ANALISIS SENTIMEN OPINI FILM MENGGUNAKAN METODE *NAÏVE BAYES* DENGAN *ENSEMBLE FEATURE* DAN SELEKSI FITUR *PEARSON CORRELATION COEFFICIENT*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fachrul Rozy Saputra Rangkuti
NIM: 145150207111111

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

Analisis Sentimen Opini Film Menggunakan Metode *Naïve Bayes* Dengan *Ensemble Feature* Dan Seleksi Fitur *Pearson Correlation Coefficient*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Fachrul Rozy Saputra Rangkuti
NIM: 145150207111111

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
M. Ali Fauzi, S.Kom., M.Kom	Yuita Arum Sari, S.Kom., M.Kom
NIK: 201502 890101 001	NIK: 201609 880715 2 001

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 (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 30 Juli 2018

Fachrul Rozy Saputra Rangkuti

NIM:145150207111111

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Allah SWT yang telah memberikan rahmat, hidayah serta inayah-Nya kepada penulis, sehingga penulis dapat menyelesaikan penelitian berjudul "Analisis Sentimen Opini Film Menggunakan Metode *Naïve Bayes* dengan *Ensemble Feature* dan Seleksi Fitur *Pearson Correlation Coefficient*". Melalui serangkaian tulisan ini penulis juga ingin menyampaikan rasa terima kasih kepada seluruh pihak yang berperan dalam memberikan bantuan serta dukungan kepada penulis dalam menyelesaikan skripsi ini, diantaranya:

1. Bapak M. Ali Fauzi, S.Kom, M.Kom selaku dosen pembimbing I yang telah membagikan ilmu dan meluangkan waktu serta arahan dalam proses pengerjaan skripsi ini.
2. Ibu Yuita Arum Sari, S.Kom., M.Kom selaku dosen pembimbing II yang juga telah membagikan ilmu dan meluangkan waktu serta arahan dalam proses pengerjaan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
6. Bapak Ir. Heru Nurwasito, M.Kom dan Bapak M. Ali Fauzi, S.Kom, M.Kom selaku dosen pendamping akademik yang telah memberikan solusi dan nasihat kepada penulis selama menempuh studi.
7. Segenap Bapak dan Ibu Dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis dari awal kuliah hingga skripsi ini dapat terselesaikan.
8. Orang tua penulis H. S. Khoir, Drs dan Ibu Masturo Harahap yang senantiasa memberikan dukungan baik dalam bentuk materi maupun do'a kepada penulis.
9. Ibu Eka Dewi Lukmana Sari, M.Pd selaku Guru Bahasa Indonesia SMA Negeri 6 Balikpapan yang telah memberikan bantuan dan masukan dalam proses pengerjaan skripsi.
10. Teman – teman seperjuangan yang telah banyak membantu dalam drama skripsi baik secara langsung maupun tidak langsung yaitu Rosy Indah, Amalia Kusuma, Dyah Ayu, Adefani Tia a.k.a Nonce Magdalena, Syafitri Hidayatul, Fakhruddin, Azmi Makarima, Ema Agasta dan sahabat Informatika yang lainnya.
11. Semua pihak yang telah membantu dalam proses penyusunan skripsi ini.

Penulis menyadari bahwa dalam skripsi ini masih terdapat banyak kekurangan, sehingga dengan terbuka penulis menerima seluruh kritik dan saran yang membangun. Akhir kata, penulis ucapkan terima kasih kepada para pembaca yan telah membaca skripsi ini.

Malang, 30 Juli 2018

Penulis
fsaputrar@student.ub.ac.id

ABSTRAK

Fachrul Rozy Saputra Rangkuti, Analisis Sentimen Opini Film Menggunakan Metode *Naïve Bayes* dengan *Ensemble Feature* dan Seleksi Fitur *Pearson Correlation Coefficient*.

Pembimbing: M. Ali Fauzi, S.Kom., M.Kom dan Yuita Arum Sari, S.Kom., M.Kom

Microblogging telah menjadi media informasi yang sangat populer di kalangan pengguna internet. Oleh karena itu, *microblogging* menjadi sumber data yang kaya untuk opini dan ulasan terutama pada ulasan film. Pada penelitian ini mengusulkan analisis sentimen opini film dengan menggunakan *ensemble feature* dan *Bag of Words* dan seleksi Fitur *Pearson Correlation Coefficient* untuk mengurangi dimensi fitur dan mendapatkan kombinasi fitur yang optimal. Penggunaan seleksi fitur dilakukan untuk meningkatkan performa dari klasifikasi, mengurangi dimensi fitur dan mendapatkan kombinasi fitur yang optimal. Pada penelitian ini menggunakan beberapa model dari klasifikasi *Naïve Bayes* yaitu *Bernoulli Naïve Bayes* untuk tipe data biner, *Gaussian Naïve Bayes* untuk tipe data kontinu dan *Multinomial Naïve Bayes* untuk tipe data *numeric*. Hasil penelitian ini menunjukkan bahwa dengan menggunakan kata tidak baku pada *tweet* hasil evaluasi yang didapatkan *accuracy* 82%, *precision* 86%, *recall 79,62%* dan *f-measure* 82,69% pada seleksi Fitur 20%. Kemudian setelah dilakukan pembakuan kata secara manual hasil evaluasi pada *accuracy* meningkat sebesar 8% sehingga *accuracy* menjadi 90%, *precision* 92%, *recall* 88,46% dan *f-measure* 90,19% pada seleksi fitur 85%. Berdasarkan hasil tersebut dapat disimpulkan bahwa penggunaan kata baku dapat meningkatkan performa klasifikasi dan seleksi fitur *Pearson's* memberikan kombinasi fitur yang optimal dan mengurangi jumlah total dimensi fitur.

Kata kunci: *analisis sentimen, seleksi fitur, twitter, pearson correlation coefficient, naïve bayes*

ABSTRACT

Fachrul Rozy Saputra Rangkuti, ***Sentiment Analysis on Movie Review using Naïve Bayes with Ensemble Features and Feature Selection Pearson Correlation Coefficient*****.**

Supervisors**: M. Ali Fauzi, S.Kom., M.Kom dan Yuita Arum Sari, S.Kom., M.Kom**

Microblogging has become the media information that is very popular among internet users. Therefore, the microblogging became a source of rich data for opinions and reviews especially on movie reviews. We proposed, sentiment analysis on movie review using ensemble features and Bag of Words and selection Features Pearson's Correlation to reduce the dimension of the feature and get the optimal feature combinations. Use the feature selection is done to improve the performance of the classification, reducing the dimension of the feature and get the optimal feature combinations. The process of classification using several models of Naïve Bayes i.e. Bernoulli Naïve Bayes for binary data , Gaussian Naïve Bayes for continuous data and Multinomial Naïve Bayes for numeric data. The results of this study indicate that by using the non-standard word on tweet evaluation results obtained accuracy 82%, precision 86%, recall 79.62% and f-measure 82.69% using Feature Selection 20%. Then after using manual standardization of word the evaluation results on the accuracy increased by 8% and then the accuracy becomes 90%, precision 92%, recall 88.46% and f-measure 90.19% using 85% feature selection. Based on these results it can be concluded that by using the standardization of word can improve the performance of classification and feature selection Pearson's provide optimal feature combinations and reducing the total number of dimensions feature.

Keywords: *sentiment analysis, feature selection, twitter, pearson correlation coefficient, naïve bayes*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Meningkatnya pengguna Internet di tanah air membuat pengguna media sosial di Indonesia terus bertambah. Hal tersebut dibuktikan dari hasil data Statista[1], salah satu perusahaan statistik *online* yang menyediakan data dari lembaga riset penelitian, opini dan lain-lain menyatakan bahwa pada tahun 2017 pengguna media sosial mencapai 96 juta, dan diperkirakan akan terus meningkat menjadi 25,3 juta atau lebih dari 30% pengguna pada tahun 2022[2]. Meningkatnya pengguna media sosial pada dasarnya dapat memengaruhi perilaku individu tersebut dalam berinteraksi dan berkomunikasi di masyarakat. Statistik Telekomunikasi Indonesia 2015 yang dirilis oleh Badan Pusat Statistik (BPS) menyatakan bahwa sebanyak 73,5% masyarakat memakai layanan internet untuk mendapatkan berita dan informasi, 45,1% untuk memperoleh hiburan dan 35,08% untuk mengerjakan tugas. Selebihnya, digunakan untuk keperluan mengirim email, jual beli, hingga finansial[3]. Berita dan Informasi saat ini banyak tersedia pada *microblogging*. *Microblogging* telah menjadi media informasi yang sangat populer di kalangan pengguna internet (Pak & Paroubek, 2010). Oleh karena itu, *microblogging* menjadi sumber data yang kaya untuk opini dan ulasan terutama pada ulasan film. Salah satu *microblogging* yang paling banyak digunakan untuk menyampaikan opini film di Indonesia adalah Twitter (Sitorus, et al., 2017).

Twitter merupakan layanan *microblogging* yang populer yang mana pengguna memposting pesan status, yang disebut *tweet*, dengan tidak lebih dari 140 karakter (Silva, Hruschka, & Jr., 2014) akan tetapi pada tanggal 7 November 2017 mengalami pembaharuan pada kicauan *tweet* menjadi 280 karakter untuk semua Bahasa (The Open University, 2018). Twitter memungkinkan penggunanya untuk menulis berbagai topik dan berdiskusi masalah yang terjadi seperti opini film. Opini mengenai film yang diposting melalui media sosial Twitter dapat digunakan sebagai sarana untuk memberikan penilaian terhadap film yang diproduksi untuk masyarakat. Hal tersebut dapat dimanfaatkan oleh produser Film untuk mengetahui kualitas yang diproduksi layak atau tidak dimasyarakat, begitu juga dengan masyarakat dapat meninjau terlebih dahulu film yang akan ditonton. Oleh karena itu, analisis sentimen memiliki peran penting. Analisis sentimen atau *Opinion Mining* dapat digunakan untuk memperoleh gambaran umum persepsi masyarakat terhadap kualitas film, apakah film tersebut layak untuk ditonton (positif) atau tidak layak untuk ditonton (negatif).

[1] https://www.statista.com/

[2] https://databoks.katadata.co.id/datapublish/2017/08/22/2022-pengguna-media-sosial-indonesia-mencapai-125-juta

[3] https://databoks.katadata.co.id/datapublish/2016/12/14/media-sosial-alasan-utama-penduduk-indonesia-akses-internet

Analisis sentimen pada *microblogging*, seperti Twitter, dilakukan untuk pencarian informasi, dengan tujuan utamanya adalah untuk mengklasifikasikan sentimen *tweet* positif, negatif atau netral berdasarkan isinya. Masalah spesifik yang harus ditangani dalam jenis analisis sentimen menggunakan *tweet* adalah pengguna Twitter memposting pesan pada berbagai topik, tidak seperti blog, berita, dan situs lainnya, yang disesuaikan dengan topik tertentu (Silva, Hruschka, & Jr., 2014). Pengklasifikasian analisis sentimen dapat dilakukan pada kata, frasa atau kalimat, dokumen, dan fitur (Siddiqua, Ahsan, & Chy, 2016). Analisis sentimen dengan fitur, fitur yang sering digunakan untuk analisis sentimen adalah fitur *Bag of Words* (BoW) (Fauzi & Yuniarti, 2018). Penggunaan fitur *Bag-of-Words* saja belum dapat bekerja secara maksimal untuk melakukan analisis sentimen karena sebuah *short text* termasuk *tweet* sangat kaya dengan fitur dan terbukti bahwa penggabungan beberapa fitur dapat meningkatkan akurasi, pada peneltian yang yang membahas tentang analisis sentimen menggunakan *ensemble feature*, penggunaan *Bag-of-Words* saja untuk beberapa *tweet* yang pendek sangat bergantung pada statistik kata yang terdapat pada data latih sehingga dibutuhkan *ensemble feature* untuk meningkatkan akurasi dan dapat melengkapi kelemahan masing-masing fitur (Irfan, Fauzi, & Tibyani, 2018). Untuk meningkatkan akurasi dari analisis sentimen maka pada penelitian ini terdapat beberapa fitur yang dilakukan untuk mengekstrak *tweet* yaitu *Twitter spesific*, *Textual Featurres*, *Part of Speech* (PoS) *Features*, *Lexicon Based Features*, dan *Bags of Words* (BoW) dengan metode klasifikasi yang digunakan adalah *Naïve Bayes*.

Pada beberapa penelitian sebelumnya analisis sentimen dengan menggunakan *Naïve Bayes* telah banyak digunakan salah satunya, Penelitian yang dilakukan oleh Alexander Pak dan Patrick Paroubek pada tahun 2010. Penelitian tersebut menunjukkan bahwa klasifikasi *Naïve Bayes* lebih baik daripada klasifikasi menggunakan *Support Vector Machine* dan *Conditional Random Field*, dengan salah satu fitur yang digunakan adalah *POS tags* (Pak & Paroubek, 2010). Terdapat beberapa macam klasifikasi *Naïve Bayes* yaitu *Bernoulli Naïve Bayes* (McCallum & Nigam, 1998), *Gaussian Naïve Bayes* (Capdevila & Flórez, 2009) dan Multinomial *Naïve Bayes* (McCallum & Nigam, 1998). Penelitian yang dilakukan oleh Ang Yang, Jun Zhang, Lei Pan dan Yang Xiang menunjukkan bahwa klasifikasi dengan menggunakan *Multinomial Naïve Bayes* dan *Naïve bayes* proses komputasi diselesaikan dalam hitungan menit dibandingkan dengan metode *SVM*, *k-NN*, *Decision Tree C4.5* dan *Logistic Regression Model* (Yang, et al., 2016). *Naïve Bayes* memiliki kekurangan pada jumlah fitur yang terlalu banyak, sehingga membuat performa klasifikasi menjadi kurang optimal. Oleh karena itu, dalam penelitian ini menggunakan seleksi fitur untuk mencari kombinasi fitur yang optimal dan mengurangi dimensi fitur (Utami & Wahono, 2015).

Seleksi fitur adalah teknik untuk memilih fitur penting dan relevan terhadap data dan mengurangi fitur yang tidak relevan (Onan & Korukoğlu, 2015). Seleksi fitur merupakan bagian penting untuk mengoptimalkan kinerja dari klasifikasi

suatu metode (Shardlow, 2007). Tujuan utama dari seleksi fitur adalah mengurangi kompleksitas, meningkatkan akurasi dan untuk memilih fitur terbaik dari suatu kumpulan fitur data (Sharma & Dey, 2012). Metode seleksi fitur yang digunakan dalam penelitian ini adalah *Pearson Correlation Coefficient*. Penelitian yang dilakukan oleh Aytug Onan dan Serdar Korukoglu membahas tentang perbandingan dari beberapa metode seleksi fitur, metode seleksi fitur dengan menggunakan *Pearson Correlation Coefficient* masih lebih optimal dibandingkan dengan seleksi fitur menggunakan *Information Gain* dan *Chi Square* (Onan & Korukoğlu, 2015). Penelitian yang dilakukan oleh Matthew Shardlow pada tahun 2007. Seleksi fitur dengan menggunakan metode *filter*, *Pearson Correlation Coeffiecient* dapat meningkatkan akurasi dan memiliki kelebihan yaitu efektif dan cepat untuk jumlah fitur yang sangat banyak (Shardlow, 2007). Sedangkan *Ensemble Feature* merupakan fitur gabungan dengan memperhatikan berbagai aturan-aturan tertentu (Siddiqua, Ahsan, & Chy, 2016). Tujuan dari *ensemble feature* diterapkan pada analisis sentimen untuk mengintegrasikan fitur secara efisien pada berbagai jenis fitur dan pada algoritme klasifikasi untuk membuat proses klasifikasi yang lebih akurat (Xia, Zong, & Li, 2011). Penelitian yang dilakukan oleh Umme Aymun Siddiqua, Tanveer Ahsan, dan Abu Nowshed Chy pada tahun 2016 mmengungkapkan bahwa penelitian dengan menggabungkan *ensemble feature* dan *Bag of Words* dapat membantu meningkatkan akurasi pada *tweet* yang memiliki kalimat yang pendek dan menghasilkan akurasi sebesar 87,7% dengan fitur yang paling optimal adalah *lexicon-based features* (Siddiqua, Ahsan, & Chy, 2016).

Untuk meningkatkan kinerja dari klasifikasi menggunakan pendekatan *ensemble* dengan menerapkan metode seleksi fitur, penelitian ini dapat mengidentifikasi dan menghilangkan fitur yang tidak penting dan tidak relevan. Berdasarkan uraian diatas, maka penulis mengusulkan penelitian dengan judul "Analisis Sentimen Opini Film mmenggunakan Metode *Naïve Bayes* dengan *Ensemble Feature* dan Seleksi Fitur *Pearson Correlation Coefficient*".

1.2 Rumusan masalah

Rumusan masalah yang dapat dirangkum berdasarkan uraian permasalahan di atas adalah sebaagai berikut:

1. Bagaimana pengaruh seleksi fitur *Pearson Correlation Coefficient* pada analisis sentimen opini film menggunakan *Naïve Bayes* dengan *ensemble feature*?
2. Bagaimana pengaruh kata tidak baku dan kata baku terhadap analisis sentimen opini film menggunakan *Naïve Bayes* dengan *ensemble feature* dan seleksi fitur *Pearson Correlation Coefficient*?

1.3 Tujuan

Tujuan yang ingin dicapai dari penelitian ini adalah:

1. Mengetahui pengaruh seleksi fitur *Pearson Correlation Coefficient* pada analisis sentimen opini film menggunakan *Naïve Bayes* dengan *ensemble feature*.
2. Mengetahui pengaruh kata tidak baku dan kata baku terhadap analisis sentimen opini film menggunakan *Naïve Bayes* dengan *ensemble feature* dan seleksi fitur *Pearson Correlation Coefficient*.

1.4 Manfaat

Penelitian yang dilakukan diharapkan dapat memberikan manfaat bagi para pembaca, antara lain yaitu:

a. Bagi Produsen
    1. Sebagai wadah untuk mengetahui tanggapan masyarakat mengenai film yang telah diproduksi dan menjadi bahan untuk mengetahui opini masyarakat terhadap film tersebut.

b. Bagi Konsumen
    1. Sebagai bahan peninjauan kembali pada masyarakat untuk menonton film yang akan ditonton dengan melihat hasil dari sentimen.

c. Bagi Penulis
    1. Penelitian ini dapat menjadi pembelajaran dan mampu mengimplementasikan ilmu yang sudah dipelajari.
    2. Mendapatkan pemahaman mengenai penerapan metode *Ensemble Feature* dan *Naïve Bayes* dan Seleksi Fitur *Pearson Correlation Coefficient* untuk analisis sentimen opini film pada *microblogging* Twitter.

1.5 Batasan masalah

Untuk menghindari permasalahan baru, diberikan batasan masalah sebagai berikut:

1. Opini Film menggunakan dokumen teks Twitter berbahasa Indonesia.
2. Sentimen yang digunakan adalah sentimen positif dan sentimen negatif.
3. *Tweet* diambil dari situs Twitter menggunakan *Twitter REST API*.
4. Pembakuan kata pada dokumen *tweet* dilakukan secara manual.

5. *Dataset* yang digunakan sebanyak 500 data yang terdiri dari 400 data latih dan 100 data uji.

1.6 Sistematika pembahasan

BAB 1 PENDAHULUAN

Bab pendahuluan berisi latar belakang dari permasalahan yang diangkat, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Bab landasan kepustakaan berisi penjelasan tentang konsep dasar yang digunakan pada penelitian-penelitian sebelumnya, selain itu juga dijelaskan teori – teori yang mencakup dengan penelitian ini seperti metode *Naïve Bayes* dan metode seleksi fitur *Pearson Correlation Coefficient*.

BAB 3 METODOLOGI PENELITIAN

Bab metodologi menjelaskan tentang metodologi yang akan digunakan dalam penelitian seperti tipe penelitian, strategi penelitian, partisipan penelitian, lokasi penelitian, teknik pengumpulan data, implementasi algoritme, teknik analisis data dan jadwal penelitian.

BAB 4 PERANCANGAN

Bab ini membahas mengenai perancangan algoritme, perancangan antarmuka dan perancangan pengujian yang diperlukan dalam pembuatan sistem.

BAB 5 HASIL

Bab Hasil berisi mengenai hasil dari penelitian ini seperti perancangan algoritme yang akan dibangun, manualisasi, perancangan *database* yang diperlukan dalam pembuatan sistem dan proses implementasi dan kode program sesuai metode yang digunakan yaitu seleksi fitur *Pearson Correlation Coefficient* dan *Naïve Bayes*.

BAB 6 PEMBAHASAN

Bab Pembahasan berisi mengenai pengujian algoritme serta analisis hasil pengujian.

BAB 7 PENUTUP

Bab ini menjelaskan kesimpulan dari perancangan, implementasi dan pengujian sistem, serta saran untuk pengembang dan penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Kajian pustaka yang akan dibahas pada penelitian ini meliputi penelitian yang sudah ada sebelumnya dengan objek yang sama maupun metode yang sama dengan penelitian ini. Penelitian yang pertama dilakukan oleh Aytug Onan dan Serdar Korukoglu pada tahun 2015. Penelitian ini membahas tentang banyaknya fitur yang tidak relevan pada metode klasifikasi sehingga menurunkan kinerja dari metode klasifikasi dan akurasi, dengan menggunakan fitur *Bag of Word* dan *unigrams*. Hasil akurasi yang paling optimal dengan menggunakan metode *Naïve Bayes* dengan seleksi fitur *Pearson correlation coefficient* dengan pemilihan fitur 40% dengan akurasi 91,44% (Onan & Korukoğlu, 2015).

Penelitian yang kedua dilakukan oleh Matthew Shardlow pada tahun 2007. Penelitian ini membahas tentang seleksi fitur pada analisis data untuk mengurangi fitur yang tidak relevan. Jumlah fitur yang digunakan pada penelitian ini sebanyak 325 fitur. Penelitian tersebut menunjukkan bahwa metode seleksi fitur *Pearson coefficient* lebih optimal dan dapat meningkatkan akurasi dari metode klasifikasi dan mengurangi jumlah fitur. Seleksi fitur dengan menggunakan *Pearson's* masih lebih optimal dibandingkan dengan metode seleksi fitur yang digunakan pada penelitian ini. Hasil yang didapatkan dengan menggunakan metode seleksi fitur *Pearson Coefficient* yaitu sekitar 85% - 90% (Shardlow, 2007).

Penelitian yang ketiga dilakukan oleh Ang Yang, Jun Zhang, Lei Pan dan Yang Xiang pada tahun 2015. Penelitian tersebut menunjukkan bahwa analisis sentimen dengan menggunakan sentimen *lexicon* dan beberapa metode pembelajaran dengan hasil paling optimal ketika menggunakan *Multinomial Naïve Bayes*. Hasil yang didapatkan adalah 84,60% dengan hasil kompleksitas sebesar 30 detik, *Naïve Bayes* 136 detik, *K-Nearest Neighbour* (*K-NN*) sekitar 3 jam, *Decission Tree* (*C4.5*) 4 jam, *Logistic Regression Model* (*LRM*) 7 jam dan *Support Vector Machine* (*SVM*) 11 jam. Oleh karena itu pada penelitian tersebut menggunakan *Naïve Bayes* sebagai klasifikasi Twitter untuk analisisi sentimen (Yang, et al., 2016).

Penelitian yang keempat tentang kombinasi *rule-based classifier* dengan *ensemble feature* pada *microblog* yang dilakukan pada tahun 2016. Pada penelitan ini terdapat 5 tipe fitur yaitu, *Twitter Specific Feature*, *Textual Feature*, *Part-of-Speech Feature*, *Lexicon based Feature*, dan *Bag of Word*. Penggunaan dari fitur *ensemble* secara lengkap dapat meningkatkan akurasi dan mendapati akurasi tertinggi jika dibandingkan dengan penggunaan fitur secara terpisah. Berdasarkan penelitian yang dilakukan diperoleh akurasi sebesar 87,7% pada percobaan ke-9. (Siddiqua, Ahsan, & Chy, 2016).

Tabel 2.1 Kajian Pustaka

No	Judul	Obyek	Metode	Hasil
1	*A feature selection model based on genetic rank aggregation for text sentiment classification (Onan & Korukoğlu, 2015)*	Kumpulan *dataset* ulasan berdasarkan domain, Kamera, Laptop dan Musik diambil dari situs Amazon.com, untuk ulasan Kamp diambil dari CampRatingz.com, *dataset* Dokter diambil dari RateMDs.com, *dataset* Pengacara diambil dari LawyerRatingz.com dan untuk *dataset* Radio diambil dari RadioRatingz.com dan *dataset* TV diperoleh dari TVRatingz.com.	Klasifikasi *Naïve Bayes, K-Nearest Neighbour* Seleksi Fitur *Information Gain, Chi-square, Gain Ratio, Symmetrical uncertainty coefficient, Pearson correlation coefficient, Relief F algorithm* dan *Probabilistic significance measure.* Fitur yang digunakan *Bag of Words* dan *unigram*	Hasil akurasi yang paling optimal dengan menggunakan metode *Naïve Bayes* dengan seleksi fitur *Pearson correlation coefficient* pada pemilihan fitur 40% dan didapatkan akurasi sebesar akurasi 91,44%
2	*An Analysis of Feature Selection Techniques* (Shardlow, 2007)	*Dataset* yang digunakan adalah kanker paru – paru dengan fitur sebanyak 325 fitur dan fitur tersebut bersifat diskrit	*Pearson's correlation coefficient, Mutual Information, Relief, Ensemble with data permutation* dan *Ensemble of methods*	Hasil akurasi yang diperoleh ketika menggunakan 100 fitur pada metode *Pearson's* berkisar antara 85% hingga 90%

Tabel 2.1 Kajian Pustaka (lanjutan)

No	Judul	Obyek	Metode	Hasil
3	*Enhanced Twitter Sentiment Analysis by Using Feature Selection and Combination* (Yang, et al., 2016)	50.000 *tweets* diambil menggunakan *Twitter API* dan membuat data seimbang 5000 *tweet* positif dan 5000 *tweet* negatif.	*Multinomial Naïve Bayes, Naïve Bayes, k-Nearest Neighbour, Decision Tree C4.5, Support Vector Machine,* dan *Logistic Regression Model*.	*Multinomial Naïve Bayes* menghasilkan *accuracy* sebesar 84,6% dan proses waktu komputasi 30 detik hasil ini lebih baik daripada menggunakan hasil akurasi *SVM* 82,3% dan waktu komputasi 11 jam.
4	*Combining a Rule-based Classifier with Ensemble of Feature Sets and Machine Learning Techniques for Sentiment Analysis on Microblog* (Siddiqua, Ahsan, & Chy, 2016).	Dataset yang digunakan berasal dari *Stanford* Twitter sentimen *140* sebanyak 1,6 juta tweet untuk data *training*, dan untuk data *testing* 182 positif dan 177 negatif.	*Naïve Bayes* (Probabilitas dan Multinomial), *Multiclass SVM*, dan *Multiclass SMO, Ensemble Feature* (*twitter* specific *features*, *textual features, Parts of Speech (PoS) features*, *lexicon-based features*) dan *Bag of Words* (BoW)	Hasil yang diperoleh pada penelitian diperoleh akurasi sebesar 87,7% pada percobaan ke-9.
5	Analisis Sentimen Kurikulum 2013 pada Twitter menggunakan *Ensemble Feature* dan	*Dataset* yang digunakan berupa Twitter sebanyak 200 *tweet* dengan kata kunci 'kurikulum2013'	*K-Nearest Neighbour, Ensemble Feature* (*twitter* specific *features*, *textual features, Parts of Speech (PoS) features*, *lexicon*	Penggabungan kedua fitur dapat meningkatkan akurasi mencapai 96% berbeda jika fitur

Tabel 2.1 Kajian Pustaka (lanjutan)

No	Judul	Obyek	Metode	Hasil
5	Metode *K-Nearest Neighbor* (Irfan, Fauzi, & Tibyani, 2018)		*based features*) dan *Bag of Words* (BoW)	dilakukan secara independent hasil yang didapat 80% pada fitur *Bag of Words* dan 82% fitur *ensemble tanpa Bag of Words*.
6	*Twitter as a Corpus for Sentiment Analysis and Opinion Mining* (Pak & Paroubek, 2010)	300.000 *tweets* dengan memisahkan beberapa *tweet* yaitu: 1. *Tweet* yang mengandung perasaan poisitif (*happiness*, *joy*) 2. *Tweet* yang mengandung perasaan negative (*sad*, *anger*) 3. *Tweet* fakta tidak mengekspresikan emosi apapun (netral)	*Naïve Bayes*, *Support Vector Machine, Conditional Problem Field*.	*Naïve Bayes* bekerja lebih baik daripada metode lainnya terutama pada *Multinomial Naïve Bayes*.

2.2 Opini Film

Menurut Kamus Besar Bahasa Indonesia Opini merupakan pendapat, ide atau pikiran untuk menjelaskan sebuah pernyataan yang tidak bersifat objektif. Dalam banyak kasus keputusan yang kita buat dipengaruhi oleh opini dari orang lain. Sedangkan Film merupakan media campuran dari media audio dan visual, untuk menyampaikan suatu pesan kepada sekelompok orang yang berkupul disuatu tempat tertentu. Opini Film memiliki karakteristik yang unik. Ketika seseorang menulis opini mengenai *film*, ia mungkin berkomentar beberapa elemen dari sebuah film (*screen-play, vision effect, music*) dan ada juga dengan orang-orang yang terlibat di dalam film (sutradara, penulis skenario, aktor). (Andilala, 2016).

2.3 Twitter

Twitter merupakan layanan *microblogging* yang populer dimana pengguna memposting pesan status, yang disebut *tweet*, dengan tidak lebih dari 140 karakter (Silva, Hruschka, & Jr., 2014). Twitter menjadi salah satu tempat bagi pengguna untuk saling berbagi pengalaman antar sesama penggunanya tanpa adanya sekat penghalang. Dengan menggunakannya, pengguna akan mudah untuk mengikuti tren, cerita, informasi dan berita dari seluruh penjuru dunia. Ada beberapa alasan menggunakan analisis sentimen menggunkan *corpus* Twitter diantaranya yaitu (Pak & Paroubek, 2010):

1. Twitter berisi kicauan *tweet* yang berupa teks dan akan selalu bertambah setiap harinya dan *corpus* yang terkumpul bisa dalam jumlah besar.
2. Pengguna Twitter sangat bervariasi mulai dari selebritas, perwakilan perusahaan, politisi dan bahkan Presiden negara. Oleh karena itu, dimungkinkan untuk mengumpulkan postingan teks pengguna dari berbagai kalangan sosial dan beberapa kelompok tertentu.

2.4 Analisis Sentimen

Analisis Sentimen atau biasa yang disebut dengan *Opinion Mining* adalah proses mengekstraksi informasi subjektif, seperti pendapat, sentimen dan sikap dalam materi sumber terhadap entitas. Opini atau pendapat adalah factor penting dari proses pengambilan keputusan. Penentuan pendapat seseorang terhadap acara tertentu dapat menjadi sangat penting dalam berbagai bidang, seperti ilmu manajemen, ilmu politik, ekonomi, dan disiplin ilmu sosial lainnya. Tujuannya adalah untuk mengetahui pendapat seseorang mengenai topik atau target tertentu yang sedang dibahas. Orang selalu lebih suka mendengar sentimen orang lain sebelum mengambil keputusan, karena sangat sulit untuk mengetahui sentimen tersebut termasuk sentimen positif atau sentimen negatif. Oleh karena itu, teknik Analisis Sentimen digunakan untuk mengklasifikasikan sentimen dari

data teks di kelasnya yang sesuai baik positif maupun negatif (Onan & Korukoğlu, 2015).

2.5 *Text Mining*

Text mining merupakan proses pengetahuan intensif yang mencoba memecahkan informasi yang berlebihan dengan cara yang serupa dengan data mining untuk mencari pola dalam teks, yaitu proses menganalisis teks guna mencari informasi yang bermanfaat untuk tujuan tertentu. Tujuan dari text mining adalah untuk mendapatkan informasi dari sekumpulan dokumen yang berguna yang dapat mewakili kata- kata pada isi dokumen (Feldman & Sanger, 2007).

Teks yang belum diolah biasanya memiliki karakteristik dimensi yang tinggi, terdapat *noise* pada data dan terdapat struktur teks yang tidak baik. Untuk itu, *text mining* harus melalui beberapa tahapan yang disebut dengan *preprocessing*. *Preprocessing* adalah proses pengontrolan ukuran daftar kata – kata pada jumlah kata yang berbeda yang digunakan sebagai indeks *term* dengan kata lain *preprocessing* adalah merubah teks menjadi *term* indeks dengan tujuan menghasilkan sebuah term indeks yang bisa mewakili dokumen (Maulida, Suyatno, & Hatta, 2016). Tahapan *preprocessing* pada penelitian ini adalah *Tokenization, Stopword Removal,* dan *Stemming*.

2.5.1 *Tokenization*

Pada tahap Tokenisasi dilakukan pemecahan suatu teks menjadi kata, frasa, simbol atau elemen lainnya yang memiliki makna atau yang disebut token. Dalam proses tokenization ini, semua kata yang ada di dalam tiap dokumen dikumpulkan dan dihilangkan tanda bacanya, serta dihilangkan jika terdapat simbol atau apapun yang bukan huruf (Utami & Wahono, 2015). Pada tahap ini juga dilakukan proses *case folding,* yaitu semua huruf akan diubah menjadi huruf kecil. Berikut contoh tokenisasi pada teks dokumen Bahasa Indonesia ditunjukkan pada Tabel 2.2.

Tabel 2.2 Contoh *Tokenization*

Tokenization	
Input	***Output***
Robert Downey Jr sendiri telah memerankan karakter Tony Stark selama bertahun-tahun dengan total tujuh film. Di usia 52 tahun tidak mungkin RDJ memerankan karakter itu selamanya.	“robert”; “downey”; “jr”; “sendiri”; “telah”; “memerankan”; “karakter”; “tony”; “stark”; “selama”; “bertahun-tahun”; “dengan”; “total”; “tujuh”; “film”; “di”; “usia”; “tahun”; “tidak”; “mungkin”; “rdj”; “memerankan”; “karakter”; “itu”; “selamanya”

2.5.2 *Stopword Removal*

Pada tahap *Stopword Removal* dilakukan pemilihan kata-kata penting dari hasil token, yaitu kata – kata yang akan digunakan untuk mewakili dokumen dan kata-kata yang tidak relevan atau tidak penting akan dihapus, seperti kata "di", "yang", kepada, aku yang merupakan kata-kata yang tidak mempunyai makna tersendiri jika dipisahkan dengan kata yang lain dan tidak terkait dengan dengan kata sifat yang berhubungan dengan sentimen. (Utami & Wahono, 2015). Berikut contoh *Stopword Removal* pada teks dokumen Bahasa Indonesia ditunjukkan pada Tabel 2.3.

Tabel 2.3 Contoh *Stopword Removal*

Stopword Removal	
Input	***Output***
"robert"; "downey"; "jr"; "sendiri"; "telah"; "memerankan"; "karakter"; "tony"; "stark"; "selama"; "bertahun-tahun"; "dengan"; "total"; "tujuh"; "film"; "di"; "usia"; "tahun"; "tidak"; "mungkin"; "rdj"; "memerankan"; "karakter"; "itu"; "selamanya"	"memerankan"; "karakter"; "bertahun-tahun"; "total"; "film"; "usia"; "tahun"; "memerankan"; "karakter";

2.5.3 *Stemming*

Proses *Stemming* adalah proses pengubahan bentuk kata menjadi kata dasar atau proses menghilangkan imbuhan pada suatu kata sehingga hanya tersisa kata dasarnya saja, kata berimbuhan cenderung memiliki arti yang sama dengan kata dasarnya. (Utami & Wahono, 2015). Imbuhan (afiks) memiliki banyak jenis yaitu Prefiks, Suffiks, Konfliks, Infiks, dan Imbuhan serapan. Berikut contoh *Stemming* pada teks dokumen Bahasa Indonesia dan Imbuhan (afiks) ditunjukkan pada Tabel 2.4.

Tabel 2.4 Contoh *Stemming*

Stemming	
Input	***Output***
"memerankan"; "karakter"; "bertahun-tahun"; "total"; "tujuh"; "film"; "usia"; "tahun"; "memerankan"; "karakter";	"peran"; "karakter"; "tahun"; "total"; "film"; "usia"

2.6 *Vector Space Model*

Vector Space Model adalah model proses pencarian informasi dari *query* yang menggunakan ekspresi kemiripan berdasarkan frekuensi *term*/token/kata yang terdapat pada dokumen. Ekspresi *vector space* berupa faktor kemiripan dari kumpulan nilai frekuensi kemunculan *term*/token/kata dalam korpus yang membentuk matrik vektor (Manning, Raghavan, & Schütze, 2008). Terdapat dua tahap dalam *vector space model* yang pertama yaitu, Pengideksan Dokumen (*Document Indexing*) setelah melakukan *preprocessing*, proses ini bertujuan untuk mendapatkan sebuah set *term* yang bisa dijadikan sebagai indeks. Tahap kedua yaitu *term weighting*. Pada tahap *term weighting* dilakukan pemberian bobot/nilai pada masing – masing *term* indeks. Pembobotan dilakukan dengan menghitung frekuensi kemunculan *term* dalam dokumen. Frekuensi kemunculan (*term frequency*) adalah jumlah kemunculan term pada satu dokumen. Semakin besar kemunculan suatu *term* dalam dokumen akan memberikan nilai kesesuaian yang semakin besar.

2.6.1 *Term Frequency*

Term Frequency adalah banyaknya kemunculan *term*/token/kata *t* dalam dokumen yang bersangkutan. Semakin besar jumlah kemunculan suatu *term* (TF tinggi) dalam dokumen, semakin besar pula bobotnya atau akan memberikan nilai kesesuaian yang semakin besar (Raschka, 2014). Pada *term frequency* (TF), terdapat beberapa jenis *term frequency* yang digunakan pada penelitian ini yaitu, *Raw Term Frequency.*

2.6.1.1 *Raw Term Frequency*

Raw Term Frequency, nilai *Term Frequency (TF)* diberikan berdasarkan jumlah kemunculan suatu term dalam dokumen. Contohnya, jika nilai kemunculan *term* pada satu dokumen muncul sebanyak 8 (delapan) kali maka *term* tersebut akan bernilai 8.

2.7 Seleksi Fitur

Seleksi fitur adalah teknik untuk memilih fitur penting dan relevan terhadap data dan mengurangi fitur yang tidak relevan. Seleksi fitur merupakan bagian penting untuk mengoptimalkan kinerja dari klasifikasi suatu metode. Tujuan dari seleksi fitur ini adalah untuk memilih fitur terbaik dari suatu kumpulan fitur data. Penggunaan algoritme Seleksi fitur yang tepat dapat meningkatkan akurasi dari metode klasifikasi. Seleksi fitur sangat penting untuk analisis sentimen pada teks yang mengandung pendapat yang sangat kuat karena memiliki dimensi yang tinggi, yang dapat mempengaruhi kinerja dari klasifikasi analisis sentimen, dengan menerapkan seleksi fitur yang efektif dan efisien dapat meningkatkan kinerja analisis sentimen dalam hal akurasi dan waktu untuk melatih klasifikasi (Sharma & Dey, 2012).

Algoritme seleksi fitur dapat dibedakan menjadi dua tipe, yaitu *filter* dan *wrapper*. *Filter* menggunakan karakteristik umum bekerja secara terpisah dari algoritme pembelajaran. Secara komputasi teknik *filter* lebih cepat dan terukur. Namun pendekatan *filter* biasanya tidak mempertimbangkan dependensi fitur. Di sisi lain, *wrapper* mengevaluasi fitur dan memilih atribut berdasatkan perkiraan akurasi menggunakan model pembelajaran tertentu dan algoritme pencarian. Dengan algoritme pembelajaran tertentu, pada dasarnya *wrapper* melakukan pencarian ruang fitur dengan menghilangkan beberapa fitur (Onan & Korukoğlu, 2015).

2.7.1 *Pearson Correlation Coefficient*

Pearson Correlation Coefficient atau *Pearson's r* atau bisa juga disebut sebagai *Pearson product-moment correlation coefficient* (PPMCC) merupakan ukuran korelasi yang digunakan untuk mengukur kekuatan dan arah hubungan linier dua variabel. Metode *Pearson Correlation Coefficient* merupakan salah satu metode yang paling sederhana untuk memahami hubungan fitur dengan variabel, yang mengukur korelasi linier antara dua variabel. Nilai yang dihasilkan terletak pada [-1;1], untuk nilai -1 yang berarti korelasi negatif sempurna (karena satu variabel meningkat, yang lainnya menurun), +1 berarti korelasi positif sempurna dan 0 yang berarti tidak ada korelasi linier antara kedua variabel tersebut (Bae & Lee, 2012). *Pearson Correlation Coefficient* memiliki kelebihan yaitu cepat dan efektif untuk jumlah fitur yang sangat banyak (Shardlow, 2007). Berikut persamaan *Pearson Correlation Coefficient* ditunjukkan pada Persamaan 2.5.

$$r_{xy} = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2]}\sqrt{[n\sum y^2 - (\sum y)^2]}} \quad (2.5)$$

Keterangan:

r_{xy}	=	*Pearson's Product Moment Correlation Coefficient*
N	=	Banyaknya pasangan data *x* dan *y*
Σx	=	Total jumlah dari variabel *x*
Σy	=	Total jumlah dari variabel *y*
Σx^2	=	Kuadrat dari total jumlah variabel *x*
Σy^2	=	Kuadrat dari total jumlah variabel *y*
Σxy	=	Hasil perkalian dari total jumlah variabel *x* dan variabel *y*

2.8 *Ensemble Feature*

Ensemble feature merupakan fitur gabungan dengan memperhatikan berbagai aturan-aturan tertentu. Tujuan dari *ensemble feature* diterapkan pada analisis sentimen untuk mengintegrasikan fitur secara efisien pada berbagai jenis

fitur dan pada algoritme klasifikasi untuk membuat prosedur klasifikasi yang lebih akurat (Siddiqua, Ahsan, & Chy, 2016). Terdapat fitur F1 – F37, keseluruhan fitur dapat dilihat pada Tabel 2.7.

Tabel 2.5 *Ensemble Feature*

Type	***ID***	***Feature Description***
Twitter Specific	F1	*Whether the tweet contains a #hashtag or not.*
	F2	*Whether the tweet is a retweet or not.*
	F3	*Whether the tweet contains a user name or not.*
	F4	*Whether the tweet contains a URL or not.*
Textual Features	F5	*TweetLength: Number of words in the tweet.*
	F6	*AvgWordLength: Average character length of words.*
	F7	*Number of question marks available in the tweet.*
	F8	*Number of exclamation marks available in the tweet.*
	F9	*Number of quotes available in the tweet.*
	F10	*Number of words start with the uppercase letter in tweet.*
	F11	*Whether the tweet contains a positive emoticon or not.*
	F12	*Whether the tweet contains a negative emoticon or not.*
Part of Speech (PoS) Features	F13	*Number of noun PoS available in the tweet.*
	F14	*Number of adjective PoS available in the tweet.*
	F15	*Number of verb PoS available in the tweet.*
	F16	*Number of adverb PoS available in the tweet.*
	F17	*Number of interjection PoS available in the tweet*
	F18	*Percentage of noun PoS in the tweet.*
	F19	*Percentage of adjective PoS in the tweet.*
	F20	*Percentage of verb PoS in the tweet.*
	F21	*Percentage of adverb PoS in the tweet.*
	F22	*Percentage of interjection PoS in the tweet.*
Lexicon Based Features	F23	*Number of positive words available in the tweet.*
	F24	*Number of negative words available in the tweet.*
	F25	*Number of positive words with adjective PoS.*

Tabel 2.5 *Ensemble Feature* (lanjutan)

Type	*ID*	*Feature Description*
Lexicon Based Features	F26	*Number of negative words with adjective PoS.*
	F27	*Number of positive words with verb PoS.*
	F28	*Number of negative words with verb PoS.*
	F29	*Number of positive words with adverb PoS.*
	F30	*Number of negative words with adverb PoS.*
	F31	*Percentage of positive words with adjective PoS.*
	F32	*Percentage of negative words with adjective PoS.*
	F33	*Percentage of positive words with verb PoS.*
	F34	*Percentage of negative words with verb PoS.*
	F35	*Percentage of positive words with adverb PoS.*
	F36	*Percentage of negative words with adverb PoS.*
	F37	*Number of intensifier words available in the tweet.*

Sumber: (Siddiqua, Ahsan, & Chy, 2016)

Untuk sistem analisis sentimen menggunakan fitur dari penelitian Umme Aymun Siddiqua, Tanveer Ahsan, dan Abu Nowshed Chy yang dikelompokkan kedalam beberapa tipe yang berbeda yaitu, *Twitter Specific Feature*, *Textual Feature*, *Part-of-Speech Feature*, *Lexicon based Feature*, dan *Bag of Word*. Karakteristik untuk mengekstrak fitur pada *Twitter Specific* seperti *#hashtag*, *username*, *retweet*¸*dan URL*. Untuk fitur *Textual,* fitur yang diekstrak hanya menggunakan informasi yang terdapat dalam teks seperti jumlah kata, kemunculan tanda kutip atau tanda seru, *emoticon*, dan sebagainya. Fitur *Part of Speech* menggunakan kamus dengan *tag noun*, *adjective*, *verb*, *adverb* dan *interjection*. Fitur *Lexicon-based* menggunakan kamus pada penelitian (Wahid & Azhari, 2016) . Untuk fitur *Bag of Words (BoW)* menghitung kemunculan *term* dengan menggunakan *raw term frequency* (Siddiqua, Ahsan, & Chy, 2016).

2.9 *Naïve Bayes Classifier*

Naïve Bayes merupakan salah satu algoritme klasifikasi statistik sederhana karena kemudahan penggunaannya serta waktu pemrosesan yang cepat bila diterapkan pada dataset yang besar, mudah diimplementasikan dengan strukturnya yang cukup sederhana dan tingkat efektivitas yang tinggi. Algoritme *Naïve Bayes* menggunakan keseluruhan probabilitas, yaitu probabilitas dokumen terhadap kategori (prior). Kemudian teks akan terkategori berdasarkan probabilitas maksimum (posterior). Dengan kata lain metode ini mengasumsikan bahwa ada atau tidaknya fitur tertentu dari kelas tidak berhubungan dengan ada

atau tidaknya fitur yang lain. Dalam berbagai macam penerapannya, estimasi parameter untuk model *Naive Bayes* menggunakan metode *maximum likelihood*; yang artinya pengguna dapat bekerja menggunakan model *Naive Bayes* tanpa perlu mempercayai probabilitas *Bayesian* atau tanpa menggunakan metode *Bayesian* (Raschka, 2014). Berikut persamaan Naïve Bayes ditunjukkan pada Persamaan 2.6.

$$P(c|d_i) = \frac{P(c)\,P(d_i|c)}{P(d_i)} \tag{2.6}$$

Dalam Persamaan 2.7 merupakan penyederhanaan perhitungan posterior pada proses klasifikasi dengan peluang kemunculan kata/*evidence* dihilangkan, karena tidak akan berpengaruh pada perbandingan hasil klasifikasi pada tiap kategori.

$$P(c|d_i) = P(c)\,P(d_i|c) \tag{2.7}$$

Perhitungan peluang kemunculan kata (*likelihood/conditional probability*) dilakukan dengan menggunakan multinomial *Naïve Bayes*. Dalam kasus ini *conditional probability* yang berawal dari i=1 sampai dengan i=k sehingga perhitungan nilai *Posterior* dapat dilakukan dalam Persamaan 2.8.

$$P(c|d_i) = P(c) \times P(d_1|c) \times P(d_2|c) \times P(d_3|c) \times \ldots \times P(d_k|c) \tag{2.8}$$

Keterangan:

$P(c\|d_i)$	=	Probabilitas *posterior* (*c* merupakan kelas dan *d* merupakan dokumen)
$P(c)$	=	Probabilitas *prior* dari kelas
$P(d_i\|c)$	=	Probabilitas *likelihood*
$P(d_i)$	=	Probabilitas *prior* dari dokumen

2.9.1 *Gaussian Naïve Bayes*

Gaussian Naive Bayes adalah salah satu model klasifikasi dari naïve bayes merepresentasikan probabilitas bersyarat dari fitur dengan tipe data kontinu pada sebuah kelas yang setiap kategori mencirikan *Multivariate Gaussian* atau Normal *Probability Density Function* (PDF) (Capdevila & Flórez, 2009). PDF adalah kemungkinan munculnya suatu nilai dalam range tertentu. *Gaussian Naïve Bayes* memiliki 2 parameter *mean* dan *variance*. Berikut persamaan *Gaussian Naïve Bayes* untuk nilai input *x* ditunjukkan pada Persamaan 2.9.

$$P(d_i|c) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{2.9}$$

Keterangan:

$P(d_i \mid c)$	=	setiap kelas c, probabilitas besyarat kelas c *untuk fitur* d_i

μ_y = didapat dari *mean* sampel x_i ($\bar{x}$) dari semua data latih yang menjadi milik kelas *c*

σ_y^2 = *varian* sampel (s^2) dari data latih

2.9.2 *Multinomial Naïve Bayes*

Multinomial Naïve Bayes merupakan salah satu metode spesifik dari metode Naïve Bayes yang menggunakan *conditional probability*. *Conditional probability* dapat dilakukan dengan menggunakan frekuensi kemunculan suatu kata pada suatu kelas (*raw term frequency*). *Multinomial Naïve Bayes* ini juga merupakan salah satu machine learning dalam *supervised learning* pada proses pengklasifikasian teks dengan menggunakan nilai probabilitas suatu kelas dalam suatu dokumen (Raschka, 2014). *Multinomial Naïve Bayes* pada teks dilakukan tanpa memperhitungkan jumlah kata yang muncul dalam dokumen. Berikut persamaan *Multinomial Naïve Bayes* menggunakan *conditional probability* menggunakan *add one* dan *Laplace smoothing* ditunjukkan pada Persamaan 2.10.

$$P(d_i|c) = \frac{count(d_i, c_j) + 1}{\left(\sum_{w \in V} count(d, c_j) + |V|\right)} \quad (2.10)$$

Keterangan:

$count(d_i,c_j) +1$ = Jumlah dari suatu kata *query* yang muncul dalam satu kelas atau kategori, penambahan angka 1 digunakan untuk menghindari nilai 0.

$\sum_{w \epsilon V} count(d,c_j)$ = Jumlah kata yang ada pada kelas c_j

V = Jumlah seluruh kata unik yang ada pada semua kelas

2.9.3 *Bernoulli Naïve Bayes*

Bernoulli Naïve Bayes menerapkan algoritme pelatihan dan klasifikasi, untuk klasifikasi dengan menggunakan *binary* (0 dan 1) pada fitur biner. Nilai 1 pada *Bernoulli Naïve Bayes* menyatakan fitur tersebut terdapat dalam dokumen dan 0 fitur tersebut terdapat dalam dokumen (Raschka, 2014). Dalam kasus klasifikasi teks, vektor kata kejadian (bukan jumlah kata vektor) dapat digunakan untuk melatih dan menggunakan pengklasifikasi ini. *Bernoulli Naïve Bayes* mungkin tampil lebih baik di beberapa dataset, terutama yang memiliki dokumen lebih pendek. Berikut persamaan *Bernoulli Naïve Bayes* ditunjukkan pada Persamaan 2.11 dan 2.12.

$$P(d_i|c) = \prod_{t=1}^{m} P(w_t|C)^b \,.\, (1 - P(w_t|C))^{(1-b)} \qquad b \in (0,1) \quad (2.11)$$

Keterangan:

$P(w_t|C)$ = probabilitas kata yang terdapat dalam dokumen kelas *C*

$(1 - P(w_t\|C))$	=	probabilitas kata yang tidak terdapat dalam dokumen kelas
b_{it}	=	Kemunculan kata w_t yang terdapat dalam dokumen, Jika terdapat kata dalam dokumen maka b_{it} = 1 dan probabilitas yang dibutuhkan adalah $P(w_t\|C)$
$(1-b_{it})$	=	Kemunculan kata w_t tidak terdapat dalam dokumen maka b_{it} = 0 dan probabilitas yang dibutuhkan adalah $(1 - P(w_t\|C))$

$$P(w_t|C) = \frac{df_{wt,C} + 1}{df_C + 2} \quad (2.12)$$

Keterangan:

$df_{wt,C}$	=	Jumlah dokumen dalam *dataset training* yang mengandung fitur w_t dan termasuk dalam kelas *C.*
df_C	=	Jumlah dokumen dalam *dataset training* yang termasuk dalam kelas *C.*
+1 dan *+2*	=	Parameter *Laplace Smoothing*.

2.10 Evaluasi

Evaluasi digunakan untuk mengukur hasil uji coba kinerja sistem yang telah dibuat dengan metode tertentu dan juga agar dapat mengetahui sistem yang dibuat sudah dapat berjalan dengan baik terhadap kondisi lingkungan atau parameter yang lebih kompleks, variatif dan berbeda. Data yang digunakan hanya berlabel positif dan negatif maka evaluasi dilakukan dengan menggunakan *confusion matrix*. Evaluasi dengan *confusion matrix* hanya menggunakan perhitungan *Preccission, Recall,* dan Akurasi. Evaluasi *confussion matrix* ditunjukkan pada Tabel 2.6.

Tabel 2.6 *Confusion Matrix*

	Classified Positive	***Classified Negative***
Actual Positive	TP (*True Positive*)	FN (*False Negative*)
Actual Negative	FP (*False Positive*)	TN (*True Negative*)

Evaluasi *Precisson* melakukan perhitungan dengan tingkat ketepatan yang diminta oleh pengguna dengan hasil jawaban yang diberikan oleh *system* (Manning, Raghavan, & Schütze, 2008). Berikut persamaan *Precission* ditunjukkan pada Persamaan 2.13.

$$P = \frac{tp}{tp + fp} \quad (2.13)$$

Keterangan:

P	=	*Precission*
tp	=	Jumlah nilai aktual pada dokumen, berkategori positif (*retrieved*) dan jumlah nilai *classified* berkategori positif (relevan)
fp	=	Jumlah nilai aktual pada dokumen, berkategori negatif (*not retrieved*) dan jumlah nilai *classified* berkategori positif (relevan)

Evaluasi *Recall* adalah tingkat jumlah banyak dan sedikitnya kesesuaian informasi yang didapatkan dari hasil percobaan berdasarkan sudut pandang kelas atau label yang dilakukan. Perhitungan ini dilakukan dengan cara Proporsi dari semua dokumen yang relevan di koleksi termasuk dokumen yang diperoleh (Manning, Raghavan, & Schütze, 2008). Berikut persamaan *Precission* ditunjukkan pada Persamaan 2.14.

$$R = \frac{tp}{tp + fn} \tag{2.14}$$

Keterangan:

R	=	*Recall*
tp	=	Jumlah nilai aktual pada dokumen, berkategori positif (*retrieved*) dan jumlah nilai *classified* berkategori positif (relevan)
fn	=	Jumlah nilai aktual pada dokumen, berkategori positif (*retrieved*) dan jumlah nilai *classified* berkategori negative (tidak relevan)

Evaluasi *F-Measure* adalah bobot *harmonic mean* pada *recall* dan *precission* atau pengukuran yang menilai timbal balik antara *precission* dan *recall* (bobot *harmonic mean*) (Manning, Raghavan, & Schütze, 2008). Rumus untuk menghitung *F-Measure* bisa dilihat di Persamaan 2.15.

$$F = \frac{1}{\frac{1}{2}\left(\frac{1}{P} + \frac{1}{R}\right)} = \frac{2 \times P \times R}{P + R} \tag{2.15}$$

Keterangan:

F	=	*F - Measure*
P	=	Nilai hasil dari *Precission*
R	=	Nilai hasil dari *Recall*

Evaluasi *Accuracy* adalah perhitungan dengan melakukan kesesuaian nilai hasil prediksi pengujian dengan nilai actual (*ground truth*) yang dibandingkan (Manning, Raghavan, & Schütze, 2008). Berikut persamaan perhitungan *Accuracy* yang ditunjukkan pada Persamaan 2.16.

$$Accuracy = \frac{tp + tn}{tp + fp + fn + tn} \quad (2.16)$$

Keterangan:

tp	=	Jumlah nilai aktual pada dokumen, berkategori positif (*retrieved*) dan jumlah nilai *classified* berkategori positif (relevan)
fp	=	Jumlah nilai aktual pada dokumen, berkategori negatif (*not retrieved*) dan jumlah nilai *classified* berkategori positif (relevan)
fn	=	Jumlah nilai aktual pada dokumen, berkategori positif (*retrieved*) dan jumlah nilai *classified* berkategori negatif (tidak relevan)
tn	=	Jumlah nilai aktual pada dokumen, berkategori negatif (not *retrieved*) dan jumlah nilai *classified* berkategori negatif (tidak relevan)

BAB 3 METODOLOGI PENELITIAN

Tahapan penelitian ini dirancang untuk analisis sentimen opini film menggunakan metode *naïve bayes* dengan *ensemble feature* dan seleksi fitur *pearson correlation coefficient.* Alur penelitian yang digunakan dalam proses analisis sentimen ini secara umum meliputi Tipe Penelitian, Strategi Penelitian, Partisipan Penelitian, Lokasi Penelitian, Teknik Pengumpulan Data, Implementasi Algoritme, dan Teknik Analisis Data.

3.1 Tipe Penelitian

Pada penelitian ini menggunakan tipe penelitian nonimplementatif. Tipe penelitian nonimplementatif berfokus pada investigasi terhadap fenomena atau situasi tertentu, atau analisis terhadap hubungan antar fenomena yang sedang dikaji untuk kemudian menghasilkan hasil investigasi atau hasil analisis ilmiah sebagai produk utamanya. Metode yang digunakan untuk menghasilkan produk utama bisa berupa survei, eksperimentasi, studi kasus, penelitian tindakan (*action research*), studi etnografi, wawancara, kuisioner, observasi, dan sebagainya. Jika ditinjau dari kegiatan penelitiannya, pendekatan pada penelitian ini menggunakan nonimplementatif analitik (*analytical*/*explanatory*), penelitian dengan menggunakan pendekatan ini bertujuan untuk menjelaskan derajat hubungan antar elemen dalam objek penelitian dengan fenomena/situasi tertentu yang sedang diteliti, dengan produk yang dihasilkan berupa analisis.

3.2 Strategi Penelitian

Penelitian ini termasuk penelitian yang menggunakan metode eksperimen. Penelitian eksperimen adalah penelitian yang melakukan investigasi hubungan sebab akibat dengan menggunakan uji coba yang dikontrol oleh peneliti dengan melibatkan pengembangan dan evaluasi dan pada umumnya dilakukan di laboratorium. Metode eksperimen yang digunakan dalam penelitian ini yaitu *Naïve Bayes* dengan *ensemble features* dan seleksi fitur *Pearson Correlation Coefficient.*

3.3 Partisipan Penelitian

Partisipan yang terlibat dalam penelitian ini adalah Ibu Eka Dewi Lukmana Sari selaku Guru Bahasa Indonesia SMA Negeri 6 Balikpapan. Alasan memilih partisipan penelitian ini karena partisipan dapat memberikan label opini film pada Twitter sesuai dengan spesialisasi partisipan sehingga dapat menghasilkan data yang valid.

3.4 Lokasi Penelitian

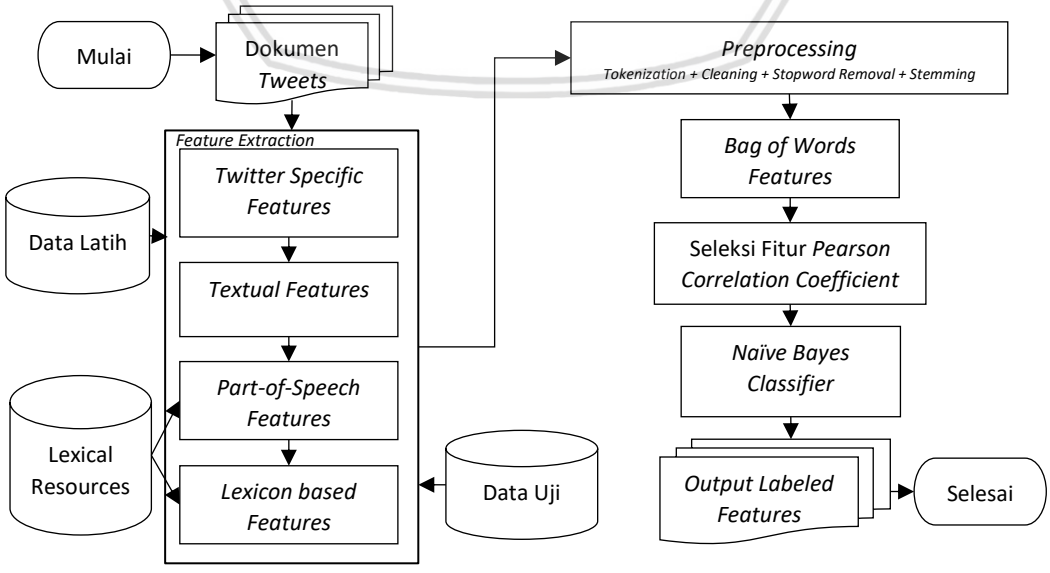
Penelitian ini dilakukan di Laboratorium Komputasi Cerdas Fakultas Ilmu Komputer Universitas Brawijaya. Alasan pemilihan lokasi penelitian dilakukan untuk mengukur pengaruh penggunaan seleksi fitur pada *ensemble features* terhadap analisis sentimen opini film berbahasa Indonesia pada media sosial Twitter.

3.5 Teknik Pegumpulan Data

Tahapan ini bertujuan untuk mengumpulkan data yang akan dapat digunakan untuk penelitian. Data yang digunakan dalam penelitian ini merupakan data primer yang berupa *tweet*. Sumber dataset yang digunakan berasal dari situs web sosial media yaitu Twitter, dataset yang diambil berupa data uji dan data latih. Pada penelitian ini untuk mengambil data *tweet* menggunakan *Twitter REST API Search* untuk mengambil data *tweet* yang akan digunakan pada penelitian ini. Dari data tersebut mengandung dua kelas yaitu positif dan negatif yang terdiri dari 5 fitur yaitu, *Twitter Specific, Textual Features, Part-of-Speech (POS) Features, Lexicon Based Features* dan *Bag of Words*. Penentuan kelas positif dan negatif akan ditentukan oleh seseorang yang menyukai film dan pengamat film yang selanjutnya akan diverifikasi oleh seorang pakar.

3.6 Implementasi Algoritme

Implementasi Algoritme merupakan penjabaran dari proses-proses dalam membangun sebuah sistem. Sistem yang akan dibangun adalah detail dari sistem analisis sentimen. Tujuan dari sistem ini adalah untuk memberikan label sentimen positif atau negatif ke dokumen *tweet* berdasarkan isinya. Gambaran umum dari proses perancangan algoritme ditunjukkan pada Gambar 3.1.



Gambar 3.1 Perancangan Algoritme

Dari Gambar 3.1 implementasi algoritme, dokumen *tweet* merupakan hasil dari *crawling* Twitter *REST API* yang sudah diolah menjadi data latih dan data uji. Selanjutnya akan dilakukan ekstraksi fitur untuk semua fitur *ensemble* untuk mendapatkan nilai fitur dengan memasukkan seluruh data latih dan data uji dokumen *tweet*.

Pada tipe fitur *Twitter Specific* akan mengekstrak seluruh nilai fitur berdasarkan aturannya yaitu *hashtag*, *retweet*, *username*, dan *URL*. Selanjutnya bagian *textual features*, yaitu pengambilan nilai fitur berdasarkan informasi eksplisit yang terdapat di dalam suatu *tweet,* seperti jumlah tanda tanya, tanda seru, jumlah huruf besar pada setiap kata dalam *tweet* dan sebagainya. Setelah itu masuk bagian *Parts of Speech (PoS) features* kamus yang digunakan pada *PoS* menggunakan API dari Kateglo (Kamus, Tesaurus, Glosarium)[4] dan *lexicon-based features* yang melakukan pengambilan fitur berdasarkan *lexicon* atau kamus[5]. Kamus yang digunakan berasal dari penelitian (Wahid & Azhari, 2016).

Pada bagian fitur *Bag of Words* (BoW), proses yang dilakukan pertama adalah melakukan *preprocessing* pada seluruh dokumen *tweet* yang meliputi *tokenization, cleaning, filtering, stemming*. Pada proses *filtering stopword* yang digunakan modifikasi dari *stopword* Tala dan *stemming* menggunakan Sastrawi. Setelah seluruh proses pengambilan nilai fitur dilakukan, maka akan dilakukan untuk mencari nilai *Pearson's* untuk mendapatkan fitur – fitur dan dilakukan proses seleksi fitur berdasarkan nilai *Pearson's* tertinggi dan mendapatkan fitur – fitur yang terseleksi berdasarkan nilai *Pearson's*. Selanjutnya melakukan proses *Naïve Bayes Classifier*, perhitungan *Naïve Bayes* dihitung berdasarkan tipe data dan menggunakan distribusi *Naïve Bayes* sesuai tipe datanya, sehingga dari hasil klasifikasi dapat diperoleh label kelas *tweet*, yaitu positif atau negatif.

3.7 Teknik Analisis Data

Pengujian dilakukan setelah dilakukan kode program dari rancangan algoritme. Pengujian dilakukan agar dapat mengetahui sistem yang dibuat sudah dapat berjalan dengan baik sesuai dengan kebutuhan dan perancangan yang telah dibuat sebelumnya. Pengujian dilakukan dengan membuktikan hasil pelabelan kelas yang dilakukan oleh pakar dan yang dilakukan oleh sistem dan pengujian kombinasi fitur yang optimal dengan *dataset* Twitter. Berikut skenario pengujian yang dilakukan, yaitu:

1. Pengujian yang ditujukan untuk mengetahui pengaruh seleksi fitur *Pearson Correlation Coefficient* pada kata tidak baku setiap masing – masing fitur yaitu *Bag of Words* dan *ensemble* (*twitter* specific *features*, *textual features, Parts*

[4] http://kateglo.com/api.php?format=json&phrase=bahtera
[5] https://github.com/masdevid/sentistrength_id

of Speech (PoS) features, *lexicon-based features)* terhadap akurasi metode *Naïve Bayes*.

2. Pengujian yang ditujukan untuk mengetahui pengaruh seleksi fitur *Pearson Correlation Coefficient* pada kata baku setiap masing – masing fitur yaitu *Bag of Words* dan *ensemble* (*twitter* specific *features*, *textual features, Parts of Speech (PoS) features*, *lexicon-based features)* terhadap akurasi metode *Naïve Bayes*.

Setelah proses pengujian selesai, akan dilakukan analisis terhadap hasil pengujian. Kemudian, tahap analisis dilakukan dengan dengan melihat nilai akurasi yang dihasilkan oleh *Naïve Bayes* dari skenario pengujian untuk menilai pengaruh semua fitur terhadap metode *Naïve Bayes* dan menilai pengaruh seleksi fitur *Pearson Correlation Coefficient* terhadap akurasi metode *Naïve Bayes*.

3.8 Kesimpulan dan Saran

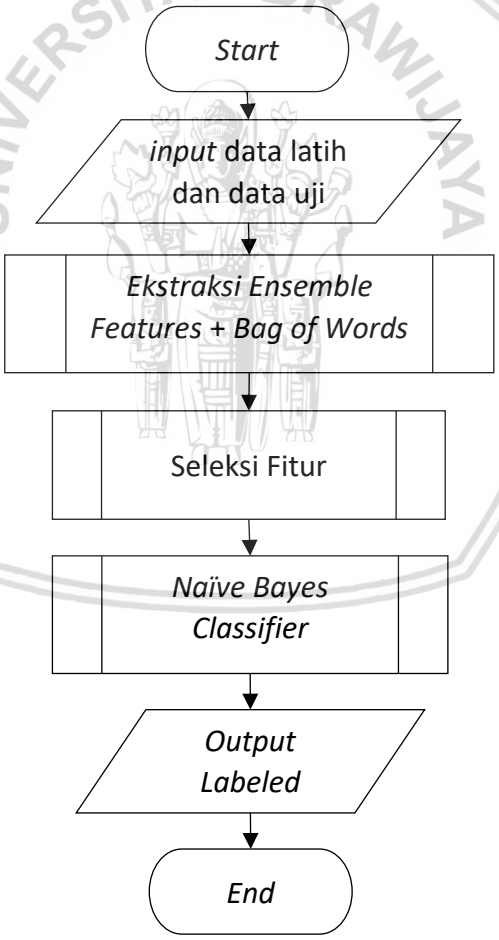
Pembuatan kesimpulan dilakukan sesudah pengujian dilakukan, kesimpulan akan menjawab pertanyaan dari masalah yang telah didapatkan meliputi tahapan pengaruh seleksi fitur *Pearson Correlation Coefficient* terhadap analisis sentimen opini film menggunakan metode *Naïve* Bayes dengan *ensemble feature* kemudian pengaruh kata tidak baku dan kata baku terhadap analisis sentimen opini film menggunakan *Naïve Bayes* dengan *ensemble feature* dan seleksi fitur *Pearson Correlation Coefficient*.

Bagian saran diperoleh dari analisis yang dilakukan pada pengujian dan juga penulisan saran dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan untuk penelitian selanjutnya.

BAB 4 PERANCANGAN

4.1 Perancangan Tahapan Proses

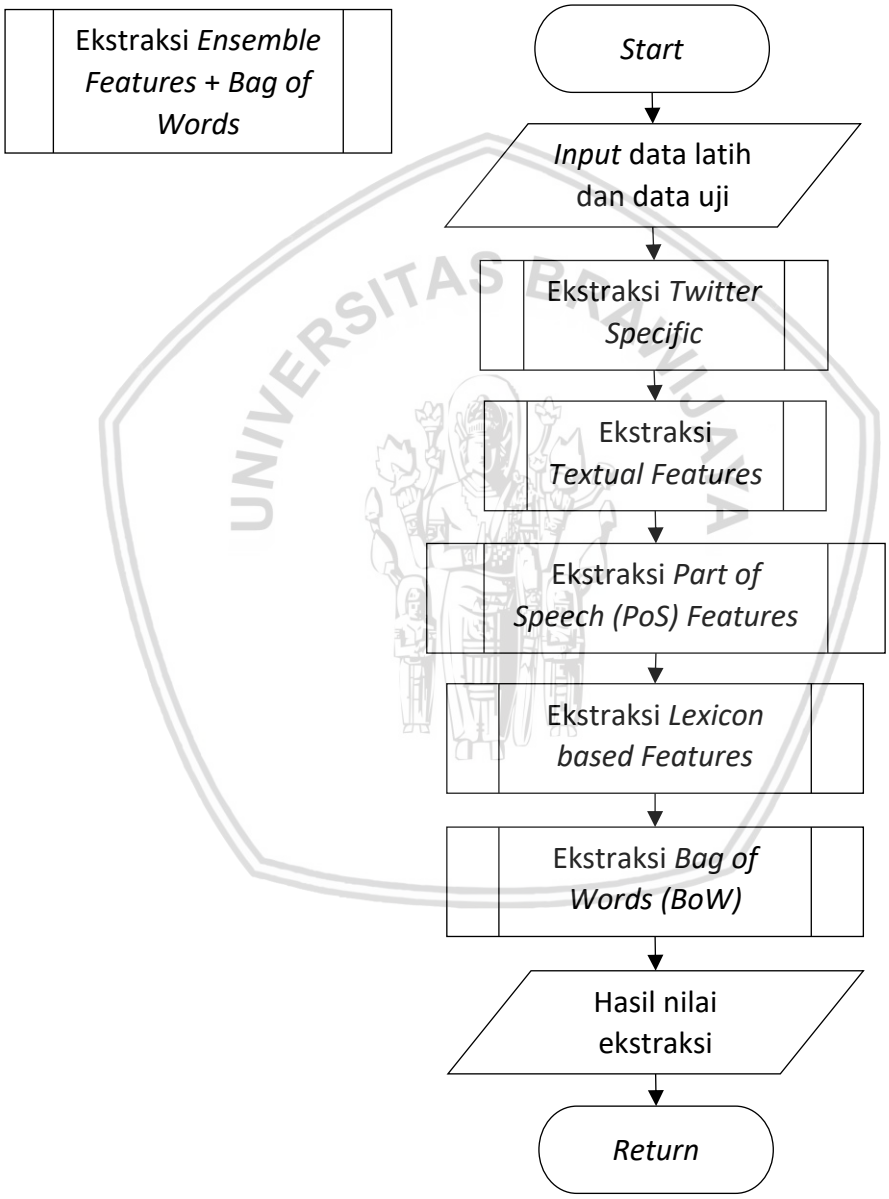
Perancangan dalam analisis sentimen opini film untuk sistem ini adalah dengan memasukkan dokumen *tweet* opini film yaitur dengan memasukkan data latih dan data uji. Data tersebut akan dilakukan proses ekstraksi terlebih dahulu sesuai dengan *ensemble feature* dan *Bag of Words* sehingga akan didapatkan nilai sesuai dengan fiturnya. Setelah melakukan ekstraksi fitur, maka akan dilakukan seleksi fitur dengan menggunakan *Pearson Correlation Coefficient* untuk memilih fitur yang relevan dan memilih kombinasi fitur yang optimal. Setelah dilakukan proses seleksi fitur maka langkah selanjutnya adalah melakukan klasifikasi menggunakan *Naïve Bayes* yang sesuai dengan tipe datanya. Keluaran dari sistem ini berupa analisis sentimen opini film positif dan negatif. Gambar 4.1 menunjukkan tahapan proses perhitungan yang dilakukan.



Gambar 4.1 Diagram Alir Proses *Naïve Bayes* dengan *Ensemble Feature* dan Seleksi Fitur

4.1.1 Tahapan Proses *Ensemble Features*

Pada tahapan ini, dilakukan proses dari ekstraksi fitur menggunakan *ensemble feature* dan *Bag of Words*, proses ekstraksi fitur *ensemble* dilakukan dari F1 sampai dengan F38, pada proses ekstraksi *ensemble features* terdapat 4 bagian proses utama yaitu, dengan melakukan ekstraksi fitur terlebih dahulu pada fitur *twitter specific*, *textual features*, *Part of Speech (PoS) features*, *Lexicon-based features* dan *Bag of Words*. Diagram alir tahapan proses pengambilan nilai fitur ditunjukkan pada Gambar 4.2.

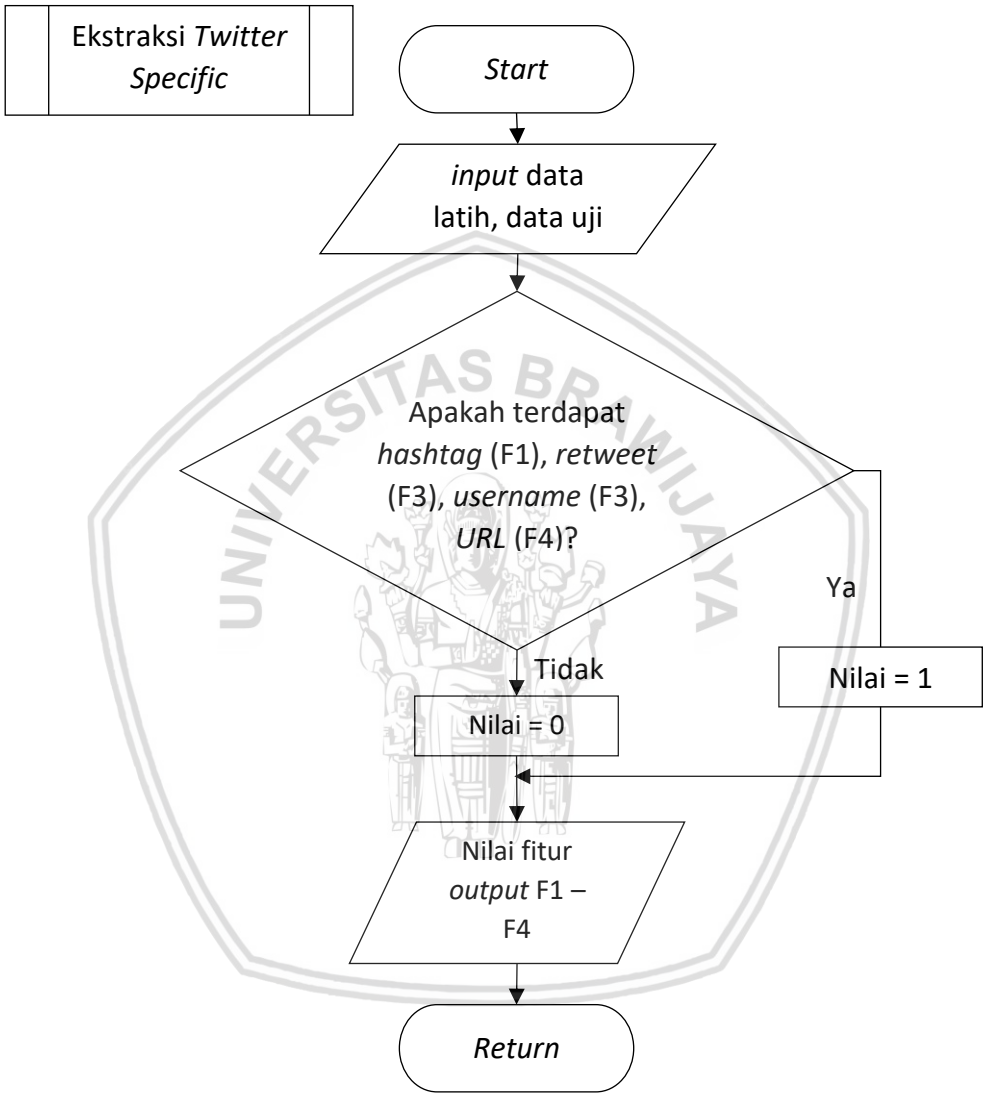


Gambar 4.2 Diagram Alir *Ensemble Feature*

4.1.1.1 Tahapan Proses *Twitter Specific*

Pada tahap ini, proses pengambilan nilai fitur (F) dilakukan berdasarkan spesifikasi tertentu yang dimiliki pada satu dokumen *tweet*, terdapat F1 – F4

dalam tipe *twitter specific* yang meliputi beberapa pengecekan, untuk F1 dilakukan pengecekan pada *tweet* apakah memiliki *hashtag*, untuk F2 dilakukan pengecekan pada *tweet* apakah memiliki *retweet*, untuk F3 apakah terdapat *username* dan F4 dilakukan pengecekan pada *tweet* apakah mengandung *URL*. Diagram alir *twitter specific* ditunjukkan pada Gambar 4.3.

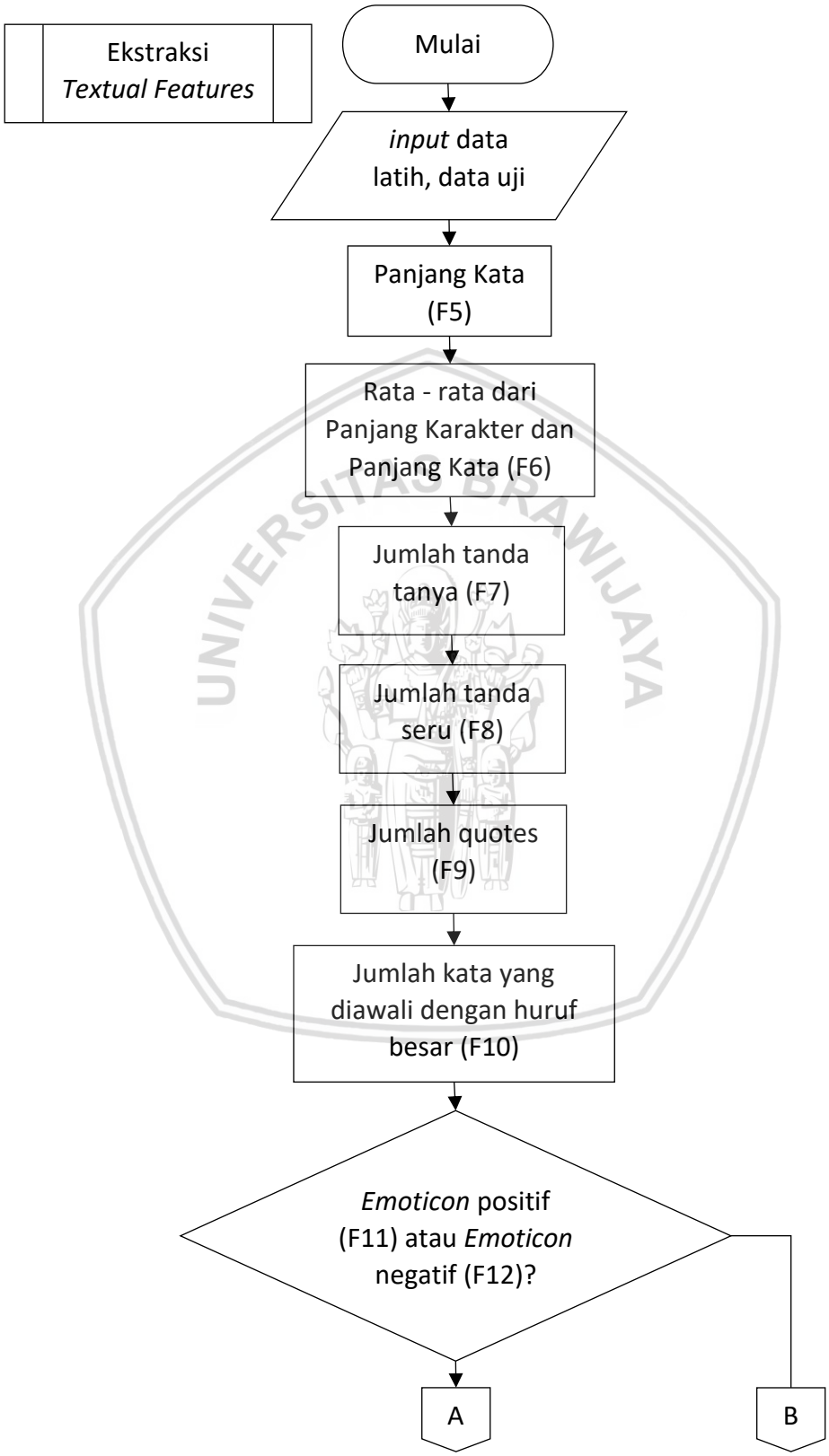


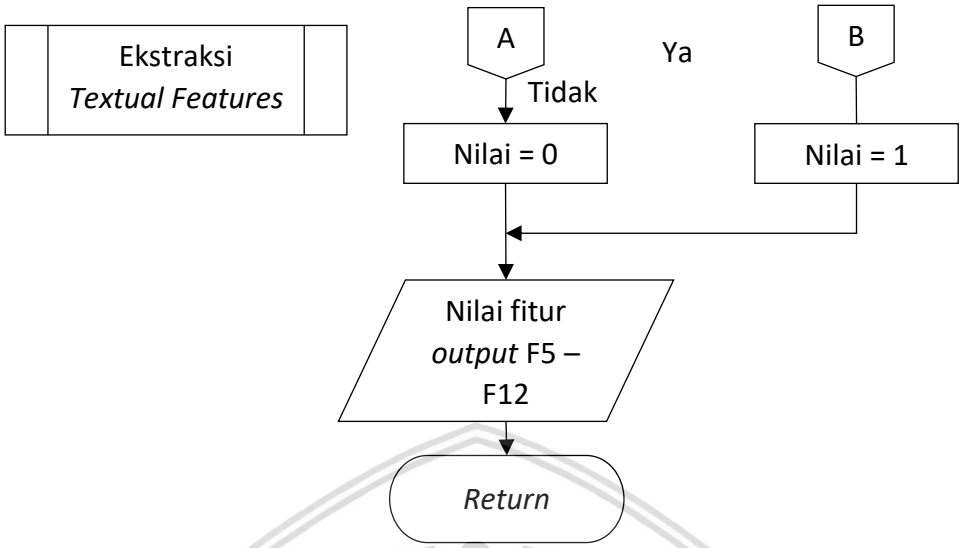
Gambar 4.3 Diagram Alir *Twitter Specific*

4.1.1.2 Tahapan Proses *Textual Features*

Pada tahap ini, proses pengambilan nilai fitur dilakukan berdasarkan informasi spesifik yang dalam satu dokumen *tweet*, terdapat fitur F5 – F12 dalam tipe ini yaitu, menghitung panjang kata (F5), menghitung rata – rata dari panjang karakter dan panjang kata (F6), jumlah tanda tanya pada *tweet* (F7), jumlah tanda seru pada *tweet* (F8), jumlah *quotes* pada *tweet* (F9), jumlah kata yang diawali dengan huruf besar (F9), apakah *tweet* mengandung *emoticon* positif atau tidak

(F10), apakah *tweet* mengandung *emoticon* negatif atau tidak. Diagram alir *textual informasi* ditunjukkan pada Gambar 4.4





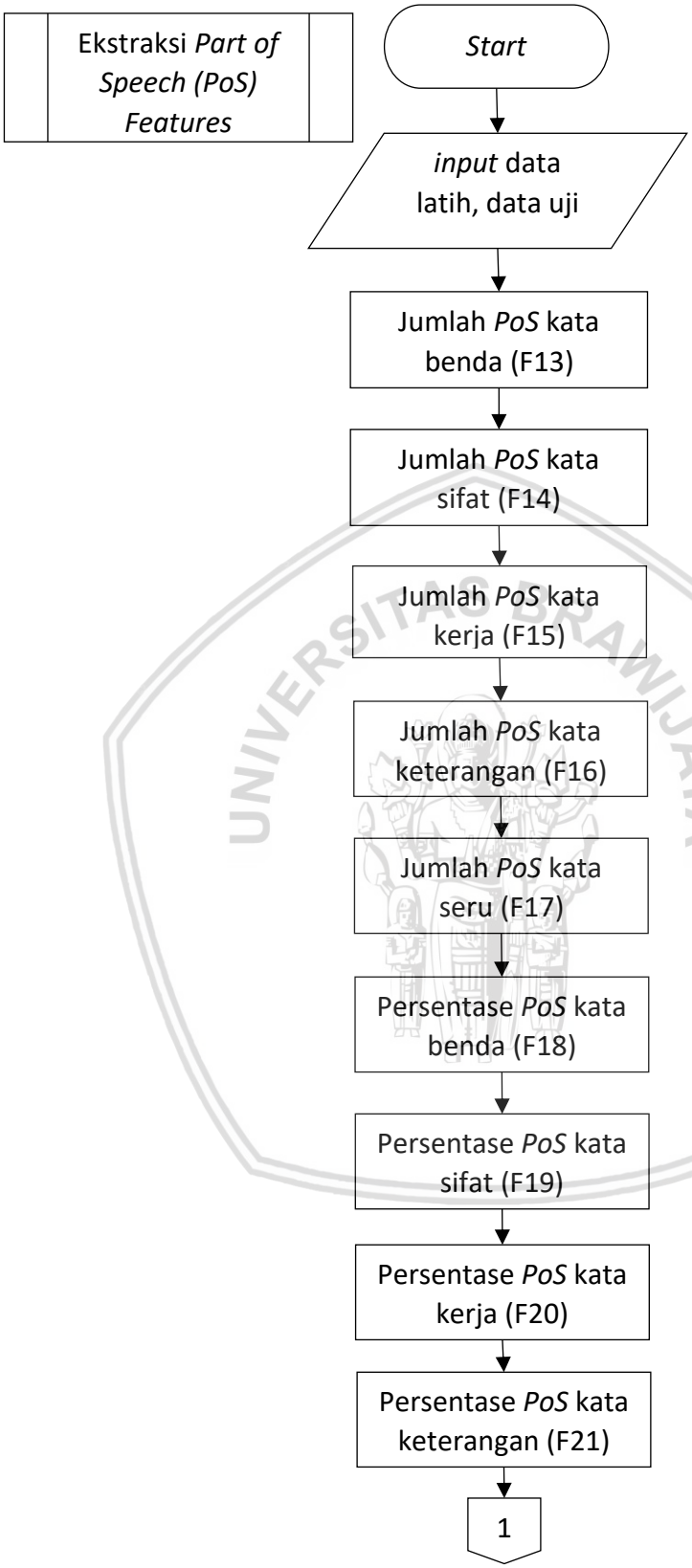
Gambar 4.4 Diagram Alir *Textual Features*

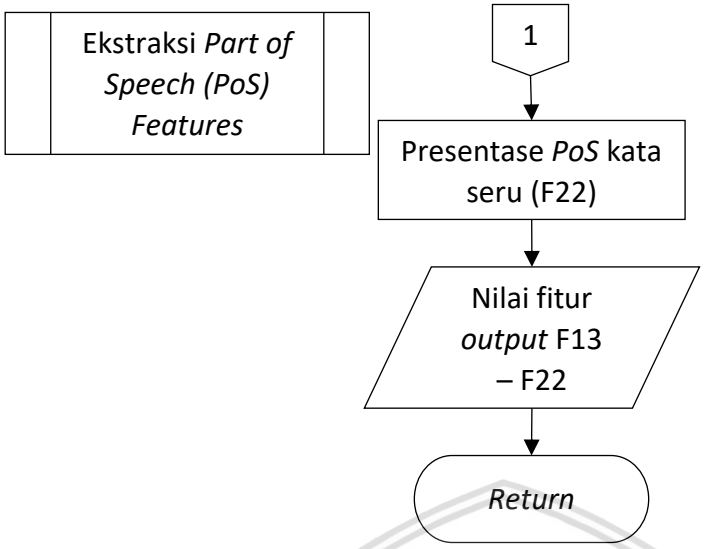
4.1.1.3 Tahapan Proses *Part of Speech*

Tahapan ini merupakan proses ekstraksi fitur berdasarkan informasi penanda kata/*PoS tagging* pada dokumen *tweet,* digunakan bantuan dengan kamus yang mengandung *PoS tagging*. Terdapat fitur F13 – F22 dalam tipe jumlah *PoS* sebagai kata benda/*noun* (F13), jumlah *PoS* sebagai kata sifat/*adjective* (F14), jumlah *PoS* sebagai kata kerja/*verb* (F15), jumlah *PoS* sebagai kata keterangan/*adverb* (F16), jumlah *PoS* sebagai kata seru/*interjection* (F17), persentase *PoS* sebagai kata benda/*noun* (F18), persentase *PoS* sebagai kata sifat/*adjective* (F19), persentase *PoS* sebagai kata kerja/*verb* (F20), persentase *PoS* sebagai kata keterangan/*adverb* (F21), persentase *PoS* sebagai kata seru/*interjection* (F22). Diagram alir *Part of Speech* ditunjukkan pada Gambar 4.5.

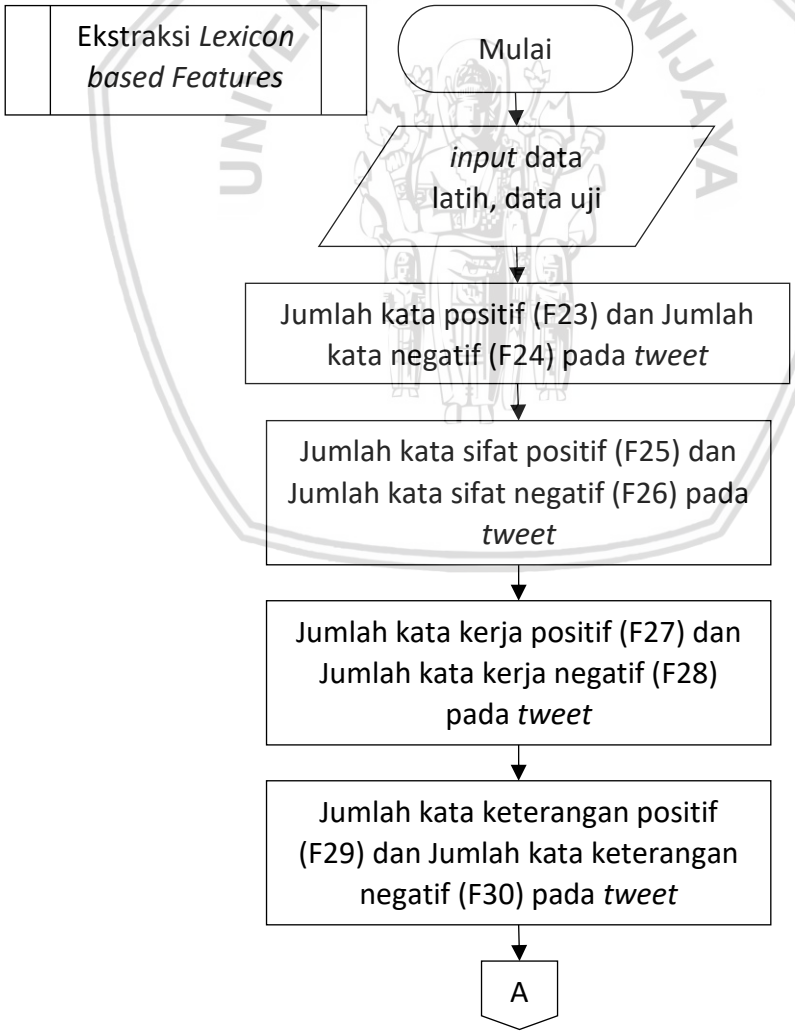
4.1.1.4 Tahapan Proses *Lexicon based Features*

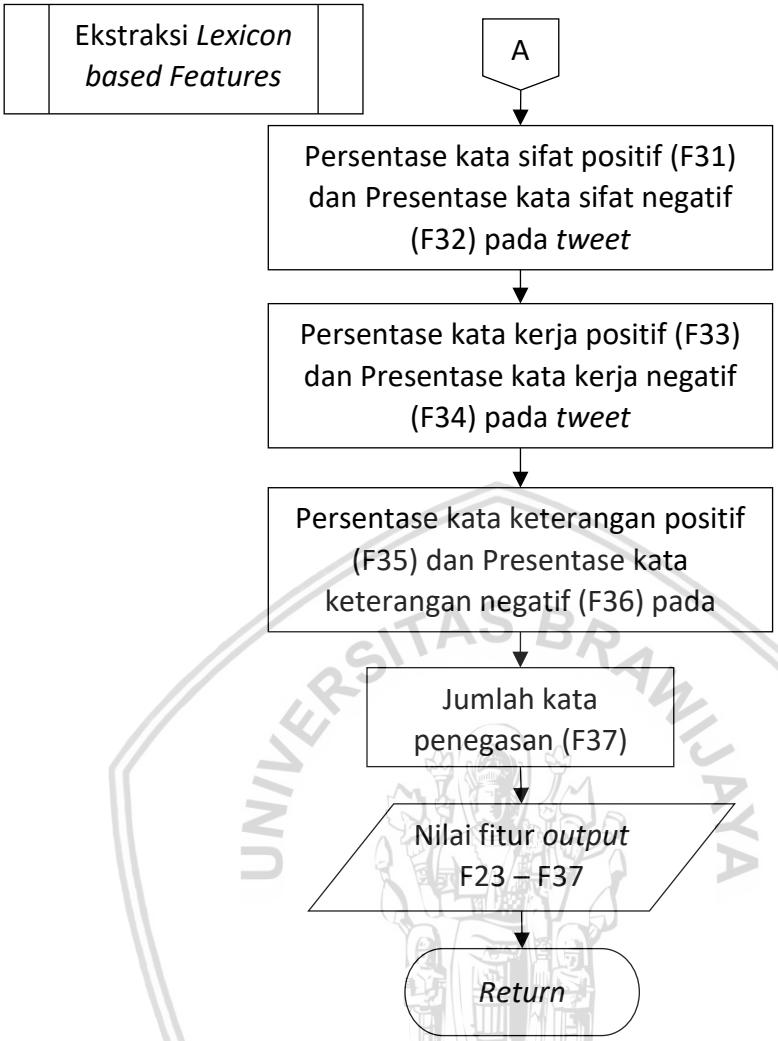
Tahapan ini merupakan proses ekstraksi fitur berdasarkan *lexicon based* pada dokumen *tweet,* dengan menggunakan bantuan kamus kata atau *lexicon*. Terdapat fitur F23 – F37 pada tipe *lexicon* yaitu, jumlah yang mengandung kata positif (F23) dan negatif (F24) di dalam *tweet*, jumlah kata sifat/*adjective* positif (F25) dan negatif (F26), jumlah kata kerja/*verb* positif (F27) dan negatif (F28), jumlah kata keterangan/*adverb* positif (F29) dan negatif (F30), persentase kata sifat/*adjective* positif (F31) dan negatif (F32), persentase kata kerja/*verb* positif (F33) dan negatif (F34), persentase kata keterangan/*adverb* positif (F35) dan negatif (F36), jumlah kata penegasan (*intensifier word)* (F37). Diagram alir *lexicon-based features* ditunjukkan pada Gambar 4.6.


Ekstraksi *Part of Speech (PoS)* Features
Start
input data latih, data uji
Jumlah *PoS* kata benda (F13)
Jumlah *PoS* kata sifat (F14)
Jumlah *PoS* kata kerja (F15)
Jumlah *PoS* kata keterangan (F16)
Jumlah *PoS* kata seru (F17)
Persentase *PoS* kata benda (F18)
Persentase *PoS* kata sifat (F19)
Persentase *PoS* kata kerja (F20)
Persentase *PoS* kata keterangan (F21)
1



Gambar 4.5 Diagram Alir *Part of Speech*





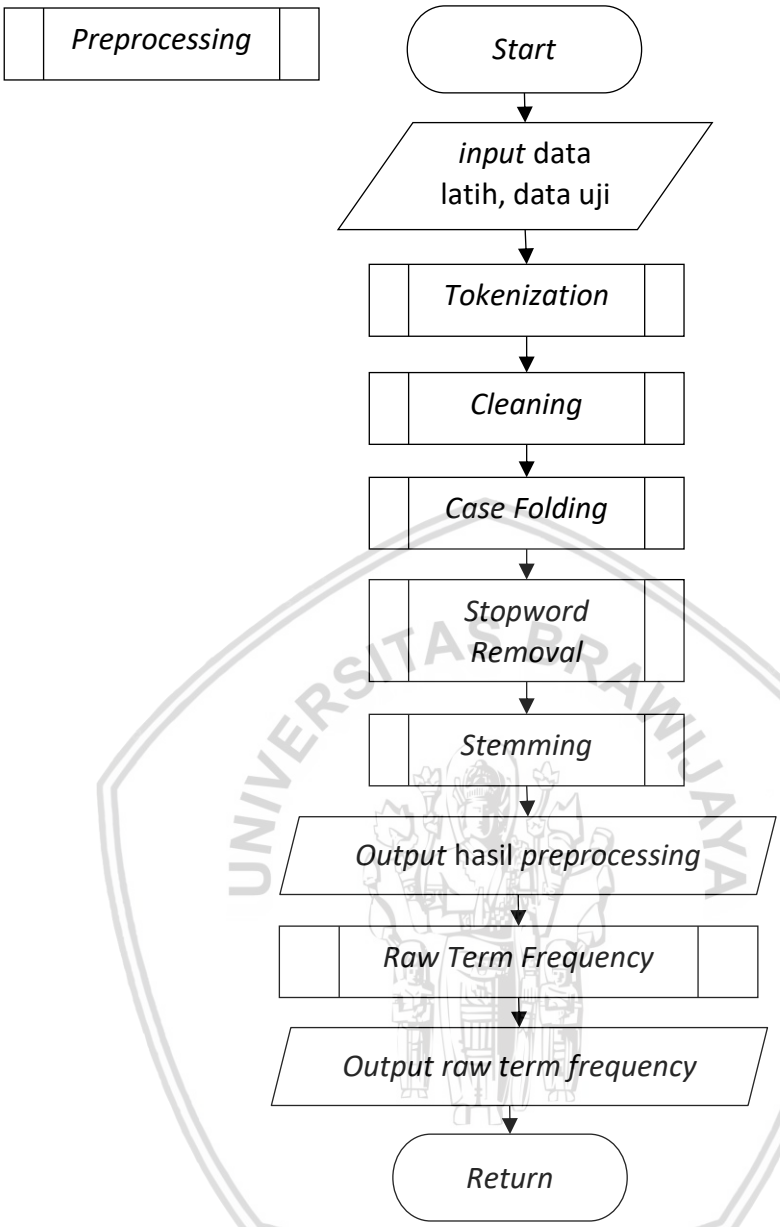
Gambar 4.6 Diagram Alir *Lexicon based features*

4.1.2 Tahapan Proses *Bag of Words*

Pada tahapan ini, proses pengambilan nilai fitur menggunakan kumpulan kata atau kumpulan *term* yang bisa disebut sebagai *Bag of Word (BoW),* pada pengambilan nilai fitur ini memerlukan *preprocessing* terlebih dahulu pada *tweet*, setelah itu akan dilakukan proses perhitungan *term frequency* berdasarkan kemunculan kata pada dokumen *tweet* untuk mendapatkan nilai bobot kata dari setiap *tweet*.

4.1.2.1 Tahapan Proses *Preprocessing*

Tahap *preprocessing* merupakan tahap pemrosesan awal dari dokumen untuk mengolah teks menjadi teks yang terstruktur. Pengolahan teks dilakukan beberapa tahap yaitu, *tokenization*, *cleaning*, *case folding*, *stopword removal*, dan *stemming*. Diagram Alir *preprocessing* ditunjukkan pada Gambar 4.7.



Gambar 4.7 Diagram Alir *Preprocessing*

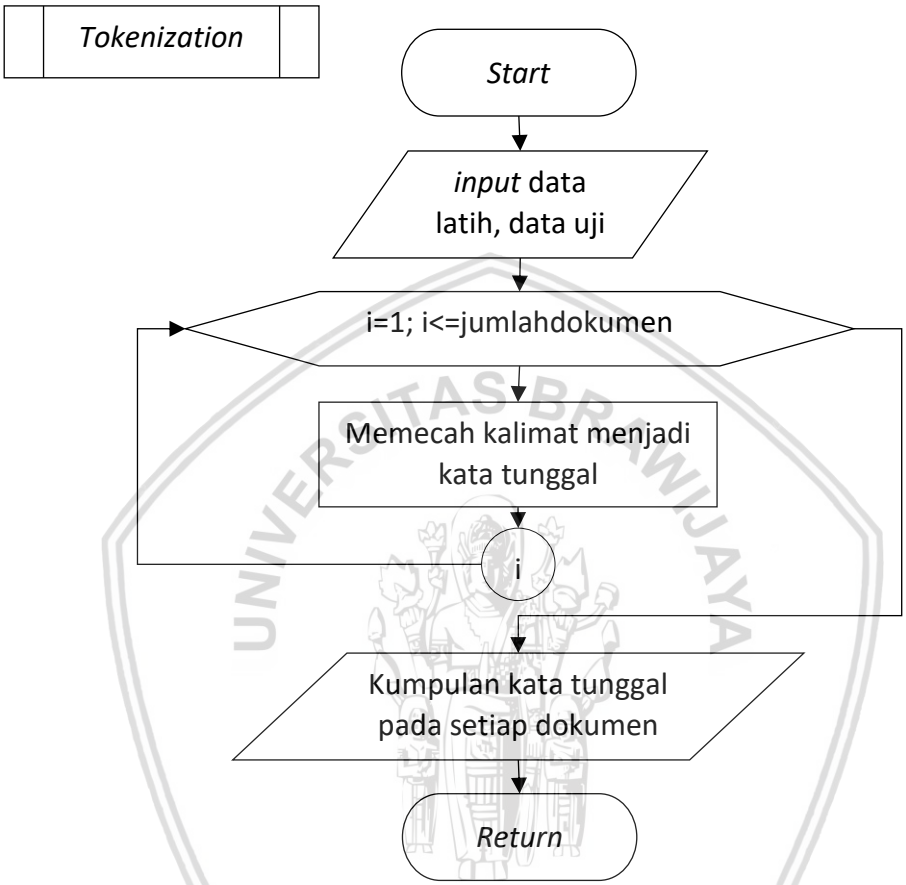
4.1.2.2 Tahapan Proses *Tokenization*

Tokenization merupakan proses pertama yang dilakukan pada *preprocessing* teks. Pada proses ini dilakukan pemisahan kumpulan kalimat menjadi kumpulan kata (token). Proses tersebut akan terus berulang hingga semua kata pada kalimat dipisahkan. Diagram Alir *tokenization* ditunjukkan pada Gambar 4.8.

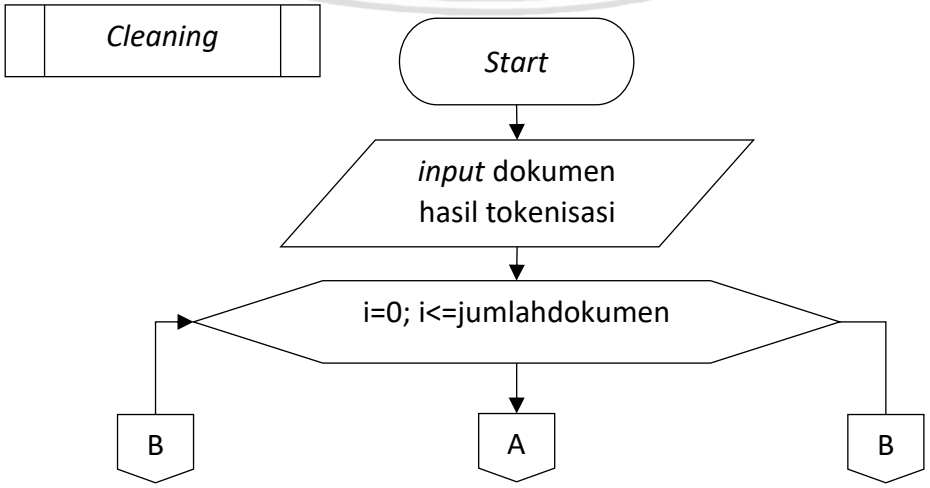
4.1.2.3 Tahapan Proses *Cleaning*

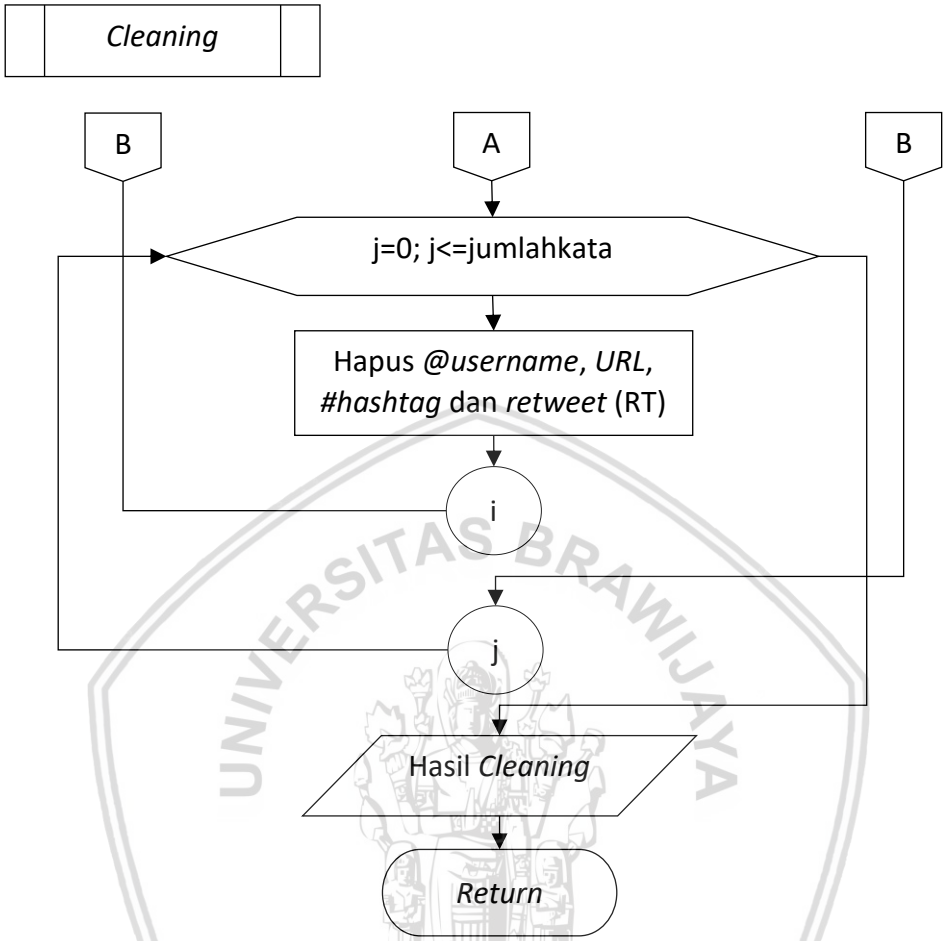
Cleaning merupakan tahapan yang dilakukan setelah *tokenization* dengan tujuan yaitu menghilangkan *noise* (karakter yang tidak penting) pada dokumen

teks. Proses yang akan dihilangkan pada tahap ini yaitu, *username* yang ditandai dengan karakter '@', *tag html, script, URL* yang ditandai dengan 'http://' atau 'https://', menghilangkan tanda *retweet* yang ditandai dengan "RT", menghilangkan karakter '#', menghilangkan semua karakter selain huruf dan menghilangkan tanda baca. Diagram Alir *cleaning* ditunjukkan pada Gambar 4.9.



Gambar 4.8 Diagram Alir *Tokenization*





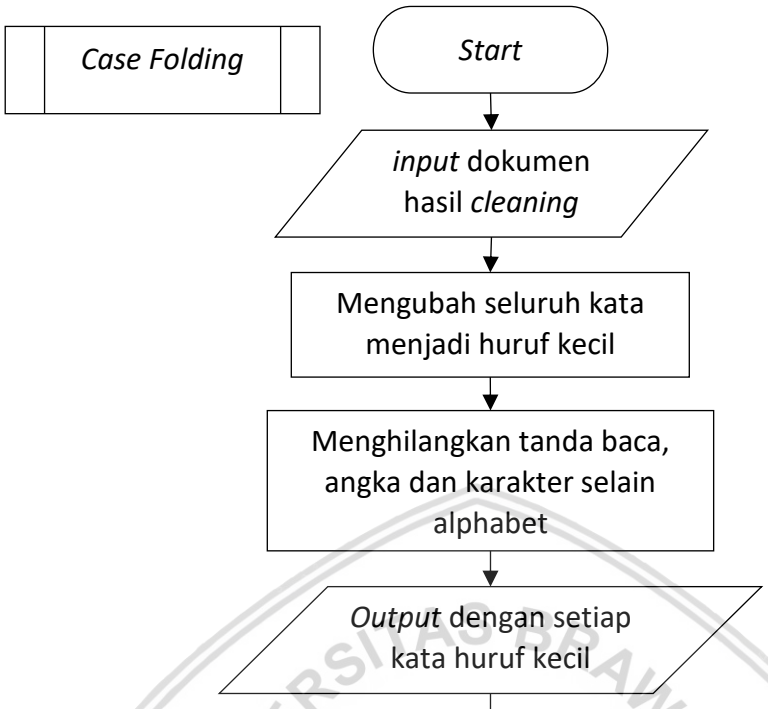
Gambar 4.9 Diagram Alir *Cleaning*

4.1.2.4 Tahapan Proses *Case Folding*

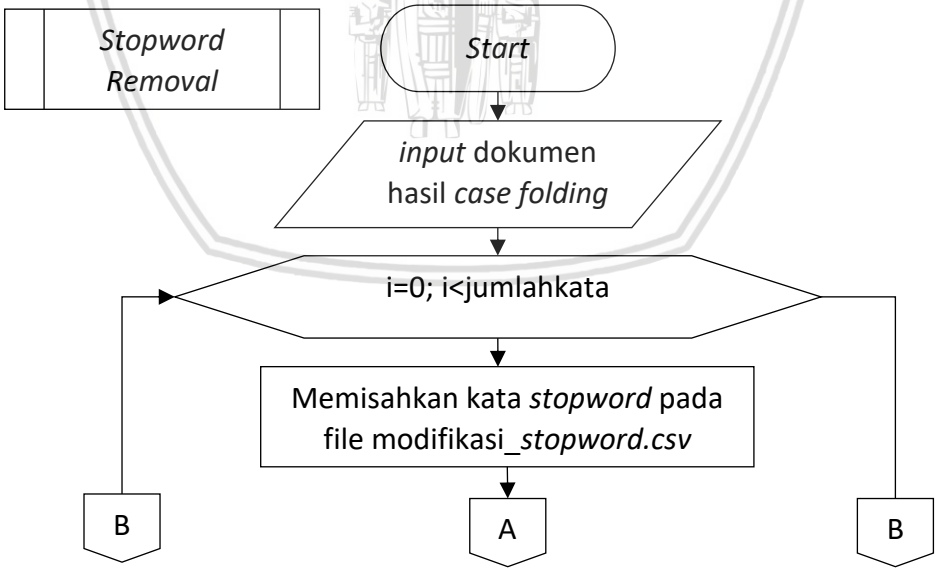
Tahap *case folding* digunakan untuk mengubah seluruh teks pada dokumen menjadi huruf kecil atau *lowercase*. Tahap *case folding* perlu dilakukan karena dokumen pada *twitter* tidak konsisten dengan penggunaan huruf capital atau *uppercase*. Tahap *case folding* juga dilakukan penghilangan tanda baca, angka dan penghilangan karakter selain alphabet pada seluruh dokumen. Diagram Alir *case folding* ditunjukkan pada Gambar 4.10.

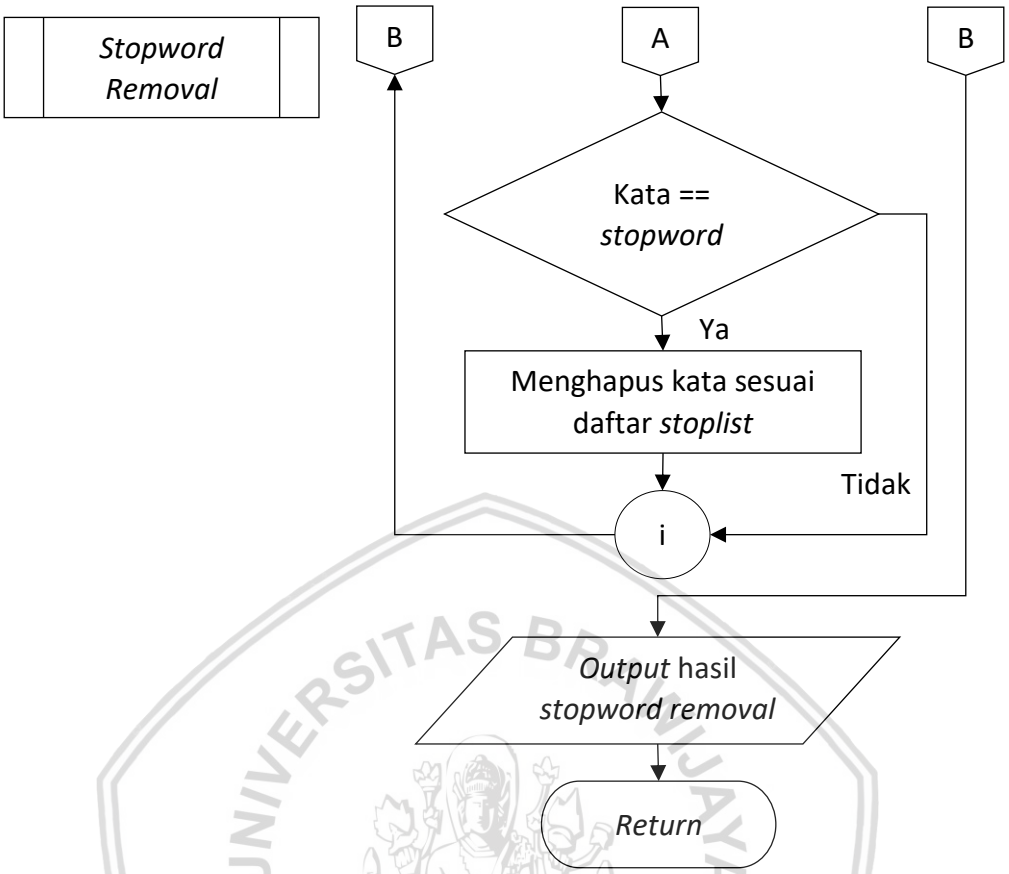
4.1.2.5 Tahapan Proses *Stopword Removal*

Tahap *Stopword Removal* adalah tahapan yang dilakukan untuk menghilangkan kata yang tidak memiliki arti atau tidak relevan. Kata yang akan dihilangkan bersumber dari *stopword Tala*. Hasil dari tahap ini adalah kumpulan kata yang akan mewakili suatu dokumen. Diagram Alir *stopword removal* ditunjukkan pada Gambar 4.11.



Gambar 4.10 Diagram Alir *Case Folding*





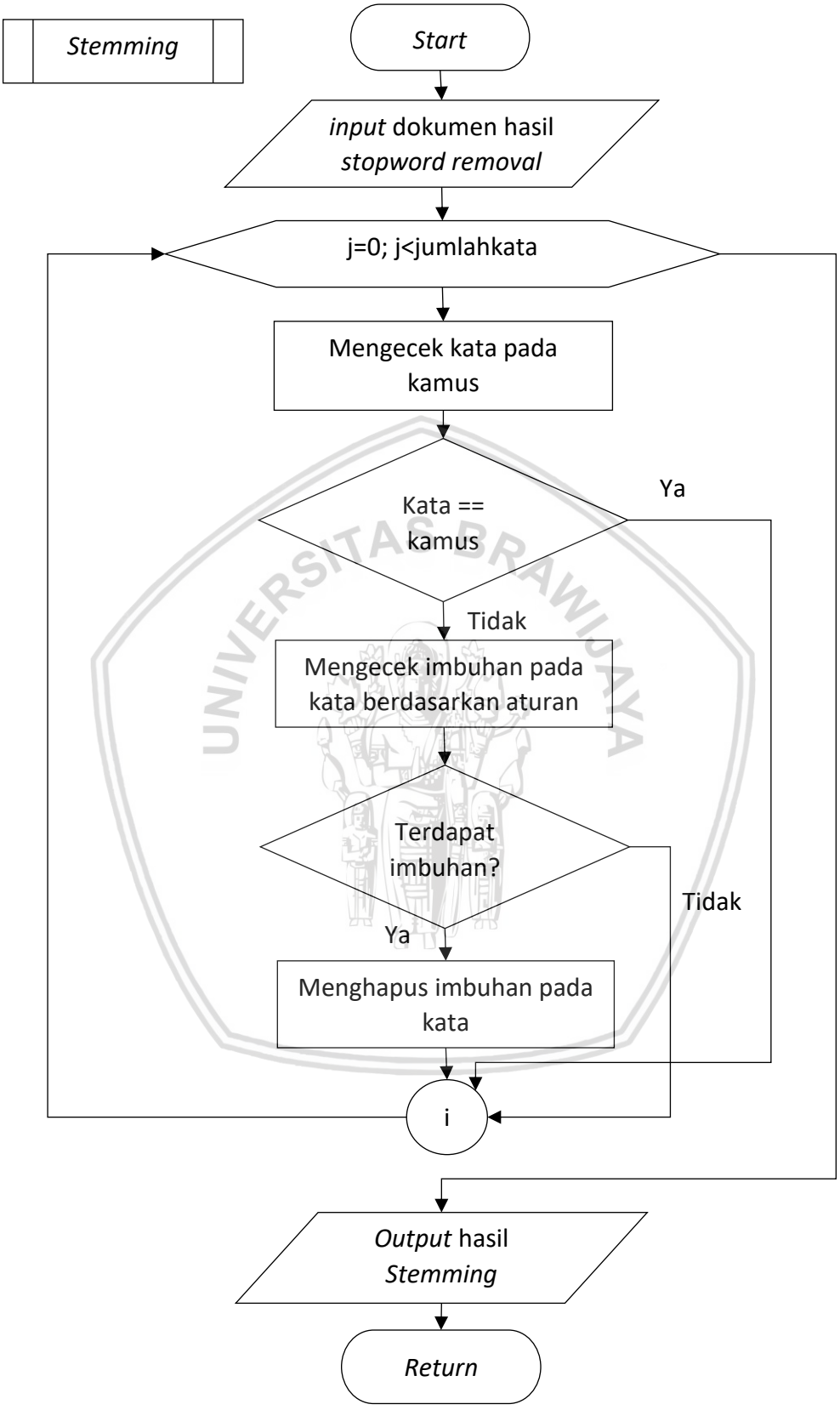
Gambar 4.11 Diagram Alir *Stopword Removal*

4.1.2.6 Tahapan Proses *Stemming*

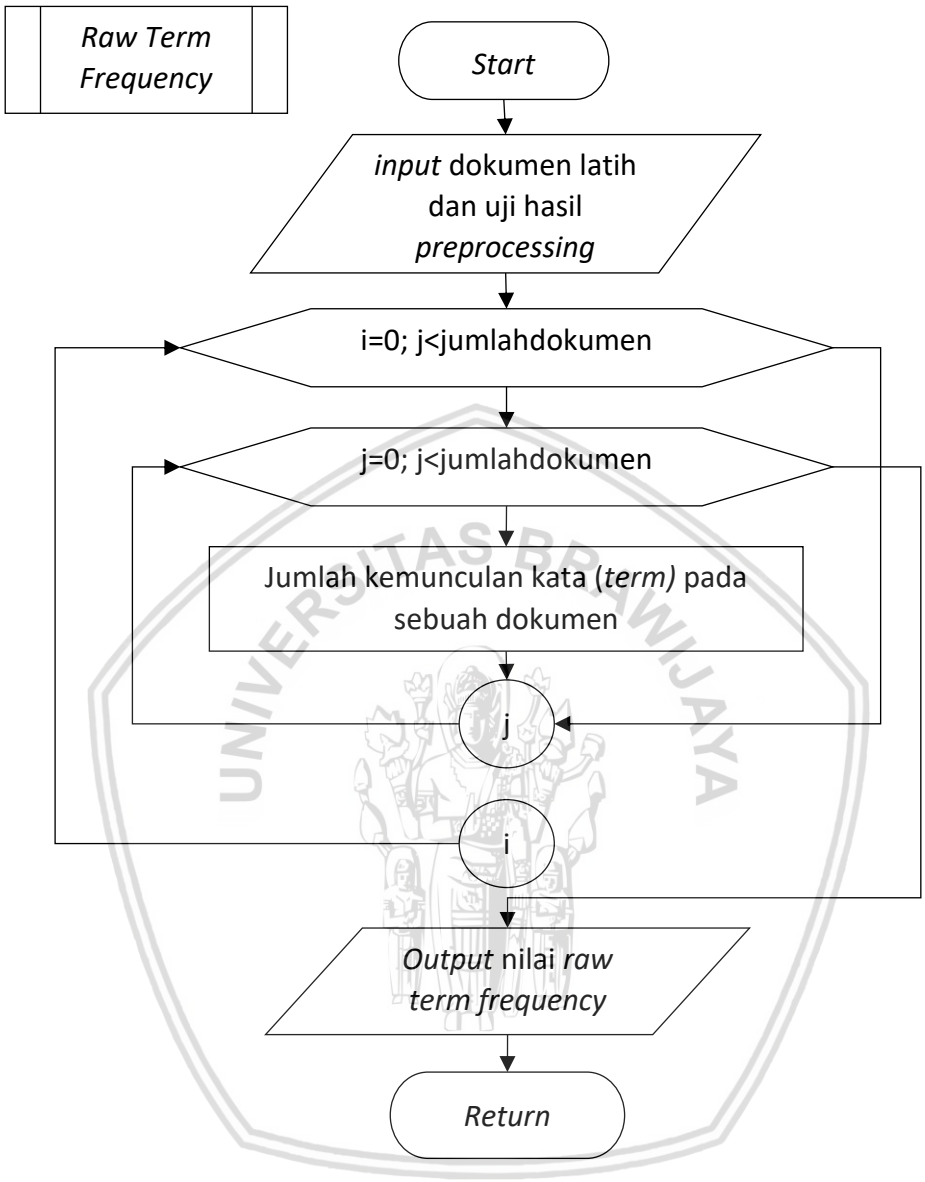
Stemming merupakan tahapan kelima pada tahapan *preprocessing* teks. Proses ini akan mengubah bentuk kata menjadi kata dasar atau proses menghilangkan imbuhan pada suatu kata sehingga hanya tersisa kata dasarnya saja. Proses pertama yang dilakukan adalah memasukkan dokumen hasil dari *stopword* removal, selanjutnya akan dilakukan pengecekkan apakah kata terdapat pada kamus jika ya maka kata tersebut merupakan hasil *stemming* jika tidak maka akan dicek terlebih dahulu apakah terdapat imbuhan jika ya maka kata tersebut masuk kedalam hasil *stemming*. Diagram Alir *stemming* ditunjukkan pada Gambar 4.12.

4.1.2.7 Tahapan Proses *Raw Term Frequency*

Raw Term Frequency, nilai *Term Frequency (TF)* diberikan berdasarkan jumlah kemunculan suatu term dalam dokumen. Proses ini dimulai ketika menerima masukan berupa hasil *pre-processing* teks, proses selanjutnya adalah akan menghitung jumlah kemuculan kata/*term* sampai semua dokumen dilakukan proses jumlah kemunculan kata. Diagram Alir *Raw Term Frequency* ditunjukkan pada Gambar 4.13.



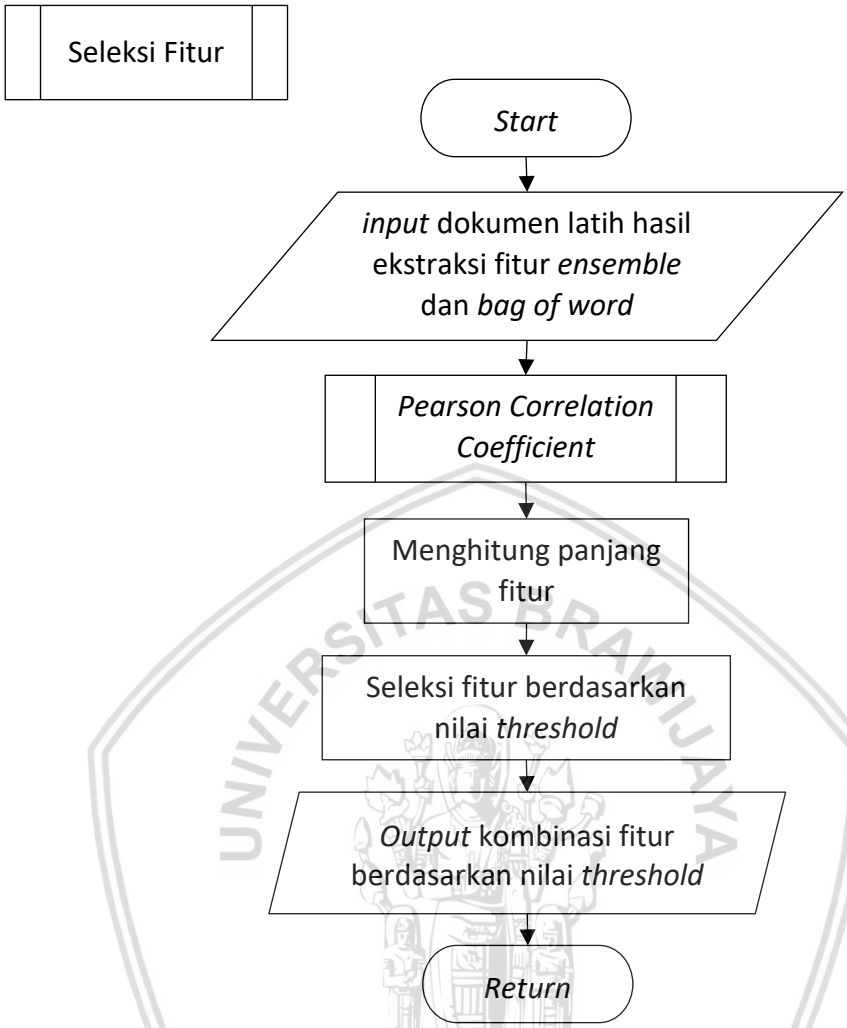
Gambar 4.12 Diagram Alir *Stemming*



Gambar 4.13 Diagram Alir *Raw Term Frequency*

4.1.3 Tahapan Proses Seleksi Fitur

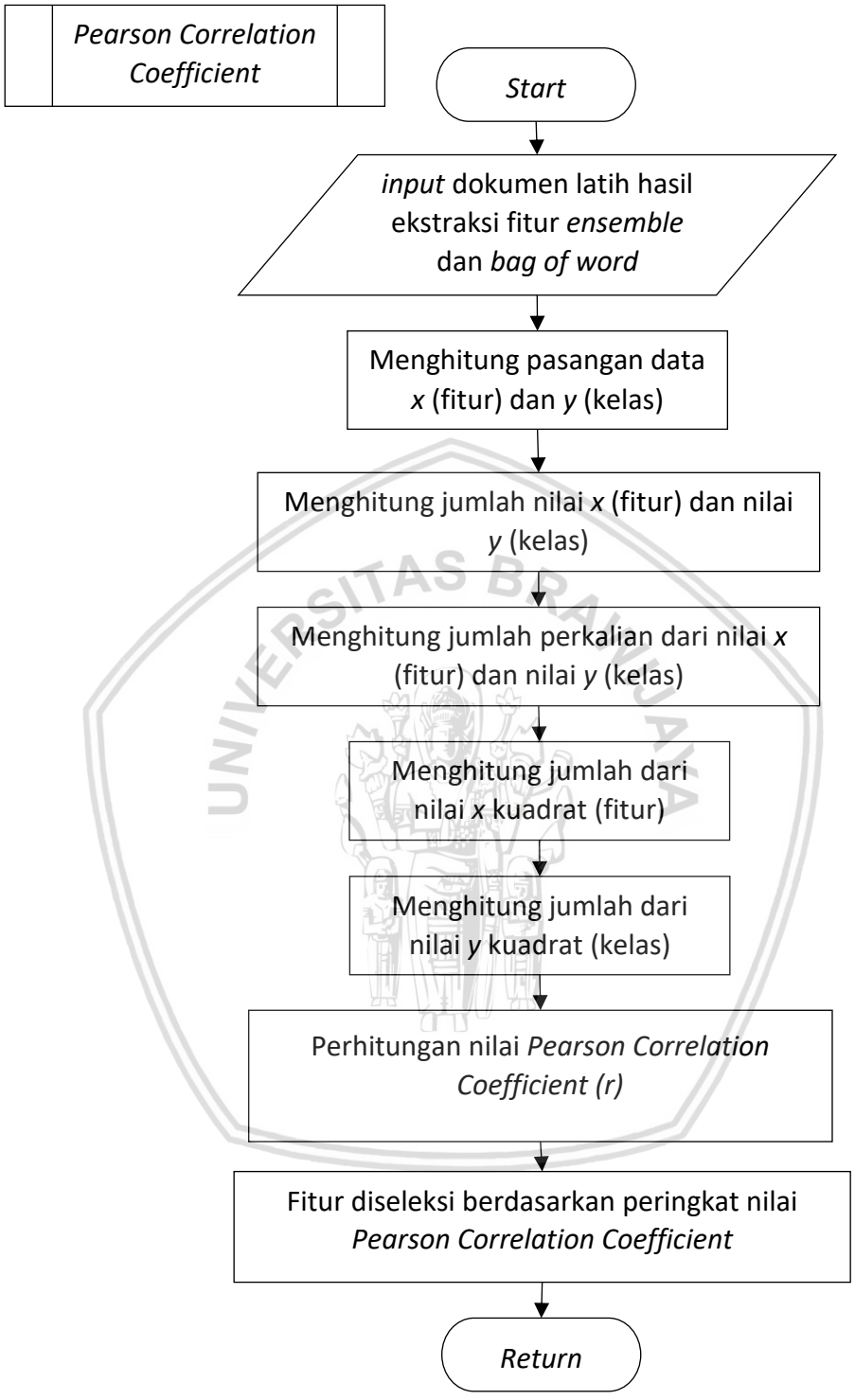
Tahapan Seleksi Fitur merupakan tahapan yang dilakukan setelah semua fitur *ensemble* dan *Bag of Words* selesai dilakukan, tahapan ini digunakan untuk mendapatkan kombinasi fitur yang optimal dan relevan pada setiap dokumen. Metode yang digunakan untuk seleksi fitur ini adalah *Pearson Correlation Coefficient*. Diagram Alir Seleksi Fitur ditunjukkan pada Gambar 4.14.



Gambar 4.14 Diagram Alir Seleksi Fitur

4.1.3.1 *Tahapan Proses Pearson Correlation Coefficient*

Tahapan *Pearson Correlation Coefficient* merupakan metode yang digunakan untuk menyeleksi fitur – fitur yang digunakan sehingga mendapatkan fitur – fitur yang optimal dan relevan. Selain itu, *Pearson Correlation Coefficient* merupakan salah satu metode yang paling sederhana untuk memahami hubungan fitur dengan variabel, yang mengukur korelasi linier antara dua variabel Diagram Alir *Pearson Correlation Coefficient* ditunjukkan pada Gambar 4.14.

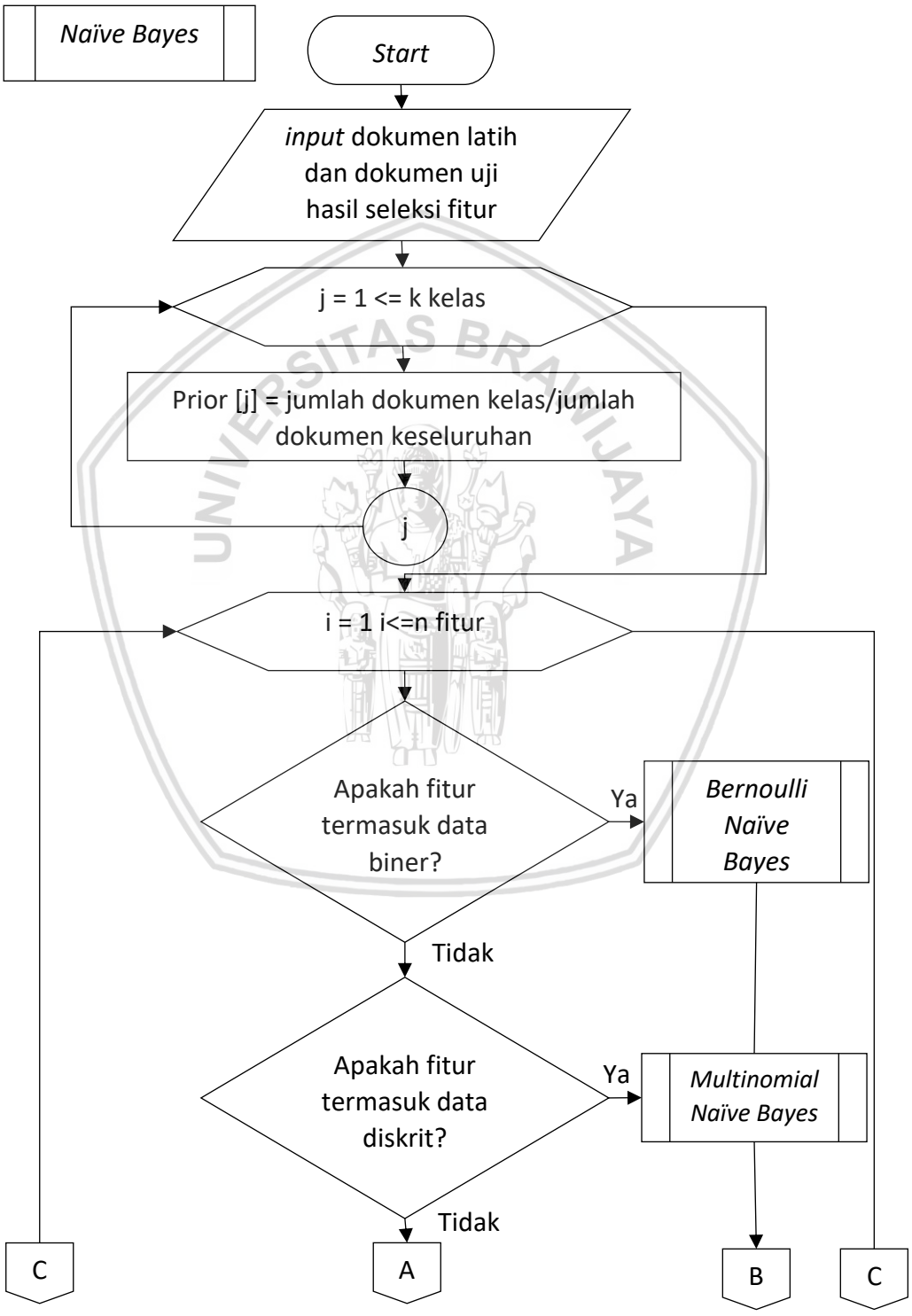


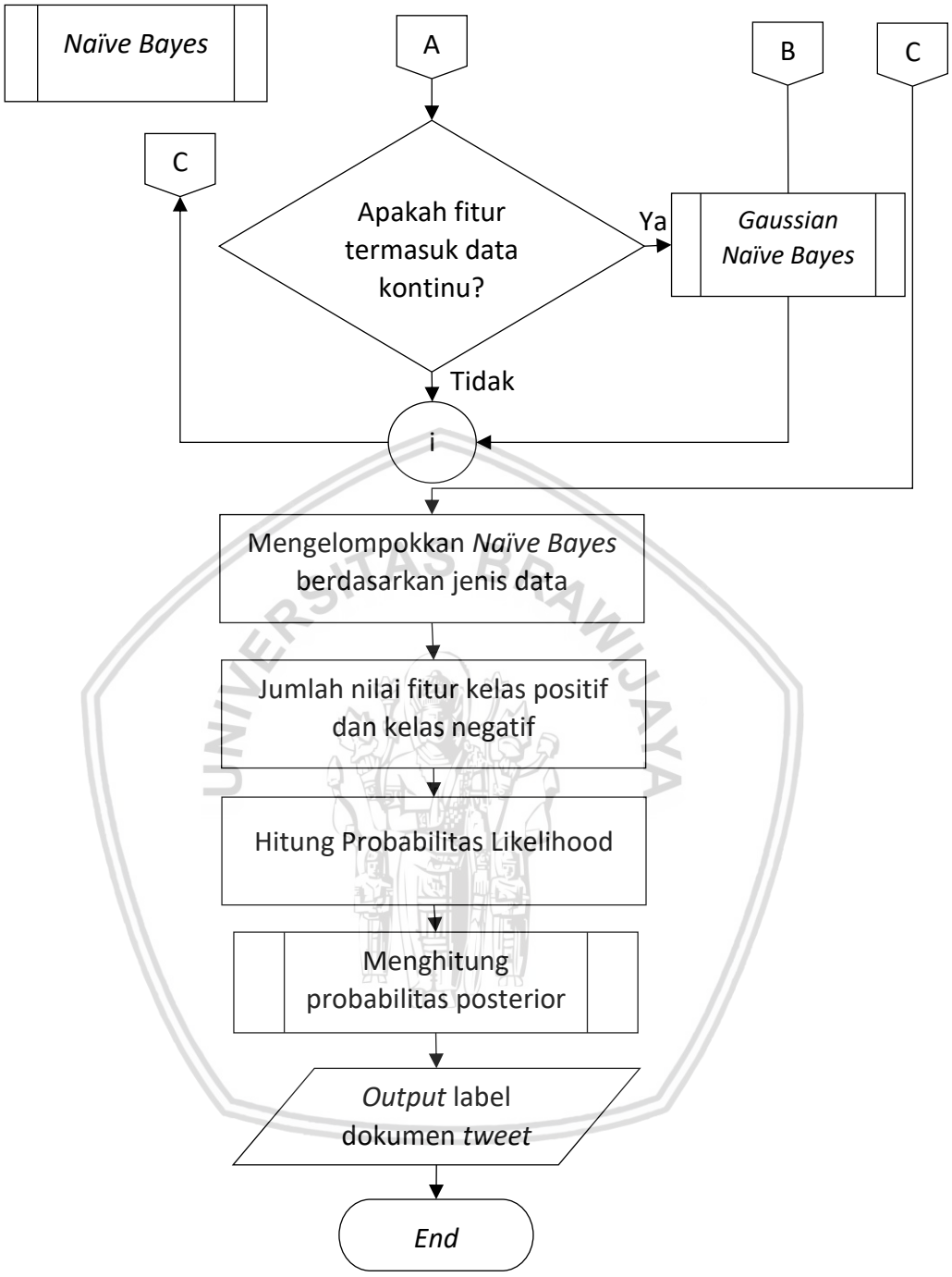
Gambar 4.15 Diagram Alir *Pearson Correlation Coefficient*

4.1.4 Tahapan Proses *Naïve Bayes*

Tahapan proses *Naïve Bayes* dimulai dengan memasukkan data uji yang diperoleh dari ekstraksi fitur *ensemble* dan *Bag of Word*. Selanjutnya menghitung prior kemudian melakukan pengelompokkan klasifikasi *Naïve Bayes* berdasarkan

jenis datanya, selanjutnya perhitungan probabiltias masing-masing fitur untuk kelas positif dan kelas negatif (likelihood). Hasil likelihood digunakan untuk menghitung posterior dengan cara mengalikannya dengan prior. Label kelas dengan nilai posterior terbesar akan menjadi label dari data yang baru. Diagram Alir *Naïve Bayes* ditunjukkan pada Gambar 4.15.

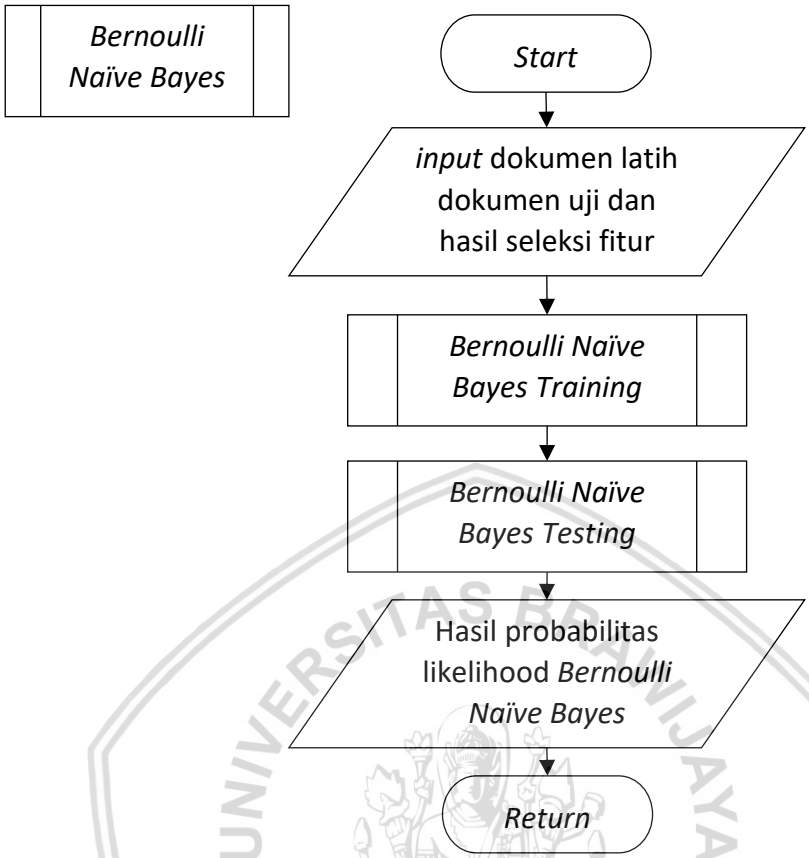




Gambar 4.16 Diagram Alir *Naive Bayes*

4.1.4.1 Tahapan Proses *Bernoulli Naïve Bayes*

Tahapan Proses *Bernoulli Naïve Bayes* dimulai dengan memasukkan hasil ekstraksi fitur dokumen latih, dokumen uji dan fitur hasil dari seleksi fitur *Pearson's*. Pada tahapan *Bernoulli Naïve Bayes* dibagi menjadi 2 yaitu: proses *traning* dan proses testing. Diagram Alir *Bernoulli Naïve Bayes* ditunjukkan pada Gambar 4.16.



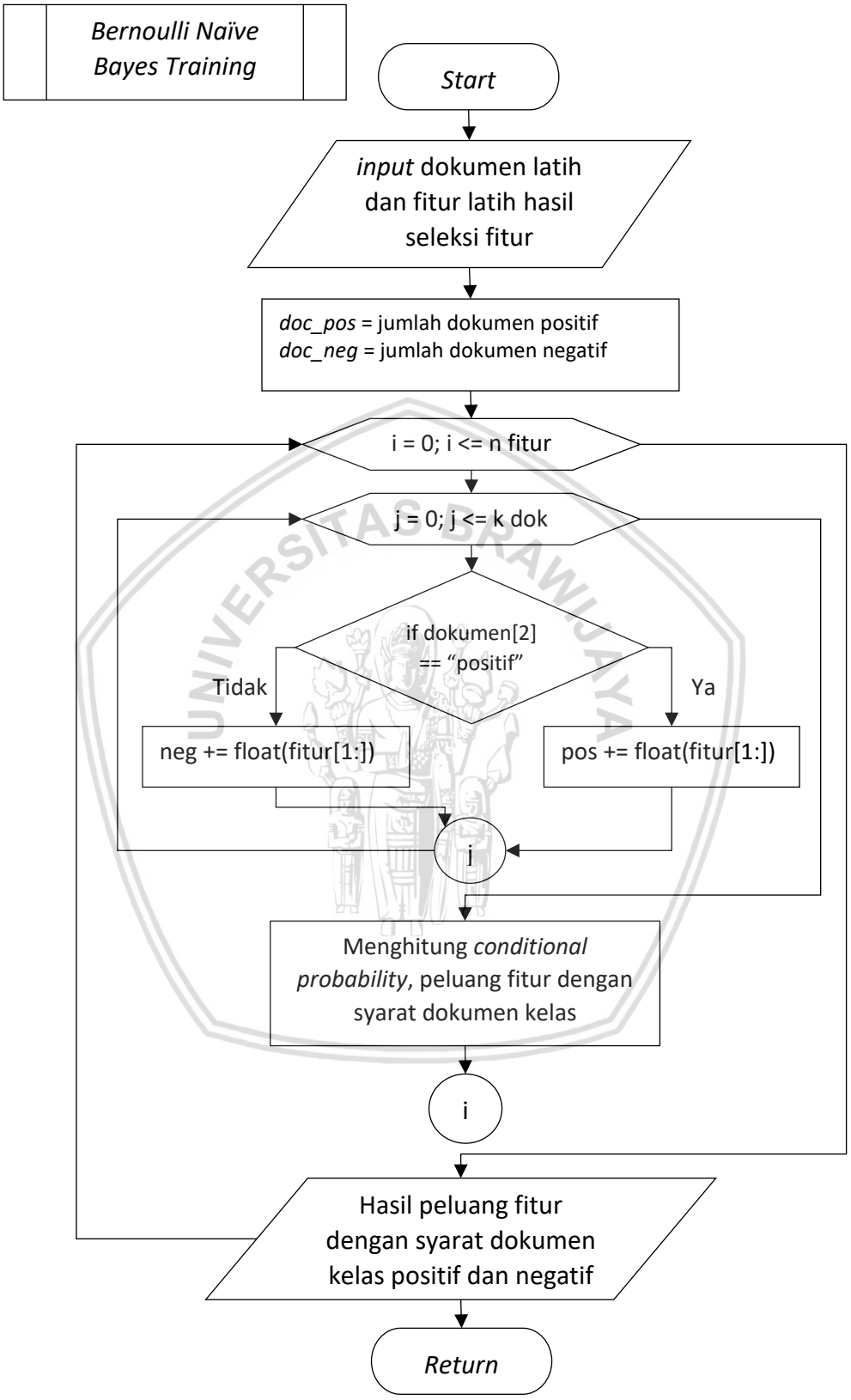
Gambar 4.17 Diagram Alir *Bernoulli Naive Bayes*

4.1.4.2 Tahapan Proses *Bernoulli Naïve Bayes Training*

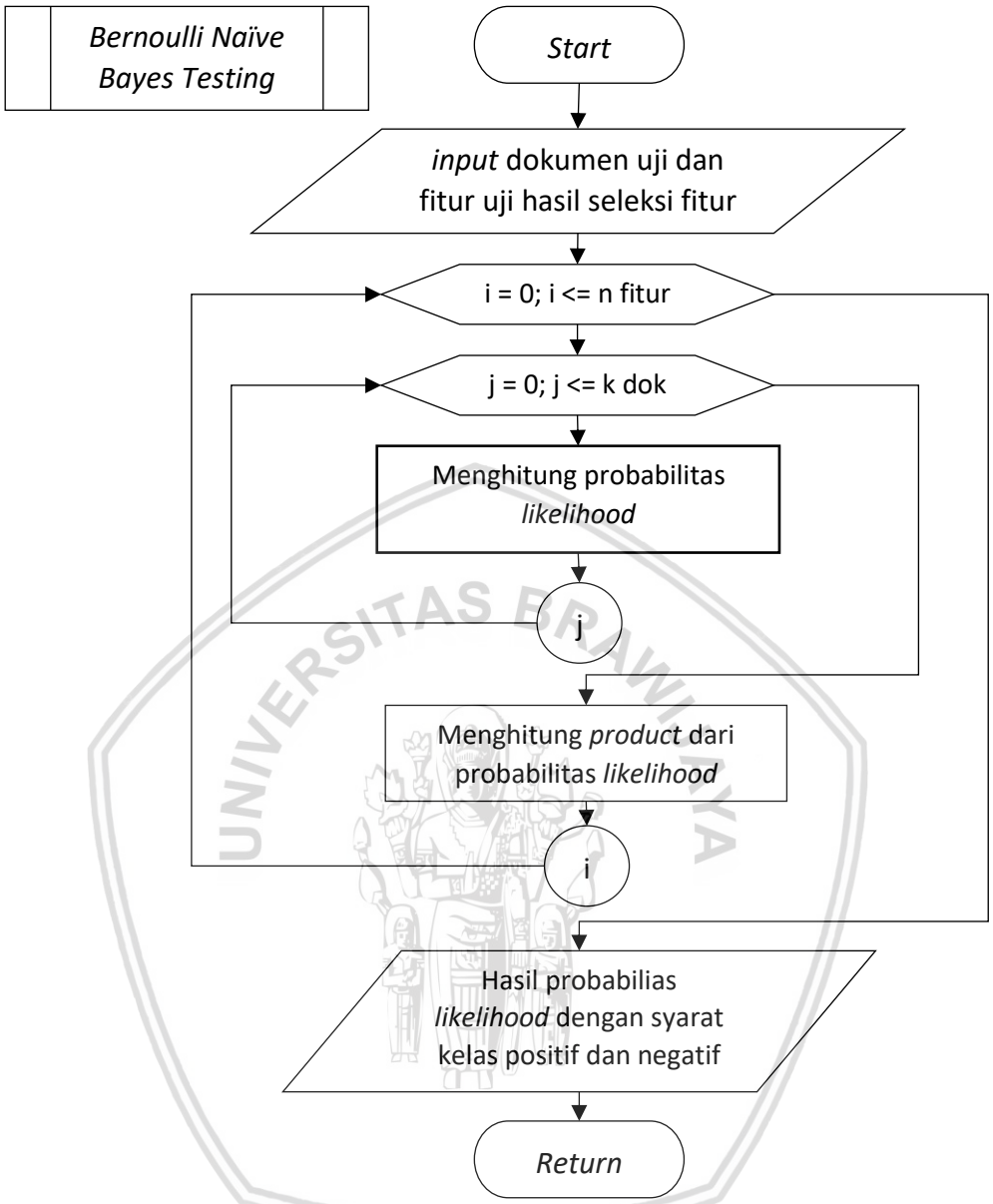
Tahapan Proses *Bernoulli Naïve Bayes Training* dimulai dengan memasukkan semua hasil fitur latih terseleksi berdasarkan *threshold* seleksi fiturnya. Selanjutnya menghitung jumlah seluruh fitur berdasarkan kelas positif dan negatif. Selanjutnya melakukan perhitungan *conditional probability* pada *Bernoulli Naïve Bayes*. Diagram Alir *Bernoulli Naïve Bayes Training* ditunjukkan pada Gambar 4.18.

4.1.4.3 Tahapan Proses *Bernoulli Naïve Bayes Testing*

Tahapan Proses *Bernoulli Naïve Bayes Testing* dimulai dengan memasukkan semua hasil fitur uji terseleksi berdasarkan *threshold* seleksi fiturnya. Selanjutnya menghitung nilai probabilitas *likelihood* pada masing – masing fitur uji. Selanjutnya nilai probabilitas *likelihood* setiap dokumen dan melakukan perkalian (*product*) pada setiap dokumen uji dengan syarat kelas positif dan negatif. *Diagram* Alir *Bernoulli Naïve Bayes Testing* ditunjukkan pada Gambar 4.19.



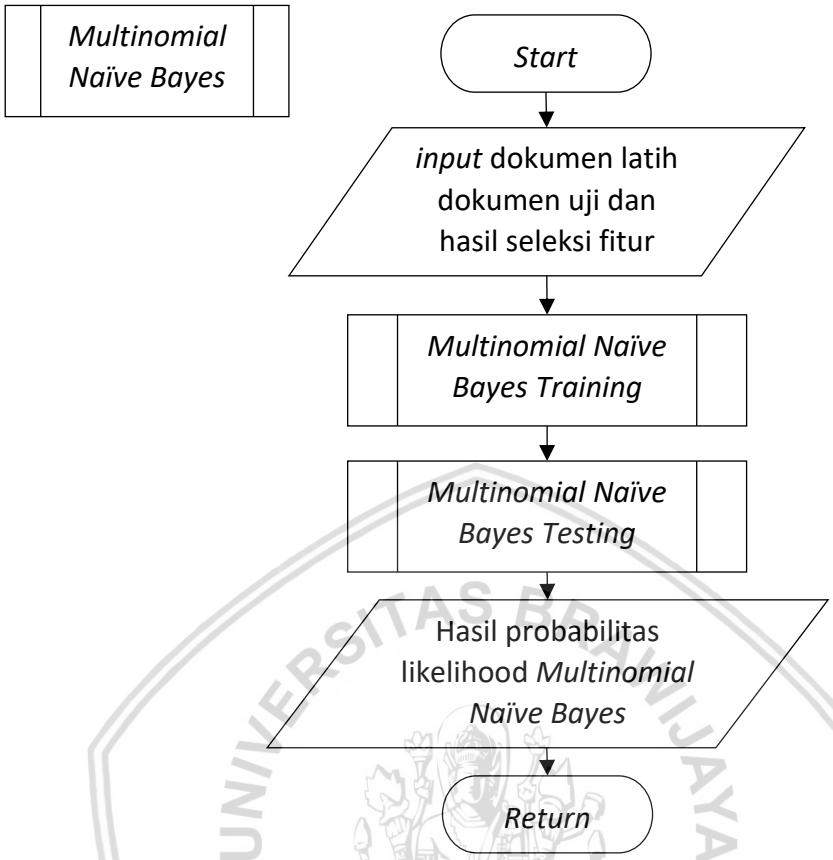
Gambar 4.18 Diagram Alir *Bernoulli Naïve Bayes Training*



Gambar 4.19 Diagram Alir *Bernoulli Naïve Bayes Testing*

4.1.4.4 Tahapan Proses *Multinomial Naïve Bayes*

Tahapan Proses *Multinomial Naïve Bayes* dimulai dengan memasukkan hasil ekstraksi fitur dokumen latih, dokumen uji dan fitur hasil dari seleksi fitur *Pearson's*. Pada tahapan *Multinomial Naïve Bayes* dibagi menjadi 2 yaitu: proses *traning* dan proses testing. Diagram Alir *Multinomial Naïve Bayes* ditunjukkan pada Gambar 4.20.



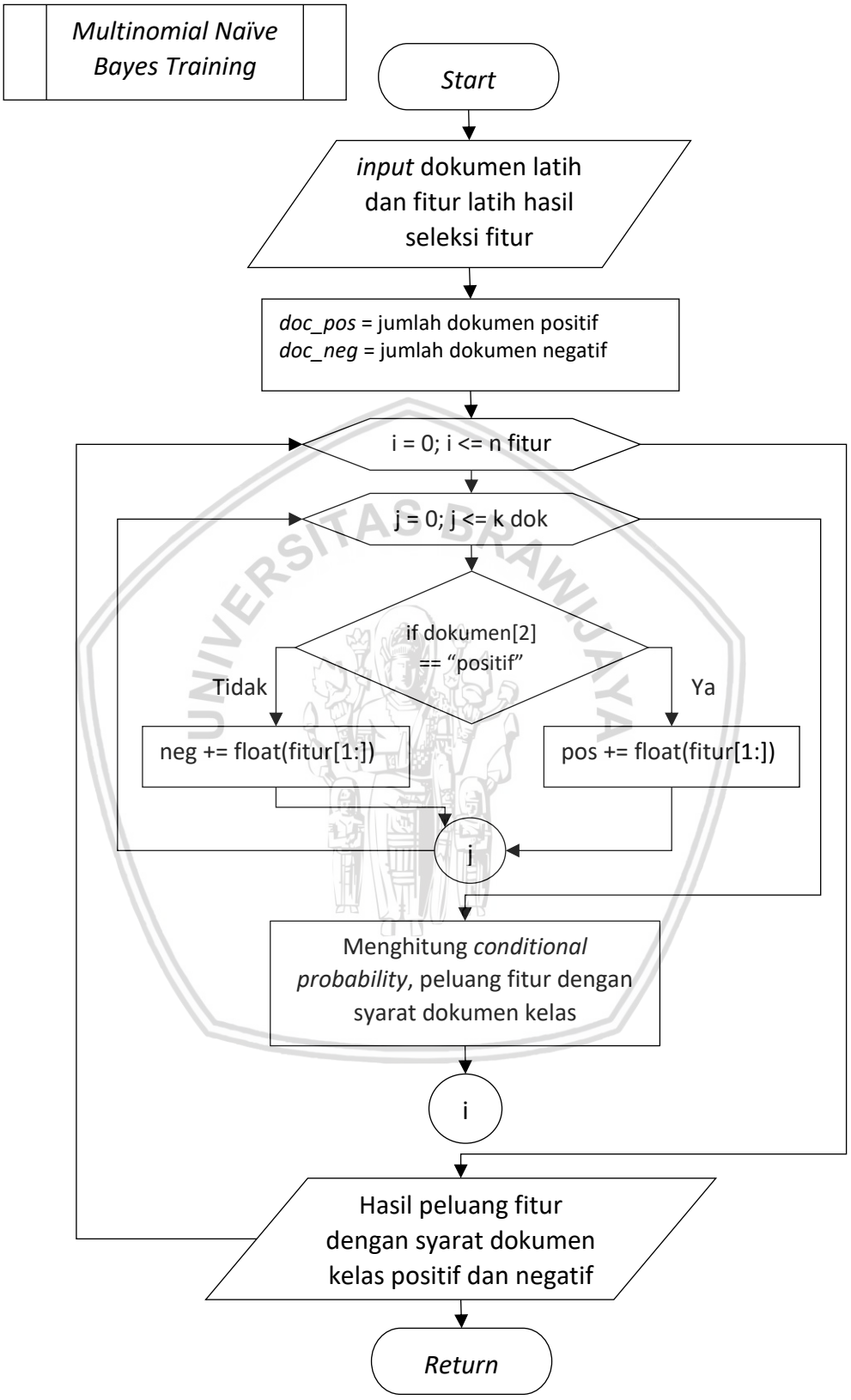
Gambar 4.20 Diagram Alir *Multinomial Naive Bayes*

4.1.4.5 Tahapan Proses *Multinomial Naïve Bayes Training*

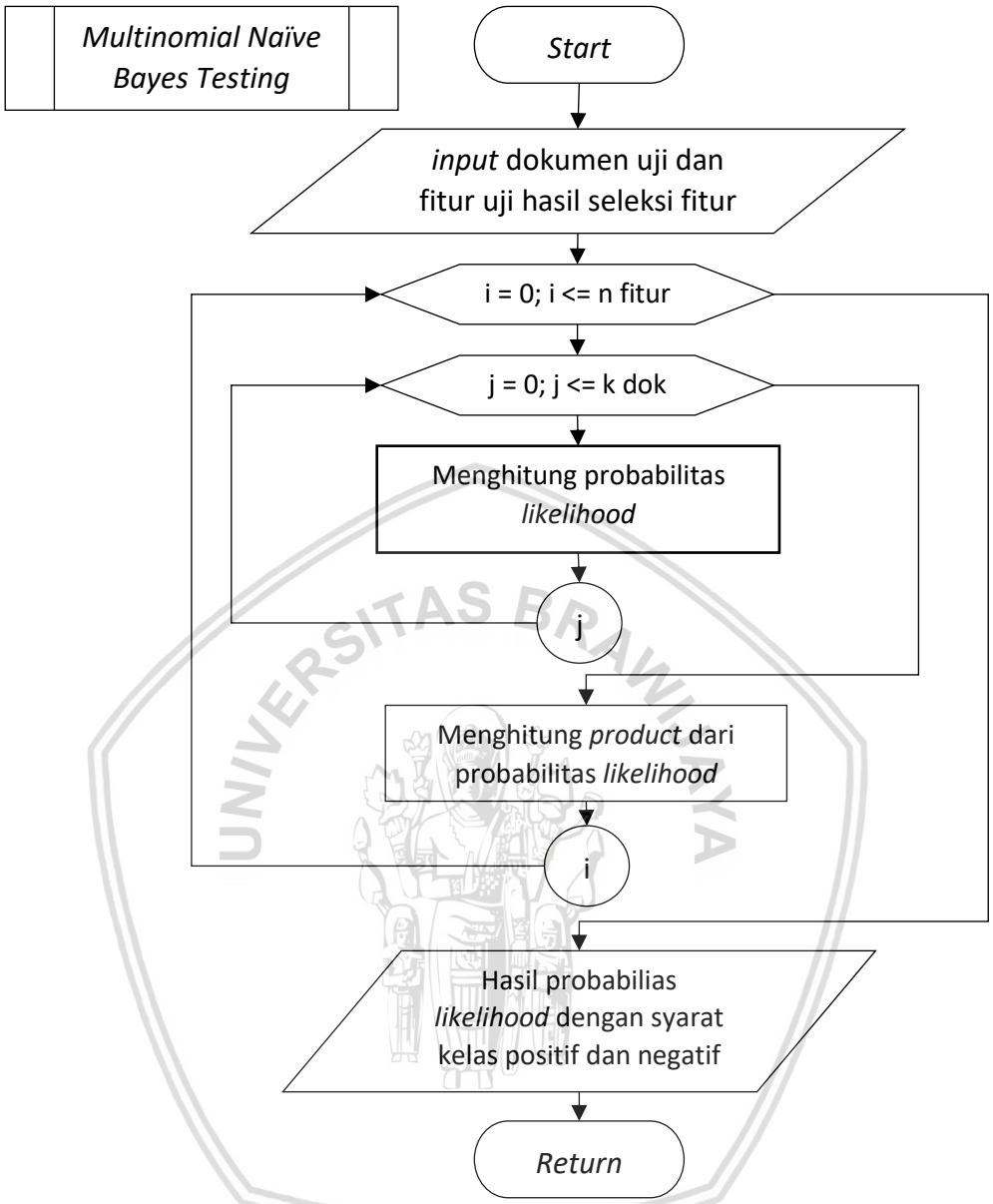
Tahapan Proses *Multinomial Naïve Bayes Training* dimulai dengan memasukkan semua hasil fitur latih terseleksi berdasarkan *threshold* seleksi fiturnya. Selanjutnya menghitung jumlah seluruh fitur berdasarkan kelas positif dan negatif. Selanjutnya melakukan perhitungan *conditional probability* pada *Multinomial Naïve Bayes*. Diagram Alir *Multinomial Naïve Bayes Training* ditunjukkan pada Gambar 4.21.

4.1.4.6 Tahapan Proses *Multinomial Naïve Bayes Testing*

Tahapan Proses *Multinomial Naïve Bayes Testing* dimulai dengan memasukkan semua hasil fitur uji terseleksi berdasarkan *threshold* seleksi fiturnya. Selanjutnya menghitung nilai probabilitas *threshold* pada masing – masing fitur uji. Selanjutnya nilai probabilitas *likelihood* setiap dokumen dan melakukan perkalian (*product*) pada setiap dokumen uji dengan syarat kelas positif dan negatif. *Diagram* Alir *Multinomial Naïve Bayes Testing* ditunjukkan pada Gambar 4.22.



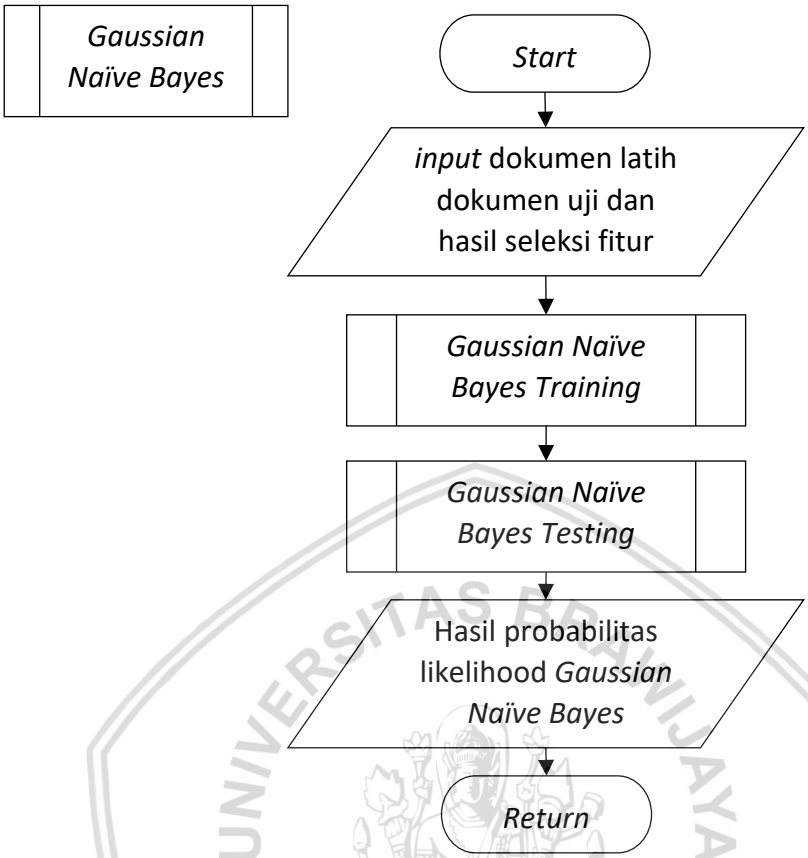
Gambar 4.21 Diagram Alir *Multinomial Naïve Bayes Training*



Gambar 4.22 Diagram Alir *Multinomial Naïve Bayes Testing*

4.1.4.7 Tahapan Proses *Gaussian Naïve Bayes*

Tahapan Proses *Gaussian Naïve Bayes* dimulai dengan memasukkan hasil ekstraksi fitur dokumen latih, dokumen uji dan nilai fitur hasil dari seleksi fitur *Pearson's*. Pada tahapan *Gaussian Naïve Bayes* dibagi menjadi 2 yaitu: proses *traning* dan proses testing. Diagram Alir *Gaussian Naïve Bayes* ditunjukkan pada Gambar 4.23.



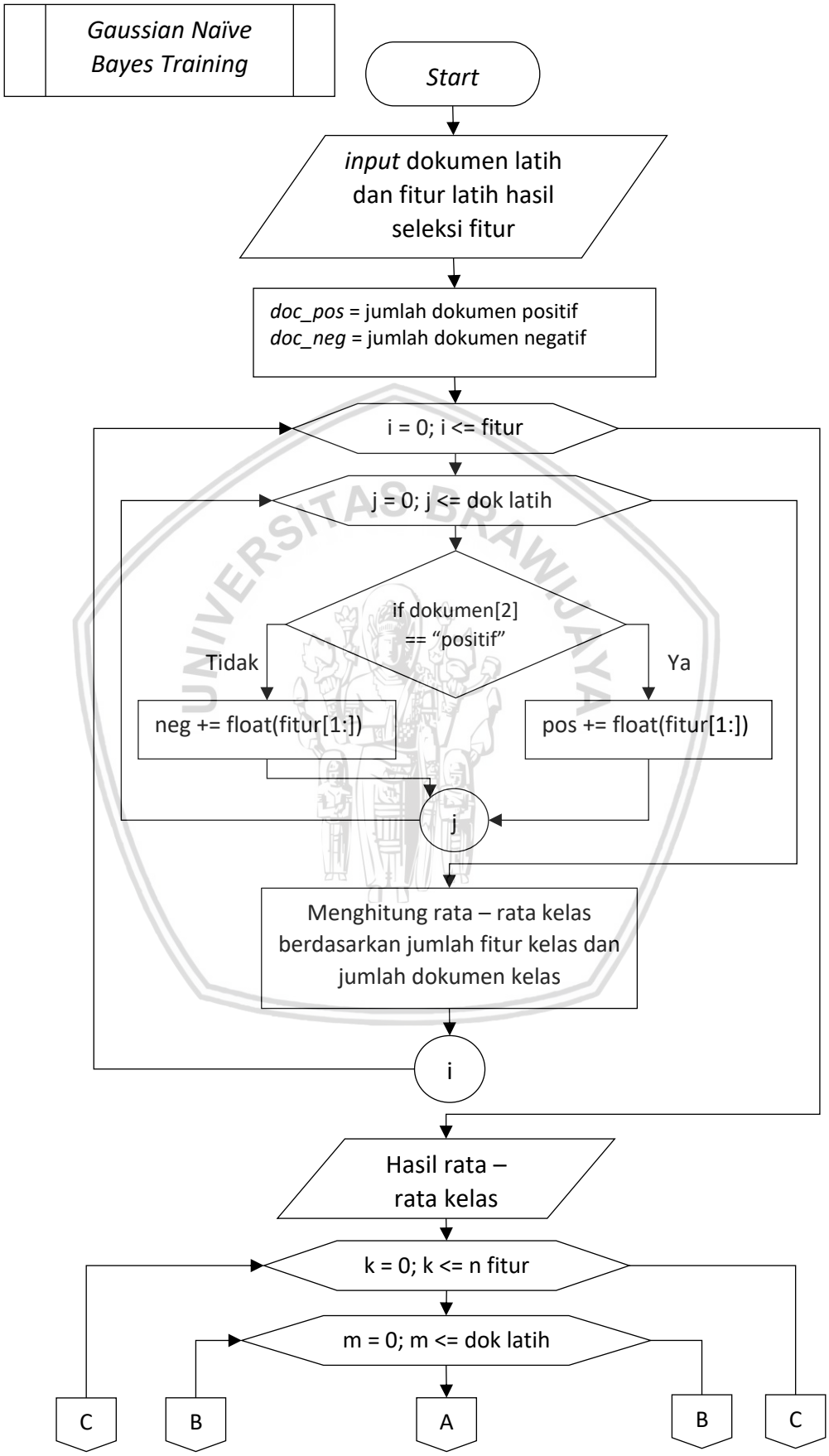
Gambar 4.23 Diagram Alir *Gaussian Naive Bayes*

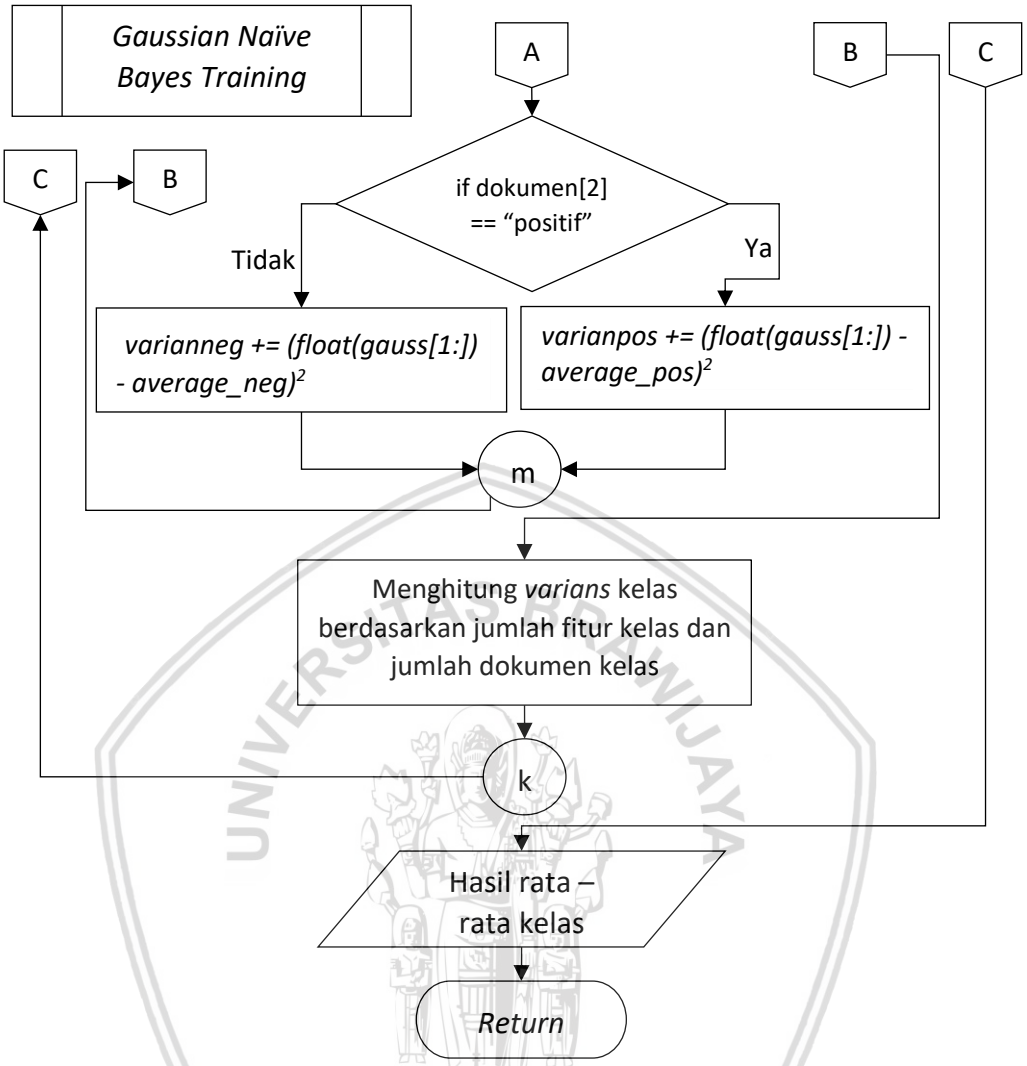
4.1.4.8 Tahapan Proses *Gaussian Naïve Bayes Training*

Tahapan Proses *Gaussian Naïve Bayes Training* dimulai dengan memasukkan semua hasil fitur latih terseleksi berdasarkan *threshold* seleksi fiturnya. Selanjutnya menghitung nilai rata – rata dan nilai *varians* berdasarkan kelas positif dan negatif sehingga didapat nilai rata – rata dan nilai *varians* pada setiap kelas. Diagram Alir *Gaussian Naïve Bayes Training* ditunjukkan pada Gambar 4.24.

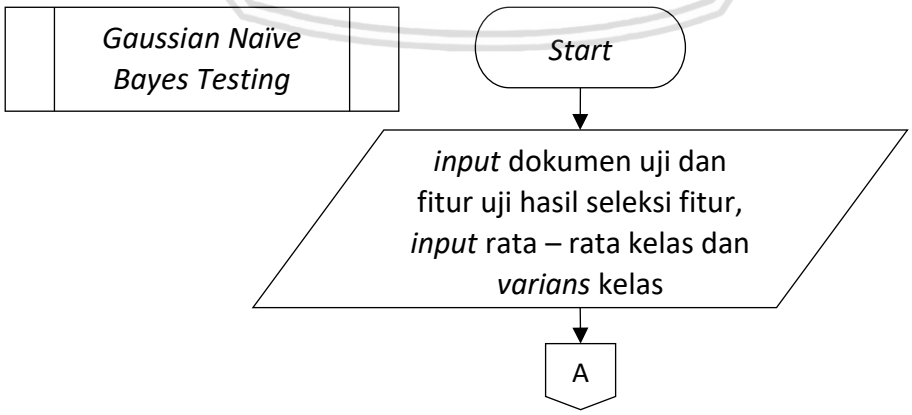
4.1.4.9 Tahapan Proses *Gaussian Naïve Bayes Testing*

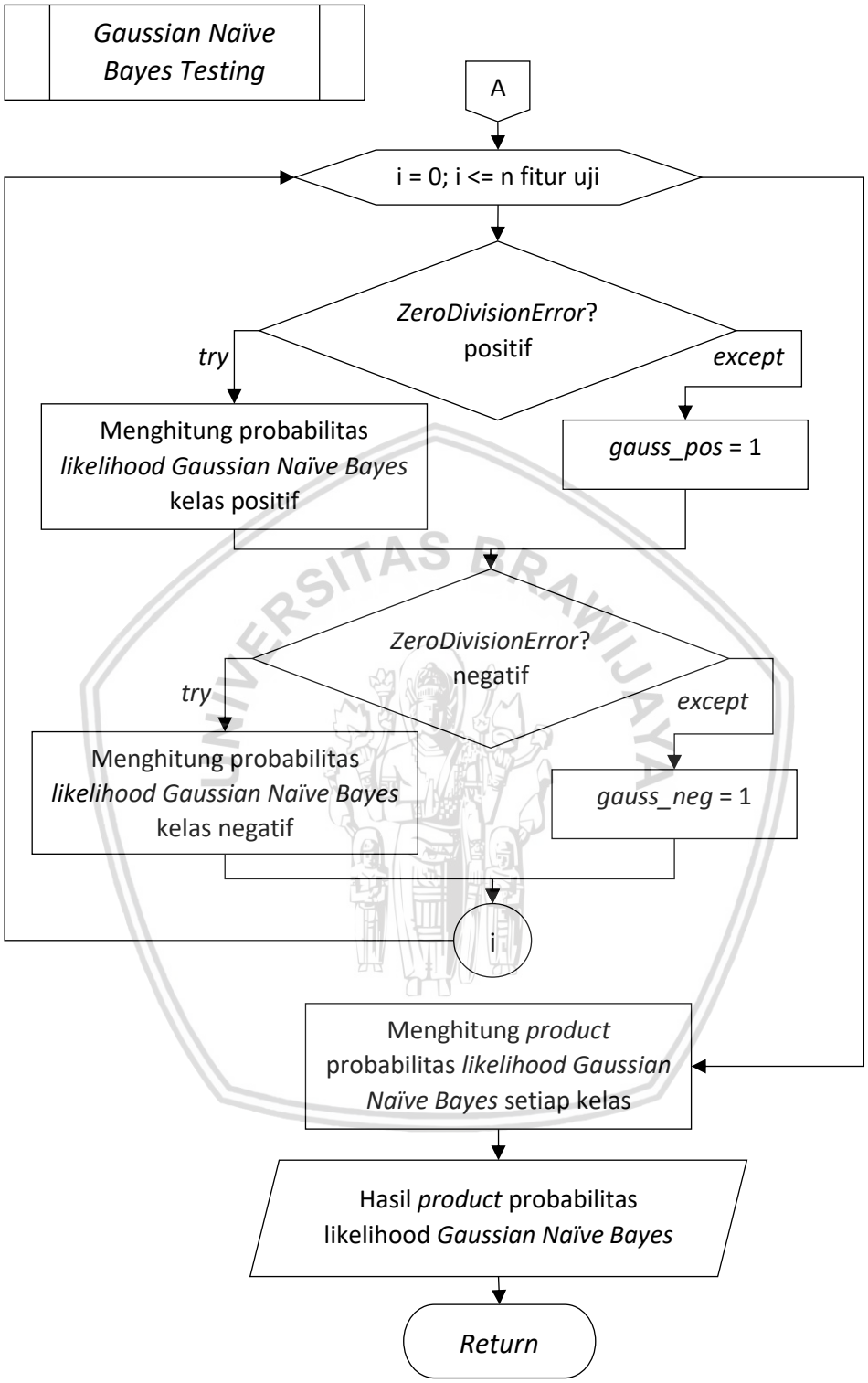
Tahapan Proses *Gaussian Naïve Bayes Testing* dimulai dengan memasukkan semua hasil fitur uji terseleksi berdasarkan *threshold* seleksi fiturnya. Selanjutnya menghitung nilai probabilitas *likelihood* pada masing – masing fitur uji. Selanjutnya nilai probabilitas *likelihood* setiap dokumen akan dilakukan perkalian (*product*) pada setiap dokumen uji dengan syarat kelas positif dan negatif. *Diagram* Alir *Gaussian Naïve Bayes Testing* ditunjukkan pada Gambar 4.25.


Gaussian Naïve Bayes Training
Start
input dokumen latih dan fitur latih hasil seleksi fitur
doc_pos = jumlah dokumen positif
doc_neg = jumlah dokumen negatif
i = 0; i <= fitur
j = 0; j <= dok latih
if dokumen[2] == "positif"
Tidak
Ya
neg += float(fitur[1:])
pos += float(fitur[1:])
j
Menghitung rata – rata kelas berdasarkan jumlah fitur kelas dan jumlah dokumen kelas
i
Hasil rata – rata kelas
k = 0; k <= n fitur
m = 0; m <= dok latih
C
B
A
B
C



Gambar 4.24 Diagram Alir *Gaussian Naïve Bayes Training*

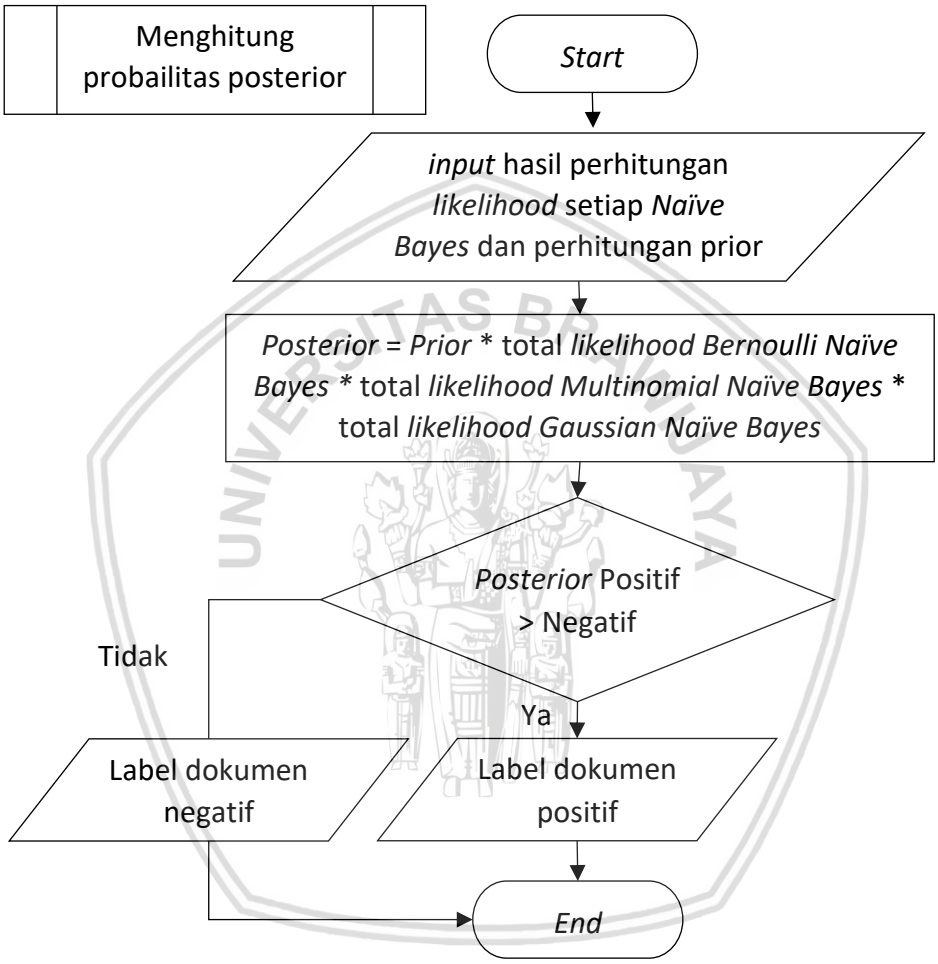




Gambar 4.25 Diagram Alir *Gaussian Naïve Bayes Testing*

4.1.4.10 Tahapan Proses Probabilitas *Posterior*

Tahapan proses untuk menghitung *posterior* dimulai dengan memasukkan hasil dari total perhitungan probabilitas *likelihood* setiap *Naïve Bayes* dan hasil perhitungan *prior*. Keluaran dari proses ini adalah kelas dari dokumen uji dengan membandingkan hasil nilai sentimen positif dan sentimmen negatif. Diagram Alir perhitungan Probabilitas *Posterior* ditunjukkan pada Gambar 4.26.



Gambar 4.26 Diagram Alir Probabilitas *Posterior*

4.2 Perhitungan Manual

Perhitungan manual pada proses analisis sentimen opini film menggunakan metode *Naïve Bayes* dengan *Ensemble Feature* dan Selesi fitur *Pearson Correlation Coefficient*. Berikut ini adalah langkah – langkah dalam melakukan proses perhitungan:

1. Melakukan ekstraksi fitur pada *ensemble feature*

   Proses pertama yang dilakukan adalah mengekstrak fitur pada *ensemble feature* sehingga didapatkan nilai pada setiap fitur. Pada tahap ini akan

dilakukan ekstraksi fitur pada setiap tipe fitur *ensemble* yaitu *twitter specific*, *textual features*, *part of speech features*, dan *lexicon-based features*.

2. Melakukan *Preprocessing*

   Preprocessing teks dilakukan setelah tahap ekstraksi fitur pada *ensemble* selesai untuk setiap tipe *ensemble*. Selanjutnya dilakukan *preprocessing* teks untuk membuat teks tidak terstruktur pada suatu dokumen menjadi terstruktur. Tahap *Preprocessing* bertujuan untuk mendapatkan kata – kata yang terstruktur sehingga dapat digunakan menjadi fitur pada *Bag of Word*.

3. Menghitung nilai *raw term frequency*

   Perhitungan *raw term frequency* dilakukan dengan menghitung jumlah kemunculan kata pada sebuah dokumen.

4. Menghitung nilai *Pearson Correlation Coefficient*

   Pearson Correlation Coefficient digunakan sebagai metode seleksi fitur untuk mendapatkan kombinasi fitur yang relevan. Nilai *Pearson's* didapaatkan dengan mengukur kekuatan antara dua variabel yaitu variabel *x* (fitur) dan variabel *y* (kelas). Selanjutnya akan didapatkan nilai *Pearson's* dan dilakukan proses pengurutan nilai dari terbesar sampai terkecil.

5. Melakukan Seleksi Fitur

   Proses seleksi Fitur dilakukan setelah semua nilai *Pearson's* pada fitur didapatkan. Proses Seleksi menggunakan beberapa *threshold* yaitu 10% sampai dengan 95%.

6. Pengelompokkan Naïve Bayes

   Melakukan pengelompokkan Naïve Bayes berdasarkan jenis datanya, untuk jenis data biner akan menggunakan *Bernoulli Naïve Bayes*, untuk jenis data diskrit akan menggunakan *Multinomial Naïve Bayes* dan jenis data kontinu menggunakan *Gaussian Naïve Bayes*.

7. Menghitung prior masing-masing kelas

   Proses menghitung prior dilakukan dengan cara menghitung total masing - masing label kelas pada data latih dan membaginya dengan total data latih.

8. Menghitung likelihood

   Proses ini merupakan perhitungan probabilitas masing - masing fitur dengan mengelompokkan jumlah kelas positif dan kelas negatif pada masing – masing fitur, untuk *Gaussian Naïve Bayes* menghitung rata – rata positif dan negatif, menghitung nilai varian positif dan negatif.

9. Menghitung posterior
   Perhitungan posterior dilakukan dengan cara mengalikan likelihood dengan prior berdasarkan jenis data dan metode *Naïve Bayes* yang digunakan.

Selanjutnya akan dilakukan perbandingan antar nilai posterior, label kelas dengan nilai posterior terbesar akan menjadi label untuk data yang baru.

4.2.1 *Dataset*

Dataset ini merupakan kumpulan data yang akan digunakan yaitu, terdapat data latih dan data uji. Pada contoh perhitungan kali ini, terdapat 10 data latih yang akan digunakan, contoh 10 data latih dapat dilihat pada Tabel 4.1 dan 5 data uji pada Tabel 4.2.

Tabel 4.1 Contoh Data Latih

ID	*Tweet*	Kelas
D1	Penasaran! Tonton deh besok RT @detikcom : Menjelajah Angkasa dalam Film "Gravity" #Gravity http://detik.com/tErYg via @detikhot	POSITIF
D2	#MovieReview Gravity . Buat yang suka film dengan plot yang berat dan action yang banyak, jangan nonton inilah. Pemain nya aja sedikit benar bagaimana mau ramai	NEGATIF
D3	Review #Gravity : film ini menceritakan wanita yang terjebak di luar angkasa dan dia harus berusaha untuk kembali ke bumi, sendirian!! Seru :-D"	POSITIF
D4	Tapi aku tetap mempertanyakan rating tinggi nya itu, karena dari kacamata orang awam film ini flat dan bikin bosan. Iya, terlalu realistis. #Gravity	NEGATIF
D5	"Gravity" Film yang terlalu sederhana untuk ukuran rating tinggi. Dan sampai saat ini aku masih bingung bagian mana nya yang jadi acuan rating. --"	NEGATIF
D6	Sepertinya " Gravity " bakal jadi film sci-fi paling epik tahun ini. Bahkan dapat #CertifiedFresh 98% RottenTomatoes dan 88% IMDb. :)	POSITIF
D7	Boleh juga ini RT @TraxFMJKT : Film GRAVITY sudah mulai tayang loh di Indonesia! Siapa yang sudah nonton?! #NewKompakKampus https://pic.twitter.com/pOladBbc2o	POSITIF
D8	Ya ampun ini film tidak benar banget.. Enyes-enyes. Tips harus 3d dan sedikit tips akan jadi â€¦ â˜… Gravity (at @Cinema21) â€” https://path.com/p/44OXc0	NEGATIF
D9	Ada 3D nya tidak kak? RT @realtamiii : Film Gravity itu surga buat orang seperti saya. Bisa melihat isi ISS sama Soyuz nya walaupun tidak asli. Tapi â€¦ https://twitter.com/ardictatian/status/387227902833475584	POSITIF
D10	@benakribo film nya agak membosankan karena pemain nya cuma 2 orang, cerita nya juga kurang greget. #Gravity :-& RT @benakribo: Film Gravity sepertinya keren bangetâ€¦..	NEGATIF

Tabel 4.2 Contoh Data Uji

ID	*Tweet*	Kelas
D11	RT @ulil : Sudah ada yang menonton film "Gravity"? Tubuh Sandra Bullock kok masih bagus banget ya? Minum jamu apa ya? :)" "temulawak" bagus :-) https://twitter.com/M_NatsirU/status/389298672145412096 #Gravity	POSITIF
D12	Bikin sesak napas juga hahaha tapi keren sandra bullock nya RT @juuwita : Wah film gravity ramai bukan? Wah wah wah wah	NEGATIF
D13	Habis menonton gravity .... Langsung cabut lagi lah, emang ada yang peduli? Gak ada kan... Huh huh huh film nya keren abis!!ï¿½ #Gravity	POSITIF
D14	Film nya bikin pintar karena jadi tahu bagaimana sebenarnya di luar angkasa itu ;);) RT @jflowrighthere : Lagi ada film-film menarik , Gravity , Prisoners, MSS nya @radityadika , hem #apalagi ya?	POSITIF
D15	Sudah tonton film Gravity tapi menurut aku kurang seru ... biasa aja yang main aja sih oke, sandra bullock si seksi dan cantik :*... hem @GrahaCijtung #Gravity	NEGATIF

4.2.2 *Ensemble Feature*

Pada tahap ini dilakukan manualisasi ekstraksi fitur pada fitur *ensemble* yaitu *twitter specific features, textual features, Parts of Speech (PoS) features,* dan *lexicon -based features*.

4.2.2.1 *Twitter Specific*

Pada tahap ini dilakukan proses pengambilan nilai fitur F1 – F4, untuk nilai 1 menyatakan fitur tersebut terdapat pada dokumen *tweet* dan nilai 0 menyatakan fitur tersebut tidak terdapat pada *tweet*. F1 menyatakan apakah pada *tweet* terdapat *hastag* (#) atau tidak pada tweet, F2 menyatakan apakah terdapat *retweet* (RT) atau tidak, F3 menyatakan apakah terdapat *username* (@) atau tidak dan F4 menyatakan apakah terdapat URL atau tidak pada *tweet*. Hasil pengambilan nilai pada *twitter specific* ditunjukkan pada Tabel 4.3.

Tabel 4.3 Perhitungan Manual *Twitter Specific*

Fitur	Dokumen														
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F1	1	1	1	1	0	1	1	0	0	1	1	0	1	1	1
F2	1	0	0	0	0	0	1	0	1	1	1	1	0	1	0
F3	1	0	0	0	0	0	1	1	1	1	1	1	0	1	1
F4	1	0	0	0	0	0	1	1	1	0	1	0	0	0	0

4.2.2.2 *Textual Features*

Proses pengambilan nilai fitur F5 – F12, pada tipe fitur ini akan dilakukan pengambilan fitur berdasarkan informasi eksplisit pada sebuah *tweet*. F5 menyatakan panjang *tweet* yang didapatkan dari jumlah kata pada dokumen *tweet*, F6 menyatakan rata – rata panjang *tweet* yang didapatkan dari hasil pembagian antara jumlah karakter pada *tweet* dibagi dengan jumlah kata yang terdapat pada *tweet*, F7 menyatakan jumlah tanda tanya pada *tweet*, F8 menyatakan jumlah tanda seru pada *tweet*, F9 menyatakan jumlah tanda kutip pada *tweet*, F10 menyatakan jumlah kata yang diawali dengan huruf besar atau huruf kapital, F11 menyatakan apakah terdapat *emoticon* positif atau tidak dan F12 menyatakan apakah terdapat *emoticon* negatif atau tidak . Hasil pengambilan nilai pada *textual features* ditunjukkan pada Tabel 4.4.

Tabel 4.4 Perhitungan Manual *Textual Features*

Fitur	Dokumen														
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F5	14	25	22	23	23	16	18	19	26	23	23	21	22	27	25
F6	8.06	6.15	6.26	6.43	6.12	6.33	7.9	6.41	7.24	6.76	7.7	5.41	5.91	6.09	5.89
F7	0	0	0	0	0	0	1	0	1	0	3	1	1	1	0
F8	1	0	2	0	0	0	2	0	0	0	0	0	2	0	0
F9	1	0	0	0	1	1	0	0	0	0	2	0	0	0	0
F10	8	4	3	3	3	6	8	5	8	4	8	4	5	6	4
F11	0	0	1	0	0	1	0	0	0	0	1	0	0	1	1
F12	1	0	0	0	1	0	1	1	1	1	1	0	0	0	0

4.2.2.3 *Part of Speech Features*

Proses pengambilan nilai fitur F13 – F22, pada tipe fitur ini akan dilakukan pengambilan fitur berdasarkan penandaan kata (*tagging*) pada sebuah *tweet*. F13 menyatakan jumlah kata benda pada dokumen *tweet*. F14 menyatakan jumlah kata sifat pada dokumen *tweet*. F15 menyatakan jumlah kata kerja pada dokumen *tweet*. F16 menyatakan jumlah kata keterangan pada dokumen *tweet*. F17 menyatakan jumlah kata seru pada dokumen *tweet*. Selanjutnya dilakukan perhitungan persentase dari setiap *tag* tersebut yaitu banyaknya *tag* kata pada fitur yang dimaksud (F18 – F22) dibagi dengan banyaknya seluruh kata dalam *tweet* (F5) selanjutnya dikalikan dengan 100%. F18 menyatakan persentase kata benda dalam *tweet,* F19 menyatakan persentase kata sifat*,* F20 menyatakan persentase kata kerja*,* F21 menyatakan persentase kata keterangan dan F22 menyatakan persentase kata seru dalam *tweet*. Hasil pengambilan nilai pada *Part of Speech* ditunjukkan pada Tabel 4.5.

Tabel 4.5 Perhitungan Manual *Part of Speech Features*

Fitur	Dokumen														
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F13	3	4	6	4	5	4	3	2	6	7	4	2	4	3	4
F14	2	5	0	4	3	0	0	2	1	1	2	2	1	1	2
F15	2	1	4	3	2	2	2	1	4	1	3	1	6	5	3
F16	0	2	1	1	2	2	4	3	2	4	3	2	2	2	2
F17	0	0	0	0	0	0	0	0	0	0	0	5	3	1	1
F18	0.21	0.16	0.27	0.17	0.22	0.25	0.17	0.11	0.23	0.3	0.17	0.09	0.18	0.11	0.16
F19	0.14	0.2	0	0.17	0.13	0	0	0.11	0.03	0.04	0.08	0.09	0.04	0.03	0.08
F20	0.14	0.04	0.18	0.13	0.08	0.12	0.11	0.05	0.15	0.04	0.13	0.04	0.27	0.18	0.12
F21	0	0.08	0.04	0.04	0.08	0.12	0.22	0.15	0.07	0.17	0.13	0.09	0.09	0.07	0.08
F22	0	0	0	0	0	0	0	0	0	0	0	0.24	0.13	0.03	0.04

4.2.2.4 *Lexicon based Features*

Proses pengambilan nilai fitur F23 – F37, pada tipe fitur ini akan dilakukan pengambilan fitur berdasarkan kamus/*lexicon* pada sebuah *tweet*. F23 menyatakan jumlah kata positif pada dokumen *tweet*. F24 menyatakan jumlah kata negatif pada dokumen *tweet*. F25 dan F26 menyatakan jumlah kata positif dan negatif pada *PoS* kata sifat. F27 dan F28 menyatakan jumlah kata positif dan negatif pada *PoS* kata kerja. F29 dan F30 menyatakan jumlah kata positif dan negatif pada *PoS* kata keterangan. Selanjutnya dilakukan perhitungan persentase dari setiap kata tersebut yaitu positif dan negatif pada setiap jenis kata (F25 – F30) dibagi dengan banyaknya seluruh kata positif dalam *tweet* (F23) dan kata negatif (F24) selanjutnya dikalikan dengan 100%. F31 dan F32 menyatakan presentase kata positif dan negatif pada *PoS* kata sifat. F33 dan F34 menyatakan presentase kata positif dan negatif pada *PoS* kata kerja. F35 dan F36 menyatakan presentase kata positif dan negatif pada *PoS* kata keterangan. Hasil pengambilan nilai pada *Lexicon based Features* ditunjukkan pada Tabel 4.6.

Tabel 4.6 Perhitungan Manual *Lexicon based Features*

Fitur	Dokumen														
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F23	0	2	1	1	2	0	0	1	3	1	2	6	2	3	4
F24	0	1	2	1	2	0	0	0	0	3	1	0	0	0	2
F25	0	2	0	1	2	0	0	1	0	1	2	1	1	1	1
F26	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1
F27	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
F28	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
F29	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
F30	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1
F31	0	1	0	1	1	0	0	1	0	1	1	0.17	0.5	0.33	0.25
F32	0	1	0	1	0.5	0	0	0	0	0	0	0	0	0	0.5
F33	0	0	0	0	0	0	0	0	0	0	0	0	0.5	0.33	0
F34	0	0	0.5	0	0	0	0	0	0	0.33	0	0	0	0	0

Tabel 4.6 Perhitungan Manual *Lexicon based Features* (lanjutan)

Fitur	Dokumen														
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F35	0	0	0	0	0	0	0	0	0	0	0	0	0	0.33	0
F36	0	0	0	0	0	0	0	0	0	0.67	0	0	0	0	0.5
F37	0	1	0	1	1	1	0	1	0	2	1	0	1	0	1

4.2.3 *Bag of Word* (*BoW*) *Features*

Pada langkah ini proses ekstraksi fitur dilakukan dengan mendapatkan nilai bobot kata dari dokumen *tweet*. Nilai bobot didapatkan dari banyaknya jumlah kemunculan kata pada sebuah dokumen dengan menggunakan *raw term frequency*. Untuk mendapatkan bobot kata tersebut akan dilakukan *preprocessing* terlebih dahulu pada *tweet. Preprocessing* meliputi *tokenization, cleaning, case folding, stopword removal* dan *stemming*. Untuk melakukan *preprocessing* akan mengambil 1 contoh dari dokumen tweet. Contoh proses *tokenization* yang dilakukan pada salah satu *tweet* ditunjukkan pada Tabel 4.7.

Tabel 4.7 Hasil *Tokenization*

Tokenization	
Input	*Output*
Review #Gravity: film ini menceritakan wanita yang terjebak di luar angkasa dan dia harus berusaha untuk kembali ke bumi, sendirian!! Seru :-D	“Review”, “#”, ”Gravity”, ”:”, “film”, “ini”, “menceritakan”, “wanita”, “yang”, “terjebak”, “di”, “luar”, “angkasa”, “dan”, “dia”, “harus”, “berusaha”, “untuk”, “kembali”, “ke”, “bumi”, “sendirian”, ”!”, ”!”, “Seru”, “:” , ”-D”

Proses yang dilakukan selanjutnya adalah melakukan *cleaning* pada dokumen *tweet*. Contoh proses *cleaning* ditunjukkan pada Tabel 4.8.

Tabel 4.8 Hasil *Cleaning*

Cleaning	
Input	*Output*
“Review”, “#”, ”Gravity”, ”:”, “film”, “ini”, “menceritakan”, “wanita”, “yang”, “terjebak”, “di”, “luar”, “angkasa”, “dan”, “dia”, “harus”, “berusaha”, “untuk”, “kembali”, “ke”, “bumi”, “sendirian”, ”!”, ”!”, “Seru”, “:” , ”-D”	“Review”,”Gravity”, “film”, “ini”, “menceritakan”, “wanita”, “yang”, “terjebak”, “di”, “luar”, “angkasa”, “dan”, “dia”, “harus”, “berusaha”, “untuk”, “kembali”, “ke”, “bumi”, “sendirian”, “Seru”

Proses yang dilakukan selanjutnya adalah melakukan *case folding* yaitu mengubah semua huruf pada dokumen menjadi huruf kecil pada dokumen *tweet*. Contoh proses *cleaning* ditunjukkan pada Tabel 4.9.

Tabel 4.9 Hasil *Case Folding*

Case Folding	
Input	*Output*
“Review”, ”Gravity”, “film”, “ini”, “menceritakan”, “wanita”, “yang”, “terjebak”, “di”, “luar”, “angkasa”, “dan”, “dia”, “harus”, “berusaha”, “untuk”, “kembali”, “ke”, “bumi”, “sendirian”, “Seru”	“review”, ”gravity”, “film”, “ini”, “menceritakan”, “wanita”, “yang”, “terjebak”, “di”, “luar”, “angkasa”, “dan”, “dia”, “harus”, “berusaha”, “untuk”, “kembali”, “ke”, “bumi”, “sendirian”, “seru”

Proses yang dilakukan selanjutnya adalah melakukan *stopword removal* yaitu menghilangkan kata yang tidak penting yang sesuai dengan *stoplist*. Contoh proses *cleaning* ditunjukkan pada Tabel 4.10.

Tabel 4.10 Hasil *Stopword Removal*

Stopword Removal	
Input	*Output*
“review”, ”gravity”, “film”, “ini”, “menceritakan”, “wanita”, “yang”, “terjebak”, “di”, “luar”, “angkasa”, “dan”, “dia”, “harus”, “berusaha”, “untuk”, “kembali”, “ke”, “bumi”, “sendirian”, “seru”	“review”, ”gravity”, “film”, “menceritakan”, “wanita”, “terjebak”, “angkasa”, “berusaha”, “bumi”, “seru”

Proses yang dilakukan selanjutnya adalah melakukan *stemming* yaitu semua kata berimbuhan akan dikembalikan kekata dasar. Contoh proses *cleaning* ditunjukkan pada Tabel 4.11.

Tabel 4.11 Hasil *Stemming*

Stemming	
Input	*Output*
“review”, ”gravity”, “film”, “menceritakan”, “wanita”, “terjebak”, “angkasa”, “berusaha”, “bumi”, “seru”	“review”, ”gravity”, “film”, “cerita”, “wanita”, “jebak”, “angkasa”, “usaha”, “bumi”, “seru”

Setelah dilakukan *preprocessing* diatas maka akan dilakukan perhitungan berikutnya yaitu *term frequency* merupakan perhitungan frekuensi kemunculan kata pada setiap *tweet.* Contoh jika terdapat kata film muncul 2 kali pada dokumen

tweet ke 1 maka nilai *term frequency* = 2. Perhitungan kemunculan kata ditunjukkan pada Tabel 4.12.

Tabel 4.12 Perhitungan *Manual Term Frequency*

Fitur	Dokumen														
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
abis	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
action	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
acu	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
ampun	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
angkasa	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0
asli	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
at	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
awam	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
bagus	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
banget	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0
benakribo	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
benar	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0
berat	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
besok	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
bikin	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0
bingung	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
bosan	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
bullock	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
tubuh	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
ukur	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
ulil	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
usaha	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
wanita	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

4.2.4 *Pearson Correlation Coefficient*

Pada tahap ini dilakukan proses seleksi fitur menggunakan metode *Pearson Correlation Coefficient*. Proses seleksi fitur bertujuan untuk mengambil kombinasi fitur – fitur yang relevan dengan mengukur dua variabel yaitu variabel x (fitur) dan variabel y (kelas). Tabel 4.13 menunjukkan perhitungan jumlah dari variabel x dan y. Sebagai contoh, jumlah kata positif pada kata sifat yaitu fitur F25 yang akan dihitung nilai *Pearsonnya* maka:

Tabel 4.13 Perhitungan Manual *Pearson Correlation Coefficient*

	X (F25)	Y (Kelas)	X^2	Y^2	XY
Pearson's	0	1	0	1	0
	2	0	4	0	0
	0	1	0	1	0
	1	0	1	0	0
	2	0	4	0	0

Tabel 4.13 Perhitungan Manual *Pearson Correlation Coefficient* (lanjutan)

	X (F25)	Y (Kelas)	X^2	Y^2	XY
Pearson's	0	1	0	1	0
	0	1	0	1	0
	1	0	1	0	0
	0	1	0	1	0
	1	0	1	0	0
Jumlah	7	5	11	5	0

$$r_{xy} = \frac{10(0) - ((7)(5))}{\sqrt{[(10(11)) - (7)^2]}\sqrt{[(10(5)) - (5)^2]}}$$

$$= \frac{0 - 35}{\sqrt{[61]}\sqrt{[25]}} = \frac{-35}{39{,}0512} = -0{,}8962 = 0{,}896258$$

Hasil *Pearson's* pada F25 didapatkan sebesar -0,896258, untuk mempermudah pengurutan nilai *Pearson's* maka diberikan *absolute* sehingga nilai *Pearson's* pada F25 menjadi positif 0.896258. Nilai *Pearson's* yang telah didapatkan selanjutnya akan diurutkan berdasarkan nilai *Pearson's* terbesar hingga terkecil. Berikut pengurutan nilai *Pearson's* dengan fitur 25% teratas ditunjukkan pada Tabel 4.14.

Tabel 4.14 Pengurutan nilai *Pearson Correlation Coefficient* (25%)

Fitur	Nilai *Pearson's Correlation Coefficient*
F31	1
F25	0.896258
F37	0.780869
F20	0.772635
F14	0.722315
F10	0.686406
F26	0.654654
F19	0.651074
F8	0.620174
F32	0.620174
F6	0.57291
F15	0.557086
F7	0.5
F11	0.5
angkasa	0.5
banget	0.5
benar	0.5
bosan	0.5
main	0.5

Tabel 4.14 Pengurutan nilai *Pearson Correlation Coefficient* (25%) (lanjutan)

Fitur	Nilai *Pearson's Correlation Coefficient*
F24	0.478913
rating	0.468521
F5	0.452891
F2	0.408248
F4	0.408248
F36	0.333333
F30	0.333333
action	0.333333
acu	0.333333
ampun	0.333333
asli	0.333333
at	0.333333
awam	0.333333
benakribo	0.333333
berat	0.333333
besok	0.333333

4.2.5 *Naïve Bayes Classifier*

Pada proses ini dilakukan pengklasifikasian *tweet* menggunakan *Naïve Bayes*. Langkah pertama yang dilakukan pada proses ini adalah melakukan perhitungan probabilitas prior, selanjutnya akan dilakukan pengelompokkan *Naïve Bayes* berdasarkan jenis datanya untuk menentukan probabilitas likelihood, untuk jenis data biner akan menggunakan *Bernoulli Naïve Bayes*, jenis data diskrit menggunakan *Multinomial Naïve Bayes*, jenis data kontinu menggunakan *Gaussian Naïve Bayes*.

4.2.5.1 Perhitungan Probabilitas *Prior*

Perhitungan Probabilitas *Prior* dilakukan dengan menghitung jumlah dokumen kelas dibagi dengan jumlah dokumen keseluruhan. Sebagai contoh, jumlah keseluruhan data latih adalah 10, jumlah dokumen dengan kelas positif adalah 5 dan jumlah dokumen dengan kelas negatif adalah 5 maka:

$$P(Positif) = \frac{5}{10} = 0{,}5$$

$$P(Negatif) = \frac{5}{10} = 0.5$$

4.2.5.2 Perhitungan *Bernoulli Naïve Bayes*

Perhitungan *Bernoulli Naïve Bayes* dimulai dengan menjumlahkan nilai fitur pada kelas positif dan kelas negatif. Pengklasifikasin dengan menggunakan

metode *Bernoulli Naïve Bayes* digunakan hanya untuk jenis data *binary* (biner). Untuk fitur yang menggunakan *Bernoulli Naïve Bayes* adalah F1 – F4, F11, dan F12. Hasil seleksi fitur dengan menggunakan *Pearson's*, untuk jenis data biner fitur yang terseleksi adalah F11 dan F12. Berikut contoh perhitungan Manual *Bernoulli Naïve Bayes*, nilai fitur F11 dan F12 ditunjukkan pada Tabel 4.15.

Tabel 4.15 Fitur *Bernoulli Naïve Bayes* yang terseleksi

Fitur	PS	NG	PS	NG	NG	PS	PS	NG	PS	NG	PS	NG	PS	PS	NG
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F11	0	0	1	0	0	1	0	0	0	0	1	0	0	1	1
F2	1	0	0	0	0	0	1	0	1	1	1	1	0	1	0
F4	1	0	0	0	0	0	1	1	1	0	1	0	0	0	0

Langkah 1. Menghitung jumlah nilai fitur setiap kelas

Langkah pertama adalah menghitung jumlah nilai fitur pada kelas positif dan kelas negatif. Penjumlahan nilai fitur setiap kelas bertujuan untuk mencari nilai peluang fitur dengan syarat kelas positif dan negatif. Berikut contoh perhitungan penjumlahan nilai fitur setiap kelas dan hasil penjumlahan ditunjukkan pada Tabel 4.16.

$$F11\ (Positif) = 0 + 1 + 1 + 0 + 0 = 2$$

$$F11\ (Negatif) = 0 + 0 + 0 + 0 + 0 = 0$$

Tabel 4.16 Hasil Penjumlahan Fitur Kelas

Fitur	PS	NG	PS	...	PS	NG	PS	PS	NG	POSITIF	NEGATIF
	D1	D2	D3	...	D11	D12	D13	D14	D15		
F11	0	0	1	...	1	0	0	1	1	2	0
F2	1	0	0	...	1	1	0	1	0	3	1
F4	1	0	0	...	1	0	0	0	0	3	1

Langkah 2. Menghitung Peluang Fitur dengan syarat positif dan negatif

Langkah kedua adalah menghitung peluang fitur dengan syarat positif dan negatif. Perhitungan peluang dengan syarat positif dan negatif ini menggunakan peluang dengan parameter *laplace smoothing*. Berikut contoh perhitungan peluang fitur dengan syarat kelas positif dan negatif dan hasil peluang ditunjukkan pada Tabel 4.17.

$$P(F11|Positif) = \frac{(2+1)}{(5+2)} = \frac{3}{7} = 0{,}42587$$

$$P(F11|Negatif) = \frac{(0+1)}{(5+2)} = \frac{1}{7} = 0{,}14285$$

Tabel 4.17 Hasil Peluang fitur dengan syarat positif dan negatif

Fitur	PS	NG	PS	...	PS	NG	PS	PS	NG	P(F|POS)	P(F|NEG)
	D1	D2	D3	...	D11	D12	D13	D14	D15		
F11	0	0	1	...	1	0	0	1	1	0,42857	0,14285
F2	1	0	0	...	1	1	0	1	0	0,57142	0,28571
F4	1	0	0	...	1	0	0	0	0	0,57142	0,28571

Langkah 3. Menghitung Probabilitas *Likelihood*

Langkah ketiga adalah menghitung probabilitas *likelihood*, probabilitas *likelihood* digunakan untuk menghitung *posterior*. Dalam perhitungan manual ini akan dicari nilai peluang dari dokumen uji dengan syarat setiap kelasnya yaitu positif dan negatif dari hasil peluang tersebut akan dikalikan dan hasil perkalian dari peluang tersebut digunakan untuk mencari probabilitas *posterior*. Berikut contoh perhitungan probabilitas *likelihood* dan hasil peluang ditunjukkan pada Tabel 4.18.

$$P(D11|Positif) = (0{,}4285)^1 . (1 - 0{,}4285)^{(1-1)} = 0{,}4285$$

$$P(D11|Negatif) = (0{,}1428)^1 . (1 - 0{,}1428)^{(1-1)} = 0{,}1428$$

Tabel 4.18 Hasil Probabilitas *Likelihood Bernoulli Naive Bayes*

Fitur	PS	...	PS	...	PS	NG	P(D11| POS)	P(D11| NEG)	... ...	P(D15| POS)	P(D15| NEG)
	D1	...	D11	...	D14	D15					
F11	0	...	1	...	1	1	0,4285	0,1428	...	0,4285	0,1428
F2	1	...	1	...	1	0	0,5714	0,2857	...	0,4286	0,7143
F4	1	...	1	...	0	0	0,5714	0,2857	...	0,4286	0,7143
Product							0,1399	0,0116	...	0,0787	0,0728

4.2.5.3 Perhitungan *Multinomial Naïve Bayes*

Perhitungan *Multinomial Naïve Bayes* dimulai dengan menjumlahkan nilai fitur pada kelas positif dan kelas negatif. Pengklasifikasin dengan menggunakan metode *Multinomial Naïve Bayes* digunakan hanya untuk jenis data diskrit. Bebebrapa fitur yang digunakan pada *Multinomial Naïve Bayes*, untuk tipe *Textual Features* adalah F7 – F10, tipe *Part of Speech* adalah F13 – F17, tipe *Lexicon based Features* adalah F23 – F30 dan F37, tipe *Bag of Words* adalag kata/*term*. Hasil seleksi fitur dengan menggunakan *Pearson's*, untuk jenis data diskrit fitur yang terseleksi adalah *Textual Features* (F8 dan F10), *Part of Speech* (F14, F15 dan F16), *Lexicon based Features* (F24, F25, F26, F30 dan F37) dan *Bag of Word* (angkasa, pemain, benar, tinggi, banget dan rating). Berikut contoh perhitungan Manual *Multinomial Naïve Bayes* dan nilai fitur ditunjukkan pada Tabel 4.19.

Tabel 4.19 Nilai Fitur *Multinomial Naive Bayes*

Fitur	PS	NG	PS	NG	NG	PS	PS	NG	PS	NG	PS	NG	PS	PS	NG
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F7	0	0	0	0	0	0	1	0	1	0	3	1	1	1	0
F8	1	0	2	0	0	0	2	0	0	0	0	0	2	0	0
F10	8	4	3	3	3	6	8	5	8	4	8	4	5	6	4
F14	2	5	0	4	3	0	0	2	1	1	2	2	1	1	2
F15	2	1	4	3	2	2	2	1	4	1	3	1	6	5	3
F24	0	1	2	1	2	0	0	0	0	3	1	0	0	0	2
F25	0	2	0	1	2	0	0	1	0	1	2	1	1	1	1
F26	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1
F30	0	0	0	0	0	0	0	0	0	2	0	0	0	0	1
F37	0	1	0	1	1	1	0	1	0	2	1	0	1	0	1
angkasa	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0
action	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
acu	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
ampun	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
asli	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
at	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
awam	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
banget	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0
benar	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0
bosan	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
benakribo	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
berat	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
besok	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
main	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1
rating	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0

Langkah 1. Menghitung jumlah nilai fitur setiap kelas

Langkah pertama adalah menghitung jumlah nilai fitur pada kelas positif dan kelas negatif. Penjumlahan nilai fitur setiap kelas bertujuan untuk mencari nilai peluang fitur dengan syarat kelas positif dan negatif. Berikut contoh perhitungan penjumlahan nilai fitur setiap kelas dan hasil penjumlahan ditunjukkan pada Tabel 4.20.

$$F10\ (Positif) = 8 + 3 + 6 + 8 + 8 = 33$$

$$F10\ (Negatif) = 4 + 3 + 3 + 5 + 4 = 19$$

Tabel 4.20 Hasil Penjumlahan Fitur Kelas

Fitur	PS	NG	PS	...	NG	PS	NG	PS	PS	NG	POSITIF	NEGATIF
	D1	D2	D3	...	D10	D11	D12	D13	D14	D15		
F7	0	0	0	...	0	3	1	1	1	0	2	0
F8	1	0	2	...	0	0	0	2	0	0	5	0
F10	8	4	3	...	4	8	4	5	6	4	33	19
F14	2	5	0	...	1	2	2	1	1	2	3	15
F15	2	1	4	...	1	3	1	6	5	3	14	8
F24	0	1	2	...	3	1	0	0	0	2	2	7
F25	0	2	0	...	1	2	1	1	1	1	0	7
F26	0	1	0	...	0	0	0	0	0	1	0	3
F30	0	0	0	...	2	0	0	0	0	1	0	2
F37	0	1	0	...	2	1	0	1	0	1	1	6
angkasa	1	0	1	...	0	0	0	0	1	0	2	0
action	0	1	0	...	0	0	0	0	0	0	0	1
acu	0	0	0	...	0	0	0	0	0	0	0	1
ampun	0	0	0	...	0	0	0	0	0	0	0	1
asli	0	0	0	...	0	0	0	0	0	0	1	0
at	0	0	0	...	0	0	0	0	0	0	0	1
awam	0	0	0	...	0	0	0	0	0	0	0	1
banget	0	0	0	...	1	1	0	0	0	0	0	2
benar	0	1	0	...	0	0	0	0	1	0	0	2
bosan	0	0	0	...	1	0	0	0	0	0	0	2
benakribo	0	0	0	...	2	0	0	0	0	0	0	2
berat	0	1	0	...	0	0	0	0	0	0	0	1
besok	1	0	0	...	0	0	0	0	0	0	1	0
main	0	1	0	...	1	0	0	0	0	1	0	2
rating	0	0	0	...	0	0	0	0	0	0	0	3

Langkah 2. Menghitung Peluang Fitur dengan syarat positif dan negatif

Langkah kedua adalah menghitung peluang fitur dengan syarat positif dan negatif. Pada tahap manualisasi ini perhitungan dimulai dengan menghitung banyaknya fitur pada *multinomial naïve bayes* yang terseleksi selanjutnya menghitung seluruh jumlah kemunculan fitur pad akelas positif dan negatif. Perhitungan peluang dengan syarat positif dan negatif ini menggunakan peluang dengan parameter *laplace smoothing*. Berikut contoh perhitungan peluang fitur dengan syarat kelas positif dan negatif dan hasil peluang ditunjukkan pada Tabel 4.21.

$$P(F10|Positif) = \frac{(33+1)}{(30+25)} = \frac{34}{55} = 0{,}61818$$

$$P(F10|Negatif) = \frac{(19+1)}{(53+25)} = \frac{20}{78} = 0{,}25641$$

Tabel 4.21 Hasil Peluang fitur dengan syarat positif dan negatif

Fitur	PS	NG	PS	...	PS	NG	PS	PS	NG	P(F|POS)	P(F|NEG)
	D1	D2	D3	...	D11	D12	D13	D14	D15		
F7	0	0	0	...	3	1	1	1	0	0,05455	0,01282
F8	1	0	2	...	0	0	2	0	0	0,10909	0,01282
F10	8	4	3	...	8	4	5	6	4	0,61818	0,25641
F14	2	5	0	...	2	2	1	1	2	0,07273	0,20513
F15	2	1	4	...	3	1	6	5	3	0,27273	0,11538
F24	0	1	2	...	1	0	0	0	2	0,05455	0,10256
F25	0	2	0	...	2	1	1	1	1	0,01818	0,10256
F26	0	1	0	...	0	0	0	0	1	0,01818	0,05128
...	...	...	...	...	...	...	...	...	...	....	....
angkasa	1	0	1	...	0	0	0	1	0	0,05455	0,01282
action	0	1	0	...	0	0	0	0	0	0,01818	0,02564
acu	0	0	0	...	0	0	0	0	0	0,01818	0,02564
....	...	...	...	...	...	...	...	...	...	....	....
rating	0	0	0	...	0	0	0	0	0	0,01818	0,05128

Langkah 3. Menghitung Probabilitas *Likelihood*

Langkah ketiga adalah menghitung probabilitas *likelihood*, probabilitas *likelihood* digunakan untuk menghitung *posterior*. Dalam perhitungan manual ini akan dicari nilai peluang dari dokumen uji dengan syarat setiap kelasnya yaitu positif dan negatif dari hasil peluang tersebut akan dikalikan dan hasil perkalian dari peluang tersebut digunakan untuk mencari probabilitas *posterior*. Berikut contoh perhitungan probabilitas *likelihood* dan hasil peluang ditunjukkan pada Tabel 4.22.

$$P(D11|Positif) = (0{,}61818)^8 = 0{,}021327$$

$$P(D11|Negatif) = (0{,}25641)^8 = 1{,}8685 \times 10^{-5}$$

Tabel 4.22 Hasil Probabilitas *Likelihood Multinomial Naive Bayes*

Fitur	PS	...	PS	...	PS	NG	P(D11| POS)	P(D11| NEG)	... ...	P(D15| POS)	P(D15|NEG)
	D1	...	D11	...	D14	D15					
F7	0	...	3	...	1	0	0,00016	2,1E-06	...	1	1
F8	1	...	0	...	0	0	1	1	...	1	1
F10	8	...	8	...	6	4	0,02137	1,8E-05	...	0,14603	0,00432
F14	2	...	2	...	1	2	0,00528	0,04207	...	0,00528	0,04207
F15	2	...	3	...	5	3	0,02028	0,00153	...	0,02028	0,00153
F24	0	...	1	...	0	2	0,05454	0,1025	...	0,00297	0,01051
F25	0	...	2	...	1	1	0,00033	0,01051	...	0,01818	0,10256
F26	0	...	0	...	0	1	1	1	...	0,01818	0,05128

Tabel 4.22 Hasil Probabilitas *Likelihood Multinomial Naive Bayes* (lanjutan)

Fitur	PS	...	PS	...	PS	NG	P(D11| POS)	P(D11| NEG)	... ...	P(D15| POS)	P(D15|NEG)
	D1	...	D11	...	D14	D15					
...	...	...	...	...	...	...	....	....	....	....	....
angkasa	1	...	0	...	1	0	1	1	...	1	1
action	0	...	0	...	0	0	1	1	...	1	1
acu	0	...	0	...	0	0	1	1	...	1	1
....	...	....	...	...	...	...	....	....	...	....	....
rating	0		0	...	0	0	1	1	...	1	1
Product							2,1E-16	5,1E-16	...	6,9E-14	1,2E-11

4.2.5.4 Perhitungan *Gaussian Naïve Bayes*

Perhitungan *Gaussian Naïve Bayes* dimulai dengan mencari nilai rata – rata pada setiap kelas untuk setiap fiturnya dan nilai varians pada setiap kelas dan setiap fiturnya. Pengklasifikasian dengan menggunakan metode *Gaussian Naïve Bayes* digunakan hanya untuk jenis data kontinu. Bebebrapa fitur yang digunakan pada *Gaussian Naïve Bayes*, untuk tipe *Textual Features* adalah F5 – F6, tipe *Part of Speech* adalah F18 – F22, tipe *Lexicon based Features* adalah F31 – F36. Hasil seleksi fitur dengan menggunakan *Pearson's* untuk jenis data kontinu fitur yang terseleksi adalah *Textual Features* (F5 dan F6), *Part of Speech* (F18 – F22), *Lexicon based Features* (F31 – F36). Hasil seleksi fitur dengan menggunakan *Pearson's* dengan menggunakan seleksi sebanyak 25% yaitu, untuk jenis data kontinu fitur yang terseleksi adalah *Textual Features* (F6), *Part of Speech* (F19, F20 dan F21), *Lexicon based Features* (F31, F32 dan F36). Berikut contoh perhitungan manual *Gaussian Naïve Bayes* dan nilai fitur ditunjukkan pada Tabel 4.23.

Tabel 4.23 Nilai Fitur *Gaussian Naïve Bayes*

Fitur	PS	NG	PS	NG	NG	PS	PS	NG	PS	NG	PS	NG	PS	PS	NG
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15
F5	14	25	22	23	23	16	18	19	26	23	23	21	22	27	25
F6	8.06	6.15	6.26	6.43	6.12	6.33	7.9	6.41	7.24	6.76	7.7	5.41	5.91	6.09	5.89
F19	0.14	0.2	0	0.17	0.13	0	0	0.11	0.039	0.04	0.08	0.09	0.04	0.03	0.08
F20	0.14	0.04	0.182	0.13	0.08	0.12	0.11	0.05	0.15	0.043	0.13	0.04	0.27	0.18	0.12
F31	0	1	0	1	1	0	0	1	0	1	1	0.17	0.5	0.33	0.25
F32	0	1	0	1	0.5	0	0	0	0	0	0	0	0	0	0.5
F36	0	0	0	0	0	0	0	0	0	0.67	0	0	0	0	0.5

Langkah 1. Menghitung rata – rata nilai fitur setiap kelas

Langkah pertama adalah menghitung nilai rata – rata fitur pada setiap kelas positif dan kelas negatif. Nilai rata – rata fitur setiap kelas bertujuan untuk mencari nilai peluang fitur dengan syarat kelas positif dan negatif. Berikut contoh perhitungan

penjumlahan nilai fitur setiap kelas dan hasil penjumlahan ditunjukkan pada Tabel 4.24.

$$F6\ (Positif) = \frac{8{,}06 + 6{,}26 + 6{,}33 + 7{,}9 + 7{,}24}{5} = 7{,}156$$

$$F6\ (Negatif) = \frac{6{,}15 + 6{,}43 + 6{,}13 + 6{,}42 + 6{,}76}{5} = 6{,}378$$

Tabel 4.24 Nilai rata - rata fitur setiap kelas

Fitur	PS	NG	PS	...	NG	PS	NG	PS	PS	NG	RATA – RATA POSITIF	RATA – RATA NEGATIF
	D1	D2	D3	...	D10	D11	D12	D13	D14	D15		
F5	14	25	22	...	23	23	21	22	27	25	19,2	22,6
F6	8.06	6.15	6.26	...	6.76	7.7	5.41	5.91	6.09	5.89	7,156	6,378
F19	0.14	0.2	0	...	0.04	0.08	0.09	0.04	0.03	0.08	0,036	0,131
F20	0.14	0.04	0.182	...	0.043	0.13	0.04	0.27	0.18	0.12	0,142	0,0707
F31	0	1	0	...	1	1	0.17	0.5	0.33	0.25	0	1
F32	0	1	0	...	0	0	0	0	0	0.5	0	0,5
F36	0	0	0	...	0.67	0	0	0	0	0.5	0	0,134

Langkah 2. Menghitung *varians* nilai fitur setiap kelas

Langkah kedua adalah menghitung nilai *varians* fitur pada setiap kelas positif dan kelas negatif. Nilai *varians* fitur setiap kelas bertujuan untuk mencari nilai peluang fitur dengan syarat kelas positif dan negatif. Berikut contoh perhitungan penjumlahan nilai fitur setiap kelas dan hasil penjumlahan ditunjukkan pada Tabel 4.25.

$$\sigma^2(positif\ F6) = (8{,}06 - 7{,}156)^2 + (6{,}26 - 7{,}156)^2 + (7{,}33 - 7{,}156)^2$$

$$= + (7{,}9 - 7{,}156)^2 + (7{,}24 - 7{,}156)^2 = 0{,}572$$

$$\sigma^2(negatif\ F6) = (6{,}15 - 6{,}378)^2 + (6{,}43 - 6{,}378)^2 + (6{,}13 - 6{,}378)^2$$

$$= + (6{,}42 - 6{,}378)^2 + (6{,}76 - 6{,}378)^2 = 0{,}052$$

Tabel 4.25 Hasil Peluang fitur dengan syarat positif dan negatif

Fitur	PS	NG	PS	...	NG	PS	NG	PS	PS	NG	*VARIAN* POSITIF	*VARIAN* NEGATIF
	D1	D2	D3	...	D10	D11	D12	D13	D14	D15		
F5	14	25	22	...	23	23	21	22	27	25	18,56	3,84
F6	8.06	6.15	6.26	...	6.76	7.7	5.41	5.91	6.09	5.89	0,572	0,052
F19	0.14	0.2	0	...	0.04	0.08	0.09	0.04	0.03	0.08	0,003	0,002
F20	0.14	0.04	0.182	...	0.04	0.13	0.04	0.27	0.18	0.12	0,0005	0,0011
F31	0	1	0	...	1	1	0.17	0.5	0.33	0.25	0	0
F32	0	1	0	...	0	0	0	0	0	0.5	0	0,2
F36	0	0	0	...	0.67	0	0	0	0	0.5	0	0,07

Langkah 3. Menghitung Probabilitas *Likelihood*

Langkah ketiga adalah menghitung probabilitas *likelihood*, probabilitas *likelihood* digunakan untuk menghitung *posterior*. Dalam perhitungan manual ini akan dicari nilai peluang dari dokumen uji dengan syarat setiap kelasnya yaitu positif dan negatif dari hasil peluang tersebut akan dikalikan dan hasil perkalian dari peluang tersebut digunakan untuk mencari probabilitas *posterior*. Berikut contoh perhitungan probabilitas *likelihood* dan hasil peluang ditunjukkan pada Tabel 4.26.

$$P(D11|Positif) = \frac{1}{\sqrt{2 \times 3{,}14 \times 0{,}572}} \exp\left(-\frac{(7{,}703 - 7{,}1596)^2}{2 \times 0{,}572}\right) = 0{,}4073$$

$$P(D11|Negatif) = \frac{1}{\sqrt{2 \times 3{,}14 \times 0{,}052}} \exp\left(-\frac{(7{,}703 - 6{,}378)^2}{2 \times 0{,}052}\right) = 1{,}0886E-07$$

Tabel 4.26 Hasil Probabilitas *Likelihood Gaussian Naive Bayes*

Fitur	PS	...	PS	...	PS	NG	P(D11| POS)	P(D11| NEG)	...	P(D15| POS)	P(D15|NEG)
	D1	...	D11	...	D14	D15			...		
F5	14	...	23	...	27	25	0,0627	0,1994	...	0,0374	0,0961
F6	8.06	...	7.7	...	6.09	5.89	0,4073	1,1E-07	...	0,1286	0,1812
F19	0.14	...	0.08	...	0.03	0.08	4,7375	5,3109	...	5,2771	4,7551
F20	0.14	...	0.13	...	0.18	0.12	143603	2,53901	...	10,5229	4,12501
F31	0	...	1	...	0.33	0.25	1	1	...	1	1
F32	0	...	0	...	0	0.5	1	0,4776	...	1	0,8922
F36	0	...	0	...	0	0.5	1	1,3205	...	1	0,5814
Product							1,7395	1,8E-07		0,26735	0,17745

4.2.5.5 Perhitungan Probabilitas *Posterior*

Perhitungan Probabilitas *Posterior* dilakukan setelah mendapatkan nilai probabilitas *prior* dan total *product* setiap probabilitas *likelihood Naïve Bayes*. Hasil dari perhitungan probabilitas *posterior* didapatkan dari hasil perkalian antara probabilitas *prior*, *product* dari probabilitas *likelihood Bernoulli Naïve Bayes*, *product* dari probabilitas *likelihood Multinomial Naïve Bayes* dan *product* dari probabilitas *likelihood Gaussian Naïve Bayes*. Nilai dari probabilitas *posterior* akan menentukan kelas dari dari setiap data uji, jika nilai *posterior* positif lebih besar dari kelas negatif maka data uji adalah kelas positif dan sebaliknya. Berikut contoh perhitungan dari probabilitas *posterior*.

$$P(Positif|D11) = 0{,}5 \times 0{,}1399 \times 2{,}0758 \times 10^{-16} \times 1{,}7395 = 2{,}5266 \times 10^{-17}$$

$$P(Negatif|D11) = 0{,}5 \times 0{,}0116 \times 5{,}0726 \times 10^{-16} \times 1{,}8465 \times 10^{-17} =$$

$$= 5{,}4614 \times 10^{-17}$$

Dari hasil perkalian diatas nilai peluang positif dengan syarat D11 lebih besar dibandingkan dengan nilai peluang negatif dengan syarat D11 maka kelas data uji pada dokumen 11 adalah kelas positif. Berikut hasil *ground truth* setiap dokumen ditunjukkan pada Tabel 4.27.

Tabel 4.27 Hasil Probabilitas *Posterior*

Klasifikasi Naïve Bayes		Kelas (*Ground Truth)*
P(POS|D11)	2.52664E-17	POSITIF
P(NEG|D11)	5.46144E-25	
P(POS|D12)	9.68922E-12	NEGATIF
P(NEG|D12)	6.95629E-10	
P(POS|D13)	4.53532E-17	POSITIF
P(NEG|D13)	2.26307E-19	
P(POS|D14)	5.20798E-13	POSITIF
P(NEG|D14)	2.37415E-17	
P(POS|D15)	7.34193E-16	NEGATIF
P(NEG|D15)	7.98305E-14	

4.2.6 Evaluasi

Untuk mengetahui kualitas dari metode yang dilakukan maka diperlukan sistem pengujian yang sesuai sehingga bisa terlihat kinerja yang diberikan oleh metode tersebut. Pada penelitian ini akan digunakkan *precision, recall, f-measure* dan *accuracy* untuk mengevaluasi sistem. Untuk contoh manualisasi pengujian klasifikasi pada analisis sentimen ditunjukkan pada Tabel 4.28. Terdapat 5 data uji serta label *ground truth* dan label kelas prediksi.

Tabel 4.28 Label untuk Data Uji

Klasifikasi Naïve Bayes		Kelas (*Ground Truth*	Prediksi
P(POS|D11)	2.52664E-17	POSITIF	POSITIF
P(NEG|D11)	5.46144E-25		
P(POS|D12)	9.68922E-12	NEGATIF	NEGATIF
P(NEG|D12)	6.95629E-10		
P(POS|D13)	4.53532E-17	POSITIF	POSITIF
P(NEG|D13)	2.26307E-19		
P(POS|D14)	5.20798E-13	POSITIF	POSITIF
P(NEG|D14)	2.37415E-17		
P(POS|D15)	7.34193E-16	NEGATIF	NEGATIF
P(NEG|D15)	7.98305E-14		

Berdasarkan Tabel 4.28 diatas ditunjukkan bahwa hasil Prediksi dengan hasil dilakukan oleh sistem (*ground truth*). Perhitungan manual Evaluasi ditunjukkan pada Tabel 4.29.

Tabel 4.29 Confussion Matrix Pengujian

		Prediksi	
		Positif	Negatif
Ground Truth	Positif	3	0
	Negatif	0	2

$$Precision = \frac{3}{3+0} = 1\ (100\%)$$

$$Recall\ = \frac{3}{3+0} = 1\ (100\%)$$

$$F1 - Measure = \frac{2 \times 1 \times 1}{1+1} = 1\ (100\%)$$

$$Acuracy = \frac{3+2}{3+0+0+2} = 1\ (100\%)$$

4.3 Perancangan Pengujian

Pada bagian ini dilakukan perancangan pengujian dengan tujuan untuk mendapatkan hasil klasifikasi berupa opini positif atau opini negatif dengan menggunakan *naïve bayes* dengan *ensemble feature* dan seleksi fitur *pearson correlation coefficient* pada analisis sentimen opini film. Terdapat 2 skenario pengujian yaitu Pengujian Seleksi Fitur dengan Kata Tidak Baku dan Pengujian Seleksi Fitur dengan Kata Baku.

4.3.1 Pengujian Seleksi Fitur dengan Kata Tidak Baku

Pengujian Seleksi Fitur dengan menggunakan Kata Tidak Baku dilakukan dengan cara tanpa mengubah kalimat atau isi pada dokumen Twitter. Pengujian ini menggunakan nilai *threshold* seleksi Fitur 10% sampai dengan 95%. Berikut perancangan tabel yang digunakan pada pengujian kata tidak baku ditunjukkan pada Tabel 4.30.

Tabel 4.30 Pengujian Seleksi Fitur dengan Kata Tidak Baku

Threshold **Seleksi Fitur**	**Evaluasi**			
	Accuracy	***Precision***	***Recall***	***F-Measure***
10%	**%%	**%%	**%%	**%%
15%	**%%	**%%	**%%	**%%
20%	**%%	**%%	**%%	**%%

Tabel 4.30 Pengujian Seleksi Fitur dengan Kata Tidak Baku (lanjutan)

Threshold Seleksi Fitur	Evaluasi			
	Accuracy	***Precision***	***Recall***	***F-Measure***
25%	**%	**%	**%	**%
…	**%	**%	**%	**%
95%	**%	**%	**%	**%
100%	**%	**%	**%	**%

4.3.2 Pengujian Seleksi Fitur dengan Kata Baku

Pengujian Seleksi Fitur dengan menggunakan Kata Baku dilakukan dengan cara mengubah kalimat atau isi pada dokumen Twitter menjadi kata baku. Pengujian ini menggunakan nilai *threshold* seleksi Fitur 10% sampai dengan 90%. Berikut perancangan tabel yang digunakan pada pengujian kata tidak baku ditunjukkan pada Tabel 4.31.

Tabel 4.31 Pengujian Seleksi Fitur dengan Kata Baku

Threshold Seleksi Fitur	Evaluasi			
	Accuracy	***Precision***	***Recall***	***F-Measure***
10%	**%	**%	**%	**%
15%	**%	**%	**%	**%
20%	**%	**%	**%	**%
25%	**%	**%	**%	**%
…	**%	**%	**%	**%
95%	**%	**%	**%	**%
100%	**%	**%	**%	**%

BAB 5 HASIL

Pada bab ini menjelaskan tentang implementasi sistem yang dibuat berdasarkan perancangan yang telah dilakukan. Implementasi yang akan dibahas dalam bab ini adalah implementasi algoritme.

5.1 Implementasi Algoritme

Berdasarkan perancangan yang telah dijelaskan pada Bab 4, maka pada bab ini akan membahas implementasi program, sesuai dengan perancangan yang telah dibuat. Program di implementasikan dengan menggunakan bahasa pemrograman *Python* 2.7.

5.1.1 Implementasi *Twitter Specification*

Proses ini diawali dengan mendapatkan nilai fitur F1 sampai F4, kemudian nilai tersebut hanya berisi nilai 1 jika terdapat fitur dan nilai 0 jika tidak terdapat fitur di dalam *tweet*. Kode Program pada proses ini dapat dilihat pada Kode Program 5.1 hinga 5.4.

5.1.1.1 Impelementasi Kode Program Fitur F1

Proses ini dilakukan untuk mendapatkan nilai fitur F1, apakah *tweet* terdapat *hashtag* atau tidak jika terdapat *hashtag* dalam satu dokumen maka diberi nilai 1 jika tidak diberi nilai 0. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.1.

Kode Program 5.1 Implementasi Proses Fitur F1

```
1  def F1(self):
2      list = []
3      list.append("F1")
4      for dokumen in self.docs:
5          if '#' in dokumen[1]:
6              list.append(1)
7          else:
8              list.append(0)
9      return list
```

Penjelasan dari Kode Program 5.1 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F1" pada `list`.
3. Baris 4 – 8 melakukan perulangan pada seluruh dokumen. Jika terdapat '#' pada dokumen *array* ke-1 maka dokumen akan bernilai 1 jika tidak maka bernilai 0.
4. Baris 9 mengembalikan nilai dari `list`.

5.1.1.2 Impelementasi Kode Program Fitur F2

Proses ini dilakukan untuk mendapatkan nilai fitur F2, apakah *tweet* terdapat *retweet* atau tidak jika terdapat *retweet* dalam satu dokumen maka diberi nilai 1 jika tidak diberi nilai 0. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.2.

Kode Program 5.2 Implementasi Proses Fitur F2

```
1   def F2(self):
2       list = []
3       list.append("F2")
4       for dokumen in self.docs:
5           if 'RT' in dokumen[1]:
6               list.append(1)
7           else:
8               list.append(0)
9       return list
```

Penjelasan dari Kode Program 5.2 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F2" pada `list`.
3. Baris 4 – 8 melakukan perulangan pada seluruh dokumen. Jika terdapat 'RT' pada dokumen *array* ke-1 maka dokumen akan bernilai 1 jika tidak maka bernilai 0.
4. Baris 9 mengembalikan nilai dari `list`.

5.1.1.3 Impelementasi Kode Program Fitur F3

Proses ini dilakukan untuk mendapatkan nilai fitur F3, apakah *tweet* terdapat *username* atau tidak jika terdapat *username* dalam satu dokumen maka diberi nilai 1 jika tidak diberi nilai 0. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.3.

Kode Program 5.3 Implementasi Proses Fitur F3

```
1   def F3(self):
2       list = []
3       list.append("F3")
4       username = "(?<=^|(?<=[^a-zA-Z0-9-_\.]))@([\w+[\w0-9-
5   _]+)"
6       for dokumen in self.docs:
7           if re.findall(username, dokumen[1]):
8               list.append(1)
9           else:
10              list.append(0)
11      return list
```

Penjelasan dari Kode Program 5.3 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F3" pada `list`.
3. Baris 4-5 inisialisasi *regex username* untuk menyeleksi *username* pada dokumen.
4. Baris 6 – 8 melakukan perulangan pada seluruh dokumen. Jika terdapat *regex username* pada dokumen *array* ke-1 maka dokumen akan bernilai 1 jika tidak maka bernilai 0.
5. Baris 9 mengembalikan nilai dari `list`.

5.1.1.4 Impelementasi Kode Program Fitur F4

Proses ini dilakukan untuk mendapatkan nilai fitur F4, apakah *tweet* terdapat *URL* atau tidak jika terdapat *URL* dalam satu dokumen maka diberi nilai 1 jika tidak diberi nilai 0. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.4.

Kode Program 5.4 Implementasi Proses Fitur F4

```
1   def F4(self):
2       list = []
3       list.append("F4")
4       rgx = "((http|https|ftp)\:\/\/)?[a-zA-Z0-9\.\/\?\:@\-
5   _=#]+\.([a-zA-Z]){2,6}([a-zA-Z0-9\.\&\/\?\:@\-_=#])*"
6       for dokumen in self.docs:
7           if re.findall(rgx, dokumen[1]):
8               list.append(1)
9           else:
10              list.append(0)
11      return list
```

Penjelasan dari Kode Program 5.4 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F4" pada `list`.
3. Baris 4-5 inisialisasi *regex URL* untuk menyeleksi *URL* pada dokumen
4. Baris 6-10 melakukan perulangan pada seluruh dokumen. Jika terdapat *regex URL* pada dokumen *array* ke-1 maka dokumen akan bernilai 1 jika tidak maka bernilai 0.
5. Baris 11 mengembalikan nilai dari `list`.

5.1.2 Implementasi *Textual Information*

Proses ini dilakukan untuk mendapatkan nilai fitur F5 sampai F12. Kode Program pada proses ini dapat dilihat pada Kode Program 5.1 hinga 5.4.

5.1.2.1 Impelementasi Kode Program Fitur F5

Proses ini dilakukan untuk mendapatkan nilai fitur F5 dari *tweet* berupa panjang kata yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.5.

Kode Program 5.5 Implementasi Proses Fitur F5

```
1   def F5(self):
2       list = []
3       list.append("F5")
4       for dokumen in self.docs:
5           rgxurl = "((http|https|ftp)\:\/\/)?[a-zA-Z0-
6   9\.\/\?\:@\-_=#]+\.([a-zA-Z]){2,6}([a-zA-Z0-9\.\&\/\?\:@\-
7   _=#])*"
8           nolink = re.sub(rgxurl, '', dokumen[1])
9           stripped = re.sub(r'[^A-Za-z\s]', '', nolink)
10          wordcount = stripped.split()
11          wordlength = len(wordcount)
12          list.append(wordlength)
13      return list
```

Penjelasan dari Kode Program 5.5 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F5" pada `list`.
3. Baris 4 – 11 melakukan perulangan pada seluruh dokumen, inisialisasi *regex url* untuk mengganti pola *url* pada dokumen menjadi `String` kosong, inisialisasi *regex* mengganti semua karakter kecuali alphabet dan spasi dan selanjutnya menghitung kata pada variabel `wordlength`
4. Baris 15 memasukkan nilai dari variabel `wordlength` kedalam *array* `list`.
5. Baris 16 mengembalikan nilai dari `list`.

5.1.2.2 Impelementasi Kode Program Fitur F6

Proses ini dilakukan untuk mendapatkan nilai fitur F6 dari *tweet* berupa rata – rata dari panjang *tweet* yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.6.

Kode Program 5.6 Implementasi Proses Fitur F6

```
1   def F6(self):
2       list = []
3       list.append("F6")
4       for dokumen in self.docs:
5           wordcount = dokumen[1].split()
6           wordlength = int(len(wordcount))
7           karakter = int(len(dokumen[1]))
8           try:
9               averageword = float(karakter) /
10  float(wordlength)
11          except ZeroDivisionError:
```

```
12                  averageword = 0
13              list.append(round(averageword, 4))
14          return list
```

Penjelasan dari Kode Program 5.6 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F6" pada `list`.
3. Baris 4 melakukan perulangan pada seluruh dokumen.
4. Baris 5-7 melakukan split kata pada dokumen *array* ke-1 dan menghitung panjang kata selanjutnya menghitung banyaknya karakter pada dokumen.
5. Baris 8-12 melakukan proses `try except`, proses `try` melakukan perhitungan nilai rata – rata panjang *tweet* dan proses `except` dilakukan ketika pembagi bernilai 0 maka `ZeroDivisionError` akan melakukan *handling* `averageword` = 0
6. Baris 13 memasukkan nilai dari variabel `averageword` kedalam *array* `list` dan mengambil 4 angka dibelakang koma.
7. Baris 14 mengembalikan nilai dari `list`.

5.1.2.3 Impelementasi Kode Program Fitur F7

Proses ini dilakukan untuk mendapatkan nilai fitur F7 dari *tweet* berupa jumlah tanda tanya yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.7.

Kode Program 5.7 Implementasi Proses Fitur F7

```
1   def F7(self):
2       list = []
3       list.append("F7")
4       for dokumen in self.docs:
5           list.append(dokumen[1].count("?"))
6       return list
```

Penjelasan dari Kode Program 5.7 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `list`.
2. Baris 3 menambahkan `String` "F7" pada `list`.
3. Baris 4 – 5 melakukan perulangan pada seluruh dokumen, selanjutnya menghitung jumlah tanda tanya pada semua dokumen indeks pertama dan memasukkannya ke dalam `list`.
4. Baris 6 mengembalikan nilai dari `list`.

5.1.2.4 Impelementasi Kode Program Fitur F8

Proses ini dilakukan untuk mendapatkan nilai fitur F8 dari *tweet* berupa jumlah tanda seru yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.8.

Kode Program 5.8 Implementasi Proses Fitur F8

```
1   def F8(self):
2       list = []
3       list.append("F8")
4       for dokumen in self.docs:
5           list.append(dokumen[1].count("!"))
6       return list
```

Penjelasan dari Kode Program 5.8 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F8" pada `list`.
3. Baris 4 – 5 melakukan perulangan pada seluruh dokumen, selanjutnya menghitung jumlah tanda seru pada semua dokumen indeks pertama dan memasukkannya ke dalam `list`.
4. Baris 6 mengembalikan nilai dari `list`.

5.1.2.5 Impelementasi Kode Program Fitur F9

Proses ini dilakukan untuk mendapatkan nilai fitur F9 dari *tweet* berupa jumlah tanda petik atau *quotes* yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.9.

Kode Program 5.9 Implementasi Proses Fitur F9

```
1   def F9(self):
2       list = []
3       list.append("F9")
4       rgx = '"([^"]*)"'
5       for dokumen in self.docs:
6           a = len(re.findall(rgx, dokumen[1]))
7           list.append(a)
8       return list
```

Penjelasan dari Kode Program 5.9 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan string "F9" pada `list`.
3. Baris 4 inisialisasi dari *regex quotes*
4. Baris 5-7 melakukan perulangan pada seluruh dokumen, mencari semua kemunculan pola pada `String` *regex* yang terdapat pada dokumen dan memasukkan nilai dari variabel `a` ke dalam *array*.

5. Baris 8 mengembalikan nilai dari `list`.

5.1.2.6 Impelementasi Kode Program Fitur F10

Proses ini dilakukan untuk mendapatkan nilai fitur F10 dari *tweet* berupa jumlah kemunculan kata diawali dengan huruf besar yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.10.

Kode Program 5.10 Implementasi Proses Fitur F10

```
1  def F10(self):
2      list = []
3      list.append("F10")
4      startwithUppercase = "\\b([A-Z][A-Z]+|[A-Z][a-z]+|[A-
5  Z][0-9]+|[A-Z][a-z0-9]+){1,}\\b"
6      for dokumen in self.docs:
7          a = len(re.findall(startwithUppercase, dokumen[1]))
8          list.append(a)
9      return list
```

Penjelasan dari Kode Program 5.10 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3 menambahkan `String` "F10" pada `list`.
3. Baris 4-5 inisialisasi dari *regex* kata yang diawali dengan huruf besar
4. Baris 6-8 melakukan perulangan pada seluruh dokumen, mencari semua kemunculan pola `String` pada dokumen indeks ke-1 dan memasukkan nilai kemuncullannya kedalam `list`.
5. Baris 9 mengembalikan nilai dari `list`.

5.1.2.7 Impelementasi Kode Program Fitur F11

Proses ini dilakukan untuk mendapatkan nilai fitur F11 dari *tweet* berupa *emoticon* positif yang terdapat dalam satu dokumen tweet. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.11.

Kode Program 5.11 Implementasi Proses Fitur F11

```
1  def F11(self):
2      with
3  open("C:\Users\ACER\PycharmProjects\ProgramBismillah
4  \kamus\emoticon_id.txt", "rb") as f:
5          emotsReader = csv.reader(f, delimiter='|')
6          emots = [emot for emot in emotsReader]
7          list = []
8          list.append("F11")
9          for dokumen in self.docs:
10             found = False
11             for emot in emots:
12                 if emot[1] is not '' and emot[0] in
13 dokumen[1].lower() and int(emot[1]) > 0:
```

```
14                          list.append(1)
15                          found = True
16                          break
17                  if not found:
18                      list.append(0)
19          return list
```

Penjelasan dari Kode Program 5.11 adalah sebagai berikut:

1. Baris 2-6 membuka file *emoticon_id.txt* dan menyimpannya kedalam variabel `emots`
2. Baris 7 menyimpan data sementara dalam bentuk `array` dengan variabel `list`.
3. Baris 8 menambahkan `String` "F11" pada `list`.
4. Baris 9-16 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `found` bernilai `False`, selanjutnya melakukan perulangan pada `emots` dan melakukan seleksi kondisi jika `emot` pada indeks pertama bukan string dan `emot` indeks ke-0 terdapat pada dokumen indeks pertama dan `emot` pada indeks pertama lebih besar daripada 0 maka akan bernilai 1 dan dimasukkan kedalam `list` dan variabel `found` akan bernilai `True`. Selanjutnya melakukan `break`
5. Baris 17-18 Jika tidak variabel `found` maka akan bernilai 0 dan dimasukkan kedalam `list`.
6. Baris 19 mengembalikan nilai dari `list`.

5.1.2.8 Impelementasi Kode Program Fitur F12

Proses ini dilakukan untuk mendapatkan nilai fitur F12 dari *tweet* berupa *emoticon* negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.12.

Kode Program 5.12 Implementasi Proses Fitur F12

```
1   def F12(self):
2       with
3   open("C:\Users\ACER\PycharmProjects\ProgramBismillah
4   \kamus\emoticon_id.txt", "rb") as f:
5           emotsReader = csv.reader(f, delimiter='|')
6           emots = [emot for emot in emotsReader]
7           list = []
8           list.append("F12")
9           for dokumen in self.docs:
10              found = False
11              for emot in emots:
12                  if emot[1] is not '' and emot[0] in
13  dokumen[1].lower() and int(emot[1]) < 0:
14                      list.append(1)
15                      found = True
16                      break
17              if not found:
18                  list.append(0)
19          return list
```

Penjelasan dari Kode Program 5.12 adalah sebagai berikut:

1. Baris 2-6 membuka file *emoticon_id.txt* dan menyimpannya kedalam variabel `emots`.
2. Baris 7 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
3. Baris 8 menambahkan `String` "F12" pada `list`.
4. Baris 9-16 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `found` bernilai `False`, selanjutnya melakukan perulangan pada `emots` dan melakukan seleksi kondisi jika `emot` pada indeks pertama bukan `String` dan `emot` indeks ke-0 terdapat pada dokumen indeks pertama dan `emot` pada indeks pertama kurang dari 0 maka akan bernilai 1 dan dimasukkan kedalam `list` dan variabel `found` akan bernilai `True`. Selanjutnya melakukan `break`.
5. Baris 17-18 Jika tidak sama dengan variabel `found` maka akan bernilai 0 dan dimasukkan kedalam `list`.
6. Baris 19 mengembalikan nilai dari `list`.

5.1.3 Implementasi *Part of Speech*

Proses ini dilakukan untuk mendapatkan nilai fitur F13 sampai F22. Kode Program pada proses ini dapat dilihat pada Kode Program 5.13 hinga 5.22.

5.1.3.1 Impelementasi Kode Program Fitur F13

Proses ini dilakukan untuk mendapatkan nilai fitur F13 dari *tweet* berupa banyaknya kata dengan *postag noun* atau banyaknya kata benda yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.13.

Kode Program 5.13 Implementasi Proses Fitur F13

```
1   def F13(self):
2       with
3   open("C:\Users\ACER\PycharmProjects\ProgramBismillah
4   \Postag\Noun.csv",
5                    "rb") as n:
6           nounReader = csv.reader(n)
7           noun = []
8           for word in nounReader:
9               noun.append(word)
10          list_noun = []
11          list_noun.append("F13")
12          for dokumen in self.docs:
13              jumlah_noun = 0
14              kata =
15  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
16              for k in kata:
17                  ktbenda = True
18                  for word in noun:
```

```
19                          if k == word[0]:
20                              ktbenda = False
21                              break
22                      if not ktbenda:
23                          jumlah_noun += 1
24                  list_noun.append(jumlah_noun)
25              return list_noun
```

Penjelasan dari Kode Program 5.13 adalah sebagai berikut:

1. Baris 2-9 membuka file *Noun.csv* dan menyimpannya kedalam variabel `noun`.
2. Baris 10 menyimpan data sementara dalam bentuk *array* dengan variabel `list_noun`.
3. Baris 11 menambahkan `String` "F13" pada `list_noun`.
4. Baris 12-15 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_noun` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
5. Baris 16-21 melakukan perulangan pada kata, inisialisasi variabel `ktbenda` bernilai `True`, melakukan perulangan kembali pada `noun` Jika `k` sama dengan `word` indeks ke-0 maka `ktbenda` bernilai `False`. Selanjutnya melakukan `break`.
6. Baris 22-24 Jika tidak `ktbenda` maka `jumlah_noun` akan ditambahkan 1 setiap kemunculannya.
7. Baris 25 mengembalikan nilai dari `list_noun`.

5.1.3.2 Impelementasi Kode Program Fitur F14

Proses ini dilakukan untuk mendapatkan nilai fitur F14 dari *tweet* berupa banyaknya kata dengan *postag adjective* atau banyaknya kata sifat yang terdapat dalam satu dokumen tweet. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.14.

Kode Program 5.14 Implementasi Proses Fitur F14

```
1   def F14(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \Postag\Adjective.csv",
4                 "rb") as adjv:
5           adjReader = csv.reader(adjv)
6           adj = []
7           for word in adjReader:
8               adj.append(word)
9           list_adj = []
10          list_adj.append("F14")
11          for dokumen in self.docs:
12              jumlah_adj = 0
13              kata =
14  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
```

```
15              for k in kata:
16                  ktsifat = True
17                  for word in adj:
18                      if k == word[0]:
19                          ktsifat = False
20                          break
21                  if not ktsifat:
22                      jumlah_adj += 1
23              list_adj.append(jumlah_adj)
24          return list_adj
```

Penjelasan dari Kode Program 5.14 adalah sebagai berikut:

1. Baris 2-8 membuka file *Adjective.csv* dan menyimpannya kedalam variabel `adj`.
2. Baris 9 menyimpan data sementara dalam bentuk array dengan variabel `list_adj`.
3. Baris 10 menambahkan `String` "F14" pada `list_adj`.
4. Baris 11-14 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_adj` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
5. Baris 15-20 melakukan perulangan pada kata, inisialisasi variabel `ktsifat` bernilai `True`, melakukan perulangan kembali pada `adj` Jika `k` sama dengan `word` indeks ke-0 maka `ktsifat` bernilai `False`. Selanjutnya melakukan `break`.
6. Baris 21-23 Jika tidak `ktsifat` maka `jumlah_adj` akan ditambahkan 1 setiap kemunculannya.
7. Baris 24 mengembalikan nilai dari `list_adj`

5.1.3.3 Impelementasi Kode Program Fitur F15

Proses ini dilakukan untuk mendapatkan nilai fitur F15 dari *tweet* berupa banyaknya kata dengan *postag verb* atau banyaknya kata kerja yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.15.

Kode Program 5.15 Implementasi Proses Fitur F15

```
1   def F15(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \Postag\Verb.csv",
4                 "rb") as v:
5           verbReader = csv.reader(v)
6           verb = []
7           for word in verbReader:
8               verb.append(word)
9           list_verb = []
10          list_verb.append("F15")
```

```
11          for dokumen in self.docs:
12              jumlah_verb = 0
13              kata =
14  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
15              for k in kata:
16                  ktkerja = True
17                  for word in verb:
18                      if k == word[0]:
19                          ktkerja = False
20                          break
21                  if not ktkerja:
22                      jumlah_verb += 1
23              list_verb.append(jumlah_verb)
24          return list_verb
```

Penjelasan dari Kode Program 5.15 adalah sebagai berikut:

1. Baris 2-8 membuka file *Verb.csv* dan menyimpannya kedalam variabel `verb`.
2. Baris 9 menyimpan data sementara dalam bentuk *array* dengan variabel `list_verb`.
3. Baris 10 menambahkan `String` "F15" pada `list_verb`.
4. Baris 11-14 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_verb` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
5. Baris 15-20 melakukan perulangan pada kata, inisialisasi variabel `ktkerja` bernilai `True`, melakukan perulangan kembali pada `adj` Jika `k` sama dengan `word` indeks ke-0 maka `ktkerja` bernilai `False`. Selanjutnya melakukan `break`.
6. Baris 21-23 Jika tidak `ktkerja` maka `jumlah_verb` akan ditambahkan 1 setiap kemunculannya.
7. Baris 24 mengembalikan nilai dari `list_verb`.

5.1.3.4 Impelementasi Kode Program Fitur F16

Proses ini dilakukan untuk mendapatkan nilai fitur F16 dari *tweet* berupa banyaknya kata dengan *postag* ad*verb* atau banyaknya kata keterangan yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.16.

Kode Program 5.16 Implementasi Proses Fitur F16

```
1   def F16(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \Postag\Adverb.csv",
4                 "rb") as adv:
5           adverbReader = csv.reader(adv)
6           adverb = []
7           for word in adverbReader:
```

```
8               adverb.append(word)
9           list_adverb = []
10          list_adverb.append("F16")
11          for dokumen in self.docs:
12              jumlah_adverb = 0
13              kata =
14  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
15              for k in kata:
16                  ktktg = True
17                  for word in adverb:
18                      if k == word[0]:
19                          ktktg = False
20                          break
21                  if not ktktg:
22                      jumlah_adverb += 1
23              list_adverb.append(jumlah_adverb)
24          return list_adverb
```

Penjelasan dari Kode Program 5.16 adalah sebagai berikut:

1. Baris 2-8 membuka file *Adverb.csv* dan menyimpannya kedalam variabel `adverb`.
2. Baris 9 menyimpan data sementara dalam bentuk *array* dengan variabel `list_adverb`.
3. Baris 10 menambahkan `String` "F16" pada `list_adverb`.
4. Baris 11-14 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_adverb` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
5. Baris 15-20 melakukan perulangan pada kata, inisialisasi variabel `ktktg` bernilai `True`, melakukan perulangan kembali pada `adverb` Jika `k` sama dengan `word` indeks ke-0 maka `ktktg` bernilai `False`. Selanjutnya melakukan `break`.
6. Baris 21-23 Jika tidak `ktkerja` maka `jumlah_adverb` akan ditambahkan 1 setiap kemunculannya.
7. Baris 24 mengembalikan nilai dari `list_adverb`.

5.1.3.5 Impelementasi Kode Program Fitur F17

Proses ini dilakukan untuk mendapatkan nilai fitur F17 dari *tweet* berupa banyaknya kata dengan *postag interjection* yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.16.

Kode Program 5.17 Implementasi Proses Fitur F17

```
1   def F17(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \Postag\Interjection.csv",
4                 "rb") as i:
```

```
5           interReader = csv.reader(i)
6           inter = []
7           for word in interReader:
8               inter.append(word)
9           list_i = []
10          list_i.append("F17")
11          for dokumen in self.docs:
12              jumlah_inter = 0
13              kata =
14  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
15              for k in kata:
16                  ktinter = True
17                  for word in inter:
18                      if k == word[0]:
19                          ktinter = False
20                          break
21                  if not ktinter:
22                      jumlah_inter += 1
23              list_i.append(jumlah_inter)
24          return list_i
```

Penjelasan dari Kode Program 5.17 adalah sebagai berikut:

1. Baris 2-8 membuka file *Interjection.csv* dan menyimpannya kedalam variabel `inter`.
2. Baris 9 menyimpan data sementara dalam bentuk *array* dengan variabel `list_i`.
3. Baris 10 menambahkan `String` "F16" pada `list_i`.
4. Baris 11-14 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_inter` bernilai *0*, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
5. Baris 15-20 melakukan perulangan pada kata, inisialisasi variabel `ktinter` bernilai `True`, melakukan perulangan kembali pada `adverb` Jika `k` sama dengan `word` index ke-0 maka `ktinter` bernilai `False`. Selanjutnya melakukan `break`.
6. Baris 21-23 Jika tidak `ktkerja` maka `jumlah_inter` akan ditambahkan 1 setiap kemunculannya.
7. Baris 24 mengembalikan nilai dari `list_inter`.

5.1.3.6 Impelementasi Kode Program Fitur F18

Proses ini dilakukan untuk mendapatkan nilai fitur F18 dari *tweet* berupa persentase *postag noun* kata benda yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.18.

Kode Program 5.18 Implementasi Proses Fitur F18

```
1   def F18(self):
2       list = []
3       noun = self.F13()
4       F5 = self.F5()
5       list.append("F18")
6       for i in range(len(noun[1:])):
7           try:
8               percentage = float(((noun[1:][i]) / F5[1:][i])
9   * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.18 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F13 dan F5
3. Baris 5 menambahkan `String` "F18" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `noun` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses `try except`, proses `try` melakukan perhitungan `percentage` dari *noun* (kata benda) dan proses `except` dilakukan ketika pembagi bernilai 0 maka `ZeroDivisionError` akan melakukan *handling* `percentage` = 0. Selanjutnya akan dimasukkan ke dalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.3.7 Impelementasi Kode Program Fitur F19

Proses ini dilakukan untuk mendapatkan nilai fitur F19 dari *tweet* berupa persentase *postag adjecive* kata sifat yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.19.

Kode Program 5.19 Implementasi Proses Fitur F19

```
1   def F19(self):
2       list = []
3       adj = self.F14()
4       F5 = self.F5()
5       list.append("F19")
6       for i in range(len(adj[1:])):
7           try:
8               percentage = float(((adj[1:][i]) / F5[1:][i]) *
9   1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.19 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F14 dan F5
3. Baris 5 menambahkan `String` "F19" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `adj` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses `try except`, proses `try` melakukan perhitungan `percentage` dari *adjective* (kata sifat) dan proses `except` dilakukan ketika pembagi bernilai 0 maka `ZeroDivisionError` akan melakukan *handling* `percentage` = 0. Selanjutnya akan dimasukkan ke dalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.3.8 Impelementasi Kode Program Fitur F20

Proses ini dilakukan untuk mendapatkan nilai fitur F20 dari *tweet* berupa persentase *postag verb* kata kerja yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.20.

Kode Program 5.20 Implementasi Proses Fitur F20

```
1   def F20(self):
2       list = []
3       verb = self.F15()
4       F5 = self.F5()
5       list.append("F20")
6       for i in range(len(verb[1:])):
7           try:
8               percentage = float(((verb[1:][i]) / F5[1:][i])
9   * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.20 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F15 dan F5
3. Baris 5 menambahkan `String` "F20" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `verb` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses `try except`, proses `try` melakukan perhitungan `percentage` dari *verb* (kata kerja) dan proses `except` dilakukan ketika pembagi bernilai 0 maka `ZeroDivisionError` akan melakukan

handling `percentage` = 0. Selanjutnya akan dimasukkan ke dalam *array* dan mengambil 4 angka dibelakang koma.

6. Baris 13 mengembalikan nilai dari `list`.

5.1.3.9 Impelementasi Kode Program Fitur F21

Proses ini dilakukan untuk mendapatkan nilai fitur F21 dari *tweet* berupa persentase *postag adverb* kata keterangan yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.21.

Kode Program 5.21 Implementasi Proses Fitur F21

```
1   def F21(self):
2       list = []
3       adv = self.F16()
4       F5 = self.F5()
5       list.append("F21")
6       for i in range(len(adv[1:])):
7           try:
8               percentage = float(((adv[1:][i]) / F5[1:][i]) *
9   1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.21 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F16 dan F5
3. Baris 5 menambahkan `String` "F21" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `adv` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses `try except`, proses `try` melakukan perhitungan `percentage` dari *adverb* (kata keterangan) dan proses `except` dilakukan ketika pembagi bernilai 0 maka `ZeroDivisionError` akan melakukan *handling* `percentage` = 0. Selanjutnya akan dimasukkan ke dalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.3.10 Impelementasi Kode Program Fitur F22

Proses ini dilakukan untuk mendapatkan nilai fitur F22 dari *tweet* berupa persentase *postag interjection* kata interjeksi yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.22.

Kode Program 5.22 Implementasi Proses Fitur F22

```
1   def F22(self):
2       list = []
3       interjection = self.F17()
4       F5 = self.F5()
5       list.append("F22")
6       for i in range(len(interjection[1:])):
7           try:
8               percentage = float(((interjection[1:][i]) /
9   F5[1:][i]) * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.22 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F16 dan F5
3. Baris 5 menambahkan `String` "F21" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `interjection` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses `try except`, proses `try` melakukan perhitungan `percentage` dari *interjection* (kata penegasan) dan proses `except` dilakukan ketika pembagi bernilai 0 maka `ZeroDivisionError` akan melakukan *handling* `percentage` = 0. Selanjutnya akan dimasukkan ke dalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.4 Implementasi *Lexicon-based Features*

Proses ini dilakukan untuk mendapatkan nilai fitur F22 sampai F37. Kode Program pada proses ini dapat dilihat pada Kode Program 5.22 hinga 5.37.

5.1.4.1 Impelementasi Kode Program Fitur F23

Proses ini dilakukan untuk mendapatkan nilai fitur F23 dari *tweet* berupa kata – kata positif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.23.

Kode Program 5.23 Implementasi Proses Fitur F23

```
1   def F23(self):
2       with
3   open("C:\Users\ACER\PycharmProjects\ProgramBismillah
4   \kamus\sentiwords_id.txt", "rb") as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
```

```
9           list = []
10          list.append("F23")
11          for dokumen in self.docs:
12              jumlah_kata_positif = 0
13              kata =
14  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
15              for k in kata:
16                  positif = True
17                  for word in words:
18                      if k == word[0] and word[1] is not ''
19  and int(word[1]) > 0:
20                          positif = False
21                          break
22                  if not positif:
23                      jumlah_kata_positif += 1
24              list.append(jumlah_kata_positif)
25          return list
```

Penjelasan dari Kode Program 5.23 adalah sebagai berikut:

1. Baris 2-8 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 9 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
3. Baris 10 menambahkan `String` "F23" pada `list`.
4. Baris 11-14 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_kata_positif` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
5. Baris 15-21 melakukan perulangan pada kata, inisialisasi variabel `positif` bernilai `True`, selanjutnya melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan `string` dan `word` indeks pada indeks pertama lebih besar daripada 0 maka `positif` sama dengan `False`. Selanjutnya melakukan `break`
6. Baris 21-23 Jika tidak `positif` maka `jumlah_kata_positif` akan ditambahkan 1 setiap kemunculannya.
7. Baris 24 mengembalikan nilai dari `list`.

5.1.4.2 Impelementasi Kode Program Fitur F24

Proses ini dilakukan untuk mendapatkan nilai fitur F24 dari *tweet* berupa kata – kata negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.24.

Kode Program 5.24 Implementasi Proses Fitur F24

```
1   def F24(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\sentiwords_id.txt","rb") as f:
4           wordsReader = csv.reader(f, delimiter=":")
5           words = []
6           for word in wordsReader:
7               words.append(word)
8           list = []
9           list.append("F24")
10          for dokumen in self.docs:
11              jumlah_kata_negatif = 0
12              kata =
13  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
14              for k in kata:
15                  negatif = True
16                  for word in words:
17                      if k == word[0] and word[1] is not ''
18  and int(word[1]) < 0:
19                          negatif = False
20                          break
21                  if not negatif:
22                      jumlah_kata_negatif += 1
23              list.append(jumlah_kata_negatif)
24          return list
```

Penjelasan dari Kode Program 5.24 adalah sebagai berikut:

1. Baris 2-7 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 8 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
3. Baris 9 menambahkan `String` "F24" pada `list`.
4. Baris 11-13 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_kata_negatif` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen index pertama yang mengandung `utf-8`.
5. Baris 14-20 melakukan perulangan pada kata, inisialisasi variabel `negatif` bernilai `True`, selanjutnya melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan `String` dan `word` indeks pada indeks pertama kurang dari 0 maka `negatif` sama dengan `False`. Selanjutnya melakukan `break`
6. Baris 21-23 Jika tidak `negatif` maka `jumlah_kata_negatif` akan ditambahkan 1 setiap kemunculannya.
7. Baris 24 mengembalikan nilai dari `list`

5.1.4.3 Impelementasi Kode Program Fitur F25

Proses ini dilakukan untuk mendapatkan nilai fitur F25 dari *tweet* berupa jumlah kata dengan *postag adjective* (kata sifat) positif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.25.

Kode Program 5.25 Implementasi Proses Fitur F25

```
1   def F25(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\sentiwords_id.txt",
4                 "rb") as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
9
10      with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
11  \Postag\Adjective.csv",
12                "rb") as adj:
13          adjReader = csv.reader(adj)
14          adj = []
15          for a in adjReader:
16              adj.append(a)
17
18          list_adj = []
19          list_adj.append("F25")
20          for dokumen in self.docs:
21              jumlah_adj = 0
22              kata =
23  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
24              for k in kata:
25                  for sifat in adj:
26                      if k == sifat[0]:
27                          pos = True
28                          for word in words:
29                              if k == word[0] and word[1] is
30  not '' and int(word[1]) > 0:
31                                  pos = False
32                                  break
33                          if not pos:
34                              jumlah_adj += 1
35              list_adj.append(jumlah_adj)
36          return list_adj
```

Penjelasan dari Kode Program 5.25 adalah sebagai berikut:

1. Baris 2-8 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 10-16 membuka file *Adjective.csv* dan menyimpannya kedalam variabel `adj`
3. Baris 18 menyimpan data sementara dalam bentuk *array* dengan variabel `list_adj`.
4. Baris 19 menambahkan `String` "F25" pada `list_adj`.

5. Baris 20-23 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_adj` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
6. Baris 24-27 melakukan perulangan pada kata, melakukan perulangan pada `adj`. Jika `k` sama dengan `sifat` pada `adj` indeks ke-0 maka `pos` sama dengan `True`.
7. Baris 28-32 melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan `String` dan `word` indeks pada indeks pertama lebih besar daripada 0 maka `pos` sama dengan `False`. Selanjutnya melakukan `break`.
8. Baris 33-35 Jika tidak *pos* maka `jumlah_adj` akan ditambahkan 1 setiap kemunculannya.
9. Baris 36 mengembalikan nilai dari `list_adj`.

5.1.4.4 Impelementasi Kode Program Fitur F26

Proses ini dilakukan untuk mendapatkan nilai fitur F26 dari *tweet* berupa jumlah kata dengan *postag adjective* (kata sifat) negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.26.

Kode Program 5.26 Implementasi Proses Fitur F26

```
1   Def F26(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\sentiwords_id.txt",
4                 "rb") as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
9
10      with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
11  \Postag\Adjective.csv",
12                "rb") as adj:
13          adjReader = csv.reader(adj)
14          adj = []
15          for a in adjReader:
16              adj.append(a)
17
18          list_adj = []
19          list_adj.append("F26")
20          for dokumen in self.docs:
21              jumlah_adj = 0
22              kata =
23  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
24              for k in kata:
25                  for sifat in adj:
26                      if k == sifat[0]:
```

```
27                              neg = True
28                              for word in words:
29                                  if k == word[0] and word[1] is
30  not '' and int(word[1]) < 0:
31                                          neg = False
32                                          break
33                              if not neg:
34                                  jumlah_adj += 1
35              list_adj.append(jumlah_adj)
36          return list_adj
```

Penjelasan dari Kode Program 5.26 adalah sebagai berikut:

1. Baris 2-8 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 10-16 membuka file *Adjective.csv* dan menyimpannya kedalam variabel `adj`
3. Baris 18 menyimpan data sementara dalam bentuk *array* dengan variabel `list_adj`.
4. Baris 19 menambahkan `String` "F26" pada `list_adj`.
5. Baris 20-23 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_adj` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
6. Baris 24-27 melakukan perulangan pada kata, melakukan perulangan pada `adj`. Jika `k` sama dengan `sifat` pada `adj` indeks ke-0 maka `neg` sama dengan `True`
7. Baris 28-32 melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan `String` dan `word` indeks pada indeks pertama kurang dari 0 maka `neg` sama dengan `False`. Selanjutnya melakukan `break`
8. Baris 33-35 Jika tidak `neg` maka `jumlah_adj` akan ditambahkan 1 setiap kemunculannya.
9. Baris 36 mengembalikan nilai dari `list_adj`

5.1.4.5 Impelementasi Kode Program Fitur F27

Proses ini dilakukan untuk mendapatkan nilai fitur F27 dari *tweet* berupa jumlah kata dengan *postag verb* (kata kerja) positif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.27.

Kode Program 5.27 Implementasi Proses Fitur F27

```
1   def F27(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\sentiwords_id.txt",
4                 "rb") as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
9
10      with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
11  \Postag\Verb.csv",
12                "rb") as v:
13          verbReader = csv.reader(v)
14          verb = []
15          for a in verbReader:
16              verb.append(a)
17
18          list_verb = []
19          list_verb.append("F27")
20          for dokumen in self.docs:
21              jumlah_verb = 0
22              kata =
23  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
24              for k in kata:
25                  for kerja in verb:
26                      if k == kerja[0]:
27                          pos = True
28                          for word in words:
29                              if k == word[0] and word[1] is
30  not '' and int(word[1]) > 0:
31                                  pos = False
32                                  break
33                          if not pos:
34                              jumlah_verb += 1
35              list_verb.append(jumlah_verb)
36          return list_verb
```

Penjelasan dari Kode Program 5.27 adalah sebagai berikut:

1. Baris 2-8 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 10-16 membuka file *Verb.csv* dan menyimpannya kedalam variabel `verb`
3. Baris 18 menyimpan data sementara dalam bentuk *array* dengan variabel `list_verb`.
4. Baris 19 menambahkan `String` "F27" pada `list_verb`.
5. Baris 20-23 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_verb` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.

6. Baris 24-27 melakukan perulangan pada kata, melakukan perulangan pada `verb`. Jika `k` sama dengan `kerja` pada `verb` index ke-0 maka `pos` sama dengan `True`.
7. Baris 28-32 melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan `String` dan `word` indeks pada indeks pertama kurang dari 0 maka `pos` sama dengan `False`. Selanjutnya melakukan `break`
8. Baris 33-35 Jika tidak `pos` maka `jumlah_verb` akan ditambahkan 1 setiap kemunculannya.
9. Baris 36 mengembalikan nilai dari `list_verb`

5.1.4.6 Impelementasi Kode Program Fitur F28

Proses ini dilakukan untuk mendapatkan nilai fitur F28 dari *tweet* berupa jumlah kata dengan *postag verb* (kata kerja) negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.28.

Kode Program 5.28 Implementasi Proses Fitur F28

```
1   def F28(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\sentiwords_id.txt",
4                 "rb") as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
9
10      with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
11  \Postag\Verb.csv",
12                "rb") as v:
13          verbReader = csv.reader(v)
14          verb = []
15          for a in verbReader:
16              verb.append(a)
17
18          list_verb = []
19          list_verb.append("F28")
20          for dokumen in self.docs:
21              jumlah_verb = 0
22              kata =
23  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
24              for k in kata:
25                  for kerja in verb:
26                      if k == kerja[0]:
27                          neg = True
28                          for word in words:
29                              if k == word[0] and word[1] is
30  not '' and int(word[1]) < 0:
31                                  neg = False
32                                  break
33                          if not neg:
```

```
34                                  jumlah_verb += 1
35                          list_verb.append(jumlah_verb)
36                  return list_verb
```

Penjelasan dari Kode Program 5.28 adalah sebagai berikut:

1. Baris 2-8 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 10-16 membuka file *Verb.csv* dan menyimpannya kedalam variabel `verb`
3. Baris 18 menyimpan data sementara dalam bentuk *array* dengan variabel `list_verb`.
4. Baris 19 menambahkan `String` "F28" pada `list_verb`.
5. Baris 20-23 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_verb` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
6. Baris 24-27 melakukan perulangan pada kata, melakukan perulangan pada `verb`. Jika `k` sama dengan `kerja` pada `verb` indeks ke-0 maka `neg` sama dengan `True`
7. Baris 28-32 melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan string dan `word` indeks pada indeks pertama kurang dari 0 maka `neg` sama dengan `False`. Selanjutnya melakukan `break`
8. Baris 33-35 Jika tidak `pos` maka `jumlah_verb` akan ditambahkan 1 setiap kemunculannya.
9. Baris 36 mengembalikan nilai dari `list_verb`

5.1.4.7 Impelementasi Kode Program Fitur F29

Proses ini dilakukan untuk mendapatkan nilai fitur F29 dari *tweet* berupa jumlah kata dengan *postag adverb* (kata keterangan) positif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.29.

Kode Program 5.29 Implementasi Proses Fitur F29

```
1   def F29(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\sentiwords_id.txt",
4                 "rb") as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
9
10      with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
```

```
11  \Postag\Adverb.csv",
12                "rb") as adv:
13          adverbReader = csv.reader(adv)
14          adverb = []
15          for a in adverbReader:
16              adverb.append(a)
17
18          list_adverb = []
19          list_adverb.append("F29")
20          for dokumen in self.docs:
21              jumlah_adverb = 0
22              kata =
23  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
24              for k in kata:
25                  for keterangan in adverb:
26                      if k == keterangan[0]:
27                          pos = True
28                          for word in words:
29                              if k == word[0] and word[1] is
30  not '' and int(word[1]) > 0:
31                                  pos = False
32                                  break
33                          if not pos:
34                              jumlah_adverb += 1
35              list_adverb.append(jumlah_adverb)
36          return list_adverb
```

Penjelasan dari Kode Program 5.29 adalah sebagai berikut:

1. Baris 2-8 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 10-16 membuka file *Aderb.csv* dan menyimpannya kedalam variabel `adverb`
3. Baris 18 menyimpan data sementara dalam bentuk *array* dengan variabel `list_adverb`.
4. Baris 19 menambahkan `String` "F29" pada `list_adverb`.
5. Baris 20-23 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_adverb` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
6. Baris 24-27 melakukan perulangan pada kata, melakukan perulangan pada `adverb`. Jika `k` sama dengan `keterangan` pada `adverb` indeks ke-0 maka `pos` sama dengan `True`
7. Baris 28-32 melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan `String` dan `word` indeks pada indeks pertama kurang dari 0 maka `pos` sama dengan `False`. Selanjutnya melakukan `break`
8. Baris 33-35 Jika tidak `pos` maka `jumlah_adverb` akan ditambahkan 1 setiap kemunculannya.

9. Baris 36 mengembalikan nilai dari `list_adverb`

5.1.4.8 Impelementasi Kode Program Fitur F30

Proses ini dilakukan untuk mendapatkan nilai fitur F30 dari *tweet* berupa jumlah kata dengan *postag adverb* (kata keterangan) negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.30.

Kode Program 5.30 Implementasi Proses Fitur F30

```
1   def F30(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\sentiwords_id.txt",
4                 "rb") as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
9
10      with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
11  \Postag\Adverb.csv",
12                "rb") as adv:
13          adverbReader = csv.reader(adv)
14          adverb = []
15          for a in adverbReader:
16              adverb.append(a)
17
18          list_adverb = []
19          list_adverb.append("F30")
20          for dokumen in self.docs:
21              jumlah_adverb = 0
22              kata =
23  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
24              for k in kata:
25                  for keterangan in adverb:
26                      if k == keterangan[0]:
27                          neg = True
28                          for word in words:
29                              if k == word[0] and word[1] is
30  not '' and int(word[1]) < 0:
31                                  neg = False
32                                  break
33                          if not neg:
34                              jumlah_adverb += 1
35              list_adverb.append(jumlah_adverb)
36          return list_adverb
```

Penjelasan dari Kode Program 5.30 adalah sebagai berikut:

1. Baris 2-8 membuka file *sentiwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 10-16 membuka file *Aderb.csv* dan menyimpannya kedalam variabel `adverb`
3. Baris 18 menyimpan data sementara dalam bentuk *array* dengan variabel `list_adverb`.

4. Baris 19 menambahkan `String` "F29" pada `list_adverb`.
5. Baris 20-23 melakukan perulangan pada seluruh dokumen, dengan membuat inisialisasi awal pada variabel `jumlah_adverb` bernilai 0, selanjutnya melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`.
6. Baris 24-27 melakukan perulangan pada kata, melakukan perulangan pada `adverb`. Jika `k` sama dengan `keterangan` pada `adverb` index ke-0 maka `neg` sama dengan `True`
7. Baris 28-32 melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 dan `word` indeks pertama bukan `String` dan `word` indeks pada indeks pertama kurang dari 0 maka `neg` sama dengan `False`. Selanjutnya melakukan `break`
8. Baris 33-35 Jika tidak `pos` maka `jumlah_adverb` akan ditambahkan 1 setiap kemunculannya.
9. Baris 36 mengembalikan nilai dari `list_adverb`

5.1.4.9 Impelementasi Kode Program Fitur F31

Proses ini dilakukan untuk mendapatkan nilai fitur F31 dari *tweet* berupa peersentase kata dengan *postag adjective* (kata sifat) positif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.31.

Kode Program 5.31 Implementasi Proses Fitur F31

```
1   def F31(self):
2       list = []
3       tagadj_pos = self.F25()
4       F23 = self.F23()
5       list.append("F31")
6       for i in range(len(tagadj_pos[1:])):
7           try:
8               percentage = float(((tagadj_pos[1:][i]) /
9   F23[1:][i]) * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.31 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F25 dan F23
3. Baris 5 menambahkan `String` "F31" pada `list`.

4. Baris 6 melakukan perulangan pada panjang `tagadj_pos` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses *handling* `try except`, kondisi `try` adalah melakukan proses persentase, dan kondisi `except` menangani jika terjadi error pada pembagian 0 maka persentase bernilai 0. proses perhitungan persentase dari `percentage` dimasukkan kedalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.4.10 Impelementasi Kode Program Fitur F32

Proses ini dilakukan untuk mendapatkan nilai fitur F32 dari *tweet* berupa peersentase kata dengan *postag adjective* (kata sifat) negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.32.

Kode Program 5.32 Implementasi Proses Fitur F32

```
1   def F32(self):
2       list = []
3       tagadj_neg = self.F26()
4       F24 = self.F24()
5       list.append("F32")
6       for i in range(len(tagadj_neg[1:])):
7           try:
8               percentage = float(((tagadj_neg[1:][i]) /
9   F24[1:][i]) * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.32 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F25 dan F23
3. Baris 5 menambahkan `String` “F32” pada `list`.
4. Baris 6 melakukan perulangan pada panjang `tagadj_neg` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses *handling* `try except`, kondisi `try` adalah melakukan proses persentase, dan kondisi `except` menangani jika terjadi error pada pembagian 0 maka persentase bernilai 0. proses perhitungan persentase dari `percentage` dimasukkan kedalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.4.11 Impelementasi Kode Program Fitur F33

Proses ini dilakukan untuk mendapatkan nilai fitur F33 dari *tweet* berupa peersentase kata dengan *postag verb* (kata kerja) positif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.33.

Kode Program 5.33 Implementasi Proses Fitur F33

```
1   def F33(self):
2       list = []
3       tagverb_pos = self.F27()
4       F23 = self.F23()
5       list.append("F33")
6       for i in range(len(tagverb_pos[1:])):
7           try:
8               percentage = float(((tagverb_pos[1:][i]) /
9   F23[1:][i]) * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.33 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F25 dan F23
3. Baris 5 menambahkan `String` "F33" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `tagverb_pos` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses *handling* `try except`, kondisi `try` adalah melakukan proses persentase, dan kondisi `except` menangani jika terjadi error pada pembagian 0 maka persentase bernilai 0. proses perhitungan persentase dari `percentage` dimasukkan kedalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.4.12 Impelementasi Kode Program Fitur F34

Proses ini dilakukan untuk mendapatkan nilai fitur F34 dari *tweet* berupa peersentase kata dengan *postag verb* (kata kerja) negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.34.

Kode Program 5.34 Implementasi Proses Fitur F34

```
1   def F34(self):
2       list = []
3       tagverb_neg = self.F28()
4       F24 = self.F24()
5       list.append("F34")
6       for i in range(len(tagverb_neg[1:])):
7           try:
8               percentage = float(((tagverb_neg[1:][i]) /
9   F24[1:][i]) * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.34 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F25 dan F23
3. Baris 5 menambahkan `String` "F34" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `tagverb_neg` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses *handling* `try except`, kondisi `try` adalah melakukan proses persentase, dan kondisi `except` menangani jika terjadi error pada pembagian 0 maka persentase bernilai 0. proses perhitungan persentase dari `percentage` dimasukkan kedalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.4.13 Impelementasi Kode Program Fitur F35

Proses ini dilakukan untuk mendapatkan nilai fitur F35 dari *tweet* berupa peersentase kata dengan *postag adverb* (kata kerja) positif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.35.

Kode Program 5.35 Implementasi Proses Fitur F35

```
1   def F35(self):
2       list = []
3       tagadv_pos = self.F29()
4       F23 = self.F23()
5       list.append("F35")
6       for i in range(len(tagadv_pos[1:])):
7           try:
8               percentage = float(((tagadv_pos[1:][i]) /
```

```
9   F23[1:][i]) * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.35 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.
2. Baris 3-4 melakukan pemanggilan *function* F25 dan F23
3. Baris 5 menambahkan `String` "F35" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `tagadv_pos` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses *handling* `try except`, kondisi `try` adalah melakukan proses persentase, dan kondisi `except` menangani jika terjadi error pada pembagian 0 maka persentase bernilai 0. proses perhitungan persentase dari `percentage` dimasukkan kedalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.4.14 Impelementasi Kode Program Fitur F36

Proses ini dilakukan untuk mendapatkan nilai fitur F36 dari *tweet* berupa peersentase kata dengan *postag adverb* (kata kerja) negatif yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.36.

Kode Program 5.36 Implementasi Proses Fitur F36

```
1   def F36(self):
2       list = []
3       tagadv_neg = self.F30()
4       F24 = self.F24()
5       list.append("F36")
6       for i in range(len(tagadv_neg[1:])):
7           try:
8               percentage = float(((tagadv_neg[1:][i]) /
9   F24[1:][i]) * 1)
10          except ZeroDivisionError:
11              percentage = 0
12          list.append(round(percentage, 4))
13      return list
```

Penjelasan dari Kode Program 5.36 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.

2. Baris 3-4 melakukan pemanggilan *function* F25 dan F23
3. Baris 5 menambahkan `String` "F36" pada `list`.
4. Baris 6 melakukan perulangan pada panjang `tagadv_neg` indeks pertama hingga selesai.
5. Baris 7-12 melakukan proses *handling* `try except`, kondisi `try` adalah melakukan proses persentase, dan kondisi `except` menangani jika terjadi error pada pembagian 0 maka persentase bernilai 0. proses perhitungan persentase dari `percentage` dimasukkan kedalam *array* dan mengambil 4 angka dibelakang koma.
6. Baris 13 mengembalikan nilai dari `list`.

5.1.4.15 Impelementasi Kode Program Fitur F37

Proses ini dilakukan untuk mendapatkan nilai fitur F37 dari *tweet* berupa jumlah kemunculan kata penegasan (*intensifier*) yang terdapat dalam satu dokumen *tweet*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.37.

Kode Program 5.37 Implementasi Proses Fitur F37

```
1   def F37(self):
2       with open('C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \kamus\penegasan_boosterwords_id.txt',
4                 'rb') as f:
5           wordsReader = csv.reader(f, delimiter=":")
6           words = []
7           for word in wordsReader:
8               words.append(word)
9           list = []
10          list.append("F37")
11          for dokumen in self.docs:
12              kata =
13  word_tokenize(dokumen[1].lower().decode('utf-8', 'ignore'))
14              jumlah_kata_intensifier = 0
15              for k in kata:
16                  for word in words:
17                      if k == word[0]:
18                          jumlah_kata_intensifier += 1
19                          break
20              list.append(jumlah_kata_intensifier)
21          return list
```

Penjelasan dari Kode Program 5.37 adalah sebagai berikut:

1. Baris 2-8 membuka file *penegasan_boosterwords_id.txt* dan menyimpannya kedalam variabel `words`
2. Baris 9 menyimpan data sementara dalam bentuk *array* dengan variabel `list`.

3. Baris 10 menambahkan `String` "F37" pada `list`.
4. Baris 11-14 melakukan perulangan pada seluruh dokumen, dengan melakukan tokenisasi pada kata dan menjadikan huruf kecil. Selanjutnya melakukan konversi kata pada dokumen indeks pertama yang mengandung `utf-8`, selanjutnya membuat inisialisasi awal pada variabel `jumlah_kata_intensifier` bernilai *0*
5. Baris 15-20 melakukan perulangan pada kata, selanjutnya melakukan perulangan pada `words` dan melakukan seleksi kondisi jika `k` sama dengan `word` pada indeks ke-0 maka `jumlah_kata_intensifier` ditambahkan 1. Selanjutnya melakukan `break`, Selanjutnya memasukkan nilai sementara `jumlah_kata_intensifier` kedalam array.
6. Baris 21 mengembalikan nilai dari `list`.

5.1.5 Implementasi *Bag of Words*

Pada implementasi *Bag of Words* (BoW) *features*, terdapat beberapa proses yang dilakukan meliputi proses preprocessing seperti tokenisasi, filterisasi, *stemming*, kemudian proses menghitung *term frequency* dengan menggunakan *raw term frequency* yaitu berdasarkan kemunculan kata pada suatu dokumen. Kode Program pada proses ini dapat dilihat pada Kode Program 5.38.

Kode Program 5.38 Implementasi Proses *Bag of Words*

```
1   def preprocessing(self):
2       # case folding & cleaning
3       for dokumen in self.docs:
4           casefolding = dokumen[1].lower()
5           linkRegex = "(https?:\/\/(?:www\.|(?!www))[a-zA-Z0-
6   9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,}|" \
7             "www\.[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-
8   9]\.[^\s]{2,}|" \
9             "https?:\/\/(?:www\.|(?!www))[a-zA-Z0-
10  9]\.[^\s]{2,}|www\.[a-zA-Z0-9]\.[^\s]{2,})"
11          nolink = re.sub(linkRegex, '', casefolding)
12          dokumen[1] = nolink
13
14      punctuations = '''!()-
15  []{};:'"\,<>./?@#$%^&*_~0123456789'''
16      for kata in self.docs:
17          no_punct = ""
18          for char in kata[1]:
19              if char not in punctuations:
20                  no_punct = no_punct + char
21              elif char in punctuations:
22                  no_punct = no_punct + ' '
23          kata[1] = no_punct
24
25      # stemming
26      stemmer = factory.create_stemmer()
27      for word in self.docs:
28          word[1] = stemmer.stem(str(word[1].lower()))
29          #print word[1]
30
```

```
31      # stopword
32      stopword =
33  open('C:\Users\ACER\PycharmProjects\ProgramBismillah
34  \kamus\modifikasi_stopword.txt', 'r').read()
35      stopwordlist = re.split(r'\n', stopword)
36      for kolom in self.docs:
37          wordstop = re.split(r'\s', kolom[1])
38          fil = ''
39          for b in wordstop:
40              if b not in stopwordlist:
41                  fil += ' ' + b
42          kolom[1] = fil
43
44      # tokenize
45      for value in self.docs:
46          baris = re.split(r'\s', value[1])
47          value[1] = baris
48
49      for dat in self.docs:
50          for a in dat[1]:
51              if a == '':
52                  dat[1].remove(a)
53
54      # cek kata unik as fitur
55      unik = []
56      for term in self.docs:
57          for unique in term[1]:
58              if unique not in unik:
59                  unik.append(unique)
60          termfeature = sorted(unik)
61      # print termfeature
62
63      doc_counts = []
64      for terms in termfeature:
65          doc_counts.append(0)
66          rawterm = []
67          # praproses = []
68          rawterm.append(terms)
69          for fdoc in self.docs:
70              ct = fdoc[1].count(str(terms))
71              rawterm.append(ct)
72          return rawterm
```

Penjelasan dari Kode Program 5.38 adalah sebagai berikut:

1. Baris 3-12 melakukan *casefolding* pada dokumen, selanjutnya melakukan *cleaning* dengan menghilangkan *URL* dan menghilangkan semua karakter yang bukan alphabet dan spasi yang terdapat dalam dokumen indeks pertama, dan hasil *cleaning* akan disimpan dalam `dokumen[1]`.
2. Baris 14-23 melakukan proses *cleaning* kembali dengan menghilangkan semua *punctuations* yang terdapat dalam *tweet* yang kemudian disimpan pada `kata[1]`.

3. Baris 25-29 melakukan *stemming* pada dokumen yang telah dilakukan *cleaning* sebelumnya dan melakukan split pada semua kata dan dilakukan *stemming* dan hasil dari *stemming* disimpan kedalam `word[1]`.
4. Baris 31-42 melakukan proses *stopword* dengan membukan file stopword. Selanjutnya dilakukan pengecekkan pada seluruh dokumen dan jika kata dalam dokumen sama dengan kata dalam *stopword* maka kata dalam dokumen dihapus dan diganti dengan spasi. Hasil dari *stopword* disimpan kedalam `kolom[1]`.
5. Baris 44-52 melakukan tokenisasi pada semua dokumen *tweet* sehingga menjadi *token* yang kemudian disimpan pada `token[1]` dan selanjutya menghilangkan semua spasi yang menjadi token dan disimpan pada `dat[1]`.
6. Baris 55-60 melakukan pengecekan kata duplikat pada `term[1]` untuk dijadikan fitur dan kemudian diurutkan berdasarkan abjad.
7. Baris 63-71 melakukan perhitungan kemunculan kata pada seluruh dokumen.
8. Baris 72 mengembalikan nilai `rawterm`.

5.1.6 Implementasi *Pearson Correlation Coeficient*

Pada implementasi *Pearson Correlation Coeficient*, yaitu dilakukan proses untuk mendapatkan nilai *Pearson's* yang digunakan untuk melakukan seleksi fitur. Kode Program pada proses ini dapat dilihat pada Kode Program 5.39 hingga 5.44.

5.1.6.1 Impelementasi Kode Program Nilai Variabel X

Proses ini dilakukan untuk mendapatkan total jumlah variabel X yang digunakan untuk menghitung nilai *Pearson's*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.39

Kode Program 5.39 Implementasi Nilai Variabel X

```
1  def x_value(self):
2      sumx = []
3      for fitur in self.fiturs:
4          x = sum([float(x) for x in fitur[1:401]])
5          sumx.append(x)
6      return sumx
```

Penjelasan dari Kode Program 5.39 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk *array* dengan variabel `sumx`.
2. Baris 3-5 melakukan perulangan pada seluruh fitur, dan melakukan proses penjumlahan fitur (sigma) untuk indeks *array* tertentu. Dan memasukkan nilai `x` kedalam *array* `sumx`.

3. Baris 6 mengembalikan nilai dari `sumx`.

5.1.6.2 Impelementasi Kode Program Nilai Variabel X^2

Proses ini dilakukan untuk mendapatkan total jumlah variabel X^2 yang digunakan untuk menghitung nilai *Pearson's*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.40

Kode Program 5.40 Implementasi Nilai Variabel X^2

```
1  def x_pow(self):
2      sumxp = []
3      for fitur in self.fiturs:
4          xval_pow = sum([float(x)**2 for x in fitur[1:401]])
5          sumxp.append(xval_pow)
6      return sumxp
```

Penjelasan dari Kode Program 5.40 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `sumxp`.
2. Baris 3-5 melakukan perulangan pada seluruh fitur, dan melakukan proses penjumlahan fitur (sigma) untuk indeks *array* tertentu. Dan memasukkan nilai `x` kedalam *array* `sumxp`.
3. Baris 6 mengembalikan nilai dari `sumxp`.

5.1.6.3 Impelementasi Kode Program Nilai Variabel Y

Proses ini dilakukan untuk mendapatkan total jumlah variabel Y yang digunakan untuk menghitung nilai *Pearson's*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.41.

Kode Program 5.41 Implementasi Nilai Variabel Y

```
1  def y_value(self):
2      sumy = []
3      for dokumen in self.docs:
4          if dokumen[2].lower() == "positif":
5              sumy.append(1)
6          else:
7              sumy.append(0)
8      sigma_y = sum(sumy)
9      return sigma_y
```

Penjelasan dari Kode Program 5.41 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `sumy`.
2. Baris 3-7 melakukan perulangan pada seluruh dokumen dan melakukan seleksi kondisi Jika dokumen indeks ke – 2 sama dengan positif maka akan ditambahkan kedalam `sumy` dengan nilai 1 jika tidak maka nilai 0.

3. Baris 8 melakukan penjumlahan untuk seluruh nilai pada `sumy`.
4. Baris 9 mengembalikan nilai dari `sumy`.

5.1.6.4 Impelementasi Kode Program Nilai Variabel Y^2

Proses ini dilakukan untuk mendapatkan total jumlah variabel Y yang digunakan untuk menghitung nilai *Pearson's*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.42.

Kode Program 5.42 Implementasi Nilai Variabel Y^2

```
1   def y_pow(self):
2       sumy = []
3       for dokumen in self.docs:
4           if dokumen[2].lower() == 'positif':
5               sumy.append(1)
6           else:
7               sumy.append(0)
8       ypow = []
9       for x in sumy:
10          ypow.append(x ** 2)
11      sigmapow = sum(ypow)
12      return sigmapow
```

Penjelasan dari Kode Program 5.42 adalah sebagai berikut:

1. Baris 2 menyimpan data sementara dalam bentuk array dengan variabel `sumy`.
2. Baris 3-7 melakukan perulangan pada seluruh dokumen dan melakukan seleksi kondisi Jika dokumen indeks ke – 2 sama dengan positif maka akan ditambahkan kedalam `sumy` dengan nilai 1 jika tidak maka nilai 0.
3. Baris 8 menyimpan data sementara dalam bentuk *array* dengan variabel `ypow`.
4. Baris 9-11 melakukan perulangan `x` pada `sumy`, selanjutnya melakukan nilai kuadrat pada nilai `x` dan menjumlahkan semua nilai pada `ypow`.
5. Baris 12 mengembalikan nilai dari `sigmapow`

5.1.6.5 Impelementasi Kode Program Nilai Variabel XY

Proses ini dilakukan untuk mendapatkan total jumlah variabel XY yang digunakan untuk menghitung nilai *Pearson's*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.43.

Kode Program 5.43 Implementasi Nilai Variabel XY

```
1   def xy_val(self):
2       list_y = []
3       for dokumen in self.docs:
4           if dokumen[2].lower() == 'positif':
5               list_y.append(1)
6           else:
7               list_y.append(0)
8       sigmaxy = []
9       for xyfitur in self.fiturs:
10          xy = []
11          for i, x in enumerate(list_y):
12              xy.append(float(xyfitur[1:][i]) * list_y[i])
13          sigmaxy.append(sum(xy))
14      return sigmaxy
```

Penjelasan dari Kode Program 5.43 adalah sebagai berikut:

1. Baris 2 menyimpan nilai *y* sementara dalam bentuk *array* dengan variabel `list_y`.
2. Baris 3-7 melakukan perulangan pada seluruh dokumen dan melakukan seleksi kondisi Jika dokumen indeks ke – 2 sama dengan positif maka akan ditambahkan kedalam `sumy` dengan nilai 1 jika tidak maka nilai 0.
3. Baris 9 menyimpan nilai penjumlahan `xy` sementara dalam bentuk *array* dengan variabel `sigmaxy`.
4. Baris 10-13 melakukan perulangan pada seluruh fitur latih, selanjutnya membuat peyimpanan nilai `xy` sementara dalam bentuk *array*, dan akan dilakukan perulangan kembali pada `list_y` dan melakukan proses perkalian `xyfitur` dengan `list_y`.
5. Baris 14 melakukan penjumlahan nilai `xy`.
6. Baris 15 mengembalikan nilai dari `sigmaxy`.

5.1.6.6 Impelementasi Kode Program Nilai Variabel *Pearson's*

Proses ini dilakukan untuk mendapatkan nilai *Pearson's* yang digunakan untuk seleksi fitur. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.44.

Kode Program 5.44 Implementasi Nilai Variabel XY

```
1   def pearson (self):
2       list_y = []
3       for dokumen in self.docs:
4           if dokumen[2].lower() == 'positif':
5               list_y.append(1)
6           else:
7               list_y.append(0)
8       # Assume len(x) == len(y)
```

```
9       n = len(list_y)
10      sigmax = self.x_value()
11      sigmay = self.y_value()
12      sigmaxpow = self.x_pow()
13      sigmaypow = self.y_pow()
14      sigmaxy = self.xy_val()
15      rpxy_list = []
16      for i, v in enumerate(sigmax):
17          rpxy_top = (n * sigmaxy[i]) - (sigmax[i] * sigmay)
18          rpxy_bot = math.sqrt((n * sigmaxpow[i]) -
19  (sigmax[i] ** 2)) * math.sqrt((n * sigmaypow) - (sigmay **
20  2))
21          if rpxy_bot == 0:
22              rpxy_list.append(0)
23          else:
24              rpxy_list.append(abs(rpxy_top / rpxy_bot))
25
26      x = []
27      for fitur in self.fiturs:
28          x.append(fitur[0])
29          a = [x ,rpxy_list]
30          pairs = zip(*a)
31          pearson = []
32          for set in pairs:
33              a = list(set)
34              pearson.append(a)
35      return pearson
```

Penjelasan dari Kode Program 5.44 adalah sebagai berikut:

1. Baris 2-7 melakukan pengambilan pada variabel `y` dan disimpan ke dalam `list_y`
2. Baris 8-9 menganggap panjang variabel `x` dan variabel `y` sama, maka `n` sama dengan panjang dari `list_y`
3. Baris 10-14 melakukan pemanggilan nilai dari setiap *function* `x_value`, `y_value`, `x_pow`, `y_pow` dan `xy_value`.
4. Baris 15-24 melakukan proses perhitungan nilai *Pearson's* dan menyimpannya dalam *array* `rpxy_list`
5. Baris 26-34 melakukan proses penggabungan antara `fitur[0]` dan nilai *Pearson's* untuk mengetahui nilai fitur dan nilai *Pearson's* berdasarkan `fitur[0]`.
6. Baris 35 mengembalikan nilai dari `pearson`.

5.1.7 Implementasi Seleksi Fitur

Pada Implementasi Seleksi Fitur terdapat beberapa proses yang dilakukan yaitu mengurutkan nilai *Pearson's* dan mengambil nilai fitur berdasarkan nilai *Pearson's*. Kode Program pada proses ini dapat dilihat pada Kode Program 5.45 dan 5.46.

5.1.7.1 Impelementasi Kode Program Seleksi Fitur

Proses ini dilakukan untuk mendapatkan nilai *Pearson's* yang digunakan untuk seleksi fitur. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.45.

Kode Program 5.45 Implementasi Seleksi Fitur

```
1   def seleksifitur(self):
2       sortedx = sorted(self.selection, key=lambda x: x[1],
3   reverse=True)
4       panjangfitur = len(sortedx)
5       sample = math.ceil(panjangfitur * 0.25)
6       pearsonsample = sortedx[:int(sample)]
7       return pearsonsample
```

Penjelasan dari Kode Program 5.45 adalah sebagai berikut:

1. Baris 2-3 melakukan penyortiran berdasarkan nilai *Pearson's* dari terbesar hingga terkecil
2. Baris 4 menghitung panjang dari `sorted`
3. Baris 5-6 mengambil nilai fitur sebanyak 25% dari total keseluruhan fitur
4. Baris 7 mengembalikan nilai `pearsonsample`

5.1.7.2 Impelementasi Kode Program Pengambilan Fitur

Proses ini dilakukan untuk mendapatkan nilai *Pearson's* yang digunakan untuk seleksi fitur. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.46.

Kode Program 5.46 Implementasi Pengambilan Fitur

```
1   def fitur(self):
2       with open("C:\Users\ACER\PycharmProjects\ProgramSkripsi
3   \output\extractfitur.csv", "rb") as f:
4           fiturReader = csv.reader(f)
5           fiturs = [fitur for fitur in fiturReader]
6           list_fitur = []
7           sorting = self.seleksifitur()
8           for urut in sorting:
9               found = False
10              for fitur in fiturs:
11                  if fitur[0] == urut[0]:
12                      list_fitur.append(fitur)
13                      found = True
14                      break
15              if not found:
16                  list_fitur.append(0)
17          sortedx = sorted(list_fitur, key=lambda x: x[0][0],
18  reverse=False)
19      return sorted
```

Penjelasan dari Kode Program 5.46 adalah sebagai berikut:

1. Baris 2-7 melakukan proses membuka file dari nilai ekstraksi fitur *ensemble* dan *Bag of Words* dan menyimpan nilai fitur kedalam variabel `fiturs` dan selanjutnya memnggil *function* dari `seleksifitur`.
2. Baris 8-17 melakukan proses perulangan pada variabel `sorting` dan perulangan pada variabel `fiturs`, melakukan kondisi untuk mengambil nilai fitur yang terseleksi berdasarkan fitur yang disimpan pada file *Pearson's* dan pada file ekstraksi fitur.
3. Baris 9 mengembalikan nilai `sorted`

5.1.8 Implementasi *Naïve Bayes*

Pada implementasi *Naïve Bayes* proses yang dilakukan meliputi proses menghitung nilai *Prior*, *Bernoulli Naïve Bayes Training*, *Bernoulli Naïve Bayes Testing, Gaussian Naïve Bayes Training, Gaussian Naïve Bayes Testing, Multinomial Naïve Bayes Training* dan *Multinomial Naïve Bayes Testing* dan nilai *Posterior*. Kode Program pada proses ini dapat dilihat pada Kode Program 5.47 hingga 5.54.

5.1.8.1 Impelementasi Kode Program Probabilitas *Prior*

Proses pertama pada *Naïve Bayes* adalah menghitung prior. Nilai prior diperoleh dari perhitungan peluang masing-masing kelas pada data latih. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.47.

Kode Program 5.47 Implementasi Probabilitas *Prior*

```
1   def Prior(self):
2       dokumen_pos = 0
3       dokumen_neg = 0
4       for dokumen in self.docs:
5           if dokumen[2].lower() == 'positif':
6               dokumen_pos += 1
7           else:
8               dokumen_neg += 1
9       prior_pos = float(dokumen_pos) / len(self.docs)
10      prior_neg = float(dokumen_neg) / len(self.docs)
11      self.prior_pos = prior_pos
12      self.prior_neg = prior_neg
```

Penjelasan dari Kode Program 5.46 adalah sebagai berikut:

1. Baris 2-3 Melakukan inisialisasi awal pada variabel `dokumen_pos` dan `dokumen_neg` sama dengan 0.
2. Baris 4-8 Melakukan perulangan pada seluruh dokumen dan melakukan seleksi kondisi, jika `dokumen[2]` sama dengan positif maka `dokumen_pos` akan bertambah satu (*increment*) jika tidak `dokumen_neg` akan bertambah satu (*increment*).

3. Baris 9-12 Melakukan proses perhitungan *prior* pada kelas positif dan pada kelas negatif dan masing – masing akan disimpan pada `self.prior_pos` dan `self.prior_neg`

5.1.8.2 Impelementasi Kode Program *Bernoulli Naïve Bayes Training*

Proses *training* pada *Bernoulli Naïve Bayes* dilakukan untuk mendapatkan peluang fitur dengan syarat kelas. Pada *Bernoulli Naïve Bayes Training* fitur yang digunakan hanya fitur latih dengan data *binary*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.48.

Kode Program 5.48 Implementasi *Bernoulli Naïve Bayes Training*

```
1   def BernoulliTrain(self):
2       doc_pos = 0
3       doc_neg = 0
4       for doc in self.docs:
5           if doc[2].lower() == 'positif':
6               doc_pos += 1
7           else:
8               doc_neg += 1
9       tot_neg = []
10      tot_pos = []
11      for fitur in self.fiturs:
12          if (fitur[0] == 'F1') or (fitur[0] == 'F2') or
13  (fitur[0] == 'F3') or (
14                  fitur[0] == 'F4') or (fitur[0] == 'F11') or
15  (fitur[0] == 'F12'):
16              counter = 0
17              pos = 0
18              neg = 0
19              for doc in self.docs:
20                  if doc[2].lower() == 'positif':
21                      pos += float(fitur[1:][counter])
22                  else:
23                      neg += float(fitur[1:][counter])
24                  counter +=1
25              tot_pos.append(pos)
26              tot_neg.append(neg)
27          pf_neg = []
28          for neg in tot_neg:
29              pf_neg.append((neg+1)/(doc_neg+2))
30          pf_pos = []
31          for pos in tot_pos:
32              pf_pos.append((pos+1)/(doc_pos+2))
33      self.bpf_pos = pf_pos
34      self.bpf_neg = pf_neg
```

Penjelasan dari Kode Program 5.46 adalah sebagai berikut:

1. Baris 2-8 melakukan perhitungan jumlah kelas positif dan kelas negatif pada fitur latih yang akan digunakan untuk menghitung *conditional probability* pada data latih.

2. Baris 11-26 melakukan proses perulangan pada fitur dengan melakukan kondisi jika pada file fitur terdapat F1-F4 dan F11-F12 maka akan dilakukan proses perhitungan nilai fitur positif dan nilai fitur negatif.
3. Baris 27-32 melakukan proses perhitungan peluang fitur dengan syarat kelas (*conditional probability*) pada fitur positif dan fitur negatif.
4. Baris 33-34 menyimpan nilai peluang fitur dengan syarat kelas pada `self`.

5.1.8.3 Impelementasi Kode Program *Bernoulli Naïve Bayes Testing*

Proses *testing* pada *Bernoulli Naïve Bayes* dilakukan untuk mendapatkan probabilitas *likelihood*. Nilai peluang fitur dengan syarat kelas yang telah dihitung pada *training* akan digunakan pada proses ini. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.49.

Kode Program 5.49 Implementasi *Bernoulli Naïve Bayes Testing*

```
1   def BernoulliTest(self):
2       btrain = self.BernoulliTrain()
3       bpf_pos = []
4       for c in self.bpf_pos:
5           bpf_pos.append(c)
6
7       bpf_neg = []
8       for d in self.bpf_neg:
9           bpf_neg.append(d)
10
11      ab = []
12      for fitur in self.fituruji:
13          if (fitur[0] == 'F1') or (fitur[0] == 'F2') or
14  (fitur[0] == 'F3') or (
15                  fitur[0] == 'F4') or (fitur[0] == 'F11') or
16  (fitur[0] == 'F12'):
17              ab.append(fitur)
18      d_neg = []
19      d_pos = []
20      for i, uji in enumerate(ab):
21          f_pd_neg = []
22          f_pd_pos = []
23          for x in uji[1:]:
24              f_pd_neg.append(round((bpf_neg[i] ** float(x))
25  * ((1 - bpf_neg[i]) ** (1 - float(x))), 5))
26              f_pd_pos.append(round((bpf_pos[i] ** float(x))
27  * ((1 - bpf_pos[i]) ** (1 - float(x))), 5))
28          d_neg.append(f_pd_neg)
29          d_pos.append(f_pd_pos)
30
31      positif = []
32      for i, pos in enumerate(d_pos):
33          positif.append(pos)
34
35      blikelihood_pos = [reduce(mul, i) for i in
36  zip(*positif)]
37      self.blikelihood_pos = blikelihood_pos
38
39      negatif = []
```

```
40      for i, neg in enumerate(d_neg):
41          negatif.append(neg)
42
43      blikelihood_neg = [reduce(mul, i) for i in
44  zip(*negatif)]
45      self.blikelihood_neg = blikelihood_neg
```

Penjelasan dari Kode Program 5.46 adalah sebagai berikut:

1. Baris 2-9 melakukan pemanggilan *function* `BernoulliTrain` dan memanggil variabel `bpf_pos` dan `bpf_neg`.
2. Baris 11-29 melakukan proses perulangan pada fitur dengan melakukan kondisi jika pada file fitur terdapat F1-F4 dan F11-F12 maka akan dilakukan proses perhitungan probabilitas *likelihood*.
3. Baris 31-37 melakukan proses perkalian pada seluruh fitur dengan syarat dokumen uji pada kelas positif dan disimpan kedalam `self` untuk digunakan pada perhitungan posterior.
4. Baris 39-45 melakukan proses perkalian pada seluruh fitur dengan syarat dokumen uji pada kelas negatif dan disimpan kedalam `self` untuk digunakan pada perhitungan posterior.

5.1.8.4 Impelementasi Kode Program *Gaussian Naïve Bayes Training*

Proses *training* pada *Gaussian Naïve Bayes* dilakukan untuk mendapatkan peluang fitur dengan syarat kelas. Pada *Gaussian Naïve Bayes Training* fitur yang digunakan hanya fitur latih dengan data kontinu. Pada proses ini yang dihitung adalah rata-rata setiap kelas dan *varian* setiap kelas. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.50.

Kode Program 5.50 Implementasi *Gaussian Naïve Bayes Training*

```
1   def Gaussian(self):
2       doc_pos = 0
3       doc_neg = 0
4       for doc in self.docs:
5           if doc[2].lower() == 'positif':
6               doc_pos += 1
7           else:
8               doc_neg += 1
9
10      average_neg = []
11      average_pos = []
12      for fitur in self.fiturs:
13          if (fitur[0] == 'F5') or (fitur[0] == 'F6') or
14  (fitur[0] == 'F18') or (fitur[0] == 'F19') or (
15                  fitur[0] == 'F20') or (fitur[0] == 'F21')
16  or (fitur[0] == 'F22') or (fitur[0] == 'F31') or (
17                  fitur[0] == 'F32') or (fitur[0] == 'F33')
18  or (fitur[0] == 'F34') or (fitur[0] == 'F35') or (fitur[0]
19  == 'F36'):
20              counter = 0
21              pos = 0
22              neg = 0
```

```
23              for doc in self.docs:
24                  if doc[2].lower() == 'positif':
25                      pos += float(
26                          fitur[1:][counter])
27                  else:
28                      neg += float(fitur[1:][counter])
29                  counter += 1
30              average_pos.append(pos / doc_pos)
31              average_neg.append(neg / doc_neg)
32      self.rata2_pos = average_pos
33      self.rata2_neg = average_neg
34
35      gfiturtrain = []
36      for fitur in self.fiturs:
37          if (fitur[0] == 'F5') or (fitur[0] == 'F6') or
38  (fitur[0] == 'F18') or (fitur[0] == 'F19') or (
39                  fitur[0] == 'F20') or (fitur[0] == 'F21')
40  or (fitur[0] == 'F22') or (fitur[0] == 'F31') or (
41                  fitur[0] == 'F32') or (fitur[0] == 'F33')
42  or (fitur[0] == 'F34') or (fitur[0] == 'F35') or (
43                  fitur[0] == 'F36'):
44              gfiturtrain.append(fitur)
45
46      var_pos = []
47      var_neg = []
48      for i, gauss in enumerate(gfiturtrain):
49          varianpos = 0
50          varianneg = 0
51          for j,dokumen in enumerate(self.docs):
52              if dokumen[2].lower() == "positif":
53                  varianpos += (float(gauss[1:][j]) -
54  average_pos[i]) ** 2
55              else:
56                  varianneg += (float(gauss[1:][j]) -
57  average_neg[i]) ** 2
58          var_pos.append(varianpos / doc_pos)
59          var_neg.append(varianneg / doc_neg)
60
61      self.v_pos = var_pos
62      self.v_neg = var_neg
```

Penjelasan dari Kode Program 5.46 adalah sebagai berikut:

1. Baris 2-8 melakukan perhitungan jumlah kelas positif dan kelas negatif pada fitur latih yang akan digunakan untuk menghitung rata-rata kelas dan *varian* kelas.
2. Baris 10-33 melakukan proses perulangan pada fitur dengan melakukan kondisi jika pada file fitur terdapat F5-F6, F18-F22 dan F31-F36 maka akan dilakukan proses perhitungan nilai rata-rata pada setiap kelas dan akan disimpan pada `self`.
3. Baris 35-62 melakukan proses perulangan pada fitur dengan melakukan kondisi jika pada file fitur terdapat F5-F6, F18-F22 dan F31-F36 maka akan dilakukan proses perhitungan nilai *varian* pada setiap kelas dan akan disimpan pada `self`.

5.1.8.5 Impelementasi Kode Program *Gaussian Naïve Bayes Testing*

Proses *testing* pada *Gaussian Naïve Bayes* dilakukan untuk mendapatkan probabilitas *likelihood*. Nilai rata-rata kelas dan nilai *varian yang* telah dihitung pada *training* akan digunakan pada proses ini. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.51.

Kode Program 5.51 Implementasi *Gaussian Naïve Bayes Testing*

```
def GaussianTesting(self):
    gauss = self.Gaussian()

    rt_pos = []
    for rtp in self.rata2_pos:
        rt_pos.append(rtp)

    rt_neg = []
    for rtn in self.rata2_neg:
        rt_neg.append(rtn)

    v_pos = []
    for vp in self.v_pos:
        v_pos.append(vp)

    v_neg = []
    for vn in self.v_neg:
        v_neg.append(vn)

    gfiturtest = []
    for fitur in self.fituruji:
        if (fitur[0] == 'F5') or (fitur[0] == 'F6') or
(fitur[0] == 'F18') or (fitur[0] == 'F19') or (
                fitur[0] == 'F20') or (fitur[0] == 'F21')
or (fitur[0] == 'F22') or (fitur[0] == 'F31') or (
                fitur[0] == 'F32') or (fitur[0] == 'F33')
or (fitur[0] == 'F34') or (fitur[0] == 'F35') or (
                fitur[0] == 'F36'):
            gfiturtest.append(fitur)

    d_pos = []
    d_neg = []
    for i, fitur in enumerate(gfiturtest):
        f_pd_pos = []
        f_pd_neg = []
        for x in fitur[1:]:
            try:
                exponent_pos = math.exp (- (math.pow
(float(x) - rt_pos[i],2) / (2*v_pos[i])))
                gauss_pos = (1/ (math.sqrt(2*math.pi *
v_pos[i]))) * exponent_pos
            except ZeroDivisionError:
                gauss_pos = 1

            try:
                exponent_neg = math.exp(-
(math.pow(float(x) - rt_neg[i], 2) / (2 * v_neg[i])))
                gauss_neg = (1 / (math.sqrt(2 * math.pi *
```

```
49  v_neg[i]))) * exponent_neg
50              except ZeroDivisionError:
51                  gauss_neg = 1
52              f_pd_pos.append(gauss_pos)
53              f_pd_neg.append(gauss_neg)
54          d_pos.append(f_pd_pos)
55          d_neg.append(f_pd_neg)
56
57      positif = []
58      for i, pos in enumerate(d_pos):
59          positif.append(pos)
60
61      gausslikelihood_pos = [reduce(mul, i) for i in
62  zip(*positif)]
63      self.gausslikelihood_pos = gausslikelihood_pos
64
65      negatif = []
66      for i, neg in enumerate(d_neg):
67          negatif.append(neg)
68      gausslikelihood_neg = [reduce(mul, i) for i in
69  zip(*negatif)]
70      self.gausslikelihood_neg = gausslikelihood_neg
```

Penjelasan dari Kode Program 5.46 adalah sebagai berikut:

1. Baris 2-18 melakukan pemanggilan *function* `Gaussian` dan memanggil variabel `rata2_pos`, `rata2_neg`, `v_pos`, dan `v_neg`.
2. Baris 20-51 melakukan proses perulangan pada fitur dengan melakukan kondisi jika pada file fitur terdapat F5-F6, F18-F22 dan F31-F36 maka akan dilakukan proses *handling* `try except`, kondisi `try` digunakan untuk melakukan proses perhitungan probabilitas *likelihood Gaussian* pada kelas positif dan negatif, kondisi `handling` dilakukan ketika pembagi bernilai 0 nilai `gauss_pos` dan `gauss_neg` akan bernilai 1.
3. Baris 57-63 melakukan proses perkalian pada seluruh fitur dengan syarat dokumen uji pada kelas positif dan disimpan kedalam `self` untuk digunakan pada perhitungan posterior.
4. Baris 65-70 melakukan proses perkalian pada seluruh fitur dengan syarat dokumen uji pada kelas negatif dan disimpan kedalam `self` untuk digunakan pada perhitungan posterior.

5.1.8.6 Impelementasi Kode Program *Multinomial Naïve Bayes Training*

Proses *training* pada *Multinomial Naïve Bayes* dilakukan untuk mendapatkan peluang fitur dengan syarat kelas. Pada *Multinomial Naïve Bayes Training* fitur yang digunakan hanya fitur latih dengan data *numeric*. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.52.

Kode Program 5.52 Implementasi *Multinomial Naïve Bayes Training*

```
1  def MultinomialTrain(self):
2      doc_pos = 0
3      doc_neg = 0
4      for doc in self.docs:
```

```
5           if doc[2].lower() == 'positif':
6               doc_pos += 1
7           else:
8               doc_neg += 1
9           tot_neg = []
10          tot_pos = []
11
12      with
13  open("C:\Users\ACER\PycharmProjects\ProgramBismillah\data
14  \dbaku\sf85%\DataLatihBaru85%.csv", "r") as infile:
15          for fitur in infile.readlines():
16              fitur = fitur.split(',')
17              if (fitur[0] == 'F7') or (fitur[0] == 'F8') or
18  (fitur[0] == 'F9') or (fitur[0] == 'F10') or (
19                      fitur[0] == 'F13') or (fitur[0] ==
20  'F14') or (fitur[0] == 'F15') or (fitur[0] == 'F16') or (
21                      fitur[0] == 'F17') or (fitur[0] ==
22  'F23') or (fitur[0] == 'F24') or (fitur[0] == 'F25') or (
23                      fitur[0] == 'F26') or (fitur[0] ==
24  'F27') or (fitur[0] == 'F28') or (fitur[0] == 'F29') or (
25                      fitur[0] == 'F30') or (fitur[0] ==
26  'F37') or (fitur[0] == re.findall('[A-Za-z]+',
27  fitur[0])[0]):
28                  counter = 0
29                  pos = 0
30                  neg = 0
31                  for doc in self.docs:
32                      if doc[2].lower() == 'positif':
33                          pos += float(fitur[1:][
34                                       counter])
35                      else:
36                          neg += float(fitur[1:][counter])
37                      counter += 1
38                  tot_pos.append(pos)
39                  tot_neg.append(neg)
40
41      pf_pos = []
42      for pos in tot_pos:
43          pf_pos.append(round((pos + 1) / (sum(tot_pos) +
44  len(tot_pos)), 4))
45      self.m_pf_pos = pf_pos
46
47      pf_neg = []
48      for neg in tot_neg:
49          pf_neg.append(round((neg + 1) / (sum(tot_neg) +
50  len(tot_neg)), 4))
51      self.m_pf_neg = pf_neg
```

Penjelasan dari Kode Program 5.52 adalah sebagai berikut:

1. Baris 2-8 melakukan perhitungan jumlah kelas positif dan kelas negatif pada fitur latih yang akan digunakan untuk menghitung *conditional probability* pada data latih.
2. Baris 12-37 melakukan proses perulangan pada fitur dengan melakukan kondisi jika pada file fitur terdapat F7-F10, F13-F17, F24-F30, F37 dan semua fitur *Bag of Words* maka akan dilakukan proses perhitungan nilai fitur positif dan nilai fitur negatif.

3. Baris 41-51 melakukan proses perhitungan peluang fitur dengan syarat kelas (*conditional probability*) pada fitur positif dan fitur negative dan menyimpan nilai pada `self`.

5.1.8.7 Impelementasi Kode Program *Multinomial Naïve Bayes Testing*

Proses *testing* pada *Multinomial Naïve Bayes* dilakukan untuk mendapatkan probabilitas *likelihood*. Nilai peluang fitur dengan syarat kelas yang telah dihitung pada *training* akan digunakan pada proses ini. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.53.

Kode Program 5.53 Implementasi *Multinomial Naïve Bayes Testing*

```
1   def MultinomialTesting(self):
2       mtrain = self.MultinomialTrain()
3
4       mpf_pos = []
5       for pos in self.m_pf_pos:
6           mpf_pos.append(pos)
7
8       mpf_neg = []
9       for neg in self.m_pf_neg:
10          mpf_neg.append(neg)
11      mfitur = []
12
13      with
14  open("C:\Users\ACER\PycharmProjects\ProgramBismillah\
15  data\dbaku\sf85%\DataUjiBaru85%.csv", "r") as infile:
16          for fitur in infile.readlines():
17              fitur = fitur.split(',')
18              if (fitur[0] == 'F7') or (fitur[0] == 'F8') or
19  (fitur[0] == 'F9') or (fitur[0] == 'F10') or (
20                      fitur[0] == 'F13') or (fitur[0] ==
21  'F14') or (fitur[0] == 'F15') or (fitur[0] == 'F16') or (
22                      fitur[0] == 'F17') or (fitur[0] ==
23  'F23') or (fitur[0] == 'F24') or (fitur[0] == 'F25') or (
24                      fitur[0] == 'F26') or (fitur[0] ==
25  'F27') or (fitur[0] == 'F28') or (fitur[0] == 'F29') or (
26                      fitur[0] == 'F30') or (fitur[0] ==
27  'F37') or (fitur[0] == re.findall('[A-Za-z]+',
28  fitur[0])[0]):
29                  mfitur.append(fitur)
30
31      d_pos = []
32      d_neg = []
33      for i, uji in enumerate(mfitur):
34          f_pd_neg = []
35          f_pd_pos = []
36          for x in uji[1:]:
37              f_pd_pos.append(mpf_pos[i] ** float(x))
38              f_pd_neg.append(mpf_neg[i] ** float(x))
39          d_pos.append(f_pd_pos)
40          d_neg.append(f_pd_neg)
41
42      positif = []
43      for i, pos in enumerate(d_pos):
44          positif.append(pos)
```

```
45
46      mlikelihood_pos = [reduce(mul, i) for i in
47  zip(*positif)]
48      self.mlikelihood_pos = mlikelihood_pos
49
50      negatif = []
51      for i, neg in enumerate(d_neg):
52          negatif.append(neg)
53
54      mlikelihood_neg = [reduce(mul, i) for i in
55  zip(*negatif)]
56      self.mlikelihood_neg = mlikelihood_neg
```

Penjelasan dari Kode Program 5.53 adalah sebagai berikut:

1. Baris 2-11 melakukan pemanggilan *function* `MultinomialTrain` dan memanggil variabel `m_pf_pos` dan `m_pf_neg`.
2. Baris 13-40 melakukan proses perulangan pada fitur dengan melakukan kondisi jika pada file fitur terdapat F7-F10, F13-F17, F24-F30, F37 dan semua fitur *Bag of Words* maka akan dilakukan proses perhitungan probabilitas *likelihood*.
3. Baris 42-48 melakukan proses perkalian pada seluruh fitur dengan syarat dokumen uji pada kelas positif dan disimpan kedalam `self` untuk digunakan pada perhitungan posterior.
4. Baris 50-56 melakukan proses perkalian pada seluruh fitur dengan syarat dokumen uji pada kelas negatif dan disimpan kedalam `self` untuk digunakan pada perhitungan posterior.

5.1.8.8 Impelementasi Kode Program Probabilitas *Posterior*

Perhitungan posterior merupakan tahap terakhir dalam *Naïve Bayes*, yaitu menghitung total peluang masing-masing kelas kemudian melakukan perbandingan antara kedua kelas tersebut. Kelas dengan nilai posterior terbesar akan menjadi label kelas data uji. Berikut Implementasi Kode Program dapat dilihat pada Kode Program 5.54.

Kode Program 5.54 Implementasi Probabilitas *Posterior*

```
1   def Posterior(self):
2       prior = self.Prior()
3       bernoulli = self.BernoulliTest()
4       gaussian = self.GaussianTesting()
5       multinomial = self.MultinomialTesting()
6
7       bernoulli_pos = self.blikelihood_pos
8       bernoulli_neg = self.blikelihood_neg
9
10      gauss_pos = self.gausslikelihood_pos
11      gauss_neg = self.gausslikelihood_neg
12
13      multinomial_pos = self.mlikelihood_pos
14      multinomial_neg = self.mlikelihood_neg
15
```

```
16      posterior_pos = []
17      for i, posteriorpos in enumerate(bernoulli_pos):
18          posteriorpos = self.prior_pos  * bernoulli_pos[i] *
19  gauss_pos[i] * multinomial_pos[i]
20          posterior_pos.append(posteriorpos)
21
22      posterior_neg = []
23      for j, posteriorneg in enumerate(bernoulli_neg):
24          posteriorneg = self.prior_neg * bernoulli_neg[j]
25  *gauss_neg[j] * multinomial_neg[j]
26          posterior_neg.append(posteriorneg)
27
28      klasifikasi = []
29      for k,kelas in enumerate(posterior_pos):
30          if posterior_pos[k] >= posterior_neg[k]:
31              klasifikasi.append("Positif")
32          else:
33              klasifikasi.append("Negatif")
```

Penjelasan dari Kode Program 5.53 adalah sebagai berikut:

1. Baris 2-5 Memanggil semua *function* `Prior`, `BernoulliTest`, `GaussianTesting` dan `MultinomialTesting`.
2. Baris 7-14 Melakukan inisialisasi pada variabel `bernoulli_pos`, `bernoulli_neg`, `gauss_pos`, `gauss_neg`, `multinomial_pos` dan `multinomial_neg` yang masing-masing nilai pada variabel tersebut telah disimpan pada masing – masing *function*.
3. Baris 16-20 Melakukan proses perhitungan *posterior* pada kelas positif dan nilai *posterior* akan disimpan pada `posterior_pos`.
4. Baris 22-26 Melakukan proses perhitungan *posterior* pada kelas negatif dan nilai *posterior* akan disimpan pada `posterior_neg`.
5. Baris 28-33 Melakukan proses klasifikasi dengan membandingkan nilai dari `posterior_pos` dan `posterior_neg`.

BAB 6 PEMBAHASAN

Bab ini menjelaskan hasil pengujian dan analisis dari uji coba yang telah dilakukan. Hal ini berguna untuk mengetahui kinerja dari seleksi fitur *Pearson Correlation Coefficient* dan metode *Naïve Bayes* dalam melakukan analisis sentimen opini film pada twitter. Pada bab ini terdapat 2 macam skenario pengujian yang dilakukan, yaitu:

1. Pengujian *threshold* seleksi fitur *Pearson Correlation Coefficient* dengan menggunakan kata tidak baku.
2. Pengujian *threshold* seleksi fitur *Pearson Correlation Coefficient* dengan menggunakan pembakuan kata secara manual.

6.1 Hasil dan Analisis Pengujian Seleksi Fitur dengan Kata tidak Baku

Pengujian Seleksi Fitur dengan menggunakan Kata tidak Baku untuk mengetahui akurasi dengan kombinasi fitur yang paling optimal. Pada pengujian ini menggunakan beberapa kombinasi fitur *Ensemble* dan *Bag of Word* sesuai dengan *threshold* seleksi fitur yang digunakan. Pengujian dilakukan dengan menggunakan beberapa *threshold* seleksi fitur, *threshold* seleksi fitur yang digunakan adalah 10% sampai dengan 95%. Hasil pengujian seleksi fitur dengan menggunakan Kata tidak Baku ditunjukkan pada Tabel 6.1 dan Tabel 6.2.

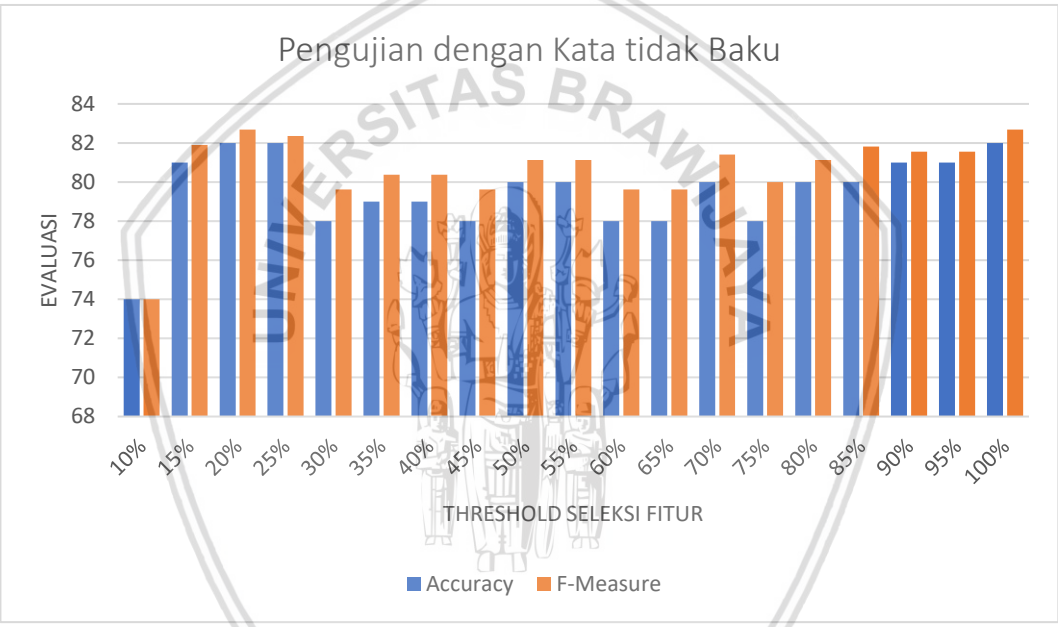
Tabel 6.1 Pengujian Seleksi Fitur dengan Kata tidak Baku

Threshold **Seleksi Fitur**	**Evaluasi**			
	Accuracy	***Precision***	***Recall***	***F-Measure***
10%	74%	74%	74%	74%
15%	81%	86%	78,18%	81,9%
20%	82%	86%	79,62%	82,69%
25%	82%	84%	80,76%	82,35%
30%	78%	86%	74,13%	79,62%
35%	79%	86%	75,43%	80,37%
40%	79%	86%	75,43%	80,37%
45%	78%	86%	74,13%	79,62%
50%	80%	86%	76,78%	81,13%
55%	80%	86%	76,78%	81,13%
60%	78%	86%	74,13%	79,62%
65%	78%	86%	74,13%	79,62%
70%	80%	88%	75,86%	81,41%
75%	78%	88%	73,33%	80%

Tabel 6.1 Pengujian Seleksi Fitur dengan Kata tidak Baku (lanjutan)

Threshold Seleksi Fitur	Evaluasi			
	Accuracy	***Precision***	***Recall***	***F-Measure***
80%	80%	86%	76,78%	81,13%
85%	80%	90%	75%	81,81%
90%	81%	84%	79,24%	81,55%
95%	81%	84%	79,24%	81,55%
100%	82%	86%	79,62%	82,69%

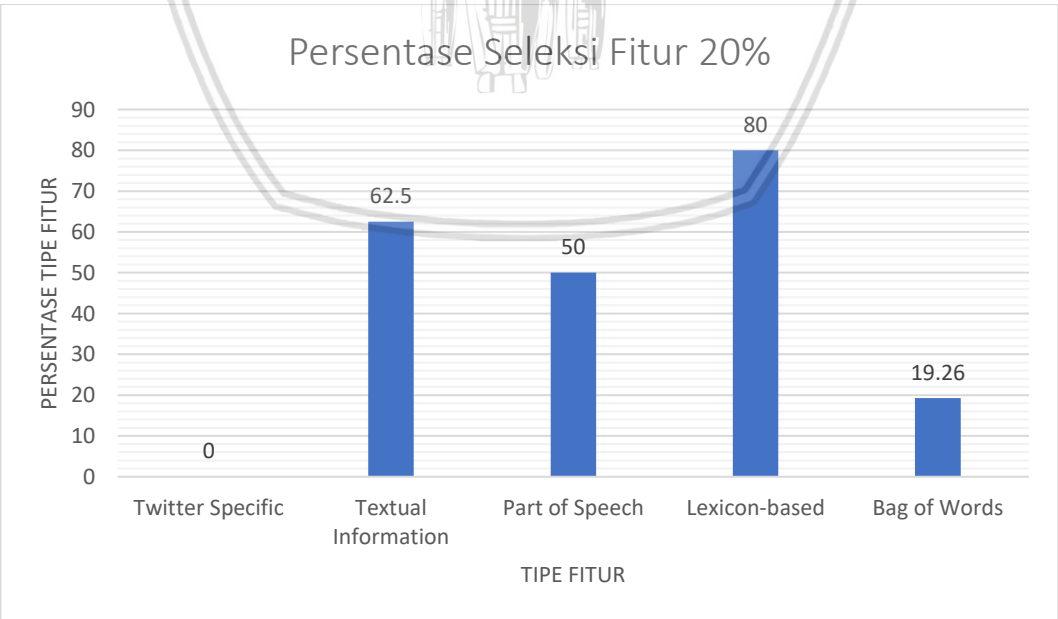
Untuk mempermudah menganalisis nilai *threshold* fitur berdasarkan Tabel 6.1 maka, hasil pada Tabel 6.1 akan dipresesentasikan dalam bentuk grafik. Grafik pada Tabel 6.1 ditunjukkan pada Gambar 6.1.



Gambar 6.1 Pengujian Seleksi Fitur Kata Tidak Baku

Berdasarkan grafik diatas seleksi fitur optimal yaitu pada *threshold* 20% dengan hasil *accuracy* 82%, *precision* 86%, *recall* 79,62% dan *f*-measure 81,9%. Ketika memakai seluruh fitur tanpa menggunakan seleksi fitur hasil yang didapatkan sama dengan seleksi fitur 20%. Pada proses seleksi fitur 20% fitur yang terseleksi adalah 387 fitur dari total keseluruhan fitur yaitu 1932 fitur. Proses seleksi fitur tidak selalu sebagai sarana untuk meningkatkan akurasi tetapi bisa juga dalam mengurangi waktu komputasi sehingga menjadi lebih cepat. Seleksi fitur menggunakan metode *Pearson's Coefficient* memberikan kombinasi fitur yang relevan dan optimal. Hal ini disebabkan ketika menggunakan fitur 20% sudah mencapai optimal yaitu, persebaran kelas pada fitur sesuai dengan kelas pada data dan fitur yang digunakan mewakili semua data uji. Pada proses menggunakan keseluruhan fitur memiliki hasil evaluasi yang sama dengan seleksi fitur 20% tetapi

pada seleksi fitur 20% memberikan komputasi yang lebih cepat dari menggunakan keseluruhan fitur. Pada proses seleksi Fitur 30% hingga 85% mengalami penurunan *Accuracy*. Ada beberapa hal yang menyebabkan penurunan tersbut. Pertama, beberapa fitur tidak efektif dalam membagi data yang artinya persebaran antara fitur dan kelas tidak merata dan metode filter seleksi fitur *Pearson's* dieksplorasi dengan melihat korelasi antara masing – masing fitur dengan label kelas dokumen. Kedua pada saat melakukan *preprocessing* ketika melakukan *filtering* dan *stemming* banyak kata – kata yang tidak dihilangkan dan tidak dibakukan sehingga fitur pada tipe *Bag of Words* menjadi banyak sebagai contoh adalah kata "horor" dengan kata "horror" memiliki makna yang sama tetapi ejaan pada kata tersebut berbeda sehingga menjadi fitur yang berbeda dan dapat menurunkan *accuracy* karena kedua *term* tersebut memiliki nilai *frequency* yang berbeda dan pada fitur *ensemble* ada beberapa fitur yang penting tetapi tidak terseleksi karena nilai *Pearson's* pada fitur tersebut terlalu rendah sebagai contoh adalah tipe *Twitter Specification*. Pengaruh banyaknya fitur berpengaruh pada nilai *Recall* semakin bertambahnya fitur maka nilai *recall* akan menurun hal ini disebabkan oleh fitur *Bag of Words*, banyaknya *term*/fitur yang tidak baku dan pada fitur *ensemble* (*Lexicon-based* dan *Part of Speech*) sedangkan pada *Precision* nilai selalu tetap dan mengalami peningkatan signifikan ketika seleksi fitur 85%. Hal ini disebabkan ketika melakukan *preprocessing* banyak fitur yang tidak ada pada *stopword* sehingga tingkat ketepatan prediksi sistem pada saat menghitung prediksi benar dari total data yang diprediksi sistem termasuk kedalam prediksi yang salah. Berikut ini adalah persentase kemunculan fitur pada seleksi fitur 20% ditunjukkan pada Gambar 6.2.



Gambar 6.2 Persentase Seleksi Fitur 20%

Berdasarkan Gambar 6.2, terdapat beberapa tipe fitur yang memengaruhi proses klasifikasi menggunakan seleksi fitur dengan *threshold* 20%. Fitur dengan tipe *Twitter Specific* tidak berpengaruh pada proses klasifikasi. Hal ini disebabkan oleh Nilai *Pearson's* pada tipe ini sangat rendah sehingga pada saat melakukan seleksi fitur 20% fitur dalam tipe ini tidak ada yang terseleksi dan persebaran kelas positif dan negatif pada fitur ini sangat merata dan nilai fitur tidak stabil sehingga menghasilkan Nilai *Pearson's* yang rendah dan tidak terseleksi pada *threshold* ini. Tipe fitur yang paling berpengaruh adalah fitur *Lexicon-based*, hal ini dikarenakan pada *tweet* opini film selalu mengandung kata sifat baik positif maupun negatif sehingga yang paling berpengaruh pada *Lexicon-based* adalah F23 fitur dengan kata positif dengan persebaran kelas pada fitur ini hampir disemua kelas positif dan negatif penyebab fitur ini terseleksi dan paling berpengaruh adalah nilai yang stabil antara variabel *x* (fitur) dan variabel *y* (kelas) dengan nilai yang diberikan pada fitur tidak terlalu rendah dan tidak terlalu tinggi, sama halnya pada tipe fitur *Part of Speech* fitur yang paling berpengaruh adalah F18 persebaran kelas pada fitur ini hampir disemua kelas positif dan negatif penyebab fitur ini terseleksi dan paling berpengaruh adalah nilai yang stabil antara variabel *x* (fitur) dan variabel *y* (kelas) dengan nilai yang diberikan pada fitur tidak terlalu rendah dan tidak terlalu tinggi. Berbeda pada tipe fitur *Textual Information* yaitu F8 persebaran kelas pada fitur ini kebanyakan adalah Positif saja. Pada fitur *Bag of Words*, fitur yang terseleksi hanya 387 fitur dari 1937 fitur tetapi fitur tersebut mewakili dokumen – dokumen yang diujikan dan persebaran kelas pada *term* kebanyakan hanya muncul di satu kelas yang sama seperti *term* "zootopia" yang kebanyakan muncul di kelas positif saja.

Pada proses pengujian dengan menggunakan kata tidak baku performa klasifikasi optimal dengan menggunakan seleksi fitur pada *threshold* 20%. Hasil evaluasi tersebut sudah baik namun terdapat banyak kata tidak baku pada dokumen yang memiliki makna sama dengan ejaan berbeda ataupun persebaran fitur dengan kelas tidak merata sehingga performa klasifikasi dan hasil evaluasi belum mencapai optimal. Oleh karena itu, untuk pengujian selanjutnya akan dilakukan pengujian dengan menggunakan kata baku secara manual untuk mengetahui pengaruh kata baku pada dokumen *tweet*. Proses pengujian ini menggunakan *threshold* seleksi Fitur 10% sampai dengan 95%.

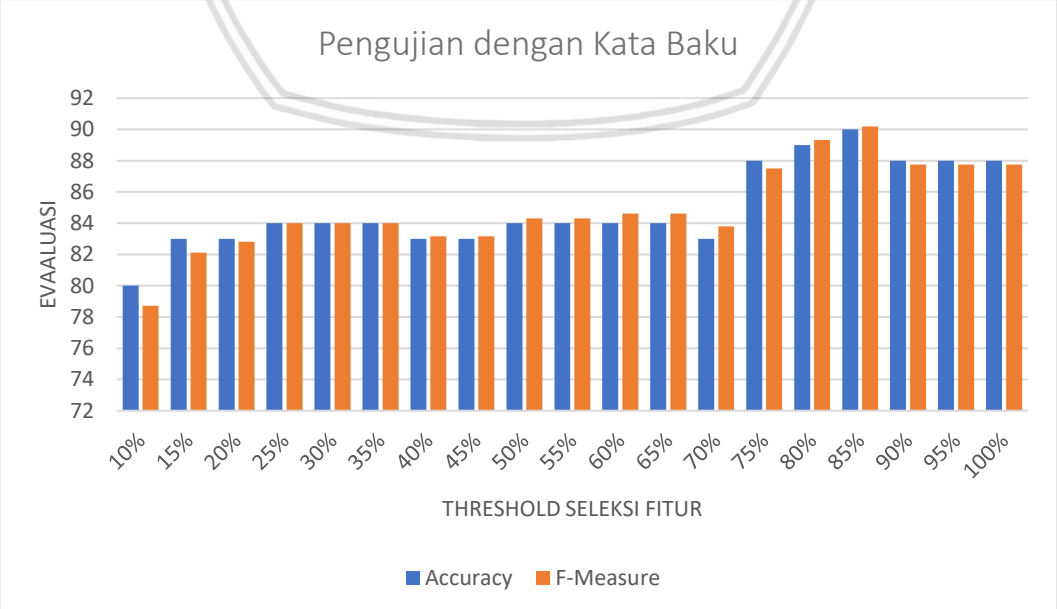
6.2 Hasil dan Analisis Pengujian Seleksi Fitur dengan Kata Baku

Pengujian Seleksi Fitur dengan menggunakan Kata Baku dilakukan untuk mengetahui perbedaan antara pengujian dengan menggunakan Kata tidak Baku dan Kata Baku pada seleksi fitur. Hasil pengujian ini ditunjukkan pada Tabel 6.2.

Tabel 6.2 Pengujian Seleksi Fitur dengan Kata Baku

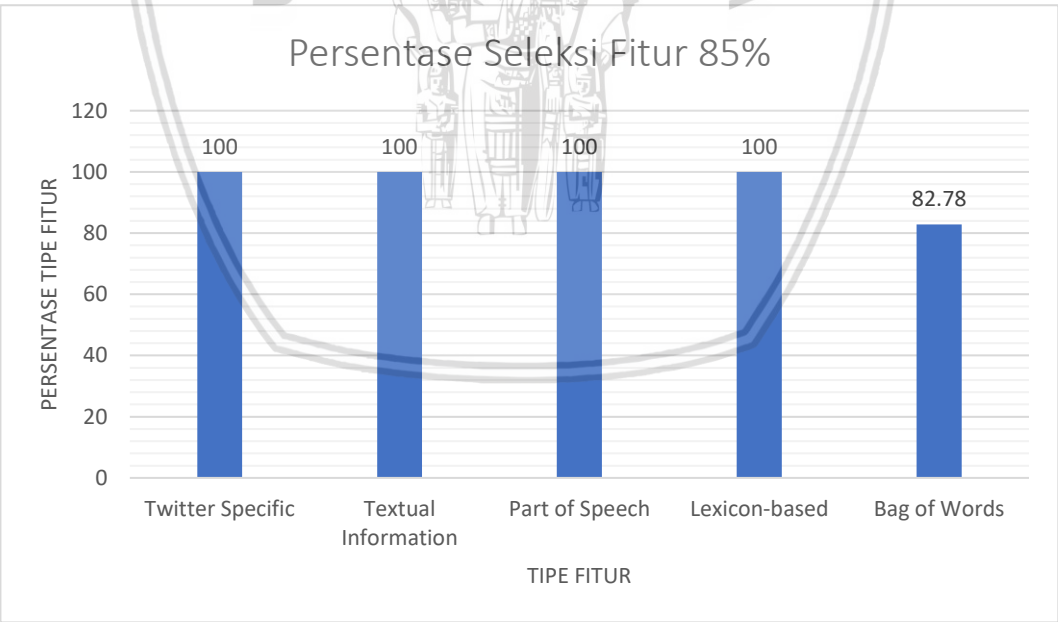
Threshold Seleksi Fitur	Evaluasi			
	Accuracy	***Precision***	***Recall***	***F-Measure***
10%	80%	74%	84,09%	78,72%
15%	83%	78%	86,67%	82,11%
20%	83%	82%	83,67%	82,82%
25%	84%	84%	84%	84%
30%	84%	84%	84%	84%
35%	84%	84%	84%	84%
40%	83%	84%	82,35%	83,16%
45%	83%	84%	82,35%	83,16%
50%	84%	86%	82,69%	84,31%
55%	84%	86%	82,69%	84,31%
60%	84%	88%	81,48%	84,61%
65%	84%	88%	81,48%	84,61%
70%	83%	88%	80%	83,8%
75%	88%	84%	91,3%	87,5%
80%	89%	92%	86,79%	89,32%
85%	90%	92%	88,46%	90,19%
90%	88%	86%	89,58%	87,75%
95%	88%	86%	89,58%	87,75%
100%	88%	86%	89,58%	87,75%

Untuk mempermudah menganalisis nilai *threshold* fitur berdasarkan Tabel 6.2 maka, hasil pada Tabel 6.2 akan dipresesentasikan dalam bentuk grafik. Grafik pada Tabel 6.2 ditunjukkan pada Gambar 6.3.



Gambar 6.3 Pengujian Seleksi Fitur dengan Kata Baku

Berdasarkan grafik Gambar 6.3 di atas seleksi fitur optimal yaitu pada *threshold* 85% dengan hasil *accuracy* 90%, *precision* 92%, *recall* 88,46% dan *f*-measure 90,19%. Pada Gambar 6.3 bisa dilihat bahwa setiap peningkatan *threshold* seleksi Fitur nilai evaluasi semakin meningkat. Peningkatan ini terbukti pada dua hal yaitu keakuratan dari klasifikasi dan yang kedua jumlah fitur yang digunakan berkurang. Pada saat *threshold* seleksi Fitur sebesar 25% sampai dengan 35% nilai evaluasi tetap dan tidak mengalami kenaikan maupun penurunan. Ada beberapa hal yang mempengaruhi hal tersebut. Pertama fitur yang terseleksi tidak terdapat pada dokumen uji sehingga nilai fitur tersebut pada data uji bernilai 0 dan tidak berpengaruh sama sekali pada perhitungan klasifikasi. Kedua persebaran kelas pada fitur tersebut hanya pada kelas positif saja atau kelas negatif saja. Pada saat seleksi Fitur 75% mengalami kenaikan signifikan pada nilai evaluasi dan mencapai kenaikan optimal ketika seleksi Fitur 85%. Seleksi Fitur 85% memberikan fitur efektif dan optimal dan dapat meningkatkan nilai dari evaluasi daripada menggunakan keseluruhan fitur. Hal ini disebabkan oleh semua fitur *ensemble* yang terseleksi sehingga membantu menaikkan nilai evaluasi, berbeda dengan menggunakan keseluruhan fitur. Fitur *Bag of Words* memberikan penurunan evaluasi ketika semua fitur tersebut digunakan. Hal ini disebabkan oleh beberapa nilai dari fitur *Bag of Words* persebaran kelasnya tidak merata dan fitur tersebut hanya mewakili label dari satu dokumen saja. Berikut seleksi Fitur 85% ditunjukkan pada Gambar 6.4.



Gambar 6.4 Persentase Seleksi Fitur 85%

Berdasarkan hasil grafik pada Gambar 6.4 seleksi fitur dengan menggunakan *threshold* fitur 85% fitur terseleksi dari *ensemble feature* adalah 100% disetiap tipe dan *Bag of Word* sebesar 82,9%. Jika dilihat dari grafik di atas *Ensemble Feature* sangat berpengaruh pada *dataset* uji dalam mendapatkan kombinasi fitur yang

relevan. Tipe fitur *Twitter Specification* yang paling berpengaruh adalah *username* (F3) persebaran kelas pada fitur ini merata yaitu pada kelas positif dan negatif walaupun semua fitur pada tipe ini memiliki nilai *Pearson's* yang rendah tetapi pada *threshold* 85% semua fitur dari tipe ini terseleksi. Pada *Textual Information* fitur yang paling berpengaruh adalah fitur *tweet* dengan tanda seru pada dokumen (F8) pada fitur ini persebaran kelas hanya pada kelas positif saja. Pada fitur *Part of Speech* fitur yang paling berpengaruh adalah fitur F19 yaitu rata – rata pada kata sifat (*percentage adjective*) dana juga pada *Lexicon-based* fitur yang paling berpengaruh adalah kata sifat positif (*adjective pos*), kedua fitur ini meningkatkan *accuracy* karena sebagian besar dari Twitter menggunakan kata sifat sehingga kedua fitur ini sangat berpengaruh. Pada fitur *Bag of Word* fitur yang terseleksi sebanyak 1531 fitur dari 1842 fitur. Sama halnya dengan pengujian pada kata tidak baku fitur persebaran kelas pada fitur *Bag of Word* hampir disemua kelas tergantung oleh *term* terseleksi dan setiap *term* pada fitur *Bag of Word* mewakili setiap kelas.

Perbandingan antara pengujian dengan menggunakan kata tidak baku dan kata baku dapat terlihat pada semua tipe fitur *ensemble* maupun pada fitur *Bag of Word*. Ketika dokumen *tweet* menggunakan kata tidak baku banyak fitur yang tidak relevan sehingga *accuracy* yang didapatkan tidak optimal. Pada tipe fitur *ensemble* seperti *Part of Speech* dan *Lexicon-based* yang tidak sesuai, sebagai contoh kata "horor" dan "horror", kata "horror" merupakan kata yang tidak baku sehingga ketika sistem mencari kata tersebut dalam kamus, sistem tidak memberikan nilai untuk fitur tersebut. Begitu juga pada fitur *Bag of Words*, yang menyebabkan fitur tersebut terlalu banyak fitur adalah pada saat melakukan *stopword* dan *stemming*. *Stopword* yang digunakan adalah modifikasi dari *stopword* Tala yang biasanya digunakan hanya untuk kata baku sehingga ketika dokumen tersebut memiliki kata tidak baku di seluruh dokumen kata tersebut tidak masuk kedalam *filtering* sehingga menimbulkan fitur fitur yang tidak relevan. Permasalahan pada *stemming* adalah ketika kata tersebut tidak terdapat pada *stopword* dan tidak juga terdapat pada *stemming* maka kata tersebut telah diangga menjadi kata dasar. Sebagai contoh pada fitur *Bag of Words* dapat dilihat pada Tabel 6.5.

Tabel 6.3 Perbandingan Pengujian Kata Baku dan tidak Baku

Kata Tidak Baku	Kata Baku
horor (17)	horor (37)
horror (18)	
horrornya (2)	

Berdasarkan Tabel 6.5 kata tersebut memiliki makna yang sama tetapi antara "horor" dan "horror" hanya saja ejaan dari 2 kata tersebut berbeda

sehingga kata tersebut menjadi fitur tidak relevan dan nilai *frequency* yang didapatkan juga berbeda pada kata "horror" memiliki *frequency* paling tinggi. Begitu juga dengan kata "horrornya", kata tersebut tidak terdapat pada *stopword* dan juga tidak terdapat pada *stemming* sehingga menimbulkan fitur yang tidak relevan dan menyebabkan nilai evaluasi terutama *accuracy* menjadi tidak optimal.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis dari penelitian analisis sentimen opini film menggunakan metode *Naïve Bayes* dengan *Ensemble Feature* dan seleksi Fitur *Pearson Correlation Coefficient* maka diperoleh kesimpulan sebagai berikut:

1. Pemilihan fitur memiliki kelebihan dan kelemahan tertentu yang telah ditunjukkan pada penelitian ini. Seleksi Fitur tidak selalu meningkatkan tingkat *accuracy* pada metode klasifikasi tetapi seleksi Fitur dapat mengurangi komputasi sehingga menjadi lebih cepat. Bahkan mengurangi jumlah fitur yang terlalu banyak dapat menghasilkan penurunan *accuracy* dan nilai evaluasi yang lain. Efek yang diinginkan dari penelitian ini adalah untuk meningkatkan performa klasifikasi dan juga membuatnya menjadi lebih efektif dengan hanya memfokuskan pada fitur yang relevan. Metode seleksi fitur yang digunakan pada penelitian ini telah dievaluasi untuk menentukan keefektifannya untuk mengurangi jumlah dimensi dalam *dataset* dan meningkatkan *accuracy* dari metode klasifikasi dengan menggunakan fitur - fitur yang relevan. Penelitian ini telah menemukan bahwa dengan menggunakan metode seleksi fitur *Pearson's* memberikan kombinasi fitur yang optimal, menghilangkan fitur – fitur yang tidak relevan dan juga mengurangi jumlah total dimensi fitur yang diperlukan untuk metode klasifikasi sehingga meningkatkan performa pada metode klasifikasi.

2. Penggunaan kata tidak baku dan kata baku pada dokumen *tweet* sangat mempengaruhi performa klasifikasi. Pada kata tidak baku banyak fitur – fitur yang memiliki makna sama tetapi ejaan pada *term* tersebut berbeda dan pada fitur *ensemble* seperti *Part of Speech* dan *Lexicon-based* banyak kata tidak baku yang tidak terdeteksi sehingga menurunkan nilai evaluasi. Seleksi Fitur optimal dengan menggunakan kata tidak baku yaitu dengan *threshold* 20% dan hasil evaluasi yang didapatkan *Accuracy* 82%, *Precision* 86%, *Recall* 79,62%, *F-Measure* 82,69%. Sedangkan dengan menggunakan kata baku Seleksi Fitur optimal dengan *threshold* 85% dan hasil evaluasi yang didapatkan *Accuracy* 90%, *Precision* 92%, *Recall* 88,46% dan *F-Measure* 90,19%. Dapat disimpulkan bahwa dengan menggunakan kata baku dapat meningkatkan nilai *accuracy* sebesar 8%.

7.2 Saran

Berdasarkan penelitian yang telah dilakukan, didapatkan beberapa saran yang dapat dikembangkan untuk penelitian selanjutnya, berikut saran-saran yang didapatkan:

1. *Microblogging* saat ini selalu mengalami pembaharuan pada keseluruhan fitur aplikasinya termasuk *emoticon* pada aplikasi tersebut sehingga pada penelitian selanjutnya melakukan penambahan data pada kamus *emoticon* dan juga penambahan kamus *sentiword* untuk kata positif dan kata negatif pada *lexicon-based features*. Sedangkan pada proses melakukan ekstraksi pada fitur *intensifier* terdapat beberapa kata yang seharusnya termasuk dalam *intensifier* tetapi tidak termasuk sehingga pada penelitian selanjutnya menambahkan kata pada kamus *intensifier word* (kata penguat).
2. Pada proses skenario pengujian kata tidak baku dan kata baku, hasil optimal didapatkan ketika menggunakan pengujian dengan kata baku. Oleh karena itu, pada penelitian selanjutnya menambahkan metode untuk mengatasi kata yang tidak baku atau kata *slang* secara otomatis sehingga dalam melakukan ekstraksi fitur dapat lebih akurat seperti metode *Levenstein Distance*, Algoritma Fonetik, atau metode yang lainnya.

DAFTAR PUSTAKA

Andilala. 2016. Movie Review Sentimen Analisis dengan Metode Naive Bayes Base on Feature Selection. *Jurnal Pseudocode, III*(1), pp. 1-9.

Bae, Y., & Lee, H. 2012. Sentiment analysis of twitter audiences: Measuring the positive or negative influence of popular twitterers. *Journal of the American Society for Information Science and Technology, Vol.63, No12*, pp. 2521 - 2535.

Capdevila, M., & Flórez, O. W. 2009. A Communication Perspective on Automatic Text Categorization. *IEEE Transactions on Knowledge and Data Engineering (Volume: 21, Issue: 7, July 2009)*, pp. 1027-1041.

Fauzi, M. A., & Yuniarti, A. 2018. Ensemble Method for Indonesian Twitter Hate Speech. *Indonesian Journal of Electrical Engineering and Computer Science Vol. 11, No. 1*, p. 294~299.

Feldman, R., & Sanger, J. 2007. *The Text Mining Handbook.* Cambridge: Cambridge University Press.

Irfan, M. R., Fauzi, M. A., & Tibyani. 2018. Analisis Sentimen Kurikulum 2013 pada Twitter menggunakan Ensemble Feature dan Metode K-Nearest Neighbor. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,Vol. 2, No. 9*, pp. 3006 - 3014.

Manning, C. D., Raghavan, P., & Schütze, H. 2008. *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Maulida, I., Suyatno, A., & Hatta, H. R. 2016. Seleksi Fitur Pada Dokumen Abstrak Teks Bahasa Indonesia Menggunakan Metode Information Gain. *JSM STMIK Mikroskil VOL 17, NO 2,*, pp. 249-258.

McCallum, A., & Nigam, K. 1998. A comparison of event models for naive bayes text classification. InAAAI-98. *InAAAI-98 workshop on learning for text categorization Vol. 752*, pp. 41-48.

Onan, A., & Korukoğlu, S. 2015. A feature selection model based on genetic rank aggregation for text sentiment classification. *Journal of Information Science Vol 43, Issue 1*, pp. 25-38.

Pak, A., & Paroubek, P. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. *Proceedings of the International Conference on Language Resources and Evaluation*, Valetta, s.n., pp. 1320-1326.

Raschka, S. 2014. Naive Bayes and Text Classification I - Introduction and Theory. *CoRR, vol. 14105329, no.14105329*, pp. 1-20.

Shardlow, M. 2007. An Analysis of Feature Selection Techniques. *The University of Manchester*, pp. 1-7.

Sharma, A., & Dey, S. 2012. Performance Investigation of Feature Selection Methods and Sentiment Lexicons for Sentiment Analysis . *Special Issue of International Journal of Computer Applications*, pp. 15-20.

Siddiqua, U. A., Ahsan, T., & Chy, A. N. 2016. Combining a Rule-based Classifier with Ensemble of Feature Sets and Machine Learning Techniques for Sentiment Analysis on Microblog. *19th International Conference on Computer and Information Technology*. Bangladesh, IEEE, pp. 304 - 309.

Silva, N. F., Hruschka, E. R., & Jr., E. R. 2014. Tweet sentiment analysis with classifier ensembles. *Decission Support System*, pp. 170-179.

Sitorus, A. P., Murfi, H., Nurrohmah, S., & Akbar, A. 2017. Sensing Trending Topics in Twitter for Greater Jakarta Area. *International Journal of Electrical and Computer Engineering (IJECE) Vol.7, No.1*, pp. 330 - 336.

University, T. O. 2018. *Being digital: Writing a good tweet* . London, United Kingdom: The Open University .

Utami, L. D., & Wahono, R. S. 2015. Integrasi Metode Information Gain Untuk Seleksi Fitur dan Adaboost Untuk Mengurangi Bias Pada Analisis Sentimen Review Restoran Menggunakan Algoritma Naïve Bayes. *Journal of Intelligent Systems, Vol. 1, No. 2*, pp. 120-126.

Wahid, D. H., & Azhari, S. 2016. Peringkasan Sentimen Esktraktif di Twitter Menggunakan Hybrid TF-IDF dan Cosine Similarity. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems),10(2)*, pp. 207-218.

Xia, R., Zong, C., & Li, S. 2011. Ensemble of feature sets and classification algorithms sentiment classification. *Information Sciences 181*, p. 1138–1152.

Yang, A., Zhang, J., Pan, L., & Xiang, Y. 2016. Enhanced Twitter Sentiment Analysis by Using Feature Selection and Combination. *2015 International Symposium on Security and Privacy in Social Networks and Big Data (SocialSec)*. Hangzhou, China, IEEE. pp. 52-57.