari pola yang unik dalam teks. Perbedaannya adalah, *data mining* dapat diartikan mencari informasi yang implisit atau tersembunyi yang sebelumnya tidak diketahui, dan berpotensi guna dalam sebuah data (Witten, et al., 2011). Sedangkan *text mining* mencari informasi yang sebenarnya sudah secara eksplisit dapat terlihat dan dimengerti oleh manusia, namun tantangannya adalah bagaimana caranya informasi ini mampu direpresentasikan dengan baik tanpa menghilangkan data penting sehingga mampu diolah oleh komputer menggunakan algoritme yang ada. Dengan potensi yang sangat besar tersebut, *text mining* dapat diaplikasikan ke dalam berbagai macam kondisi dan kasus di dunia nyata, salah satunya adalah *sentiment analysis*. Dalam hal ini prinsip *text mining* adalah ilmu yang melibatkan analisis teks, pemrosesan bahasa alami, temu kembali informasi, pengelompokan, ekstraksi informasi, pengkategorian, *visualization*, teknologi basis data, *data mining* dan *machine learning* (Utomo, 2015). *Text mining* memberikan solusi berdasarkan permasalahan seperti pemrosesan, pengelompokan, dan menganalisis teks yang tidak terstruktur dalam jumlah yang besar.

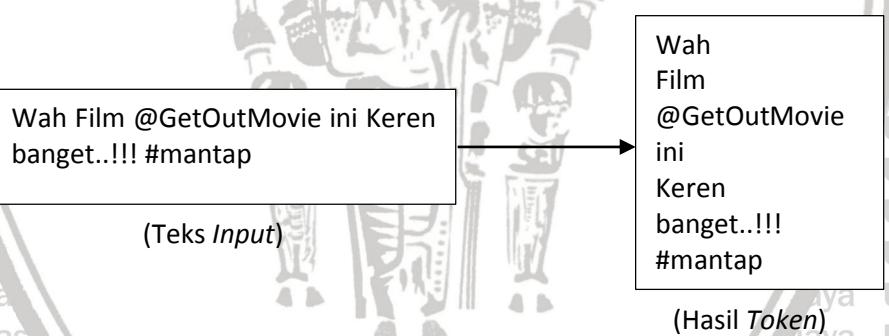
## **2.5 Text Pre-processing**

Pada *text mining* diperlukan beberapa tahapan untuk mengolah teks menjadi lebih terstruktur. Salah satu tahapan pada *text mining* adalah *text pre-processing*.

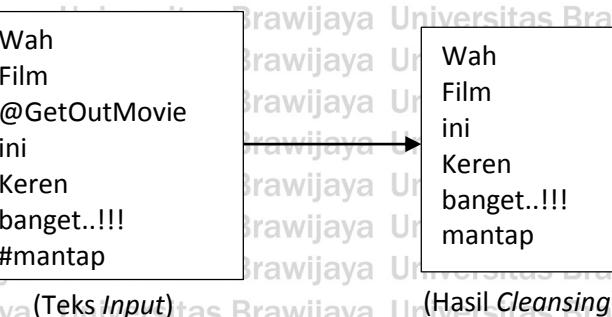
Tahap ini adalah tahapan yang mana data disiapkan agar menjadi data yang siap untuk dianalisis (Hadna, et al., 2016). Pada data yang belum dilakukan *pre-processing* masih menjadi data mentah, yaitu data yang masih belum siap untuk dilakukan analisis, karena masih banyak mengandung kata-kata yang tidak memiliki arti dan kata yang belum terstruktur. Sehingga diperlukan *text pre-processing* untuk mengolah teks menjadi lebih terstruktur. Pada tahapan *pre-processing* meliputi proses *tokenizing*, *cleansing*, *case folding*, *filtering*, dan *stemming*. Tahapan *pre-processing* ditunjukkan pada Gambar 2.1.

**Gambar 2.1 Tahapan Pre-processing****2.5.1 Tokenizing**

Pada tahap *tokenizing / parsing* merupakan tahap yang dilakukan dengan cara melakukan pemotongan string input atau memecah kalimat menjadi beberapa bagian atau kata (Manning, et al., 2009). Hasil dari pemotongan kata dinamakan dengan *token*. *Tokenizing* adalah tahap pemecahan kalimat dari inputan berdasarkan tiap kata penyusunnya. Contoh pada *tokenizing* dapat dilihat pada Gambar 2.2.

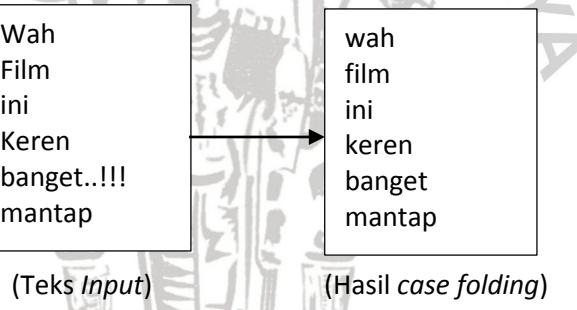
**Gambar 2.2 Tahap Tokenizing****2.5.2 Cleansing**

Tahapan *cleansing* merupakan tahapan yang dilakukan untuk menghilangkan *noise* (karakter yang tidak penting) pada dokumen teks. Pada dokumen teks Tweet terdapat varian variabel atau karakter yang perlu dihilangkan karena tidak memiliki pengaruh dalam pemrosesan teks. Varian variabel pada Tweet, diantaranya adalah *link URL* (<http://>), *username* (@), *hashtag* (#), dan *retweet* (RT). Contoh *cleansing* dapat dilihat pada Gambar 2.3.



## **ambar 2.3 Tahap *Cleansing***

### 2.5.3 Case folding



## **ambar 2.4 Tahap *Case Folding***

### **2.5.4 Filtering**

Tahapan *filtering* merupakan tahapan yang dilakukan untuk mengambil kata-kata yang dianggap penting dari hasil *token* yang sudah dilakukan proses *cleansing* dan *case folding*, kata-kata tersebut digunakan untuk mewakili suatu dokumen. Tahap *filtering* dilakukan dengan melakukan proses penghapusan *term* yang tidak memiliki arti atau tidak *relevan* (Amin, 2012). Pada *filtering* dapat menggunakan algoritme *stoplist*. Pada algoritme *stoplist* adalah membuang kata yang kurang penting atau dianggap tidak penting dalam pendekatan *bag-of-words*. Yaitu jika pada pengecekan *database* kata tersebut tidak penting maka akan dilakukan proses penghilangan kata-kata yang tidak diperlukan. *Stoplist* yang digunakan dalam penelitian ini yaitu bersumber dari Tala (2003). Contoh tahap *filtering* dapat dilihat pada Gambar 2.5.

**Gambar 2.5 Tahap Filtering**

### 2.5.5 Stemming

*Stemming* merupakan proses dalam pengubahan bentuk kata yang berimbuhan menjadi kata dasar, *stemming* selain digunakan untuk memperkecil jumlah indeks juga digunakan untuk mengelompokkan kata-kata lain yang memiliki bentuk berbeda karena perbedaan imbuhan namun memiliki kata dasar dan arti yang sama. Implementasi pada tahap *stemming* memiliki berbagai metode yang dapat digunakan, tergantung pada bahasa dari dokumen. Contoh tahap *stemming* dapat dilihat pada Gambar 2.6.

**Gambar 2.6 Tahap Stemming**

## 2.6 Perbaikan Kata Tidak Baku

Perbaikan kata tidak baku atau normalisasi bahasa adalah proses yang digunakan untuk mengubah kata-kata yang tidak baku menjadi kata baku sesuai dengan Kamus Besar Bahasa Indonesia (KBBI). Pada penelitian sebelumnya proses normalisasi meliputi (Buntoro, et al., 2014):

1. Merenggangkan tanda baca (*punctuation*) dan simbol selain *alphabet*  
Pada proses ini merenggangkan tanda baca dilakukan dengan cara memberikan jarak terhadap tanda baca dari kata-kata sebelumnya atau sesudahnya, dengan tujuan agar tanda baca dan simbol selain *alphabet* tidak menjadi satu dengan kata-kata pada saat proses tokenisasi.
2. Mengubah kata menjadi huruf kecil semua

### 3. Normalisasi Kata

Pada proses normalisasi kata dilakukan dengan cara mengubah kata yang tidak baku menjadi baku sesuai dengan pedoman yang ada pada KBBI. Sebagai contoh ketika seseorang melakukan Tweet terkadang masih banyak yang tidak menggunakan kata baku, misalnya menuliskan kata “terima kasih” menjadi makasi (terima kasih->kata baku, makasi->kata tidak baku).

### 4. Menghilangkan huruf yang berulang

Pada proses ini dilakukan dengan menghilangkan huruf yang berulang. Sebagai contoh ketika seseorang merasa senang atau kesal, terkadang mereka melakukan Tweet dengan mengulang huruf yang sama pada kata tersebut, seperti penulisan “Keeereeennnn” maka akan dinormalisasi menjadi “Keren”.

### 5. Menghilangkan *emoticon*

Pada twitter, penggunaan *emoticon* sering dilakukan untuk mengekspresikan perasaan pengguna. Namun dalam penggunaan *emoticon* tersebut terkadang tidak sesuai dengan maksud dari apa yang sebenarnya. Sebagai contoh ketika pengguna melakukan Tweet terhadap respon film “film ini bagus banget :(“ kata opini bagus namun digunakan *emoticon* :(, hal ini akan mengganggu dalam proses analisis sentimen. Sehingga dalam penelitian ini *emoticon* akan diabaikan saja atau dihapus.

Pada perbaikan kata tidak baku terdapat bermacam-macam kata yang dianggap kata tidak baku. Macam-macam kata tidak baku tersebut seperti penulisan dengan bahasa modern atau *slang* (ex : gue = saya ), penulisan dengan kaidah atau pedoman bahasa (ex : kwalitas = kualitas), penulisan dengan kata singkat (ex : adlh = adalah), penulisan dengan angka (ex : n41k = naik), dan penulisan dengan kata salah eja (ex : jelsk = jelek).

Pada penelitian ini digunakan perbaikan kata tidak baku untuk menanganai permasalahan kata tidak baku hanya pada bahasa modern atau *slang*, penyingkatan kata, dan salah eja. Perbaikan kata tidak baku dilakukan dengan proses normalisasi yang dilakukan dengan melakukan pengecekan *token-token* pada kalimat dengan menggunakan kamus, jika terdapat kata yang tidak baku maka akan diubah menjadi kata yang baku sesuai dengan kamus. Selain itu juga dilakukan dengan normalisasi Levenshtein Distance, yaitu untuk memperbaiki kesalahan dalam penulisan kata. Sehingga didapatkan hasil akhir berupa kalimat dengan menggunakan kata baku.

#### 2.6.1 Levenshtein Distance

Levenshtein distance atau yang biasa disebut dengan *edit distance* adalah suatu metode yang dapat digunakan untuk mengatasi terjadinya kesalahan ejaan.

Kesalahan ejaan terjadi apabila kata yang diketik oleh pengguna tidak terdapat pada daftar kamus Bahasa Indonesia. Fungsi metode Levenshtein Distance yaitu

untuk menghitung jarak kedekatan dari dua buah *string* melalui penambahan karakter, pengubahan karakter, dan penghapusan karakter hingga kedua *string* tersebut cocok (Freeman, et al., 2006).

Pada algoritme Levenshtein Distance terdapat 3 macam operasi utama yang dilakukan yaitu (Adriyani, et al., 2012):

### 1. Operasi Penambahan Karakter

Operasi penambahan karakter yaitu operasi yang digunakan untuk menambahkan karakter ke dalam *string*. Contoh pada penulisan *string* ‘kern’ maka diubah menjadi *string* ‘keren’ dengan menambahkan karakter ‘e’.

### 2. Operasi Pengubahan Karakter

Operasi pengubahan karakter yaitu operasi yang digunakan untuk mengubah karakter dengan cara menukar sebuah karakter dengan karakter lain. Contoh pada penulisan *string* ‘hidsp’ diubah menjadi *string* ‘hidup’ dengan mengubah karakter ‘s’ menjadi karakter ‘u’.

### 3. Operasi Penghapusan Karakter

Operasi penghapusan karakter yaitu operasi yang digunakan untuk menghapus suatu karakter pada *string*. Contoh pada penulisan *string* ‘hebatt’ di ubah menjadi *string* ‘hebat’ dengan menghilangkan karakter ‘t’.

Pada algoritme Levenshtein Distance dilakukan tahapan proses yang dimulai dari atas pojok kiri dari sebuah array dua dimensi yang telah diisi karakter *string* input dan *string* target. Selain itu juga terdapat nilai *cost* didalamnya. Nilai *cost* yang berada pada bawah pojok kanan merupakan nilai *edit distance* yang menggambarkan jumlah perbedaan dari kedua *string*. Contoh perhitungan *edit distance* dapat dilihat pada Gambar 2.7.

		F	I	L	M
F	0	1	2	3	4
I	1	0	1	2	3
L	2	1	1	1	2
M	3	2	2	2	1

Gambar 2.7 Matriks perhitungan *Edit distance*

Gambar 2.7 menunjukkan perhitungan *edit distance* menggunakan 2 *string* yang berbeda. Nilai *edit distance* yang dihasilkan adalah bernilai 1 ditandai dengan garis diagonal pada pojok kanan bawah. Pengecekan dilakukan dari iterasi awal dari kedua *string* kemudian dilakukan operasi pengubahan, penambahan, dan penghapusan karakter. Pada proses ini hanya terdapat 1 proses yaitu penyisipan karakter ‘I’ pada *string* ‘FLM’. Sehingga menjadi *string* ‘FILM’.

## 2.7 Pembobotan Kata

Pembobotan kata merupakan suatu mekanisme yang digunakan untuk memberi bobot atau skor terhadap frekuensi kemunculan sebuah kata pada dokumen teks. Salah satu metode yang populer dalam melakukan pembobotan kata adalah TF-IDF (*Term Frequency-Inverse Document Frequency*). TF-IDF merupakan sebuah metode pembobotan yang menggabungkan dua konsep, yaitu *Term Frequency* dan *Document Frequency* (Asrofi, 2015). Namun dalam penelitian ini hanya menggunakan *Term Frequency* untuk pembobotan kata yang nantinya digunakan untuk klasifikasi Naive Bayes. *Term Frequency* adalah sebuah konsep pembobotan dengan melihat seberapa sering (*frekuensi*) munculnya sebuah *term* dalam satu dokumen (Huang & Wu, 2013). Dikarenakan setiap dokumen memiliki panjang yang berbeda-beda, bisa saja terjadi sebuah kata muncul lebih banyak di dokumen yang panjang dibandingkan dengan dokumen-dokumen yang pendek. Sehingga jika semakin besar kemunculan kata dalam dokumen akan memberikan nilai kesesuaian yang semakin besar. Dan semakin tinggi nilai *Term Frequency* suatu kata pada dokumen, maka semakin besar pula pengaruh kepentingan *term* terhadap dokumen tersebut. Berikut adalah persamaan skor *Term Frequency*, dapat dirumuskan pada Persamaan 2.1.

$$w(t, d) = tf(t, d)$$

Keterangan:

$w(t, d)$  : Bobot kata  $t$  pada dokumen  $d$

$tf(t, d)$  : Jumlah kemunculan kata  $t$  pada dokumen  $d$

## 2.8 Sentiment Analysis

*Sentiment Analysis* atau dengan kata lain *opinion mining* merupakan sebuah teknik yang dapat digunakan untuk melakukan identifikasi pada sebuah sentimen yang diperlihatkan dengan menggunakan teks. Yang mana sentimen tersebut dapat dikategorikan ke dalam sentimen negatif atau positif (Nasukawa & Yi, 2003). Selain itu *sentiment analysis* adalah proses yang digunakan untuk menentukan opini, emosi dan sikap yang dicerminkan melalui teks, dan biasanya diklasifikasikan menjadi opini negatif dan positif (Coletta, et al., 2014). Dan *sentiment analysis* dapat diartikan juga sebagai daerah penelitian perhitungan untuk mengekstraksi polaritas pendapat antar kelas (positif dan negatif) dari dokumen teks (Perdana & Pinandito, 2017). Hal ini dapat disimpulkan bahwa analisis sentimen adalah sebuah proses untuk menentukan sentimen atau opini dari seseorang yang diwujudkan dalam bentuk teks dan bisa dikategorikan sebagai sentimen positif atau negatif.

(2.1)

## 2.9 Naive Bayes Classifier

*Naive Bayes Classifier* merupakan salah satu metode yang populer untuk keperluan *data mining* karena penggunaannya yang mudah (Hall, 2006) dan dalam pemrosesan memiliki waktu yang cepat, mudah diimplementasikan dengan strukturnya yang cukup sederhana dan untuk tingkat efektifitasnya memiliki efektifitas yang tinggi (Taheri & Mammadov, 2013). Klasifikasi yang berdasar pada teorema bayes sangat cocok digunakan untuk dimensi masukan yang sangat besar. Klasifikasi Naive Bayes juga memperlihatkan tingginya akurasi dan cepat ketika digunakan untuk dataset dengan jumlah besar (Aggarwal, 2015).

Secara umum proses dari klasifikasi Naive Bayes dapat dilihat pada Persamaan

### 2.2.

$$P(c_j|w_i) = \frac{P(c_j) P(w_i|c_j)}{P(w_i)} \quad (2.2)$$

Keterangan:

$P(c_j|w_i)$  : Peluang kategori  $j$  ketika terdapat kemunculan kata  $i$

$P(w_i|c_j)$  : Peluang sebuah kata  $i$  masuk ke dalam kategori  $j$

$P(c_j)$  : Peluang kemunculan sebuah kategori  $j$

$P(w_i)$  : Peluang kemunculan sebuah kata

Pada proses perhitungan klasifikasi peluang kemunculan kata sebenarnya dapat dihilangkan, hal ini dikarenakan peluang tersebut tidak berpengaruh pada perbandingan hasil klasifikasi dari setiap kategori. Sehingga proses pada klasifikasi dapat disederhanakan dengan Persamaan 2.3.

$$P(c_j|w_i) = P(c_j) P(w_i|c_j) \quad (2.3)$$

Pada perhitungan klasifikasi terdapat *prior* yang digunakan untuk menghitung peluang kemunculan kategori pada semua dokumen, perhitungan prior dapat dilihat pada Persamaan 2.4.

$$P(c_j) = \frac{N_c}{N} \quad (2.4)$$

Keterangan:

$N_c$  : Banyak dokumen berkategori  $c_j$  pada dokumen latih

$N$  : Jumlah keseluruhan dokumen latih yang digunakan

Perhitungan *posterior* merupakan perhitungan yang dilakukan dengan mengalikan *prior* dengan *total conditional probability*. Rumus perhitungan dapat dilihat pada Persamaan 2.5.

### 2.9.1 Gaussian Naive Bayes

Gaussian Naive Bayes adalah algoritme yang dapat dipilih untuk merepresentasikan probabilitas bersyarat dari fitur *continuous* pada sebuah kelas  $P(X_i | Y)$ . Gaussian Naive Bayes memiliki karakteristik dua parameter yaitu rata-rata dan varian.

### 2.9.2 Bernoulli Naive Bayes

Bernoulli Naive Bayes adalah algoritme tipe klasifikasi yang populer yang digunakan untuk mengklasifikasikan teks singkat (*short-text*). Pada klasifikasi ini menggunakan *binary* (1 dan 0) untuk pembobotan tiap *term*, berbeda dengan perhitungan *term frequency* yang melakukan pembobotan pada tiap *term*.

### 2.9.3 Multinomial Naive Bayes

Multinomial Naive Bayes adalah *conditional probability* yang dilakukan tanpa memperhitungkan urutan kata dan informasi yang ada dalam kalimat atau dokumen secara umum. Dan pada algoritme ini juga memperhitungkan jumlah kata yang muncul dalam dokumen (Destuardi dan Surya, 2009).

Algoritme *conditional probability* Multinomial Naive Bayes dalam melakukan perhitungan peluang sebuah kata  $i$  masuk ke dalam kategori  $j$  dapat dilakukan dengan menggunakan Persamaan 2.6.

$$P(w_i | c_j) = \frac{\text{count}(w_i, c_j) + 1}{(\sum_{w \in V} \text{count}(w, c_j)) + |V|}$$

Keterangan :

Bawa  $\text{count}(w_i, c_j)$  merupakan jumlah dari suatu kata *query* yang muncul dalam suatu kelas atau kategori. Dan penambahan angka 1 tersebut digunakan untuk menghindari nilai nol. Selanjutnya  $\sum_{w \in V} \text{count}(w, c_j)$  merupakan jumlah dari seluruh kata yang ada pada kelas atau kategori  $c_j$ . Dan  $|V|$  merupakan jumlah dari seluruh kata unik yang ada pada seluruh kategori.

## 2.10 Evaluasi

Evaluasi adalah sebuah aktivitas yang dilakukan untuk membandingkan hasil implementasi dengan kriteria standar yang telah ditetapkan. Tujuan dari evaluasi ini yaitu untuk mengetahui selisih antara hasil yang dicapai dengan standar yang ditetapkan. Selain itu tujuan terpenting dilakukannya evaluasi ini adalah untuk mengetahui tingkat keberhasilan. Evaluasi dapat dilakukan dengan cara menghitung *accuracy*, *precision*, *recall*, dan *f-measure* untuk mengukur kinerja sistem pada kasus klasifikasi. Kinerja sistem dapat dihitung menggunakan tabel

*kontingensi* (Feizar, 2014). Pada tabel *kontingensi* berisi data prediksi dan data aktual yang telah diklasifikasikan oleh sistem. Tabel *kontingensi* ditunjukkan oleh

Tabel 2.1

		<i>Actual class (expectation)</i>	
<i>Prediction Class (Observation)</i>	<i>Class</i>	+	-
		+	TP
	-	FN	TN

**Tabel 0.1 Tabel Kontingensi**

Tabel 2.1 menunjukkan data kelas *prediction* dan kelas *actual* pada klasifikasi, yaitu:

- *True Positive* (TP) adalah kasus dengan kelas positif yang diklasifikasikan secara benar dengan hasil positif.
- *True Negative* (TN) adalah kasus dengan kelas negatif yang diklasifikasikan secara benar dengan hasil negatif.
- *False Positive* (FP) adalah kasus kelas negatif yang diklasifikasikan pada kelas positif.
- *False Negative* (FN) adalah kasus kelas positif yang diklasifikasikan pada kelas negatif.

Beberapa indikator tersebut didefinisikan dalam persamaan di bawah ini:

$$\text{accuracy} = \frac{TN + TP}{TN + TP + FP + FN} \quad (2.7)$$

*Accuracy* adalah tingkat kedekatan antara nilai prediksi dengan nilai aktual.

Dengan semua keadaan yang diprediksi benar terhadap semua keadaan yang diprediksi. Apabila mendapat nilai *accuracy* 100%, maka hal ini menunjukkan bahwa kondisi yang diprediksi itu benar seperti pada kondisi aslinya.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.8)$$

*Precision* adalah tingkat ketepatan hasil klasifikasi dari seluruh dokumen.

*Precision* dihitung dengan cara membagi jumlah pengenalan yang bernilai benar oleh sistem dengan jumlah keseluruhan pengenalan yang dilakukan oleh sistem (Destuardi dan Surya, 2009). Secara matematis pengukuran *precision* yaitu jumlah sampel berkategori positif yang diklasifikasi benar dibagi dengan total sampel yang diklasifikasi sebagai sampel positif.



## UNIVERSITAS BRAWIJAYA



$$Recall = \frac{TP}{TP + FN}$$

*Recall* adalah tingkat keberhasilan mengenali suatu kelas yang harus dikenali.

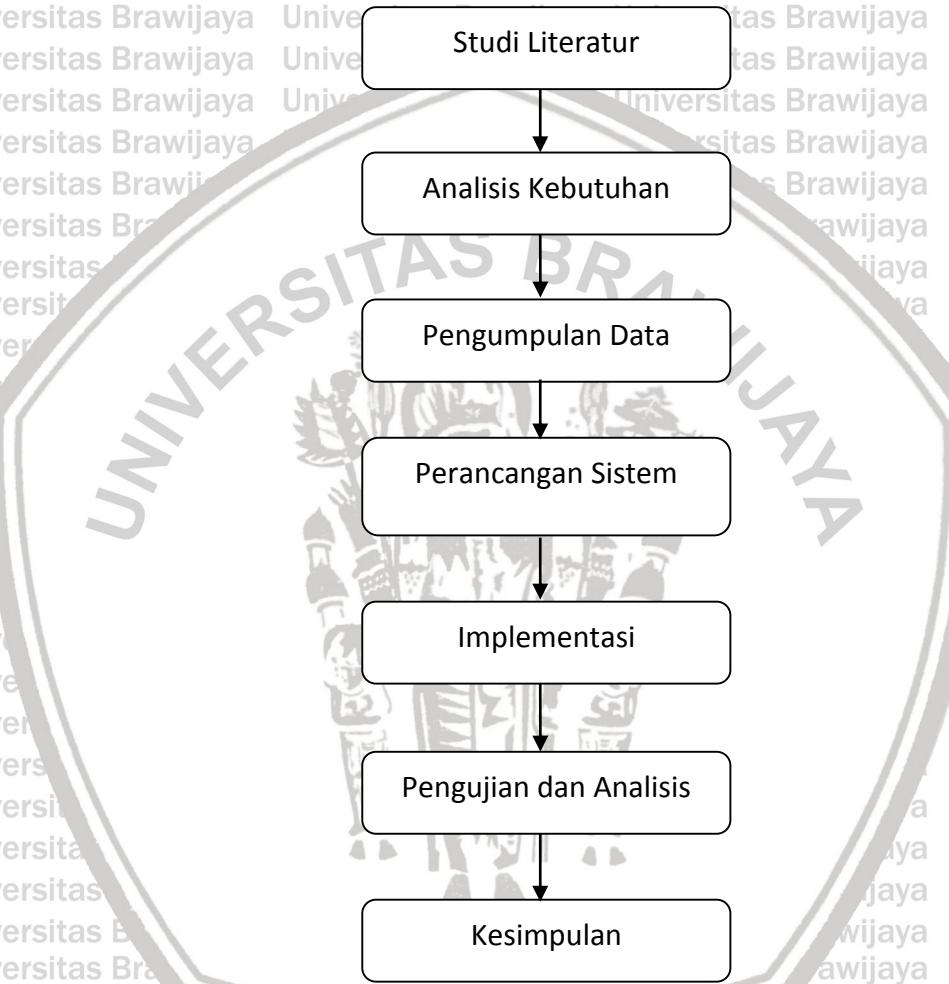
Secara matematis pengukuran *recall* yaitu jumlah sampel diklasifikasi positif dibagi total sampel dalam data uji berkategori positif.

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

*F – measure* adalah penggabungan *precision* dan *recall*. Atau nilai yang mewakili keseluruhan kinerja sistem.

### BAB 3 METODOLOGI PENELITIAN

Pada bab metode penelitian ini berisi pembahasan tentang metodologi yang digunakan dalam penelitian berjudul “Analisis Sentimen Tentang Opini Film Pada Dokumen Twitter Berbahasa Indonesia Menggunakan Naive Bayes Dengan Perbaikan Kata Tidak Baku”. Berikut adalah tahapan-tahapan metode penelitian yang dijelaskan pada Gambar 3.1, yaitu:



Gambar 3.1 Tahapan Penelitian

Berdasarkan Gambar 3.1, dijelaskan tahapannya sebagai berikut:

1. Studi literatur dilakukan untuk mempelajari hal yang berkaitan dengan analisis sentimen dengan menggunakan metode Naive Bayes dan perbaikan kata tidak baku.
2. Analisis kebutuhan dilakukan untuk menentukan apa saja yang diperlukan untuk membangun sistem, baik kebutuhan fungsional, kebutuhan perangkat, serta kebutuhan data variabel.

3. Melakukan pengumpulan data yaitu data Tweet mengenai opini film berbahasa Indonesia.
4. Melakukan perancangan sistem yaitu terkait dengan hasil dari analisis kebutuhan.
5. Melakukan implementasi sistem berdasarkan hasil dari perancangan.
6. Melakukan pengujian sistem terhadap sistem dengan menganalisis hasil keluaran sistem, apakah sistem tersebut sudah baik atau masih membutuhkan perbaikan.
7. Membuat kesimpulan dari keseluruhan metode penelitian.

### **3.1 Studi Literatur**

Pada penelitian ini dibutuhkan studi literatur yang digunakan untuk memahami teori dasar dan metode untuk menunjang pelaksanaan penelitian terhadap permasalahan yang diangkat. Studi Literatur yang diambil berupa buku, jurnal, e-book, dan penelitian terkait lainnya. Referensi yang dibutuhkan dipelajari adalah:

1. Twitter
2. Opini Film
3. *Text Mining*
4. *Text Pre-processing*
5. Perbaikan Kata Tidak Baku
6. *Sentiment Analysis*
7. *Naive Bayes Classifier*

### **3.2 Analisis kebutuhan**

Analisis kebutuhan digunakan untuk menentukan apa saja yang diperlukan untuk membangun sistem, baik kebutuhan fungsional, kebutuhan perangkat, serta kebutuhan data variabel.

#### **3.2.1 Kebutuhan fungsional**

Kebutuhan fungsional merupakan kebutuhan yang disediakan untuk pengguna dan harus ada didalam sistem yang akan dibuat. Kebutuhan fungsional yang ada dalam penelitian ini adalah sebagai berikut:

1. Sistem dapat melakukan *pre-processing*.
2. Sistem dapat mengoreksi kesalahan penulisan pada opini.
3. Sistem dapat mengklasifikasikan opini ke dalam sentimen yang telah ditentukan yaitu sentimen negatif atau positif.

### 3.2.2 Kebutuhan Perangkat

1. Kebutuhan perangkat keras (*Hardware*) yang digunakan adalah sebagai berikut:

#### a. Laptop

- Processor Intel® Core™ i3-2330M CPU @ 2.20GHz

- Hard Disk 500GB

- RAM 4 GB

- XAMPP Server

2. Kebutuhan perangkat lunak (*Software*) yang digunakan adalah sebagai berikut:

#### a. Sistem Operasi

- Menggunakan OS Windows 7 Ultimate

#### b. Bahasa Pemrograman

- Phyton

#### c. DBMS

- Menggunakan database management MySQL

#### d. Editor

- JetBrains Pycharm Community Edition 2016.3.2

#### e. Library

- Twitter, Sastrawi, pymysql

### 3.2.3 Kebutuhan Data

Data yang dibutuhkan dalam penelitian ini adalah data yang berasal dari Twitter yaitu berupa Tweet atau komentar maupun opini mengenai film berbahasa Indonesia berbentuk dokumen teks. Data ini nantinya dapat digunakan sebagai data latih dan data uji. Selain itu juga dibutuhkan kamus yang berisi daftar kata berbahasa Indonesia yang meliputi kamus\_katabaku, stoplist dan wordlist.

### 3.3 Pengumpulan Data

Pada tahap pengumpulan data dapat dilakukan dengan cara penggunaan data sekunder atau primer. Data opini film berbahasa Indonesia yang digunakan pada penelitian ini adalah jenis data sekunder, yaitu data diperoleh dari sumber yang tidak langsung. Data opini film berbahasa Indonesia didapat dengan menggunakan API Twitter terkait dengan Tweet masyarakat mengenai komentar atau *review* atau opini film. Data tersebut akan digunakan sebagai data latih dan data uji.

### 3.4 Perancangan Sistem

Pada perancangan sistem ini dilakukan setelah proses pada tahap analisis selesai dilakukan, selanjutnya akan dirancang sebuah langkah kerja secara keseluruhan dalam sistem yang akan dibuat. Terdapat beberapa tahapan yang

akan dijalankan oleh sistem. Tahapan pertama yaitu dengan memasukkan dokumen berupa sentimen pengguna. Selanjutnya dokumen tersebut akan dilakukan proses *pre-processing*, pada proses *pre-processing* terdapat tambahan perbaikan kata tidak baku yaitu dengan melakukan pencocokan dengan kamus\_katabaku. Setelah dilakukan proses *pre-processing* maka langkah selanjutnya dilanjutkan dengan koreksi kesalahan ejaan kata menggunakan normalisasi Levenshtein Distance yaitu untuk memperbaiki kata tidak baku seperti kesalahan pada penulisan kata. Setelah itu baru dapat dilakukan klasifikasi dokumen sentimen dengan menggunakan metode Naive Bayes. Kemudian didapatkan hasil klasifikasi dari dokumen Tweet apakah termasuk kelas positif atau negatif.

### **3.5 Implementasi**

Implementasi sistem merupakan tahapan yang dilakukan setelah perancangan telah selesai dilakukan. Kemudian dari hasil perancangan akan dibuat kode program. Dalam implementasi sistem ini proses yang dilakukan pertama adalah mengimplementasikan metode sesuai dengan studi literatur yang sudah dipahami. Yaitu bahasa yang digunakan dalam implementasi ini adalah bahasa pemrograman Python, dengan penggunaan *software Pycharm* dan *database management* adalah MySQL. Selanjutnya akan dilakukan implementasi *pre-processing* dokumen, perbaikan kata tidak baku, dan melakukan pengklasifikasian dengan menggunakan metode *Naive Bayes Classifier*.

### **3.6 Pengujian dan Analisis**

Pengujian dilakukan untuk melihat seberapa baik hasil akurasi dari algoritme *Naive Bayes Classifier* yang dihasilkan dengan cara memasukkan data uji ke dalam sistem untuk dapat menentukan hasil dari analisis sentimen opini film pada dokumen Twitter berbahasa Indonesia. Berbagai macam pengujian dilakukan untuk mengetahui pengaruh parameter yang diuji. Diantaranya pengujian tentang pengaruh *pre-processing* dengan perbaikan kata tidak baku terhadap hasil klasifikasi. Pengujian ini dilakukan dengan dua cara, yaitu yang pertama menguji dengan data latih dan data uji yang sama-sama dilakukan proses *pre-processing* namun tidak dilakukan perbaikan kata tidak baku dan cara selanjutnya adalah dilakukan *pre-processing* dengan perbaikan kata tidak baku. Pengujian tersebut dilakukan dengan tujuan untuk mengetahui pengaruh perbaikan kata tidak baku pada hasil klasifikasi. Setelah dilakukan pengujian maka akan dilakukan sebuah analisis, analisis ini dilakukan untuk mengetahui apakah dalam penggunaan algoritme *Naive Bayes Classifier* dalam analisis sentimen opini film berbahasa Indonesia sudah menghasilkan nilai akurasi yang terbaik.

### **3.7 Kesimpulan dan Saran**

Pada kesimpulan didapatkan setelah semua tahapan dalam penelitian ini selesai dilakukan. Yaitu yang dimulai dari kajian pustaka, penggalian kebutuhan



## **BAB 4 ANALISIS DAN PERANCANGAN**

Pada bab ini, penulis akan membahas tentang analisis dan perancangan sistem yang terdiri dari deskripsi permasalahan, deskripsi umum sistem, penyelesaian permasalahan dengan menggunakan metode Naive Bayes, perancangan database, dan perancangan pengujian.

### **4.1 Deskripsi permasalahan**

Twitter merupakan salah satu media sosial yang memungkinkan penggunanya untuk melakukan interaksi, berbagi informasi, atau bahkan untuk mengutarakan perasaan dan opini, termasuk juga dalam mengutarakan opini film. Review film atau opini film dapat dijadikan untuk evaluasi dalam perbaikan film kedepannya dan juga dapat digunakan untuk mengetahui film itu bagus atau tidak. Dalam melakukan review film, pengguna memberikan tanggapan berupa pernyataan positif maupun negatif dalam bentuk teks singkat. Banyaknya pengguna Twitter yang melakukan Tweet mengenai opini film dengan bahasa modern dan kata yang disingkat membuat pengembang mengalami kesulitan dalam mengelompokkan opini film tersebut, sehingga digunakan analisis sentimen dengan proses *pre-processing* dan perbaikan kata tidak baku didalamnya untuk menganalisis opini pengguna.

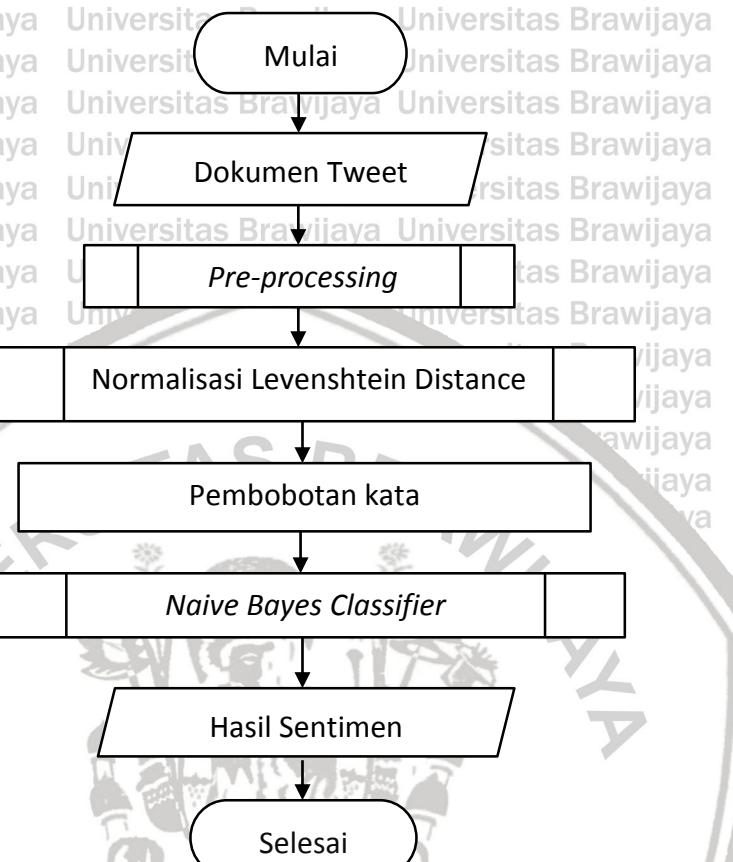
Dalam klasifikasi dokumen terdapat beberapa metode yang dapat digunakan, salah satu metode tersebut adalah Naive Bayes. Metode Naive Bayes dapat digunakan untuk mengelompokkan opini film ke dalam kategori sentimen positif atau negatif berdasarkan data latih yang tersedia. Dalam kebanyakan opini pengguna Twitter, banyak pengguna yang dalam penulisannya menggunakan bahasa modern atau melakukan penyingkatan kata. Hal tersebut menyebabkan isi

Tweet mengandung kata yang tidak baku dan sulit untuk dipahami. Hal ini berpengaruh pada proses klasifikasi yang mana hasil klasifikasi bergantung pada probabilitas frekuensi kemunculan kata. Sehingga diperlukan normalisasi bahasa dengan pencocokan kamus\_katabaku dan dengan normalisasi Levenshtein Distance yang dapat mengubah kata tidak baku menjadi kata baku.

### **4.2 Deskripsi umum sistem**

Sistem yang akan dikembangkan dalam penelitian ini yaitu bertujuan untuk menyelesaikan permasalahan analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia menggunakan Naive Bayes dengan perbaikan kata tidak baku. Dokumen yang akan diklasifikasi diawali dengan proses *pre-processing*, dalam proses *pre-processing* terdapat proses tambahan yaitu perbaikan kata tidak baku. Setelah dilakukan *pre-processing* maka langkah selanjutnya akan dilanjutkan dengan normalisasi Levenshtein Distance yaitu untuk memperbaiki kata tidak baku yang tidak terdapat dalam kamus\_katabaku. Kemudian setelah normalisasi Levenshtein Distance selesai dilakukan maka dilakukan pengklasifikasian

menggunakan Naive Bayes untuk menentukan kelas sentimen. Hasil klasifikasi pada Naive Bayes tergantung pada probabilitas frekuensi kemunculan kata pada data latih. Keseluruhan deskripsi umum sistem digambarkan pada Gambar 4.1.



**Gambar 4.1 Diagram alir sistem**

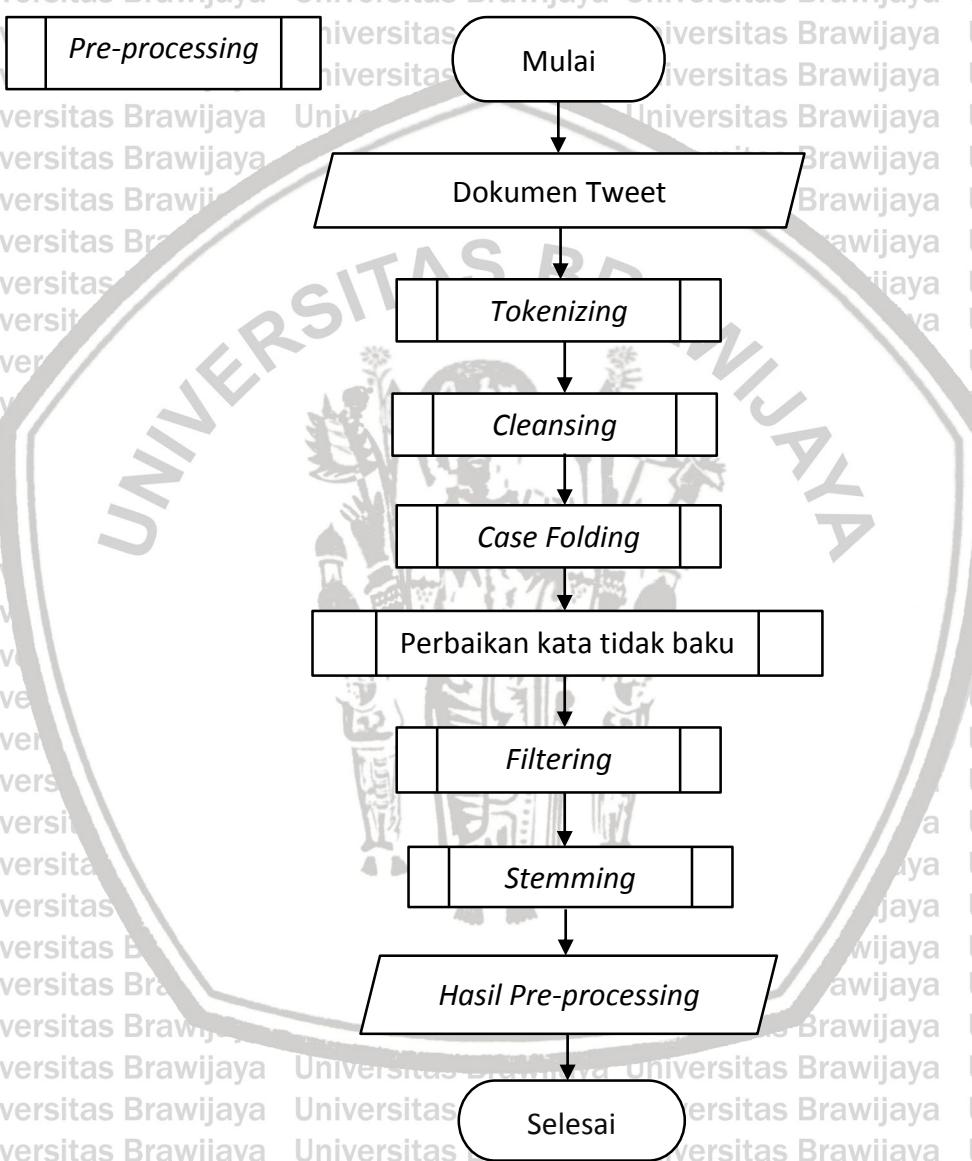
Pada Gambar 4.1 menjukkan diagram alir sistem memiliki beberapa tahapan untuk mengetahui dokumen Tweet yang dimasukkan apakah akan termasuk dalam kelas sentimen positif atau negatif. Dokumen Tweet terdiri dari data latih dan data uji. Proses yang dilakukan pertama yaitu melakukan *pre-processing* pada dokumen Tweet yang sudah dimasukkan. Pada proses *pre-processing* terdapat tambahan proses perbaikan kata tidak baku yang akan memperbaiki kata yang tidak baku menjadi baku dengan menggunakan kamus\_katabaku. Perbaikan kata tidak baku dilakukan setelah proses *case folding* dilakukan. Contoh perbaikan kata tidak baku misalnya kata “gue” menjadi “saya”. Setelah proses *pre-processing* selesai, kemudian dilanjutkan dengan normalisasi Levenshtein Distance. Yang dilakukan setelah proses *pre-processing*, yaitu apabila pada proses *pre-processing* masih terdapat kata tidak baku maka akan dilanjutkan dengan metode Levenshtein Distance untuk memperbaiki kata yang salah eja maupun penyingkatan kata. Selanjutnya setelah dilakukan proses normalisasi Levenshtein Distance maka akan masuk pada pembobotan kata yang akan digunakan untuk klasifikasi menggunakan *Naive Bayes Classifier*. Sehingga dapat ditentukan apakah

dokumen Tweet yang dimasukkan termasuk dalam kelas sentimen positif atau negatif dengan menggunakan *Naive Bayes Classifier*.

### 4.3 Pre-processing

Tahap *pre-processing* merupakan tahap pemrosesan awal dari dokumen yang bertujuan untuk mengolah teks yang tidak terstruktur menjadi teks terstruktur. Hasil dari proses *pre-processing* akan digunakan untuk proses selanjutnya.

Penjelasan alir proses dari teks *pre-processing* ditunjukkan pada Gambar 4.2.



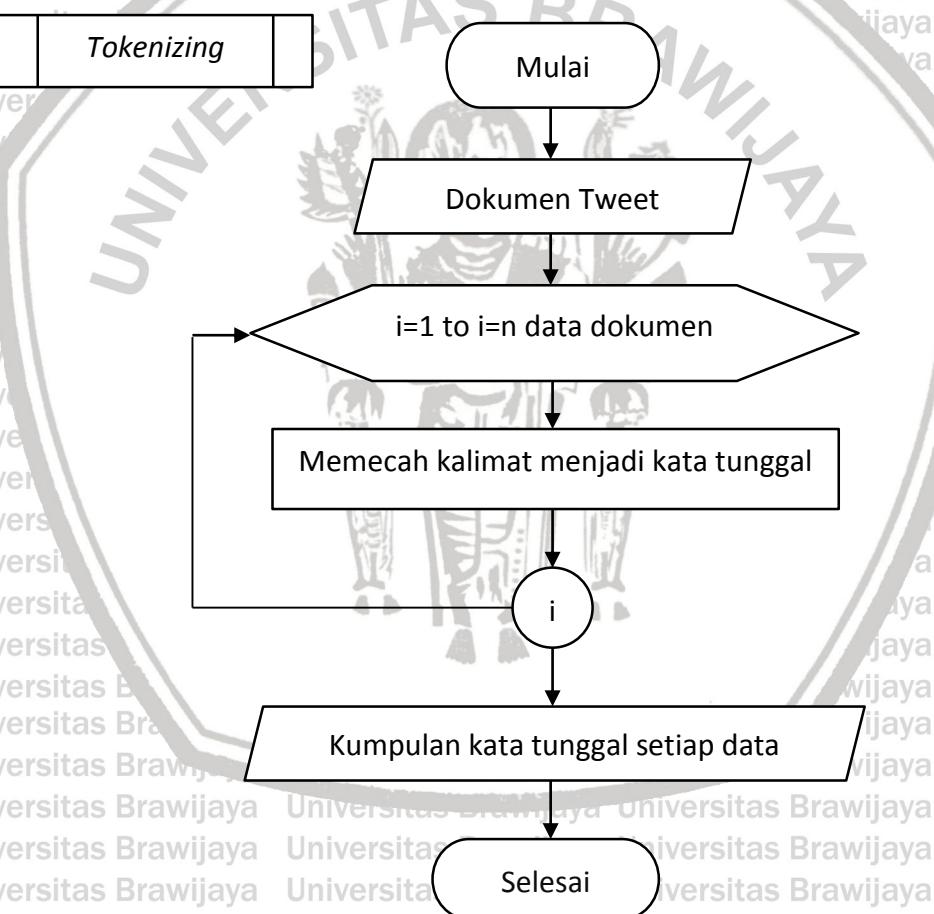
Gambar 4.2 Diagram alir *pre-processing*

Gambar 4.2 menunjukkan pada proses *pre-processing* terdapat beberapa tahapan yang mana didalam tahapan tersebut masih terdapat proses yang harus dijalankan. Pada proses awal terdapat masukan berupa data yang diperoleh dari Twitter yaitu dokumen Tweet. Selanjutnya data tersebut diproses melalui proses

*tokenizing* dahulu yaitu dengan memecah kalimat menjadi bentuk kata. Setelah itu dilanjutkan dengan proses *cleansing* dengan menghilangkan *url*, *username*, *hashtag*, maupun *retweet*. Setelah itu dilakukan *case folding* dan dilanjutkan dengan perbaikan kata tidak baku, yaitu mencocokkan kata dengan kamus\_katabaku. Setelah perbaikan kata tidak baku selesai, maka proses berikutnya adalah *filtering* dan dilanjutkan dengan *stemming*. Sehingga keluaran yang dihasilkan adalah hasil dari proses *pre-processing*.

#### 4.3.1 Tokenizing

Tahap *tokenizing* merupakan tahapan pertama yang dilakukan pada proses *pre-processing*. Tahap *tokenizing* adalah tahap untuk dilakukan pemotongan string input atau memecah kalimat menjadi beberapa bagian atau kata. Dan hasil dari *tokenizing* ini akan berguna untuk proses selanjutnya. Penjelasan alir proses dari *tokenizing* ditunjukkan pada Gambar 4.3.

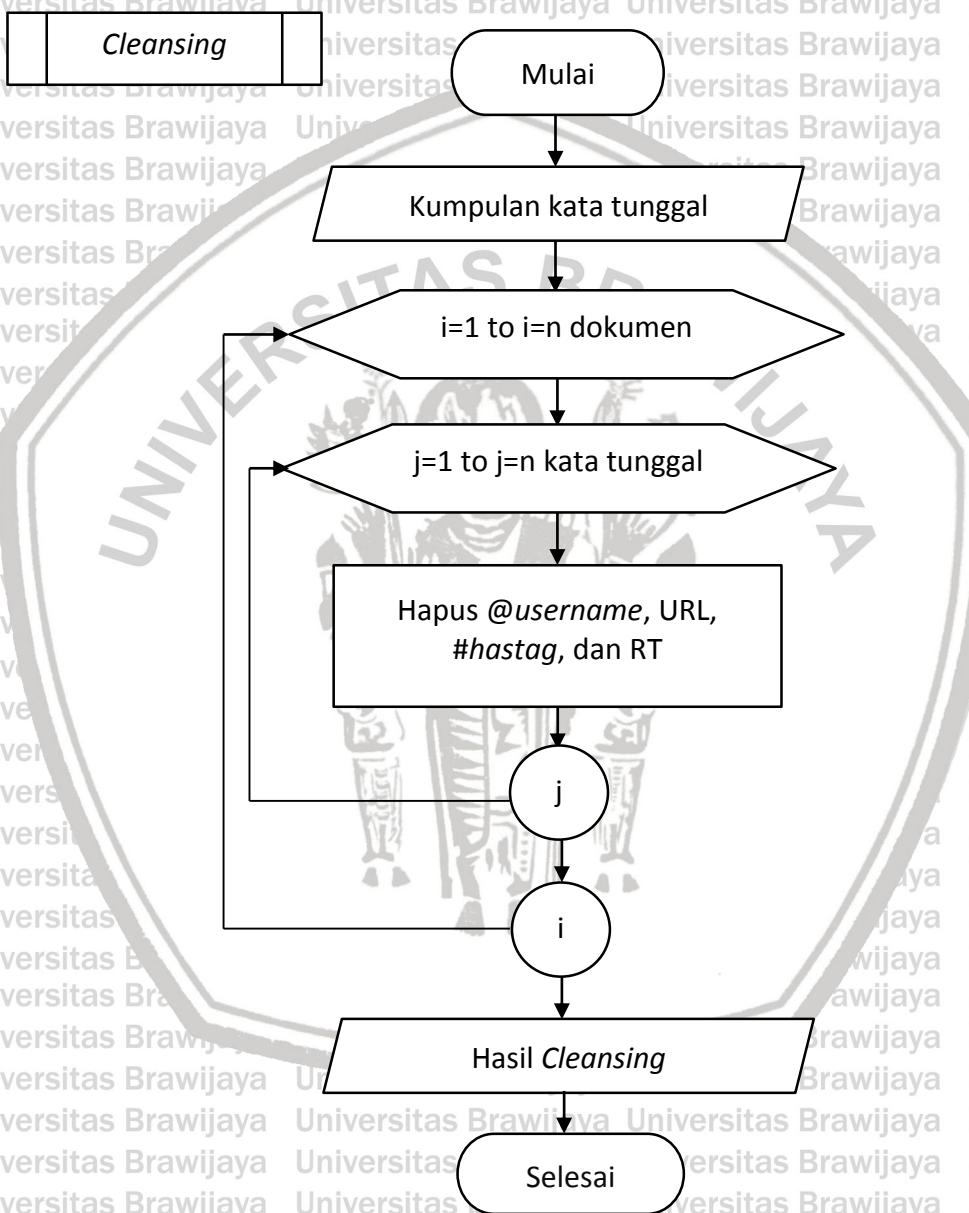


**Gambar 4.3 Diagram alir *tokenizing***

Gambar 4.3 menunjukkan pada proses *tokenizing* dimulai dengan masukan berupa data yang diperoleh dari Twitter yaitu data uji dan data latih. Kemudian data tersebut dilakukan proses pemecahan kalimat menjadi kata tunggal. Sehingga keluaran yang dihasilkan adalah kumpulan kata tunggal pada setiap data.

### 4.3.2 Cleansing

Tahap *cleansing* merupakan tahapan yang dilakukan setelah *tokenizing*, yaitu menghilangkan *noise* (karakter yang tidak penting) pada dokumen teks. Pada dokumen teks Tweet terdapat varian variabel atau karakter yang perlu dihilangkan karena tidak memiliki pengaruh dalam pemrosesan teks. Seperti *link URL* ([http](http://)), *username* (@), *hashtag* (#), dan *retweet* (RT). Penjelasan alir proses dari *cleansing* ditunjukkan pada Gambar 4.4.



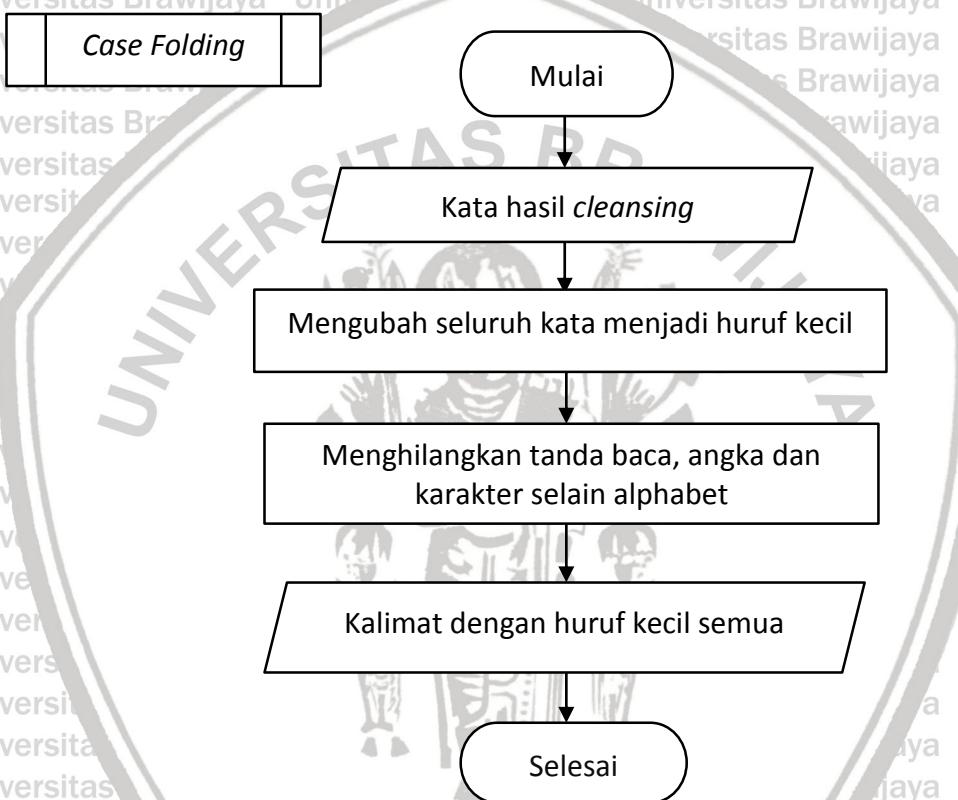
**Gambar 4.4 Diagram alir cleansing**

Gambar 4.4 menunjukkan pada proses *cleansing* dimulai dengan masukan dari hasil *tokenizing* yaitu berupa kumpulan kata tunggal. Selanjutnya dilakukan

penghapusan *username* (@), *link URL* ([http](#)), *hashtag* (#), dan *retweet* (RT). Sehingga keluaran yang dihasilkan adalah kata tanpa *noise*.

### 4.3.3 Case folding

Tahap *case folding* merupakan tahapan yang dilakukan setelah proses *cleansing*. Tahap *case folding* digunakan untuk mengubah seluruh dokumen menjadi huruf kecil semua atau *lowercase*. Tahap *case folding* perlu dilakukan karena dokumen tidak konsisten dengan huruf kapital atau *uppercase*. Selain itu pada tahap ini juga dilakukan pengilangan tanda baca, angka dan penghilangan karakter selain alphabet pada seluruh dokumen. Penjelasan alir proses dari *case folding* ditunjukkan pada Gambar 4.5.

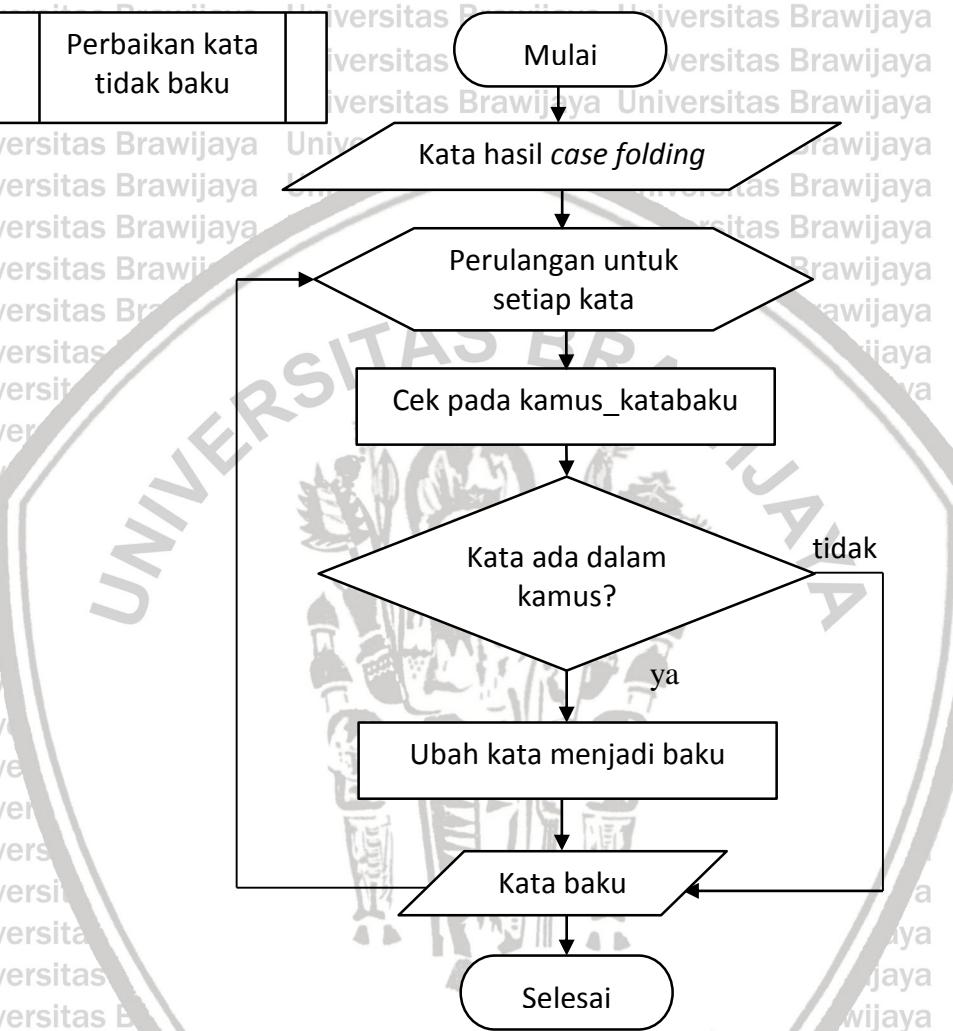


Gambar 4.5 Diagram alir *case folding*

Gambar 4.5 menunjukkan pada proses *case folding* dimulai dengan masukan berupa kata hasil dari proses *cleansing*. Kemudian kata tersebut dilakukan proses pengubahan kata menjadi huruf kecil semua dan menghilangkan angka, tanda baca maupun karakter selain alphabet. Sehingga keluaran yang dihasilkan adalah kalimat dengan huruf kecil semua.

### 4.3.4 Perbaikan kata tidak baku

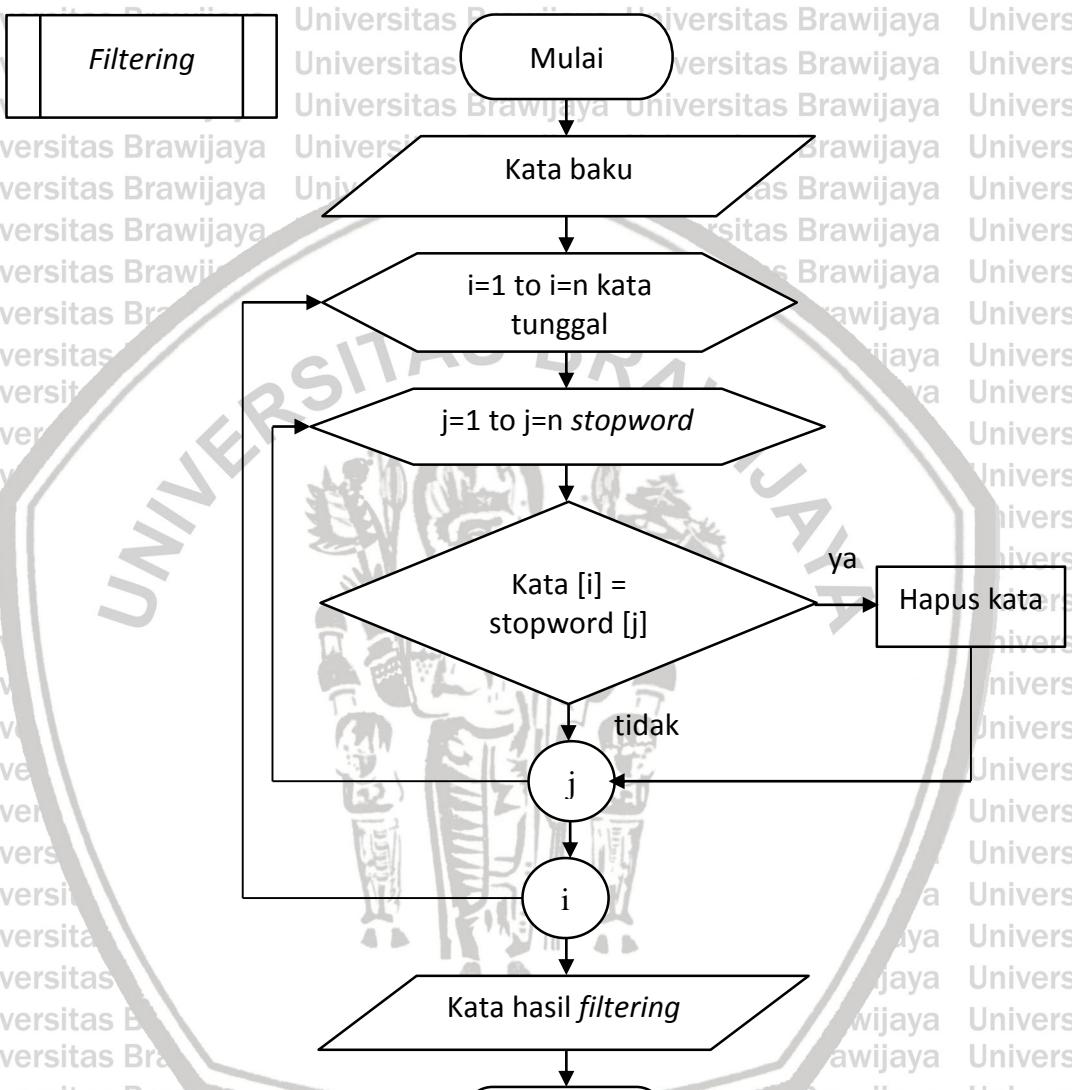
Tahap perbaikan kata tidak baku atau normalisasi bahasa merupakan tahapan untuk mengubah kata-kata yang tidak baku menjadi kata baku sesuai dengan Kamus Besar Bahasa Indonesia (KBBI). Pada perbaikan kata tidak baku ini

**Gambar 4.6 Diagram alir perbaikan kata tidak baku**

Pada Gambar 4.6 menunjukkan proses perbaikan kata tidak baku menjadi baku, proses ini dilakukan setelah proses *case folding* selesai dilakukan. Hasil *case folding* yang sudah menjadi kata tunggal dijadikan inputan untuk pengecekan kata pada kamus\_katabaku. Jika kata tidak baku ada dalam kamus maka akan diganti menjadi kata baku, namun jika tidak ada dalam kamus maka akan dianggap sebagai kata baku, sehingga hasilnya adalah kata baku.

### 4.3.5 Filtering

Tahap *Filtering* merupakan tahapan yang dilakukan untuk menghilangkan kata yang tidak memiliki arti atau tidak relevan. Kata yang akan dihilangkan bersumber dari kamus *stopword*. Hasil dari tahap ini adalah menghasilkan kumpulan kata yang dapat mewakili dokumen. Penjelasan alir proses dari *filtering* ditunjukkan pada Gambar 4.7.

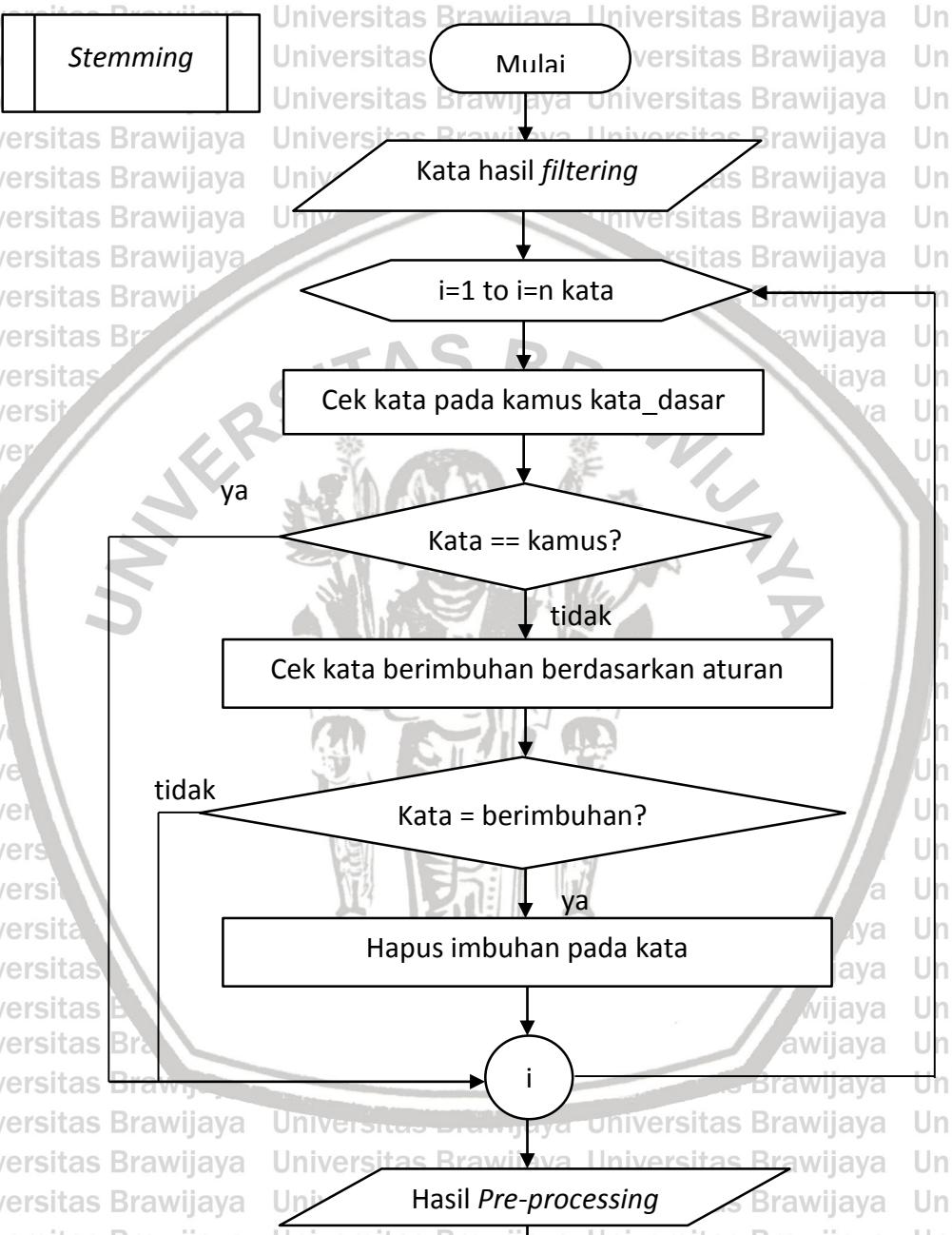


Gambar 4.7 Diagram alir *filtering*

Gambar 4.7 menunjukkan diagram alir dari proses *filtering*. Pada proses *filtering* dimulai dengan masukan kata tunggal yang sudah baku, kemudian setiap kata dilakukan pengecekan kata dengan kata yang ada pada daftar *stopword*. Apabila terdapat kata sama dengan kata yang ada pada daftar *stopword* maka kata tersebut akan dihapus. Proses ini dilakukan hingga kata terakhir diproses. Sehingga menghasilkan keluaran kata yang penting saja.

### 4.3.6 Stemming

Tahap *Stemming* merupakan tahapan yang dilakukan setelah tahap *filtering* selesai dilakukan. Hasil dari *filtering* dilakukan proses *stemming*. *Stemming* merupakan proses dalam pengubahan bentuk kata yang berimbuhan menjadi kata dasar. Penjelasan alir proses dari *stemming* ditunjukkan pada Gambar 4.8.



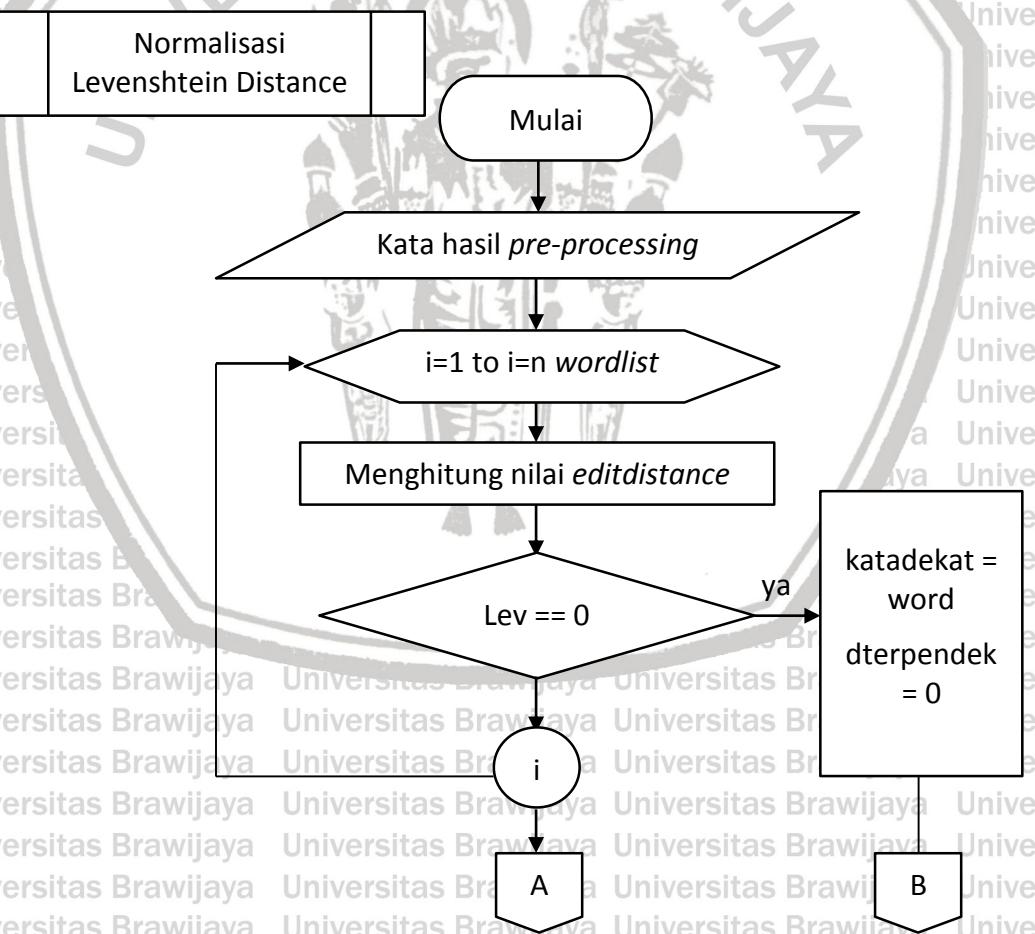
Gambar 4.8 Diagram alir *stemming*

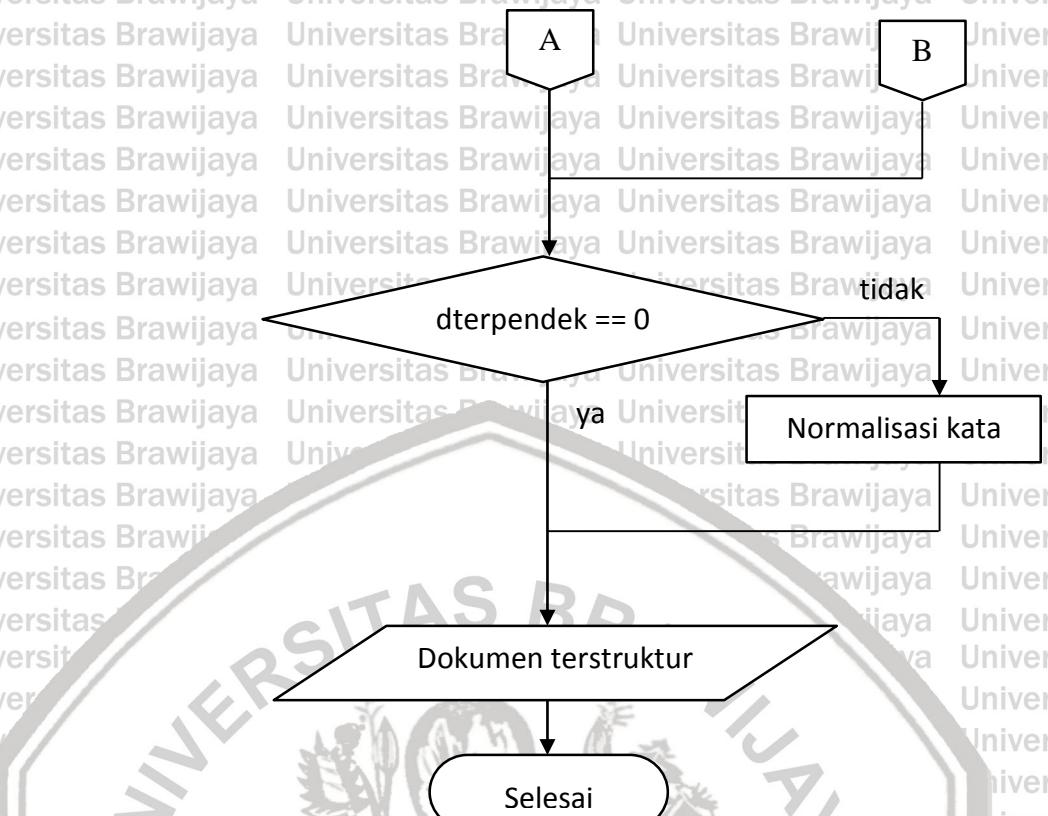
Gambar 4.8 menunjukkan diagram alir *stemming*, proses *stemming* yang digunakan adalah Sastrawi. Sastrawi merupakan *library* yang berisi proses untuk pengubahan kata jamak atau berimbuhan menjadi kata dasar. Proses ini dimulai dari masukan kata hasil *filtering* kemudian di cocokkan dengan kamus *kata\_dasar* yang ada pada Sastrawi, jika kata yang dimaksud ada dalam kamus maka proses selesai, namun jika kata tidak ada dalam kamus *kata\_dasar* maka dilakukan pengecekan kata berimbuhan. Dan apabila terdapat kata imbuhan maka imbuhan akan dihapus. Dan proses ini berulang hingga kata yang harus di *stemming* selesai. Sehingga hasil akhir adalah keluaran hasil *stemming* yaitu berupa kumpulan kata dasar yang merupakan hasil dari *pre-processing*.

#### 4.4 Normalisasi Levenshtein Distance

Tahap penyelesaian normalisasi Levenshtein Distance ini dilakukan untuk menangani kesalahan penulisan kata dengan cara menghitung jumlah *edit* suatu kata dengan kata yang lain. Proses ini dilakukan setelah proses *pre-processing*.

Penjelasan alir proses dari penyelesaian normalisasi Levenshtein Distance ditunjukkan pada Gambar 4.9.



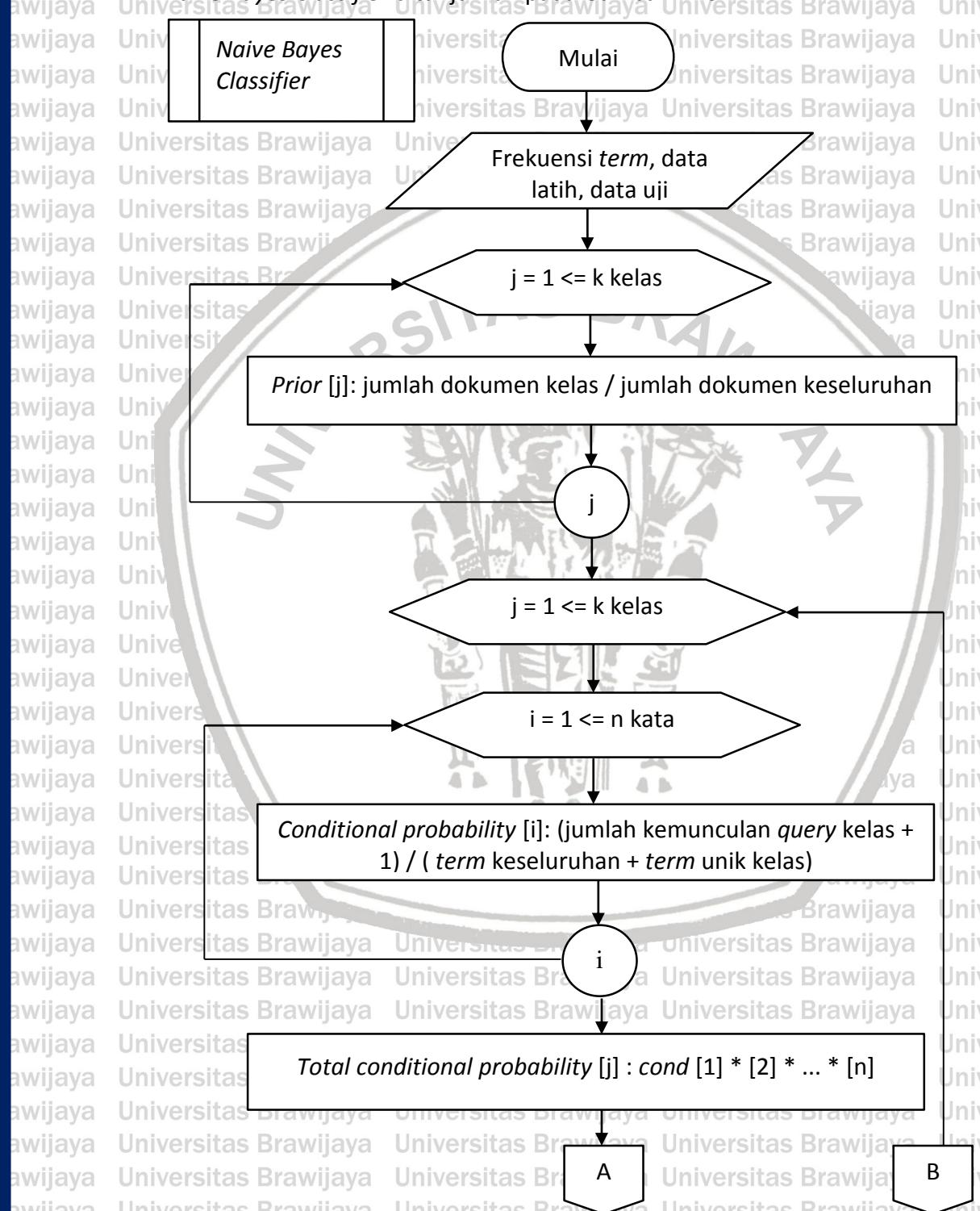


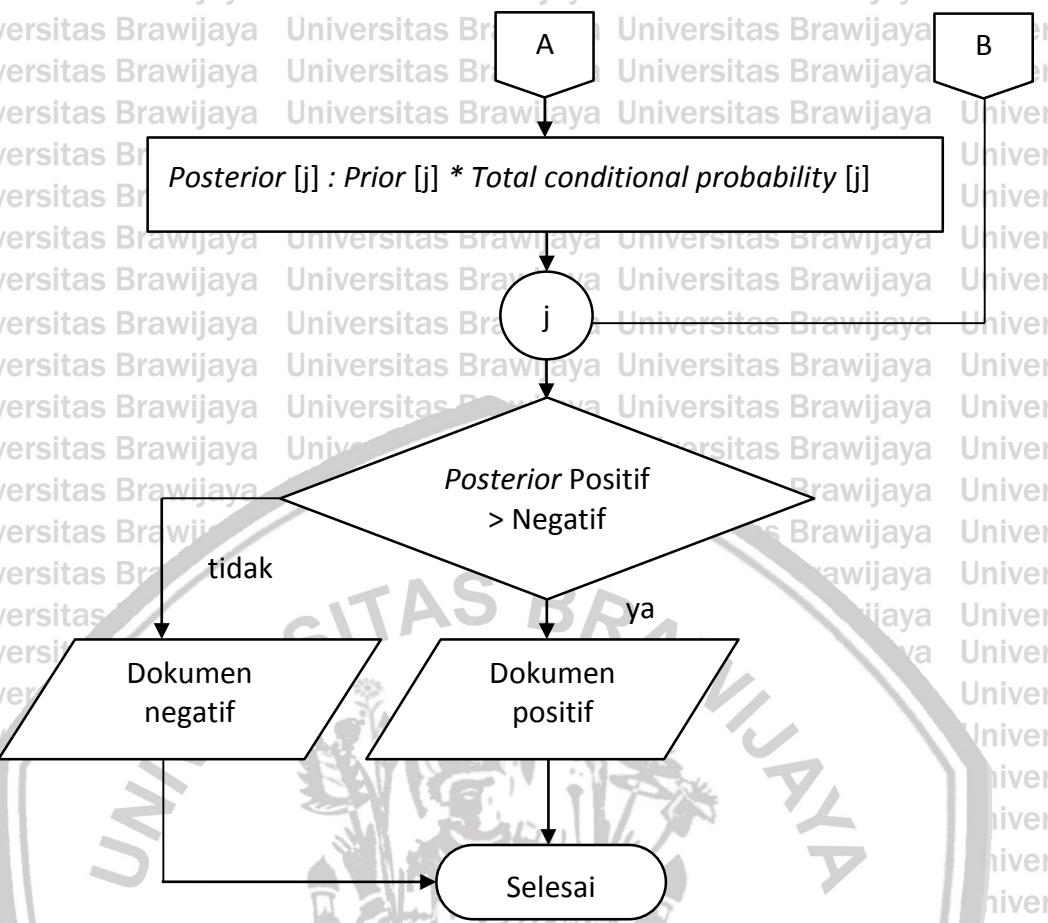
**Gambar 4.9 Diagram alir normalisasi Levenshtein Distance**

Gambar 4.9 menunjukkan diagram alir proses normalisasi Levenshtein Distance, yang mana masukan pada proses ini adalah kata dari hasil *pre-processing*. Apabila pada *pre-processing* masih terdapat kata tidak baku maka akan dilakukan perhitungan nilai *edit* dengan kamus *wordlist*. Proses ini akan berulang hingga kata akhir. Selanjutnya akan dilakukan proses menghitung nilai *editdistance*. Pada perhitungan ini apabila didapatkan nilai *lev* bernilai nol maka kata tersebut sama dengan kata yang ada pada kamus. Namun jika nilai *lev* tidak sama dengan nol maka akan dilakukan perhitungan normalisasi kata dengan kata lain pada kamus. Dalam hal ini, nilai *lev* yang terpendek menunjukkan bahwa kata tersebut mirip dengan kamus dan diganti dengan kata dekat. Sehingga hasil akhir yang didapat adalah dokumen terstruktur.

#### 4.5 Penyelesaian metode *Naive Bayes Classifier*

Tahap penyelesaian metode *Naive Bayes Classifier* dilakukan dengan Multinomial Naive Bayes, yaitu merupakan tahapan yang dapat digunakan pada pemrosesan teks, khususnya dalam hal klasifikasi dokumen. Dalam proses ini terdapat beberapa tahapan, diantaranya adalah menghitung nilai probabilitas *prior*, nilai probabilitas *prior* didapatkan dari probabilitas kelas dokumen positif dan dokumen negatif. Kemudian tahapan selanjutnya adalah *conditional probability*, tahap ini dilakukan dengan cara menghitung frekuensi kemunculan kata pada tiap dokumen yang kemudian akan dipergunakan untuk mencari nilai





**Gambar 4.10 Diagram alir proses Naive Bayes Classifier**

Gambar 4.10 menunjukkan diagram alir proses Naive Bayes Classifier dengan Multinomial Naive Bayes dimulai dengan memasukkan frekuensi term, data uji dan data latih untuk dilakukan perhitungan prior pada setiap kelas. Setelah perhitungan prior selesai dilakukan maka selanjutnya dilakukan perhitungan conditional probability pada kelas positif dan negatif. Conditional probability dihitung berdasarkan kemunculan kata pada setiap kelas masing-masing. Kemudian setelah itu menghitung total conditional probability dengan mengkalikan hasil dari setiap conditional probability per kata untuk kelas positif dan negatif. Setelah itu akan dilakukan perhitungan posterior dengan cara mengalikan hasil dari prior dan total conditional probability untuk dapat menentukan data uji yang dimasukkan akan masuk pada kelas sentimen positif atau negatif. Yaitu dapat dilihat dari besarnya nilai yang dihasilkan. Jika nilai positif lebih besar dari negatif maka data uji akan masuk ke dalam kelas sentimen positif begitu juga sebaliknya jika nilai positif lebih kecil dari nilai negatif maka data uji akan masuk ke dalam kelas sentimen negatif.

## 4.6 Perhitungan Manual

Pada penelitian ini dilakukan perhitungan manual yang berfungsi untuk memudahkan dalam memberi gambaran umum perancangan sistem analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia menggunakan Naive Bayes dengan perbaikan kata tidak baku. Pada contoh perhitungannya digunakan 6 dokumen Tweet, yaitu 5 dokumen Tweet adalah data latih dan 1 dokumen Tweet adalah data uji. Dokumen diambil dari TwitterAPI. Berikut adalah perhitungan manualisasinya yang dimulai dari dokumen Tweet awal hingga menjadi dokumen yang terstruktur untuk dapat dilakukan klasifikasi menggunakan Naive Bayes:

### 1. Dokumen Tweet Awal

Pada Tabel 4.1 menunjukkan dokumen Tweet awal yang merupakan dokumen masukan yang akan diproses untuk menjadi dokumen yang lebih terstruktur untuk dilakukan pengklasifikasian menggunakan Naive Bayes. Apakah dokumen yang diuji masuk ke dalam kelas sentimen positif ataupun sentimen negatif.

**Tabel 4.1 Manualisasi Dokumen Tweet Opini Film**

Id_dok	Opini Film	Kelas Sentimen
0001	Tadi abis nonton @GetOutMovie. Gokil sih premisnya. Keren lah itu film.	Positif
0002	Keren bang flm lo @radityadika persahabatan, keluarga dan cinta kentel banget. Tokoh pemain pas semua keren bang	Positif
0003	Gila, film Britney Ever After jelek banget parah! Nyesel nonton walau gak sampe abis.	Negatif
0004	#FactRica ~ kurang suka sama film romantis, karena endingnya pasti jadian	Negatif
0005	Ancur bgt itu film. Jelekk	Negatif
0006	Film Danur ceritanya jelek~	?

### 2. Tokenizing

Pada Tabel 4.2 menunjukkan dokumen yang sudah dilakukan *tokenizing*, yaitu dengan dilakukannya tahap pemotongan *string input* atau memecah kalimat menjadi beberapa kata tunggal.

**Tabel 4.2 Manualisasi Tokenizing**

Id_dok	Opini Film	Kelas Sentimen
0001	Tadi // abis // nonton // @GetOutMovie. // Gokil // sih // premisnya. // Keren // lah // itu // film.	Positif
0002	Keren // bang // flm // lo // @radityadika // persahabatan, // keluarga // dan // cinta // kentel // banget. // Tokoh // pemain // pas // semua // keren // bang	Positif
0003	Gila, // film // Britney // Ever // After // jelek // banget // parah! // Nyesel // nonton // walau // gak // sampe // abis.	Negatif
0004	#FactRica // ~ // kurang // suka // sama // film // romantis, // karena // endingnya // pasti // jadian	Negatif
0005	Ancur // bgt // itu // film. // Jelekk	Negatif
0006	Film // Danur // ceritanya // jelek~	?

### 3. Cleansing

Pada Tabel 4.3 menunjukkan dokumen yang sudah dilakukan *cleaning*, yaitu menghilangkan *noise* (karakter yang tidak penting) pada dokumen teks. Pada dokumen teks Tweet terdapat varian variable atau karakter yang perlu dihilangkan karena tidak memiliki pengaruh dalam pemrosesan teks. Seperti *link URL* ([http](http://)), *username* (@), *hashtag* (#), dan *retweet* (RT).

**Tabel 4.3 Manualisasi Cleaning**

Id_dok	Opini Film	Kelas Sentimen
0001	Tadi // abis // nonton // Gokil // sih // premisnya. // Keren // lah // itu // film.	Positif
0002	Keren // bang // flm // lo // persahabatan, // keluarga // dan // cinta // kentel // banget. // Tokoh // pemain // pas // semua // keren // bang	Positif
0003	Gila, // film // Britney // Ever // After // jelek // banget // parah! // Nyesel // nonton // walau // gak // sampe // abis.	Negatif
0004	FactRica // ~ // kurang // suka // sama // film // romantis, // karena // endingnya // pasti // jadian	Negatif
0005	Ancur // bgt // itu // film. // Jelekk	Negatif
0006	Film//Danur // ceritanya//jelek~	?

#### 4. Case Folding

Pada Tabel 4.4 menunjukkan dokumen yang sudah dilakukan proses *case folding* yaitu mengubah seluruh kata menjadi huruf kecil dan menghilangkan tanda baca, angka, dan karakter selain alphabet.

**Tabel 4.4 Manualisasi Case Folding**

Id_dok	Opini Film	Kelas Sentimen
0001	tadi // abis // nonton // gokil // sih // premisnya // keren // lah // itu // film	Positif
0002	keren // bang // flm // lo // persahabatan // keluarga // dan // cinta // kentel // banget // tokoh // pemain // pas // semua // keren // bang	Positif
0003	gila // film // britney // ever // after // jelek // banget // parah // nyesel // nonton // walau // gak // sampe // abis	Negatif
0004	factrica // kurang // suka // sama // film // romantis // karena // endingnya // pasti // jadian	Negatif
0005	ancur // bgt // itu // film // jelekk	Negatif
0006	film // danur // ceritanya // jelek	?

#### 5. Perbaikan kata tidak baku

Pada perbaikan kata tidak baku dilakukan pengubahan kata-kata yang tidak baku menjadi kata baku sesuai dengan Kamus Besar Bahasa Indonesia (KBBI). Pada perbaikan kata tidak baku ini dilakukan pengecekan terhadap kamus\_katabaku yang diambil dari penelitian sebelumnya dan ditambahkan oleh penulis. Berikut adalah contoh dari kamus\_katabaku ditunjukkan pada Tabel 4.5.

**Tabel 4.5 Contoh Kamus\_katabaku**

Kata Tidak Baku	Kata Baku
abis	habis
ancur	hancur
gokil	gila
kentel	kental
lo	kamu
nonton	tonton
nyesel	sesal

Hasil dari dokumen yang telah dilakukan perbaikan kata tidak baku ditunjukkan pada Tabel 4.6, yang mana mengubah kata-kata yang tidak baku menjadi baku sesuai dengan Kamus Bahasa Indonesia. Namun dalam dokumen tersebut masih terdapat kata tidak baku yaitu “jelekk”. Sehingga dapat diperbaiki lagi dengan Levenshtein Distance yang dilakukan setelah proses *pre-processing*.

**Tabel 4.6 Manualisasi Perbaikan kata tidak baku**

Id_dok	Opini Film	Kelas Sentimen
0001	tadi // habis // tonton // gila // sih // premisnya // keren // lah // itu // film	Positif
0002	keren // kakak // film // kamu // persahabatan // keluarga // dan // cinta // kental // banget // tokoh // pemain // tepat // semua // keren // kakak	Positif
0003	gila // film // britney // ever // after // jelek // banget // parah // sesal // tonton // walau // tidak // sampai // habis	Negatif
0004	factrica // kurang // suka // sama // film // romantis // karena // berakhir // pasti // bersama	Negatif
0005	hancur // banget // itu // film // jelekk	Negatif
0006	film // danur // ceritanya // jelek	?

## 6. Filtering

Pada Tabel 4.7 menunjukkan dokumen yang sudah dilakukan proses *filtering* setelah perbaikan kata tidak baku. Proses ini digunakan untuk menghilangkan kata yang tidak memiliki arti atau tidak penting.

**Tabel 4.7 Manualisasi Filtering**

Id_dok	Opini Film	Kelas Sentimen
0001	habis // tonton // gila // premisnya // keren // film	Positif
0002	keren // kakak // film // persahabatan // keluarga // cinta // kental // banget // tokoh // pemain // keren // kakak	Positif
0003	gila // film // britney // ever // after // jelek // banget // parah // sesal // tonton // habis	Negatif
0004	factrica // kurang // suka // film // romantis	Negatif
0005	hancur // banget // film // jelekk	Negatif
0006	film // danur // ceritanya // jelek	?

### 7. Stemming

Pada Tabel 4.8 menunjukkan dokumen yang sudah dilakukan proses *stemming*, yaitu tahapan yang dilakukan pengubahan bentuk kata yang berimbuhan menjadi kata dasar.

**Tabel 4.8 Manualisasi Stemming**

Id_dok	Opini Film	Kelas Sentimen
0001	habis // tonton // gila // premis // keren // film	Positif
0002	keren // kakak // film // sahabat // keluarga // cinta // kental // banget // tokoh // main // keren // kakak	Positif
0003	gila // film // britney // ever // after // jelek // banget // parah // sesal // tonton // habis	Negatif
0004	factrica // kurang // suka // film // romantis	Negatif
0005	hancur // banget // film // jelekk	Negatif
0006	film // danur // cerita // jelek	?

### 8. Normalisasi Levenshtein Distance

Pada Tabel 4.9 menunjukkan dokumen yang sudah dilakukan proses normalisasi Levenshtein Distance yaitu menangani kesalahan penulisan kata dengan cara menghitung jumlah *edit* suatu kata dengan kata yang lain.

**Tabel 4.9 Manualisasi normalisasi Levenshtein Distance**

Id_dok	Opini Film	Kelas Sentimen
0001	habis // tonton // gila // premis // keren // film	Positif
0002	keren // kakak // film // sahabat // keluarga // cinta // kental // banget // tokoh // main // keren // kakak	Positif
0003	gila // film // britney // ever // after // jelek // banget // parah // sesal // tonton // habis	Negatif
0004	factrica // kurang // suka // film // romantis	Negatif
0005	hancur // banget // film // jelekk	Negatif
0006	film // danur // cerita // jelek	?

Pada Tabel 4.9 tahap normalisasi Levenshtein Distance dilakukan pada kata “jelekk” di dokumen 0005 yang merupakan kata tidak baku yang tidak terdapat pada kamus karena kesalahan ejaan. Maka langkah selanjutnya mengubah kata

tersebut dengan menghapus karakter ‘k’ yang berada diakhir kata dengan menggunakan Levenshtein Distance. Perhitungan Levenshtein Distance dapat dilihat pada Tabel 4.10.

**Tabel 4.10 Perhitungan Levenshtein Distance**

	J	E	L	E	K	K
0	1	2	3	4	5	6
J	1	0	1	2	3	4
E	2	1	0	1	2	3
L	3	2	1	0	1	2
E	4	3	2	1	0	1
K	5	4	3	2	1	0

Tabel 4.10 menunjukkan hasil perhitungan normalisasi Levenshtein Distance.

Nilai *edit distance* yang dihasilkan adalah bernilai 1 ditandai dengan arsiran pada pojok kanan bawah. Hasil dari perhitungan Levenshtein merupakan nilai *edit* yang paling terpendek atau rendah, dalam hal ini dilakukan perhitungan dengan menggunakan 2 *string* dan hasil dengan nilai *edit* yang terpendek menunjukkan suatu kata yang dimaksud. Pengecekan dimulai dari iterasi awal dari kedua *string* kemudian dilakukan operasi penambahan, pengubahan dan penghapusan karakter. Pada proses ini hanya terdapat 1 proses yaitu penghapusan karakter ‘K’ pada *string* ‘JELEKK’. Sehingga menjadi *string* ‘JELEK’.

### 9. Pembobotan Kata

Pada pembobotan kata dilakukan untuk perhitungan klasifikasi menggunakan Multinomial Naive Bayes. Pada tahap ini dilakukan pembobotan kata dengan menghitung frekuensi kemunculan kata yang ditunjukkan pada Tabel 4.11 hasil dari proses terakhir yaitu normalisasi kata yang dilakukan sesudah *stemming*.

**Tabel 4.11 Frekuensi kemunculan kata**

Kata	Query	Frekuensi Term					Jumlah
		0001	0002	0003	0004	0005	
film	1	1	1	1	1	1	5
danur	1	0	0	0	0	0	0
cerita	1	0	0	0	0	0	0
jelek	1	0	0	1	0	1	2

Setelah diketahui jumlah frekuensi masing-masing kata uji pada dokumen latih maka dilakukan perhitungan *prior* dengan rumus Persamaan 2.4. Tabel perhitungan *prior* ditunjukkan pada Tabel 4.12.

**Tabel 4.12 Perhitungan *Prior* kelas positif dan negatif**

P(p)	P(n)
$\frac{2}{5} = 0,4$	$\frac{3}{5} = 0,6$

Setelah didapatkan hasil dari *prior* negatif dan positif maka langkah selanjutnya adalah menghitung *conditional probability* dengan menggunakan rumus Persamaan 2.6 yang ditunjukkan pada Tabel 4.13.

**Tabel 4.13 Perhitungan *Conditional Probability***

Kata	Conditional Probability (p)	Conditional Probability (n)
film	$(2+1) / (18+25) = 3/43 (0,070)$	$(3+1) / (20+25) = 4/45 (0,089)$
danur	$(0+1) / (18+25) = 1/43 (0,023)$	$(0+1) / (20+25) = 1/45 (0,022)$
cerita	$(0+1) / (18+25) = 1/43 (0,023)$	$(0+1) / (20+25) = 1/45 (0,022)$
jelek	$(0+1) / (18+25) = 1/43 (0,023)$	$(2+1) / (20+25) = 3/45 (0,067)$

Setelah didapatkan nilai dari *conditional probability* pada tiap kata masing-masing kelas, maka langkah selanjutnya yaitu menghitung *total conditional probability* dengan cara mengalikan hasil dari setiap kata pada masing-masing kelas. Perhitungan *total conditional probability* ditunjukkan pada Tabel 4.14.

**Tabel 4.14 Perhitungan *Total conditional probability***

Total conditional probability (p)	Total conditional probability (n)
$0,070 * 0,023 * 0,023 * 0,023$ $= 8,5169e-07$	$0,089 * 0,022 * 0,022 * 0,067$ $= 2,886092e-06$

Kemudian setelah didapatkan nilai dari *total conditional probability* maka langkah selanjutnya adalah menghitung *posterior*. Rumus untuk menghitung *posterior* yaitu *prior* \* *total conditional probability*. Dan hasil dari perkalian tersebut dilakukan perbandingan untuk mencari nilai tertinggi. Nilai tertinggi tersebut akan menentukan sebuah dokumen masuk ke dalam kelas sentimen positif atau negatif. Perhitungan *posterior* dihitung dengan menggunakan rumus Persamaan 2.5 yang ditunjukkan pada Tabel 4.15.

**Tabel 4.15 Perhitungan Posterior kelas Positif dan Negatif**

Posterior (p)	Posterior (n)
$0,4 * 8,5169e-07$	$0,6 * 2,886092e-06$
$= 3,40676e-07$	$= 1,731655e-06$

Berdasarkan hasil dari perhitungan Tabel 4.15 dapat disimpulkan bahwa nilai *posterior* pada kelas negatif memiliki nilai yang lebih tinggi dibandingkan dengan nilai posterior pada kelas positif. Sehingga dokumen uji pada no 0006 termasuk dalam kelas sentimen negatif.

## 4.7 Perancangan Database

Perancangan Database digunakan untuk menyimpan data yang diperlukan dalam penelitian ini untuk digunakan sebagai pengujian. Database yang digunakan pada sistem ini adalah Database MySQL. Pada penelitian ini hanya terdapat 2 tabel yaitu tabel *data\_latih* dan *data\_uji*. Dan untuk penyimpanan data lainnya seperti *stoplist*, *kamus\_katabaku*, dan *wordlist* disimpan pada *file.csv*.

Perancangan tabel Database ditunjukkan pada Gambar 4.11.

**Gambar 4.11 Perancangan Database**

### 4.7.1 Perancangan Tabel *data\_latih*

Tabel *data\_latih* digunakan untuk tempat penyimpanan data yang didapat dari hasil *search API* pada Twitter mengenai opini tentang film. Struktur tabel *data\_latih* dapat ditunjukkan oleh Tabel 4.16.

**Tabel 4.16 Perancangan Tabel *data\_latih***

No	Nama	Tipe data	Ukuran
1	<i>id_latih</i>	Integer	5
2	<i>dokumen</i>	Varchar	200
3	<i>kelas</i>	Varchar	8

## 4.7.2 Perancangan Tabel *data\_uji*

Tabel *data\_uji* digunakan untuk tempat penyimpanan data uji yang digunakan untuk proses klasifikasi. Struktur tabel *data\_uji* dapat ditunjukkan oleh Tabel 4.17.

**Tabel 4.17 Perancangan Tabel *data\_uji***

No	Nama	Tipe data	Ukuran
1	<i>id_uji</i>	Integer	5
2	dokumen	Varchar	200
3	kelas	Varchar	8

## 4.8 Perancangan Pengujian

Perancangan pengujian merupakan rancangan yang dilakukan untuk mengetahui apakah sistem yang dibuat telah sesuai dengan metode yang diterapkan yaitu metode *Naive Bayes Classifier*. Pengujian dilakukan untuk melihat seberapa baik hasil akurasi dari algoritme *Naive Bayes Classifier* yang dihasilkan dengan cara memasukkan data latih dan data uji ke dalam sistem untuk dapat menentukan hasil dari analisis sentimen opini film pada dokumen Twitter berbahasa Indonesia. Berbagai macam pengujian dilakukan untuk mengetahui pengaruh parameter yang diuji. Pada pengujian ini terdapat 3 macam pengujian yang akan dilakukan, yaitu pengujian pertama dilakukan dengan tujuan untuk mengetahui pengaruh proses penggunaan *pre-processing*. Kemudian untuk pengujian kedua dilakukan untuk mengetahui pengaruh penggunaan proses perbaikan kata tidak baku. Dan terakhir adalah pengujian untuk mengetahui pengaruh dari dilakukannya proses *pre-processing* dengan perbaikan kata tidak baku dan normalisasi Levenshtein Distance.

### 4.8.1 Skenario pengujian pengaruh penggunaan *pre-processing*

Pengujian ini menjelaskan tentang pengujian untuk mengetahui pengaruh proses *pre-processing*. Yaitu dengan menguji data uji yang dilakukan dengan *pre-processing* dan tanpa menggunakan *pre-processing*. Pada pengujian ini data yang digunakan merupakan data asli dari pengguna Twitter mengenai Tweet opini film berbahasa Indonesia. Data latih yang diambil sebanyak 140 data opini, yang terdiri dari 70 data opini positif dan 70 data opini negatif. Sedangkan untuk data uji digunakan 60 data, yang terdiri dari 30 data opini positif dan 30 data opini negatif. Skenario pengujian pengaruh penggunaan *pre-processing* dapat dilihat pada Tabel 4.18.

**Tabel 4.18 Skenario pengujian pengaruh penggunaan *pre-processing***

Klasifikasi sentimen				
Jenis Uji	Accuracy	Precision	Recall	F-Measure
Tanpa <i>Pre-processing</i> dan tanpa perbaikan kata tidak baku				
Tanpa <i>Pre-processing</i> dengan perbaikan kata tidak baku				
Dengan <i>Pre-processing</i> tanpa perbaikan kata tidak baku				

#### 4.8.2 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku

Pengujian ini menjelaskan tentang pengujian untuk mengetahui pengaruh proses perbaikan kata tidak baku. Pada perbaikan kata tidak baku dilakukan dua proses. Proses pertama dilakukan dengan pencocokan kata tidak baku pada kamus\_katabaku dan kedua adalah dengan normalisasi Levenshtein Distance. Pada pengujian ini dilakukan proses pertama saja yaitu dengan perbaikan kamus\_katabaku. Data latih dan data uji sama-sama dilakukan proses *pre-processing*. Dan dilakukan variasi dalam proses perbaikan kata tidak baku. Pada pengujian ini data yang digunakan merupakan data asli dari pengguna Twitter mengenai Tweet opini film berbahasa Indonesia. Data latih yang diambil sebanyak 140 data opini, yang terdiri dari 70 data opini positif dan 70 data opini negatif. Sedangkan untuk data uji digunakan 60 data, yang terdiri dari 30 data opini positif dan 30 data opini negatif. Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku dapat dilihat pada Tabel 4.19.

**Tabel 4.19 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku**

Klasifikasi sentimen tanpa dan dengan perbaikan kata tidak baku				
Jenis Uji	Accuracy	Precision	Recall	F-Measure
<i>Pre-processing</i> tanpa perbaikan kata tidak baku				
<i>Pre-processing</i> dengan perbaikan kata tidak baku				

#### **4.8.3 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance**

Pengujian ini menjelaskan tentang pengujian untuk mengetahui pengaruh proses perbaikan kata tidak baku. Pada perbaikan kata tidak baku dilakukan dua proses. Proses pertama dilakukan dengan pencocokan kata tidak baku pada kamus\_katabaku dan kedua adalah dengan normalisasi Levenshtein Distance. Pada pengujian ini dilakukan perbaikan kata tidak baku dengan seluruh proses. Data latih dan data uji dilakukan variasi proses *pre-processing*. Dan sama-sama dilakukan proses perbaikan kata tidak baku dan normalisasi Levenshtein Distance. Pada pengujian ini data yang digunakan merupakan data asli dari pengguna Twitter mengenai Tweet opini film berbahasa Indonesia. Data latih yang diambil sebanyak 140 data opini, yang terdiri dari 70 data opini positif dan 70 data opini negatif. Sedangkan untuk data uji digunakan 60 data, yang terdiri dari 30 data opini positif dan 30 data opini negatif. Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance dapat dilihat pada Tabel 4.20.

**Tabel 4.20 Skenario pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance**

Klasifikasi sentimen dengan perbaikan kata tidak baku dan normalisasi kata Levenshtein Distance				
Jenis Uji	Accuracy	Precision	Recall	F-Measure
Tanpa <i>Pre-processing</i> dengan perbaikan kata tidak baku dan Levenshtein Distance				
<i>Pre-processing</i> dengan perbaikan kata tidak baku dan Levenshtein Distance				

## **BAB 5 IMPLEMENTASI**

Pada bab ini, penulis membahas tentang implementasi sistem yang telah dirancang pada bab sebelumnya. Dalam implementasi ini berisi batasan implementasi, implementasi aplikasi, *pre-processing*, normalisasi Levenshtein Distance, proses analisis sentimen, dan proses evaluasi.

### **5.1 Batasan implementasi**

Batasan implementasi merupakan suatu batasan yang digunakan untuk membatasi jalannya sistem berdasarkan perancangan yang telah dilakukan pada bab sebelumnya. Tujuannya adalah untuk membuat sistem agar sesuai dengan ruang lingkup yang jelas dan sesuai dengan tujuan utama dari sistem. Berikut adalah batasan implementasi sistem dari penelitian ini:

1. Analisis sentimen tentang opini film berbahasa Indonesia pada dokumen Twitter dirancang dan dijalankan pada aplikasi Pycharm berbahasa Python.
2. Algoritme yang digunakan dalam penyelesaian masalah ini adalah Multinomial Naive Bayes dengan perbaikan kata tidak baku.
3. Perbaikan kata tidak baku dilakukan hanya pada bahasa modern atau *slang*, penyingkatan kata, perulangan kata dan salah eja. Yaitu menggunakan kamus\_katabaku dan normalisasi Levenshtein Distance.
4. Data yang digunakan adalah *data\_latih* dan *data\_uji* yang didapat dari Twitter dengan cara *search API* Twitter berupa opini mengenai film atau *review* film berbentuk teks.
5. Hasil yang dikeluarkan yaitu berupa sentimen analisis negatif ataupun positif.
6. Dalam penentuan sentimen didasarkan pada perhitungan frekuensi kemunculan kata, sistem tidak memperhatikan semantik yaitu makna kata dan kalimat.

### **5.2 Implementasi aplikasi**

Implementasi aplikasi merupakan implementasi yang digunakan untuk menjelaskan tahapan dalam membuat sistem analisis sentimen tentang opini film berbahasa Indonesia pada dokumen Twitter menggunakan Naive Bayes dengan perbaikan kata tidak baku. Tahapan-tahapan yang dilakukan sesuai dengan alur kerja yang sudah dirancang pada bab sebelumnya. Penjelasan pada sub bab ini melibatkan kode program Python yang dijalankan oleh aplikasi.

Pada aplikasi analisis sentimen tentang opini film pada dokumen Twitter terdiri dari beberapa proses, diantaranya adalah proses *pre-processing* yang meliputi *tokenizing*, *cleansing*, *casefolding*, perbaikan kata tidak baku, *filtering*, dan *stemming*. Dan juga terdapat proses normalisasi kata menggunakan Levenshtein Distance. Sedangkan proses klasifikasinya menggunakan Naive Bayes.



Sehingga aplikasi dapat melakukan sentimen dengan menggunakan proses tersebut untuk menghasilkan sentimen berupa positif atau negatif. Fungsi dari analisis sentimen dapat dilihat pada daftar Tabel 5.1.

**Tabel 5.1 Fungsi analisis sentimen**

No	Proses	Nama Fungsi	Keterangan
1.	preprocessing	tokenizing()	Berfungsi untuk memecah kalimat menjadi bentuk kata tunggal
		cleansing()	Berfungsi untuk menghilangkan noise yaitu karakter yang sering muncul pada Twitter yang dianggap tidak penting (@, #, RT, url)
		casefolding()	Berfungsi untuk mengubah seluruh kata menjadi huruf kecil dan melakukan penghilangan tanda baca, angka, dan karakter selain huruf alphabet.
		perbaikan()	Berfungsi untuk memperbaiki kata tidak baku ( <i>slang</i> ) menjadi kata baku berdasarkan kamus_katabaku.
		filtering()	Berfungsi untuk menghapus kata yang tidak relevan (tidak memiliki arti) atau tidak penting.
		stemming()	Berfungsi untuk mengubah kata berimbuhan menjadi kata dasar.
2.	levenshtein	normalisasi()	Berfungsi untuk melakukan perbaikan kata tidak baku berdasarkan <i>editdistance</i> .
		editdistance()	Berfungsi untuk menghitung jarak antara 2 string dalam pencocokan kata. Jarak terdekat dijadikan pilihan untuk mengubah kata menjadi baku.

3.	naivebayes	pembobotan()	Berfungsi untuk menghitung jumlah <i>term</i> yang ada pada data uji berdasarkan data latih.
		prior()	Berfungsi untuk menghitung nilai <i>prior</i> positif dan negatif.
		conditionalprob()	Berfungsi untuk menghitung <i>conditional probability</i> positif dan negatif.
		totalconditionalprob()	Berfungsi untuk menghitung <i>total conditional probability</i> positif dan negatif.
		posterior()	Berfungsi untuk menghitung <i>posterior</i> negatif dan positif.
		klasifikasi()	Berfungsi untuk mengklasifikasikan dokumen ke dalam kelas positif dan negatif
4.	searchtwitter	searchtwitter()	Berfungsi untuk mengambil data dengan API Twitter.

### 5.3 Pre-processing

*Pre-processing* merupakan tahapan untuk mengolah teks yang masih mentah menjadi lebih terstruktur. Data teks yang masih mentah seringkali tidak konsisten dan tidak lengkap atau mungkin mengandung banyak kesalahan. Sehingga *pre-processing* dapat digunakan untuk menyelesaikan masalah tersebut. Langkah yang dilakukan pada tahap *pre-processing* adalah dimulai dengan *tokenizing*, *cleaning*, *case folding*, perbaikan kata tidak baku, *filtering*, dan *stemming*.

#### 5.3.1 Tokenizing

*Tokenizing* merupakan proses yang dilakukan untuk memotong atau memecah kalimat menjadi bentuk kata tunggal. Kode program proses *tokenizing* dapat dilihat pada Kode Program 5.1.

```
1 def tokenizing(self, text):
2     term = text.split()
3     return term
```

#### Kode Program 5.1 Tokenizing

Pada Kode Program 5.1 dijelaskan pada baris 1 merupakan pembuatan fungsi *tokenizing*, dan baris 2 adalah pemotongan kalimat menjadi kata tunggal.



### 5.3.2 **Cleansing**

*Cleansing* merupakan proses yang dilakukan untuk menghilangkan *noise* yaitu karakter yang sering muncul pada Twitter yang dianggap tidak penting (@, #, RT, url). Kode program proses *cleansing* dapat dilihat pada Kode Program 5.2.

```

1 def cleansing(self,term):
2     scheme, netloc, url, params, query,
3         fragment = urlparse.urlparse(term)
4         if scheme and netloc or term[0]=='@' or term=='RT':
5             term = ''
6         elif term[0] == '#':
7             term = term[1:]
8         return term

```

### Kode Program 5.2 **Cleansing**

Pada Kode Program 5.2 dijelaskan pada baris 1 merupakan pembuatan fungsi *cleansing*. Kemudian baris selanjutnya dilakukan inisialisasi variabel *url* yang digunakan untuk menghilangkan url, @, RT, dan #. Pada baris 3 dilakukan statement kondisi untuk penghilangan @username dan RT, dan pada baris 5 dilakukan penghilangan hashtag (#).

### 5.3.3 **Case Folding**

*Case folding* merupakan proses yang dilakukan untuk mengubah seluruh kata menjadi huruf kecil dan melakukan penghilangan tanda baca, angka, dan karakter selain huruf alphabet. Kode program proses *case folding* dapat dilihat pada Kode Program 5.3.

```

1 def casefolding(self,term):
2     term = term.lower()
3     term = "".join([huruf for huruf in term if huruf not in
4                     string.punctuation])
5     term = "".join([huruf for huruf in term if huruf not in
6                     string.digits])
7     return term

```

### Kode Program 5.3 **Case Folding**

Pada Kode Program 5.3 dijelaskan pada baris 1 merupakan pembuatan fungsi *casefolding*. Kemudian pada baris 2 digunakan untuk mengubah kata menjadi huruf kecil, kemudian pada baris 3 dan 4 merupakan fungsi untuk menghilangkan tanda baca dan dilanjutkan dengan menghilangkan angka.

### 5.3.4 **Perbaikan kata tidak baku**

Perbaikan kata tidak baku merupakan proses yang dilakukan untuk mengubah kata-kata yang tidak baku menjadi kata baku sesuai dengan Kamus Besar Bahasa Indonesia (KBBI). Pada perbaikan kata tidak baku ini dilakukan pengecekan terhadap kamus\_katabaku yang diambil dari penelitian sebelumnya dan ditambahkan oleh penulis. Kode program proses perbaikan kata tidak baku dapat dilihat pada Kode Program 5.4.

```

1 def perbaikan(self, term):
2     kata_tdkbaku, kata_baku = self.kamus_katabaku()
3     if term in kata_tdkbaku:
4         index = kata_tdkbaku.index(term)
5         term = kata_baku[index]
6     return term

```

#### Kode Program 5.4 Perbaikan kata tidak baku

Pada Kode Program 5.4, dijelaskan pembuatan fungsi *perbaikan* kata tidak baku. Pada proses ini dilakukan pengubahan kata dengan menggunakan kamus\_katabaku. Jika kata ada dalam kata\_tdkbaku maka kata akan diubah menjadi kata yang ada dalam kata\_baku, sehingga hasil yang dikeluarkan adalah kata\_baku.

#### 5.3.5 Filtering

*Filtering* merupakan proses yang dilakukan untuk menghapus kata yang tidak relevan (tidak memiliki arti) atau tidak penting. Kode program proses *cleaning* dapat dilihat pada Kode Program 5.5.

```

1 def filtering(self, term):
2     stoplist = self.stoplist()
3     if term not in stoplist:
4         return term

```

#### Kode Program 5.5 Filtering

Pada Kode Program 5.5, dijelaskan pembuatan fungsi *filtering*. Pada proses ini dilakukan pengecekan pada kamus *stoplist*. Jika kata tidak ada pada kamus *stoplist* maka kata tetap. Namun, jika kata ada pada kamus *stoplist* maka kata akan dihapus.

#### 5.3.6 Stemming

*Stemming* merupakan proses yang dilakukan untuk mengubah kata berimbuhan menjadi kata dasar. Kode program proses *stemming* dapat dilihat pada Kode Program 5.6.

```

1 def stemming(self, term):
2     stemmer = self.factory.create_stemmer()
3     term = stemmer.stem(term)
4     return term

```

#### Kode Program 5.6 Stemming

Pada Kode Program 5.6, dijelaskan pembuatan fungsi *stemming*. Pada proses *stemming* ini digunakan proses *stemming* Sastrawi, yaitu dengan melakukan *import package* Sastrawi dan dilakukan pengubahan kata jamak atau berimbuhan menjadi kata dasar.

### 5.4 Normalisasi Levenshtein Distance

Normalisasi Levenshtein Distance merupakan proses yang dilakukan untuk menangani kesalahan penulisan kata dengan cara menghitung jumlah *edit* suatu kata dengan kata yang lain. Proses ini dilakukan setelah proses *pre-processing*.

```

1 def normalisasi(self, input):
2     list = self.wordlist()
3     if input in list:
4         return input
5     else:
6         dterpendek = -1
7         kata데kat = ''
8         for word in list:
9             lev = self.editdistance(input, word)
10            if not lev:
11                kata데kat = word
12                dterpendek = 0
13                break
14            if lev <= dterpendek or dterpendek < 0:
15                kata데kat = word
16                dterpendek = lev
17            if not dterpendek:
18                return kata데kat
19            else:
20                normkata = input.replace(input, kata데kat)
21        return normkata

```

#### Kode Program 5.7 Normalisasi

Pada Kode Program 5.7, dijelaskan pembuatan fungsi normalisasi. Pada proses ini dilakukan perbandingan kata masukan dengan kata yang ada pada kamus *wordlist*. Pada baris 2 dilakukan pemanggilan kamus *wordlist*. Jika kata inputan ada pada kamus maka keluaran adalah kata tersebut. Namun jika tidak maka dilakukan perhitungan *editdistance* pada baris ke 9. Hasil dari nilai *editdistance* digunakan untuk mengubah kata yang salah menjadi kata yang benar. Jika *lev* bernilai 0 maka kata tetap menjadi kata yang sama, namun jika *lev* bernilai kurang dari sama dengan *dterpendek* atau kurang dari 0 maka kata akan diubah dengan kata yang terdekat. Dalam hal ini, nilai *lev* yang terpendek menunjukkan bahwa kata tersebut mirip dengan kamus.

#### 5.4.2 Edit distance

*Edit distance* merupakan proses yang dilakukan untuk menghitung jarak antara 2 string dalam pencocokan kata. Jarak terdekat dijadikan pilihan untuk mengubah kata menjadi baku. Kode program proses *editdistance* dapat dilihat pada Kode Program 5.8.

```

1 def editdistance(self, o, t):
2     m = len(o)
3     n = len(t)
4     d = [[0 for i in range(n + 1)] for x in range(m + 1)]
5     for i in xrange(m+1):
6         d[i][0] = i

```

```
7     for j in xrange(n+1):
8         d[0][j] = j
9     for i in xrange(m):
10        for j in xrange(n):
11            cost = 0 if o[i] == t[j] else 1
12            d[i + 1][j + 1] = min(
13                (d[i][j + 1]) + 1,
14                (d[i + 1][j]) + 1,
15                (d[i][j]) + cost
16            )
17    return d[-1][-1]
```

## Code Program 5.8 Edit Distance

Pada Kode Program 5.8, dijelaskan pembuatan fungsi *editdistance*. Pada proses ini dilakukan pembuatan matriks yang membandingkan kata masukan dengan kata yang ada pada kamus *wordlist*. Pada baris ke 11 dijelaskan jika huruf pada kata masukan sama dengan huruf pada kamus maka *cost* bernilai 0, dan jika tidak maka bernilai 1. Pengecekan dimulai dari iterasi awal dari kedua *string* kemudian dilakukan operasi penghapusan, penambahan, dan pengubahan karakter. Sehingga hasil yang dikeluarkan adalah nilai *editdistance* minimum.

## 5.5 Proses Analisis sentimen

Proses analisis sentimen merupakan proses yang digunakan untuk menghitung klasifikasi menggunakan Multinomial Naive Bayes. Pada proses ini meliputi tahapan perhitungan *prior*, *conditional probability*, *total conditional probability*, *posterior*, dan klasifikasi sentimen.

### 5.5.1 Prior

*Prior* merupakan proses perhitungan yang dilakukan untuk menghitung nilai *prior* positif dan negatif. Kode program perhitungan *prior* dapat dilihat pada Kode Program 5.9.

```
1     def prior(self, kelas):
2         jmlpost = 0
3         jmlneg = 0
4         for k in kelas:
5             if k == 1:
6                 jmlpost +=1
7             else:
8                 jmlneg +=1
9         totalclass = jmlpost+jmlneg
10        priorpost = float(jmlpost) / float(totalclass)
11        priorneg = float(jmlneg) / float(totalclass)
12        return jmlpost,jmlneg,priorpost,priorneg
```

# **Code Program 5.9 Prior**

Pada Kode Program 5.9, dijelaskan pembuatan fungsi *prior*. Pada proses ini dilakukan perhitungan jumlah *term* positif dan negatif untuk dilakukan perhitungan prior positif dan negatif. Pada baris 10 dan 11 merupakan perhitungan prior yang menggunakan rumus pada Persamaan 2.4.

### 5.5.2 Conditional Probability

*Conditional probability* merupakan proses perhitungan yang dilakukan untuk menghitung nilai *conditional probability* positif dan negatif. Kode program perhitungan *conditional probability* dapat dilihat pada Kode Program 5.10.

```

1 def conditionalprob(self, latih, uji, kelas):
2     kataunik = []
3     jmlseluruhkatapost = 0
4     jmlseluruhkataneg = 0
5     for doc in latih:
6         for k in doc:
7             if k not in kataunik:
8                 kataunik.append(k)
9                 jmlkataunik = len(kataunik)
10                for x in range(0, len(latih)):
11                    if kelas[x] == 1:
12                        jmlseluruhkatapost += len(latih[x])
13                    else:
14                        jmlseluruhkataneg += len(latih[x])
15                doc1 = []
16                for w in uji:
17                    doc = []
18                    for kata in w:
19                        jmlkatapost = 0
20                        jmlkataneg = 0
21                        dict = {}
22                        for y in range(0, len(latih)):
23                            if kata in latih[y] and kelas[y] == 1:
24                                jmlkatapost += latih[y].count(kata)
25                            elif kata in latih[y] and kelas[y] == -1:
26                                jmlkataneg += latih[y].count(kata)
27                            hitungpost = float(jmlkatapost+1) /
28                                float(jmlseluruhkatapost + jmlkataunik)
29                            hitungneg = float(jmlkataneg + 1) /
30                                float(jmlseluruhkataneg + jmlkataunik)
31                            dict.update({"jmlpost" : round(hitungpost,3)})
32                            dict.update({"jmlneg": round(hitungneg,3)})
33                            doc.append(dict)
34                doc1.append(doc)
35            return doc1,jmlkataunik,jmlseluruhkatapost,
36            jmlseluruhkataneg

```

### Kode Program 5.10 Conditional probability

Pada Kode Program 5.10, dijelaskan pembuatan fungsi *conditional probability*.

Pada proses ini dilakukan perhitungan jumlah seluruh kata positif maupun negatif, jumlah kata unik, dan jumlah kata positif dan negatif. Tujuannya untuk dilakukan perhitungan *conditional probability* pada rumus yang ditunjukkan pada Persamaan 2.6. Baris kode yang digunakan untuk menghitung *conditional probability* ditunjukkan pada baris 27 dan 28. Pada baris 27 untuk menghitung *conditional probability* positif dan baris 28 untuk menghitung *conditional probability* negatif.

### 5.5.3 Total Conditional Probability

*Total conditional probability* merupakan proses perhitungan yang dilakukan untuk menghitung nilai *total conditional probability* positif dan negatif. Kode

program perhitungan *total conditional probability* dapat dilihat pada Kode Program 5.11.

#### Program 5.11.

```
1 def totalconditionalprob(self, cp):
2     tcp = []
3     for s in cp:
4         totcppost = 1
5         totcpneg = 1
6         dict = {}
7         for c in s:
8             totcppost *= c.get('jmlpost')
9             totcpneg *= c.get('jmlneg')
10            dict.update({"totcppost": totcppost})
11            dict.update({"totcpneg": totcpneg})
12        tcp.append(dict)
13    return tcp
```

#### Kode Program 5.11 *Total conditional probability*

Pada Kode Program 5.11, dijelaskan pembuatan fungsi *total conditional probability*. Pada proses ini dilakukan perhitungan perkalian dari hasil *conditional probability* positif maupun negatif. Yang hasilnya nanti akan digunakan untuk proses *posterior*.

#### 5.5.4 Posterior

*Posterior* merupakan proses perhitungan yang dilakukan untuk menghitung nilai *posterior* positif dan negatif. Kode program perhitungan *posterior* dapat dilihat pada Kode Program 5.12.

```
1 def posterior(self,priorpost,priorneg,tcp):
2     hasil = []
3     for k in tcp:
4         positif = priorpost * k.get('totcppost')
5         negatif = priorneg * k.get('totcpneg')
6         dict = {}
7         dict.update({"positif": positif})
8         dict.update({"negatif": negatif})
9         hasil.append(dict)
10    return hasil
```

#### Kode Program 5.12 *Posterior*

Pada Kode Program 5.12, dijelaskan pembuatan fungsi *posterior*. Pada proses ini dilakukan perhitungan hasil dari masing-masing *prior* positif dan negatif dikalikan dengan hasil dari masing-masing *total conditional probability* positif dan negatif.

#### 5.5.5 Klasifikasi

Klasifikasi merupakan proses yang dilakukan untuk mengklasifikasikan dokumen ke dalam kelas positif dan negatif. Kode program perhitungan klasifikasi dapat dilihat pada Kode Program 5.13.

```
1 def klasifikasi(self, hasil):
2     klasifikasi = []
3     for k in hasil:
4         if k.get("positif") > k.get("negatif"):
5             klas = "positif"
6         else:
```

```

7     klas = "negatif"
8     klasifikasi.append(klas)
9     return klasifikasi

```

### Kode Program 5.13 Klasifikasi

Pada Kode Program 5.13, dijelaskan proses pembuatan fungsi klasifikasi. Pada proses ini dilakukan pengklasifikasian dari hasil *posterior*. Jika nilai positif yang dihasilkan lebih besar maka sentimen masuk kelas positif. Namun jika tidak maka klasifikasi sentimen masuk kelas negatif.

## 5.6 Proses Evaluasi

Proses evaluasi merupakan proses yang digunakan untuk membandingkan hasil implementasi dengan standar dan kriteria yang telah ditetapkan untuk mengetahui tingkat keberhasilannya. Dari proses evaluasi ini akan didapatkan informasi mengenai sejauh mana hasil yang telah dicapai sehingga dapat diketahui jika terdapat selisih antara hasil dengan standar ketetapan. Evaluasi yang digunakan untuk mengukur kinerja sistem pada kasus klasifikasi adalah dengan menghitung nilai *accuracy*, *recall*, *precision*, dan *f-measure* dari hasil klasifikasi yang didapatkan dari sistem yang dibandingkan dengan klasifikasi sebenarnya.

### 5.6.1 Accuracy

*Accuracy* merupakan proses yang digunakan untuk mengukur tingkat kedekatan antara nilai prediksi dengan nilai aktual. Apabila mendapat nilai *accuracy* 100%, maka hal ini menunjukkan bahwa kondisi yang diprediksi itu benar seperti pada kondisi aslinya. Kode program perhitungan *accuracy* dapat dilihat pada Kode Program 5.14.

```

1 def accuracy(self,tp,tn,fp,fn):
2     akurasi = round((float(tn + tp) / float(tn + tp + fp + fn))
3                      * 100, 3)
4     return akurasi

```

### Kode Program 5.14 Accuracy

Pada Kode Program 5.14, dijelaskan proses pembuatan fungsi *accuracy*. Pada proses ini dilakukan perhitungan yang ditunjukkan pada baris 2 dengan menggunakan rumus yang ditunjukkan pada Persamaan 2.7.

### 5.6.2 Precision

*Precision* merupakan proses yang digunakan untuk mengukur tingkat ketepatan hasil klasifikasi dari seluruh dokumen. *Precision* dihitung dengan cara membagi jumlah pengenalan yang bernilai benar oleh sistem dengan jumlah keseluruhan pengenalan yang dilakukan oleh sistem. Secara matematis pengukuran *precision* yaitu jumlah sampel berkategori positif yang diklasifikasi benar dibagi dengan total sampel yang diklasifikasi sebagai sampel positif. Kode program perhitungan *precision* dapat dilihat pada Kode Program 5.15.

```

1 def precision(self, tp, fp):
2     presisi = round((float(tp) / float(tp + fp)) * 100, 3)
3     return presisi

```

### Kode Program 5.15 Precision

Pada Kode Program 5.15, dijelaskan proses pembuatan fungsi *precision*. Pada proses ini dilakukan perhitungan yang ditunjukkan pada baris 2 dengan menggunakan rumus yang ditunjukkan pada Persamaan 2.8.

#### 5.6.3 Recall

*Recall* merupakan proses yang digunakan untuk mengukur tingkat keberhasilan mengenali suatu kelas yang harus dikenali. Secara matematis pengukuran *recall* yaitu jumlah sampel diklasifikasi positif dibagi total sampel dalam data uji berkategori positif. Kode program perhitungan *recall* dapat dilihat pada Kode Program 5.16.

```

1 def recall(self, tp, fn):
2     recall = round((float(tp) / float(tp + fn)) * 100, 3)
3     return recall

```

### Kode Program 5.16 Recall

Pada Kode Program 5.16, dijelaskan proses pembuatan fungsi *recall*. Pada proses ini dilakukan perhitungan yang ditunjukkan pada baris 2 dengan menggunakan rumus yang ditunjukkan pada Persamaan 2.9.

#### 5.6.4 F-Measure

*F-measure* merupakan penggabungan *precision* dan *recall*. Atau nilai yang mewakili keseluruhan kinerja sistem. Kode program perhitungan *f-measure* dapat dilihat pada Kode Program 5.17.

```

1 def measure(self, presisi, recall):
2     measure = round((2*presisi*recall)/(presisi+recall), 3)
3     return measure

```

### Kode Program 5.17 F-measure

Pada Kode Program 5.17, dijelaskan proses pembuatan fungsi *f-measure*. Pada proses ini dilakukan perhitungan yang ditunjukkan pada baris 2 dengan menggunakan rumus yang ditunjukkan pada Persamaan 2.10.

## BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini penulis menjelaskan mengenai pengujian yang dilakukan terhadap sistem yang telah dibuat. Setelah pengujian dilakukan maka dilakukan analisis untuk mengetahui pengaruh masing-masing parameter yang diuji pada setiap pengujian.

### 6.1 Skenario Pengujian

Pada pengujian ini terdapat 3 macam pengujian yang dilakukan. Pengujian pertama dilakukan dengan tujuan untuk mengetahui pengaruh proses penggunaan *pre-processing*. Kemudian untuk pengujian kedua dilakukan untuk mengetahui pengaruh penggunaan proses perbaikan kata tidak baku. Dan terakhir adalah pengujian untuk mengetahui pengaruh dari dilakukannya proses *pre-processing* dengan perbaikan kata tidak baku dan normalisasi Levenshtein Distance.

#### 6.1.1 Pengujian pengaruh penggunaan *pre-processing*

Pengujian ini menjelaskan tentang pengujian untuk mengetahui pengaruh proses *pre-processing*. Pada pengujian ini dilakukan tiga tahapan proses, yang pertama yaitu dilakukan pengujian tanpa *pre-processing* dan tanpa perbaikan kata tidak baku, kedua yaitu tanpa *pre-processing* dengan perbaikan kata tidak baku, dan ketiga yaitu dilakukan dengan *pre-processing* dan tanpa perbaikan kata tidak baku. Data yang digunakan pada pengujian ini adalah data asli dari pengguna

Twitter mengenai Tweet opini film berbahasa Indonesia. Data latih yang diambil sebanyak 140 data opini, yang terdiri dari 70 data opini positif dan 70 data opini negatif. Sedangkan untuk data uji digunakan 60 data, yang terdiri dari 30 data opini positif dan 30 data opini negatif. Hasil pengujian pengaruh penggunaan *pre-processing* dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil pengujian pengaruh penggunaan *pre-processing*

Jenis Uji	Klasifikasi sentimen			
	Accuracy	Precision	Recall	F-Measure
Tanpa <i>Pre-processing</i> dan tanpa perbaikan kata tidak baku	70%	80%	53.33%	64%
Tanpa <i>Pre-processing</i> dengan perbaikan kata tidak baku	75%	71.43%	83.33%	76.92%
Dengan <i>Pre-processing</i> tanpa perbaikan kata tidak baku	86.67%	92.31%	80%	85.72%

### **6.1.2 Pengujian pengaruh penggunaan perbaikan kata tidak baku**

Pengujian ini menjelaskan tentang pengujian untuk mengetahui pengaruh proses perbaikan kata tidak baku. Pada perbaikan kata tidak baku dilakukan dua proses. Proses pertama dilakukan dengan pencocokan kata tidak baku pada kamus\_katabaku dan kedua adalah dengan normalisasi Levenshtein Distance. Pada pengujian ini dilakukan proses pertama saja yaitu perbaikan dengan kamus\_katabaku. Data latih dan data uji sama-sama dilakukan proses *pre-processing*. Dan dilakukan variasi dalam proses perbaikan kata tidak baku. Pada pengujian ini data yang digunakan merupakan data asli dari pengguna Twitter mengenai Tweet opini film berbahasa Indonesia. Data latih yang diambil sebanyak 140 data opini, yang terdiri dari 70 data opini positif dan 70 data opini negatif. Sedangkan untuk data uji digunakan 60 data, yang terdiri dari 30 data opini positif dan 30 data opini negatif. Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dapat dilihat pada Tabel 6.2.

**Tabel 6.2 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku**

Klasifikasi sentimen tanpa dan dengan perbaikan kata tidak baku				
Jenis Uji	Accuracy	Precision	Recall	F-Measure
<i>Pre-processing</i> tanpa perbaikan kata tidak baku	86.67%	92.31%	80%	85.72%
<i>Pre-processing</i> dengan perbaikan kata tidak baku	91.67%	85.71%	100%	92.31%

### **6.1.3 Pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein distance**

Pengujian ini menjelaskan tentang pengujian untuk mengetahui pengaruh proses perbaikan kata tidak baku. Pada perbaikan kata tidak baku dilakukan dua proses. Proses pertama dilakukan dengan pencocokan kata tidak baku pada kamus\_katabaku dan kedua adalah dengan normalisasi Levenshtein Distance. Pada pengujian ini dilakukan perbaikan kata tidak baku dengan seluruh proses. Data latih dan data uji dilakukan variasi proses *pre-processing*. Dan dilakukan proses perbaikan kata tidak baku dan normalisasi Levenshtein Distance. Pada pengujian ini data yang digunakan merupakan data asli dari pengguna Twitter mengenai Tweet opini film berbahasa Indonesia. Data latih yang diambil sebanyak 140 data opini, yang terdiri dari 70 data opini positif dan 70 data opini negatif. Sedangkan untuk data uji digunakan 60 data, yang terdiri dari 30 data opini positif dan 30 data opini negatif. Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance dapat dilihat pada Tabel 6.3.

**Tabel 6.3 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance**

Klasifikasi sentimen dengan perbaikan kata tidak baku dan normalisasi kata Levenshtein Distance					
Jenis Uji	Universitas	Accuracy	Precision	Recall	F-Measure
Tanpa <i>Pre-processing</i> dengan perbaikan kata tidak baku dan Levenshtein Distance	Universitas	85%	86.21%	83.33%	84.75%
<i>Pre-processing</i> dengan perbaikan kata tidak baku dan Levenshtein Distance	Universitas	98.33%	96.77%	100%	98.36%

## 6.2 Hasil Pengujian

Hasil pengujian berisi penjelasan mengenai analisis dari pengujian yang telah dilakukan. Terdapat 3 hasil pengujian, yaitu hasil pengujian pengaruh proses *pre-processing*, hasil pengujian pengaruh penggunaan perbaikan kata tidak baku, serta hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein Distance.

### 6.2.1 Hasil pengujian pengaruh penggunaan *pre-processing*

Pada pengujian pengaruh penggunaan *pre-processing* dilakukan tiga tahapan yakni, tahapan pertama data uji tanpa dilakukan proses *pre-processing* dan tanpa dilakukan perbaikan kata tidak baku menggunakan kamus\_katabaku, tahapan kedua data uji tanpa dilakukan proses *pre-processing* namun dilakukan perbaikan kata tidak baku menggunakan kamus\_katabaku, dan tahap ketiga yaitu data latih dan uji dilakukan proses *pre-processing* dan tanpa dilakukan perbaikan kata tidak baku menggunakan kamus\_katabaku. Dari pengujian tersebut diperoleh hasil dari tahap pertama dengan nilai *accuracy*, *precision*, *recall*, dan *f-measure* sebesar 70%, 80%, 53.33%, dan 64%. Hasil tersebut dikatakan rendah karena proses *pre-processing* tidak digunakan pada data uji sehingga kata yang digunakan untuk pengklasifikasian masih memiliki kata-kata asli seperti jeleknya, terjelek, jelekk. Padahal kata tersebut memiliki arti kata yang sama yaitu jelek. Akibat tidak dilakukannya proses *pre-processing* membuat semakin banyaknya kata yang tidak berguna yaitu kata yang ada dalam data uji tidak banyak muncul pada data latih. Sehingga membuat proses klasifikasi menjadi tidak sempurna dan menghasilkan nilai akurasi yang rendah.

Selanjutnya pada tahap dua sama seperti tahap satu yaitu data uji tanpa dilakukan *pre-processing* namun pada tahap ini dilakukan perbaikan kata tidak baku dengan kamus\_katabaku. Dihasilkan *accuracy*, *precision*, *recall*, dan *f-*

*measure* sebesar 75%, 71.43%, 83.33%, dan 76.92%. Hasil ini menunjukkan hasil yang lebih baik dibandingkan dengan tanpa *pre-processing* dan tanpa perbaikan. Karena kata yang ada pada data uji dilakukan perbaikan kata dengan kamus\_katabaku. Yaitu misal pada kata “bgt” diperbaiki menjadi “banget”. Sehingga kata yang ada dalam data uji akan lebih banyak muncul pada data latih dibandingkan tahap satu. Namun masih memiliki kekurangan karena kata tidak penting atau tidak memiliki arti masih tercantum akibat tidak dilakukannya *filtering*. Dan juga tanpa dilakukannya proses mengubah kata berimbuhan menjadi kata dasar (*stemming*).

Selanjutnya tahap ketiga adalah tahap dilakukannya proses *pre-processing* dan tanpa perbaikan. Hasil *accuracy*, *precision*, *recall*, dan *f-measure* yang didapat adalah sebesar 86.67%, 92.31%, 80%, dan 85.72%. Hasil ini jauh lebih baik dibandingkan dengan tahap sebelumnya yang tanpa dilakukan *pre-processing*. Dari hasil tersebut menunjukkan bahwa dengan *pre-processing*, kata yang tidak memiliki arti akan dihilangkan, dan juga kata berimbuhan yang memiliki kata dasar yang sama akan dianggap sebagai suatu kata yang sama. Sehingga hasil dari pengujian ini dapat dibuktikan bahwa *pre-processing* berpengaruh besar pada hasil klasifikasi.

### 6.2.2 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku

Pada pengujian pengaruh penggunaan perbaikan kata tidak baku dilakukan dengan cara dilakukan proses *pre-processing* namun dilakukan variasi perbaikan kata baku dengan kamus kata\_baku. Terdapat dua tahapan, tahapan pertama tidak dilakukan perbaikan kata dan tahap kedua dengan dilakukan perbaikan kata. Hasil pada tahap satu menunjukkan hasil *accuracy*, *precision*, *recall*, dan *f-measure* sebesar 86.67%, 92.31%, 80%, dan 85.72%. Dan hasil pada tahap dua menunjukkan hasil sebesar 91.67%, 85.71%, 100%, dan 92.31%. Dari perbandingan nilai tersebut dapat dikatakan bahwa klasifikasi yang dilakukan proses *pre-processing* dan perbaikan kata tidak baku mendapatkan hasil yang jauh lebih baik. Karena data uji dijadikan kata yang lebih terstruktur dahulu sebelum dilakukan proses klasifikasi.

### 6.2.3 Hasil pengujian pengaruh penggunaan perbaikan kata tidak baku dan normalisasi Levenshtein distance

Pada pengujian ini dilakukan pengujian untuk mengetahui pengaruh proses perbaikan kata tidak baku. Pada perbaikan kata tidak baku dilakukan dua proses.

Proses pertama dilakukan dengan pencocokan kata tidak baku pada kamus\_katabaku dan kedua adalah dengan normalisasi Levenshtein Distance.

Pada pengujian ini dilakukan perbaikan kata tidak baku dengan seluruh proses. Data latih dan data uji dilakukan variasi proses *pre-processing* dan sama-sama dilakukan proses perbaikan kata tidak baku dan normalisasi Levenshtein Distance.

Hasil dari pengujian tahap pertama yang tanpa dilakukan proses *pre-processing* didapatkan nilai *accuracy*, *precision*, *recall*, dan *f-measure* masing-masing dengan nilai 85%, 86.21%, 83.33%, dan 84.75%. Hasil ini menghasilkan akurasi yang baik

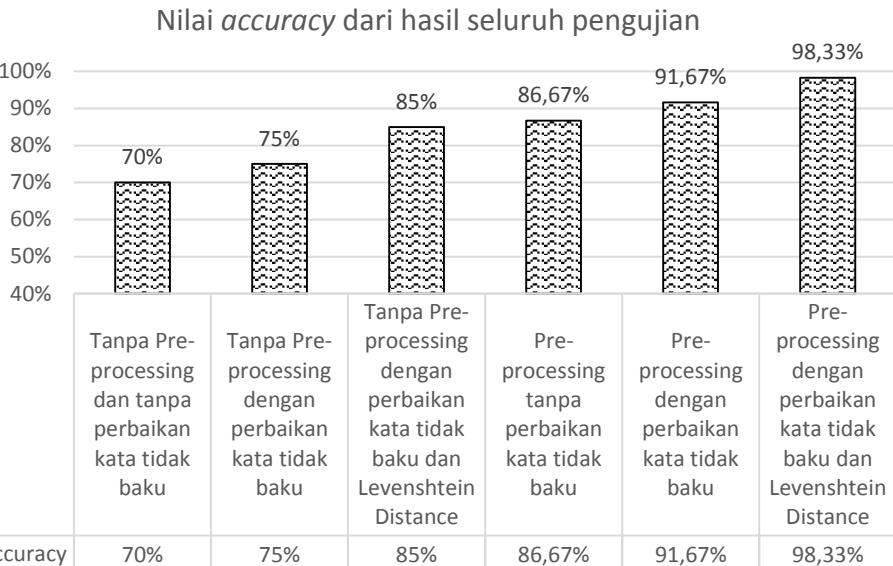
walaupun tidak dilakukan proses *pre-processing*, karena pada saat perbaikan dengan menggunakan normalisasi Levenshtein Distance setiap kata akan dicocokkan dengan kamus *wordlist*. Sehingga kata yang tidak terstruktur tergantikan oleh kata yang ada pada *wordlist* yang didapat dari perhitungan *editdistance* dengan mengambil nilai *edit* terendah untuk mengganti kata.

Kemudian tahap kedua adalah dengan dilakukannya proses *pre-processing* dan perbaikan kata dan Levenshtein Distance. Hasil dari pengujian ini didapatkan nilai *accuracy*, *precision*, *recall*, dan *f-measure* masing-masing dengan nilai 98.33%, 96.77%, 100%, dan 98.36%. Hasil ini menghasilkan akurasi terbaik dari seluruh pengujian diatas. Karena pada pengujian ini dilakukan seluruh proses mulai dari proses *pre-processing* yang menjadikan kata tidak terstruktur menjadi terstruktur dan ditambah dengan perbaikan kata tidak baku berdasarkan kamus\_katabaku yang mengubah kata singkatan atau kata *slang* menjadi kata baku. Selain itu juga dilakukan proses perbaikan kata dengan Levenshtein Distance. Yaitu memperbaiki kesalahan kata seperti salah eja yang tidak bisa diperbaiki pada kamus\_katabaku. Sehingga kata yang dihasilkan data uji memiliki banyak kemunculan kata sama dengan data latih yang berpengaruh pada proses klasifikasi. Dengan banyaknya kata yang muncul pada data latih maka menjadikan nilai akurasi menjadi lebih baik.

#### 6.2.4 Hasil dari seluruh pengujian

Pada pengujian ini dilakukan untuk mengetahui hasil dari seluruh pengujian yang telah dilakukan. Yaitu dengan menampilkan grafik dari seluruh pengujian. Hasil dari pengujian ini terdiri dari nilai *accuracy*, *precision*, *recall*, dan *f-measure*.

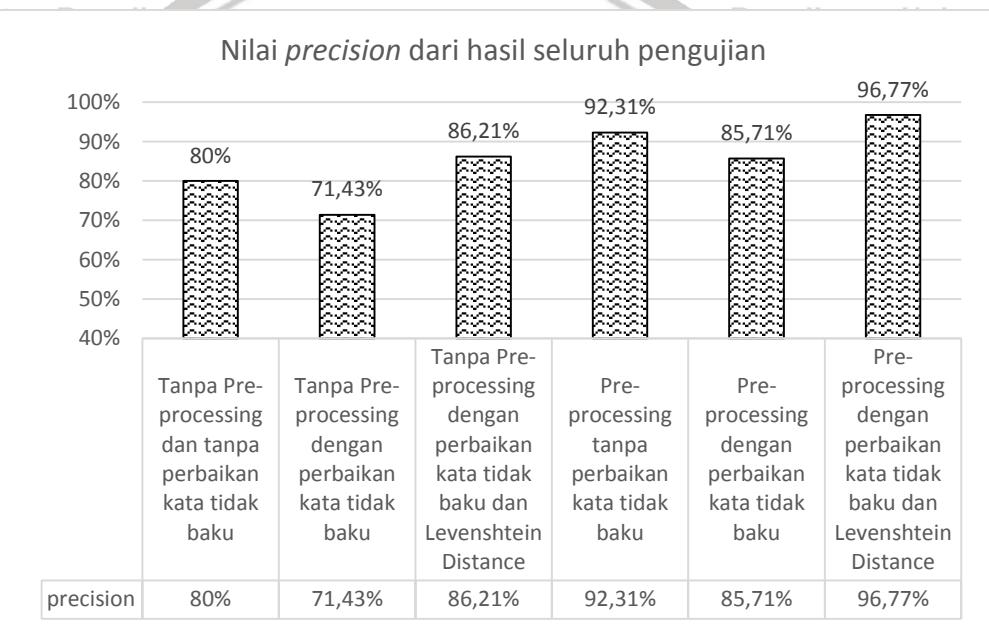
##### 1. Hasil *accuracy*



Gambar 6.1 Nilai *accuracy* dari seluruh pengujian

Pada Gambar 6.1 menunjukkan hasil dari nilai *accuracy* yaitu nilai tingkat kedekatan antara nilai prediksi dengan nilai aktual. Pada grafik tersebut dapat dilihat bahwa pengujian dengan *pre-processing* dan perbaikan kata tidak baku dan Levenshtein Distance menghasilkan akurasi tertinggi. Karena pada pengujian tersebut dilakukan berbagai tahapan mulai dari proses *pre-processing*, perbaikan kata tidak baku menggunakan kamus\_katabaku, dan perbaikan kata tidak baku dengan normalisasi Levenshtein Distance. Sedangkan pada pengujian yang lain menghasilkan akurasi yang lebih rendah karena dipengaruhi oleh beberapa faktor. Pada grafik *accuracy* semakin banyak tahapan proses yang dilakukan maka tingkat akurasinya semakin naik.

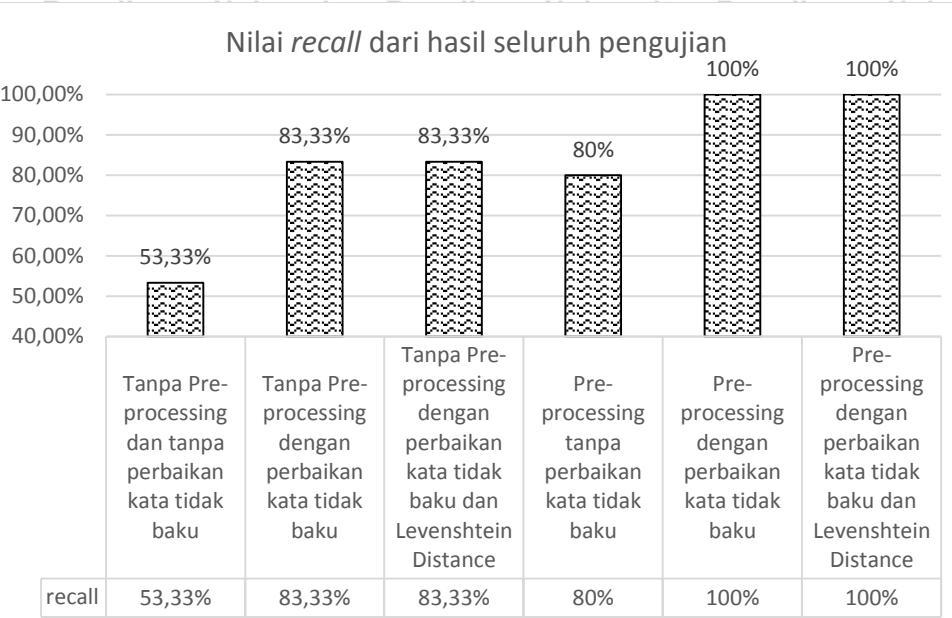
## 2. Hasil *precision*



**Gambar 6.2 Nilai *precision* dari seluruh pengujian**

Pada Gambar 6.2 menunjukkan hasil dari nilai *precision* yaitu nilai tingkat ketepatan hasil klasifikasi dari seluruh dokumen. *Precision* dihitung dengan cara membagi jumlah pengenalan yang bernilai benar oleh jumlah keseluruhan pengenalan yang dilakukan oleh sistem. Secara matematis pengukuran *precision* yaitu jumlah sampel berkategori positif yang diklasifikasi benar dibagi dengan total sampel yang diklasifikasi sebagai sampel positif. Hasil dari *precision* menunjukkan bahwa semakin banyak tahapan proses yang dilakukan nilai *precision* yang diperoleh cenderung naik.

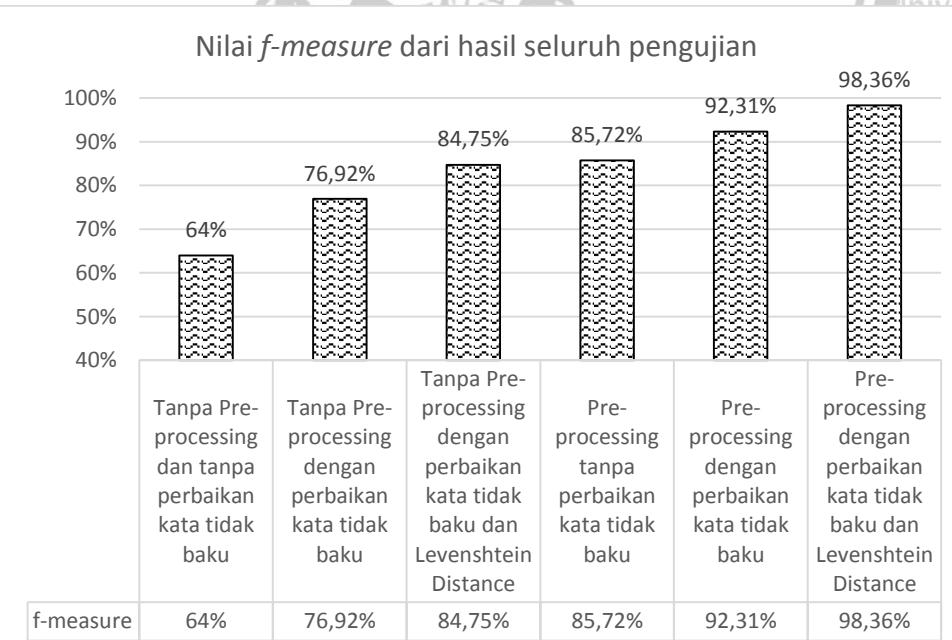
### 3. Hasil recall



**Gambar 6.3 Nilai recall dari seluruh pengujian**

Pada Gambar 6.3 menunjukkan hasil dari nilai *recall* yaitu nilai tingkat keberhasilan mengenali suatu kelas yang harus dikenali. Hasil dari *recall* menunjukkan bahwa semakin banyak tahapan proses yang dilakukan maka nilai *recall* yang diperoleh cenderung naik.

### 4. Hasil f-measure

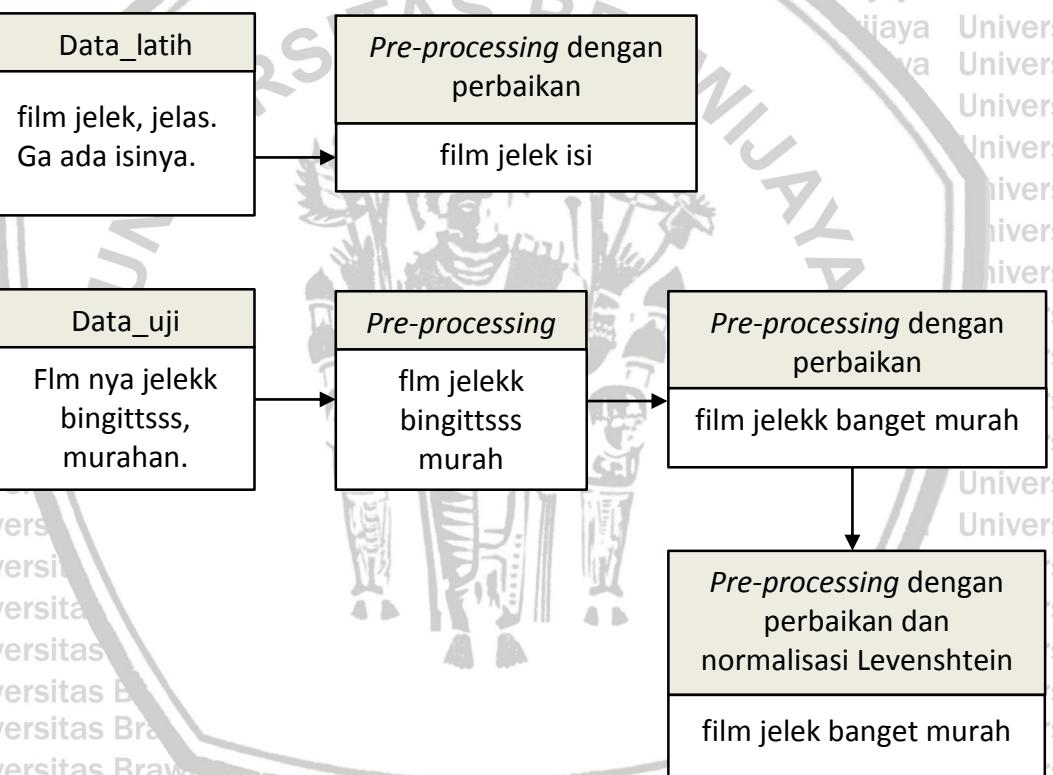


**Gambar 6.4 Nilai f-measure dari seluruh pengujian**

Pada Gambar 6.4 menunjukkan hasil dari nilai  $f - measure$  yaitu penggabungan *precision* dan *recall*. Atau nilai yang mewakili keseluruhan kinerja sistem. Hasil pada nilai *f-measure* menunjukkan nilai tertinggi berada pada pengujian dengan *pre-processing* dan perbaikan kata tidak baku dan Levenshtein Distance dan nilai terendah berada pada pengujian tanpa *pre-processing* dan tanpa perbaikan kata tidak baku. Hasil dari *f-measure* menunjukkan bahwa semakin banyak tahapan proses yang dilakukan maka nilai *f-measure* semakin naik.

### 6.3 Analisis Hasil

Berdasarkan hasil pengujian yang sudah dilakukan, diketahui bahwa hasil akurasi terbaik adalah dengan dilakukannya seluruh kombinasi yaitu proses *pre-processing* dengan perbaikan kata tidak baku dan normalisasi Levenshtein Distance. Untuk dapat menjelaskannya lebih lanjut dapat dilihat pada Gambar 6.5.



**Gambar 6.5 Ilustrasi proses pengujian**

Dari Gambar 6.5 ilustrasi di atas dapat dibuat analisis sebagai berikut:

1. Pada pengujian yang tanpa dilakukan proses *pre-processing* dan perbaikan kata, data uji tidak berubah dari data aslinya. Data yang diklasifikasi masih dalam bentuk data mentah, yaitu data yang belum terstruktur. Sehingga hasil akurasi yang diperoleh adalah rendah dengan nilai *accuracy*, *precision*, *recall*, dan *f-measure* sebesar 70%, 80%, 53.33%, dan 64%. Pada kata yang tidak dilakukan proses *pre-processing* dalam pengklasifikasian masih memiliki kata-



kata asli seperti murahnya, termurah, murahan. Padahal kata tersebut memiliki arti kata yang sama yaitu murah. Akibat tidak dilakukannya proses *pre-processing* membuat semakin banyaknya kata yang tidak berguna yaitu kata yang ada dalam data uji tidak banyak muncul pada data latih. Dan pada pengujian yang tanpa dilakukan *pre-processing* namun dilakukan perbaikan kata menghasilkan akurasi lebih baik dari pada tahap satu. Karena kata yang ada pada data uji dilakukan perbaikan kata dengan kamus\_katabaku. Yaitu misal pada kata “bingittss” diperbaiki menjadi “banget”. Sehingga kata yang ada dalam data uji akan lebih banyak muncul pada data latih dibandingkan tahap satu. Namun pada tahap ini masih memiliki kekurangan karena kata tidak penting atau tidak memiliki arti masih tercantum akibat tidak dilakukannya *filtering*. Dan juga tanpa dilakukannya proses mengubah kata berimbuhan menjadi kata dasar (*stemming*). Sehingga membuat proses klasifikasi menjadi tidak sempurna dan menghasilkan nilai akurasi yang rendah jika dibandingkan dengan dilakukan *pre-processing*. Sehingga proses *pre-processing* sangat berpengaruh pada klasifikasi sentimen analisis menggunakan Naive Bayes.

2. Pada pengujian yang dilakukan dengan proses *pre-processing* menghasilkan akurasi yang lebih tinggi dibandingkan tanpa *pre-processing*. Yaitu menghasilkan nilai *accuracy*, *precision*, *recall*, dan *f-measure* sebesar 86.67%, 92.31%, 80%, dan 85.72%. Namun dengan banyaknya data yang menggunakan kata tidak baku dalam penulisan membuat hasil akurasi tidak dapat memperoleh hasil yang lebih baik. Karena itu perbaikan kata dibutuhkan. Yaitu proses yang dapat mengubah kata yang tidak baku menjadi baku. Seperti “bgt” menjadi “banget”. Sehingga dengan ditambahkan proses perbaikan kata tidak baku pada *pre-processing* menggunakan kamus\_katabaku dapat meningkatkan nilai *accuracy*, *precision*, *recall*, dan *f-measure* menjadi sebesar 91.67%, 85.71%, 100%, dan 92.31%.
3. Pada pengujian yang dilakukan dengan proses *pre-processing* dan perbaikan kata tidak baku dan normalisasi Levenshtein Distance didapatkan hasil dengan nilai *accuracy*, *precision*, *recall*, dan *f-measure* terbaik yaitu masing-masing dengan nilai 98.33%, 96.77%, 100%, dan 98.36%. Hasil ini menghasilkan akurasi terbaik dari seluruh pengujian diatas. Karena pada pengujian ini dilakukan seluruh proses mulai dari proses *pre-processing* yang menjadikan kata tidak terstruktur menjadi terstruktur dan ditambah dengan perbaikan kata tidak baku berdasarkan kamus\_katabaku yang mengubah kata singkatan atau kata *slang* menjadi kata baku. Selain itu juga dilakukan proses perbaikan kata dengan Levenshtein Distance. Yaitu memperbaiki kesalahan kata seperti salah eja yang tidak bisa diperbaiki pada kamus\_katabaku. Sehingga kata yang dihasilkan data uji memiliki banyak kemunculan kata sama dengan data latih yang berpengaruh pada proses klasifikasi. Dengan banyaknya kata yang muncul pada data latih maka menjadikan nilai akurasi menjadi lebih baik.

Pada bab ini penulis membahas mengenai kesimpulan yang didapat dari hasil pengujian analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia menggunakan Naive Bayes dengan perbaikan kata tidak baku. Selain itu penulis juga menulis saran yang dapat digunakan untuk kepentingan penelitian selanjutnya yang berkaitan dengan *text mining* analisis sentimen atau penerapan Naive Bayes dengan perbaikan kata tidak baku.

## **7.1 Kesimpulan**

Berdasarkan hasil pengujian dan analisis yang telah selesai dilakukan, dapat disimpulkan sebagai berikut:

1. Metode klasifikasi Naive Bayes dengan perbaikan kata tidak baku dapat diterapkan pada proses analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia. Data latih dan data uji dilakukan proses *pre-processing* terlebih dahulu, yang mana pada proses *pre-processing* terdapat tambahan perbaikan kata tidak baku menggunakan kamus\_katabaku yang dilakukan setelah *case folding*. Dan perbaikan kata tidak baku menggunakan normalisasi Levenshtein Distance yang dilakukan setelah proses *pre-processing*.
2. Pada penggunaan proses *pre-processing* dan perbaikan kata tidak baku dengan penambahan normalisasi Levenshtein Distance terhadap hasil klasifikasi memberikan pengaruh akurasi yang lebih baik, yaitu sebesar 98.33% dibandingkan dengan *pre-processing* yang hanya dilakukan dengan perbaikan kata tidak baku menggunakan kamus\_katabaku saja. Yaitu dengan nilai akurasi sebesar 91.67%.
3. Berdasarkan pengujian yang telah dilakukan, diperoleh hasil akurasi terbaik dari analisis sentimen tentang opini film pada dokumen Twitter berbahasa Indonesia menggunakan Naive Bayes dengan perbaikan kata tidak baku adalah sebesar 98.33% Dan untuk *precision*, *recall*, dan *f-measure* adalah 96.77%, 100%, dan 98.36%.

## **7.2 Saran**

Berdasarkan penelitian yang telah dilakukan, penulis menganggap bahwa penelitian yang sudah selesai dilakukan masih jauh dari kata sempurna, sehingga berikut terdapat saran yang dapat digunakan untuk kepentingan penelitian selanjutnya yang berkaitan dengan *text mining* analisis sentimen atau penerapan Naive Bayes dengan perbaikan kata tidak baku:

1. Pada perbaikan kata tidak baku penulis menggunakan kamus sendiri yaitu kamus\_katabaku yang didasarkan pada penelitian sebelumnya. Pada kamus\_katabaku yang dibuat penulis mengacu pada dunia perfilman dan kata

bahasa modern (*slang*) maupun kata singkatan yang sering muncul. Sehingga dapat ditambahkan kosakata lain pada kamus\_katabaku.

2. Pada penelitian selanjutnya dapat ditambahkan fitur selain *bag-of-words*. Yaitu fitur lexicon, pembobotan emoticon, atau daftar kata positif dan negatif.

3. Pada penelitian ini juga digunakan perbaikan kata tidak baku menggunakan normalisasi Levenshtein Distance. Yaitu menghitung jarak kedekatan dari dua buah *string* melalui penambahan karakter, pengubahan karakter, dan penghapusan karakter hingga kedua *string* tersebut cocok. Dalam hal ini jarak yang dihitung adalah jarak kedekatan kata pada Tweet dan kata pada Kamus Besar Bahasa Indonesia. Hasilnya terdapat beberapa kata yang tidak sesuai harapan seperti kata ‘asik’ yang seharusnya berubah ‘asyik’ menjadi ‘usik’. Selain itu normalisasi Levenshtein Distance juga membutuhkan waktu yang lama. Sehingga dapat digunakan metode lain seperti Distributional Semantik dengan dataset yang banyak.



## DAFTAR PUSTAKA

- Adiwijaya, I., 2006. *Text Mining dan Knowledge Discovery*. [Online]. Available at : [http://web.ipb.ac.id/~ir-lab/pdf/tm%20summarization\).pdf](http://web.ipb.ac.id/~ir-lab/pdf/tm%20summarization).pdf) [Diakses 24 Februari 2017].
- Adriyani, N. M., Santiyasa, I. W. & Mulyantara, A., 2012. *Implementasi Algoritma Levenshtein Distance dan Metode Impiris untuk Menampilkan Saran Perbaikan Kesalahan Pengetikan Doukem Berbahasa Indonesia*. Fakultas Ilmu Komputer, Universitas Udayana.
- Agarwal, A. et al., 2014. *Sentiment Analysis of Twitter Data*. Department of Computer Science Columbia University. Available at: <http://www.cs.columbia.edu/~julia/papers/Agarwalaletal11.pdf>.
- Aggarwal, C., 2015. *Data Classification: Algorithms and Applications*. Berilustrasi penyunt. New York: CRC Press.
- Amin, F., 2012. *Sistem Temu Kembali Informasi dengan Metode Vector Space Model*. Jurnal Sistem Informasi Bisnis, Volume I, p. 79.
- Asih, R., 2012. *Indonesia Pengguna Twitter Terbesar Kelima Dunia*. [Online]. Available at: <http://www.tempo.co/read/news/2012/02/02/072381323/> [Diakses 21 Februari 2017].
- Asrofi, G., 2015. *Analisis Sentimen Calon Presiden Indonesia 2014 dengan Lima Class Attribute*. Universitas Gadjah Mada. Available at: [http://etd.repository.ugm.ac.id/index.php?mod=penelitian\\_detail&sub=PenelitianDetail&act=view&typ=html&buku\\_id=80122&obyek\\_id=4](http://etd.repository.ugm.ac.id/index.php?mod=penelitian_detail&sub=PenelitianDetail&act=view&typ=html&buku_id=80122&obyek_id=4) [Diakses 24 Februari 2017].
- Buntoro, G. A., Adji, T. B., & Purnamasari, A. E. 2014. *Sentiment Analysis Twitter dengan Kombinasi Lexicon Based dan Double Propagation*. CTIEE. Halaman 39-43.
- Coletta, L.F.S. et al., 2014. *Combining Classification and Clustering for Tweet Sentiment Analysis*. In 2014 Brazilian Conference on Intelligent Systems. IEEE, pp. 210–215. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6984832> [Diakses 23 Februari 2017].
- Destuardi dan Surya, S. 2009. *Klasifikasi Emosi Untuk Teks Bahasa Indonesia Menggunakan Metode Naive Bayes*. Teknik, Institut Teknologi Sepuluh Nopember, Surabaya.
- Feizar, F. H., 2014. *Analisis Sentimen Opini Film Berbahasa Indonesia Berbasis Kamus Menggunakan Metode Neighbor-Weighted K-Nearest Neighbor*. Malang: Fakultas Ilmu Komputer Universitas Brawijaya.

- Firmansyah, R. F. N., 2016. *Sentiment Analysis Pada Review Aplikasi Mobile Menggunakan Metode Naive Bayes dan Query Expansion*. Malang: Fakultas Ilmu Komputer Universitas Brawijaya.
- Freeman AT, Condon SL, Ackerman CM. 2006. *Cross linguistic name matching in English and Arabic: a “one to many mapping” extension of the levenshtein edit distance algorithm*. Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL; 2006 Jun; New York, Amerika Serikat. New York (US): Association for Computational Linguistic. hlm. 471-478.
- Hadna, N. et al., 2016. *Studi Literatur tentang perbandingan metode proses analisis sentimen di twitter*. Fakultas Teknik, Universitas Gadjah Mada.
- Hall, M., 2006. *A Decision Tree-Based Attribute Weighting Filter for Naive Bayes*. Knowledge-Based Systems, pp.120–126. Available at: <http://www.cs.waikato.ac.nz/pubs/wp/2006/uow-cs-wp-2006-05.pdf>.
- Huang, X. & Wu, Q., 2013. *Micro-blog commercial word extraction based on improved TF-IDF algorithm*. In 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013). IEEE, pp. 1–5. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6718884> [Diakses 24 Februari 2017].
- Witten, I. H., Frank, E., and Hall, M.A., 2011. *Data Mining: Practical Machine Learning Tools and Techniques*.
- Kwak, H., Lee, C., Park, H., dan Moon S. 2010. *What is Twitter, a Social Network or a News Media?*. Department of Computer Science, KAIST, Korea.
- Manalu, B.U., 2014. *Analisis Sentimen Pada Twitter Menggunakan Text Mining*. Medan : Universitas Sumatera Utara.
- Manning, C.D., Raghavan, P. & Schütze, H., 2009. *An Introduction to Information Retrieval*. [Online]. Available at : <https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf> [Diakses 26 Februari 2017]
- Moore, H. F., 1987. *Humas: Prinsip, Kasus dan Masalah*. Bandung: Remaja Rosdakarya.
- Nasukawa, T. & Yi, J., 2003. *Sentiment Analysis: Capturing Favorability Using Natural Language Processing*. In Proceedings of the 2nd International Conference on Knowledge Capture. pp. 70–77.
- Nugraha, M., 2014. *Sentiment Analysis Pada Review Film Dengan Menggunakan Metode K-Nearest Neighbor*. Bandung : Universitas Widya Tama Bandung.
- Olson, D. & Shi, Y., 2008. *Pengantar Ilmu Penggalian Data Bisnis*. Terjemahan Bahasa Inggris penyunt. Jakarta: Salemba Empat.

Perdana, R. S., & Pinandito, A. 2017. *Combining Likes-Retweet Analysis and Naive Bayes Classifier Within Twitter for Sentiment Analysis*. Malang : Fakultas Ilmu Komputer Universitas Brawijaya.

Taheri, S. & Mammadov, M., 2013. *Learning The Naive Bayes Classifier With Optimization Models*. In International Journal of Applied Mathematics and Computer Science. pp. 787–795.

Tala, Fadillah Z. 2003. *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. Institute for Logic, Language and Computation Universiteit van Amsterdam The Netherlands.

<http://www illc uva nl/Research/Reports/MoL-2003-02.text.pdf> [Diakses 26 Februari 2017]

The Twitter Government and Election Team, 2014. *The Twitter Government and Election Handbook*, San Francisco: Twitter Inc. Available at: <https://g.twimg.com/elections/files/2014/09/16/TwitterGovElectionsHandbook.pdf>. [Diakses 25 Februari 2017].

Troussas, C. et al., 2013. *Sentiment analysis of Facebook statuses using Naive Bayes classifier for language learning*. In IISA 2013. IEEE, pp. 1–6. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6623713> [Diakses 25 Februari 2017].

Utomo, D. C., 2015. *Automatic Essay Scoring Menggunakan Metode N-Gram dan Cosine Similarity*. Malang: Fakultas Ilmu Komputer Universitas Brawijaya.