PEMBENTUKAN DAFTAR *STOPWORD* MENGGUNAKAN *TERM BASED RANDOM SAMPLING* PADA ANALISIS SENTIMEN DENGAN METODE *NAÏVE BAYES*

(STUDI KASUS: KULIAH DARING DI MASA PANDEMI)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Raditya Rinandyaswara
NIM: 175150200111047

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

PENGESAHAN

PEMBENTUKAN DAFTAR STOPWORD MENGGUNAKAN TERM BASED RANDOM SAMPLING PADA ANALISIS SENTIMEN DENGAN METODE NAÏVE BAYES (STUDI KASUS: KULIAH DARING DI MASA PANDEMI)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Raditya Rinandyaswara
NIM: 175150200111047

Skripsi ini telah diuji dan dinyatakan lulus pada
13 Januari 2021
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Yuita Arum Sari, S.Kom., M.Kom.	Muhammad Tanzil Furqon, S.Kom., M.CompSc.
NIK: 2016098807152001	NIP: 19820930 200801 1 004

Mengetahui
Ketua Jurusan **Teknik Informatika**

Achmad Basuki, S.T., M.MG., Ph.D.
NIP: 19741118 200312 1 002

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 referensi.

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, 23 December 2020

Raditya Rinandyaswara

NIM: 175150200111047

PRAKATA

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Pembentukan Daftar Stopword menggunakan Term Based Random Sampling pada Analisis Sentimen dengan metode Naïve Bayes (Studi Kasus: Kuliah Daring di Masa Pandemi)" ini dapat terselesaikan.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Ibu Yuita Arum Sari, S.Kom., M.Kom. dan Bapak Muhammad Tanzil Furqon, S. Kom., M.CompSc. selaku Pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini,
2. Bapak Adhitya Bhawiyuga, S.Kom., M.Sc. selaku Ketua Program Studi Teknik Informatika,
3. Bapak Achmad Basuki, S.T., M.MG., Ph.D. selaku Ketua Jurusan Teknik Informatika,
4. Ibu Ari Kusyanti, S.T., M.Sc. selaku dosen Penasihat Akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi,
5. Ayahanda dan Ibunda dan seluruh keluarga besar atas segala nasihat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti-hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini,
6. Seluruh civitas academica Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Informatika Univesitas Brawijaya dan selama penyelesaian skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 23 Desember 2020

Penulis

radityarin@gmail.com

ABSTRAK

Raditya Rinandyaswara, Pembentukan Daftar Stopword menggunakan Term Based Random Sampling pada Analisis Sentimen dengan Metode *Naïve Bayes* (Studi Kasus: Kuliah Daring di Masa Pandemi)

Pembimbing: Yuita Arum Sari, S.Kom., M.Kom. dan Muhammad Tanzil Furqon, S.Kom., M.CompSc.

Stopword Removal merupakan bagian dari tahapan *preprocessing* teks yang bertujuan untuk menghapus kata yang tidak relevan didalam suatu kalimat berdasarkan daftar *stopword*. Daftar *stopword* yang biasa digunakan berbentuk *digital library* yang daftarnya sudah tersedia sebelumnya, namun tidak semua kata-kata yang terdapat didalam *digital library* merupakan kata yang tidak relevan dalam suatu data tertentu. Penelitian ini menggunakan daftar stopword yang dibentuk dengan algoritme *Term Based Random Sampling*. Dalam Term Based Random Sampling terdapat 3 parameter yaitu Y untuk jumlah perulangan pengambilan kata random, X untuk jumlah pengambilan bobot terendah dalam perulangan Y, dan L sebagai persentase jumlah stopword yang ingin digunakan. Sehingga penelitian ini ditujukan untuk mencari tahu kombinasi terbaik dari 3 parameter tersebut serta perbandingan *stopword Term Based Random Sampling* dengan *stopword Tala* dan tanpa proses *stopword removal* dalam analisis sentimen *tweet* mengenai kuliah daring dengan menggunakan metode *Naïve Bayes*. Hasil evaluasi dengan *stopword Term Based Random Sampling* mendapatkan akurasi tertinggi dengan kombinasi *X* sebesar 10, *Y* sebesar 10, dan *L* sebesar 40 *macroaverage accuracy* sebesar 0,758, *macroaverage precision* sebesar 0,658, *macroaverage recall* sebesar 0,636, dan *macroaverage f-measure* sebesar 0,647. Berdasarkan hasil pengujian parameter disimpulkan bahwa semakin besar nilai *X, Y*, dan *L* maka semakin tinggi kemungkinannya untuk *accuracy, precision, recall,* dan *f-measure* turun. Hasil evaluasi sistem membuktikan bahwa analisis sentimen dengan *stopword Term Based Random Sampling* berhasil mendapatkan akurasi lebih tinggi dibandingkan dengan menggunakan *stopword* Tala maupun yang tanpa menggunakan proses *stopword removal*.

Kata kunci: sentimen analisis, kuliah daring, twitter, *Naïve Bayes, Term Based Random Sampling, stopword*

ABSTRACT

Raditya Rinandyaswara, Forming a Stopword List Using Term Based Random Sampling on the Sentiment Analysis using the Naïve Bayes Method (Case Study: Online Lectures during the Pandemic)

Supervisors: Yuita Arum Sari, S.Kom., M.Kom. and Muhammad Tanzil Furqon, S.Kom., M.CompSc.

Stopword Removal is part of the text preprocessing stage which aims to remove irrelevant words in a sentence based on the stopword list. The stopword list that is commonly used is in the form of a digital library whose list is already available, but not all words contained in the digital library are irrelevant words in certain data. This study uses a stopword list formed by the Term Based Random Sampling algorithm. In Term Based Random Sampling, there are 3 parameters, namely Y for the number of random word retrieval repetitions, X for the lowest number of weights in Y repetitions, and L as the percentage of the number of stopwords you want to use. So this research is aimed at finding out the best combination of these 3 parameters as well as the comparison of term based random sampling stopword with stopword tuning and without stopword removal process in the analysis of tweet sentiment about online lectures using the Naïve Bayes method. The results of the evaluation with the Term Based Random Sampling stopword get the highest accuracy with a combination of X of 10, Y of 10, and L of 40 macroaverage accuracy of 0.758, macroaverage precision of 0.658, macroaverage recall of 0.636, and macroaverage f-measure of 0.647. Based on the results of parameter testing, it can be concluded that the greater the X, Y, and L values, the higher the probability for decreasing accuracy, precision, recall, and f-measure. The results of the system evaluation prove that sentiment analysis with the Stopword Term Based Random Sampling has succeeded in obtaining higher accuracy than using stopword tuning or without using the stopword removal process.

Keywords: sentiment analysis, online school, twitter, Naïve Bayes, Term Based Random Sampling, stopword

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bab ini terdiri dari hal yang melatarbelakangi dari penelitian ini dilaksanakan, rumusan masalah yang diperoleh dari latar belakang hingga tujuan dan manfaat dari penelitian ini serta batasan yang dijabarkan sesuai dengan cakupan dan kemampuan penulis , maupun sistematika yang menuliskan secara rangkum isi dari tiap bab.

1.1 Latar Belakang

Pada akhir tahun 2019 lalu, dunia dikejutkan dengan adanya wabah yang diakibatkan oleh virus *corona* yang berasal dari kota Wuhan, China. Penyakit yang disebut sebagai (COVID-19) ini adalah penyakit yang menyerang sistem pernapasan virus manusia. Menurut data pemerintah China, penduduk Hubei menjadi kasus pertama Covid-19 pada 17 November 2019 (Arnani, 2020). Setelah kasus pertama Covid-19 di dunia itu terjadi peningkatan pasien tiap bulannya. Hingga saat ini Indonesia sudah melewati angka 190 ribu kasus Covid-19 yang sudah terkonfirmasi yang terhitung dari sejak pasien pertama (Ramadhan, et al., 2020). Dengan adanya pandemi Covid-19 ini, pemerintah Indonesia mengadakan sistem *New Normal* dengan tujuan untuk mempercepat penanganan Covid-19 (Putsanra, 2020). Dalam penerapannya banyak kegiatan-kegiatan masyarakat yang beralih dari secara luring menjadi daring. Salah satu contohnya adalah perkuliahan.

Kebijakan kuliah daring ini ramai menjadi perbincangan seluruh masyarakat terutama mahasiswa di Indonesia. Media sosial seperti Twitter menjadi salah satu sarana mahasiswa menuliskan pendapatnya terkait kuliah daring ini. Ada yang menganggap kebijakan ini secara positif, salah satu alasannya yaitu mengingat meningkatnya kasus Covid-19 setiap harinya. Namun di sisi lain, ada yang menganggap kebijakan ini secara negatif, salah satu alasannya yaitu, ketidakpahaman terhadap materi kuliah yang diajarkan melalui daring. Selain itu, ada juga yang menganggap kebijakan ini secara netral, yaitu mereka yang melihat dari kedua sisi baik negatif maupun positif. Melihat banyaknya masyarakat menanggapi kebijakan ini, akan sulit jika proses melihat tanggapan masyarakat satu per satu. Sehingga sistem analisis sentimen diperlukan untuk menganalisa bagaimana tanggapan masyarakat serta untuk memudahkan proses analisa data.

Analisis Sentimen atau *Opinion Mining* adalah salah satu bidang studi yang menganalisis pendapat, sentimen, evaluasi, penilaian, sikap, dan emosi orang terhadap entitas seperti produk, layanan, organisasi, isu, peristiwa, topik, dan atributnya (Liu, 2012). Dengan proses analisis sentimen ini kita dapat mengetahui bagaimana pendapat orang apakah cenderung positif, negatif atau pun netral.

Sudah ada penelitian terdahulu mengenai analisis sentimen dari Twitter, salah satunya adalah oleh (Antinasari, et al., 2017) yang menggunakan metode Naïve Bayes. Data yang digunakan *tweet* terkait dengan opini film. Penelitian ini juga menambahkan perbaikan kata tidak baku menggunakan Levenshtein Distance. Berdasarkan hasil pengujian didapatkan akurasi tertinggi dengan nilai *accuracy, precision, recall,* dan *f-measure* sebesar 98.33%, 96.77%, 100%, dan 98.36%. Melihat hasil evaluasi tersebut, peneliti akan membangun sebuah sistem analisis sentimen terhadap kuliah daring yang dituliskan masyarakat di Twitter menggunakan metode klasifikasi *Naïve Bayes*. Selain itu penelitian kali ini akan membuat *stopword* dinamik dengan algoritme *Term Based Random Sampling*.

Term Based Random Sampling adalah suatu algoritme yang dapat digunakan untuk menghasilkan *stopword* secara otomatis berdasarkan seberapa informatif kata tertentu (Lo, et al., 2005). Dalam algoritme tersebut terdapat 3 parameter utama yang dapat mempengaruhi hasil dari *stopword* yang dihasilkan yaitu Y untuk jumlah perulangan pengambilan kata random, X untuk jumlah pengambilan bobot terendah dalam perulangan Y, dan L sebagai persentase jumlah stopword yang ingin digunakan. Adapun penelitian sebelumnya oleh sebelumnya (Sa'rony, et al., 2019) yang menggunakan *Term Based Random Sampling* sebagai algoritme pembentukan *stopword*. Data yang digunakan adalah tweet terkait kebijakan pemerintahan ibukota Republik Indonesia. Berdasarkan hasil pengujian didapatkan akurasi tertinggi disaat parameter L atau persentase stoplists senilai 20% dengan macroaverage akurasi, macroaverage precision, macroaverage recall, macroaverage fmeasure yang masing-masing 0,94, 0,945, 0,94, dan 0,938. Namun, dalam penelitian tersebut peneliti sebelumnya hanya meneliti terkait parameter L, dan tidak meneliti terkait 2 parameter lainnya yaitu X, dan Y. Melihat uraian diatas, penulis memutuskan untuk melakukan penelitian menggunakan Naïve Bayes dengan pembentukan stopword dengan *Term Based Random Sampling* dengan klasifikasi akan dibuat menjadi 3 kelas yaitu opini netral, positif dan negatif sesuai dari saran penelitian analisis sentimen sebelumnya (Sa'rony, et al., 2019) serta penelitian difokuskan mencari pengaruh serta kombinasi parameter terbaik dalam *Term Based Random Sampling*. Sehingga diharapkan penelitian ini dapat melihat bagaimana pengaruh dari parameter-parameter *Term Based Random Sampling* tersebut.

1.2 Rumusan Masalah

Dari latar belakang yang telah diuraikan sebelumnya, berikut adalah rumusan masalah untuk penelitian ini:

1. Bagaimana kombinasi parameter terbaik pada *Term Based Random Sampling* terhadap hasil pembentukan *stopword*?
2. Bagaimana hasil perbandingan dari pembentukan *stopword Term Based Random Sampling* dengan tanpa *stopword* pada analisis sentimen dengan *Naïve Bayes*?

3. Bagaimana hasil perbandingan dari pembentukan *stopword Term Based Random Sampling* dengan *stopword* Tala pada analisis sentimen dengan *Naïve Bayes*?

1.3 Tujuan

Tujuan penelitian ini adalah:

1. Mengetahui kombinasi parameter terbaik pada *Term Based Random Sampling* terhadap hasil pembentukan *stopword*.
2. Mengetahui hasil perbandingan dari pembentukan *stopword Term Based Random Sampling* dengan tanpa *stopword* pada analisis sentimen dengan *Naïve Bayes*.
3. Mengetahui hasil perbandingan dari pembentukan *stopword Term Based Random Sampling* dengan *stopword* Tala pada analisis sentimen dengan *Naïve Bayes*.

1.4 Manfaat

Manfaat penelitian ini adalah:

1. Dapat mengetahui kombinasi parameter terbaik pada *Term Based Random Sampling* terhadap hasil pembentukan *stopword*
2. Dapat mengetahui hasil perbandingan dari pembentukan *stopword Term Based Random Sampling* dengan tanpa *stopword* pada analisis sentimen dengan *Naïve Bayes*.
3. Dapat mengetahui hasil perbandingan dari pembentukan *stopword Term Based Random Sampling* dengan *stopword* Tala pada analisis sentimen dengan *Naïve Bayes*.

1.5 Batasan Masalah

Batasan masalah penelitian ini adalah:

1. Hanya menggunakan opini pengguna Twitter mengenai Kuliah Daring.
2. Algoritme yang digunakan hanya *Naïve Bayes Classifier* tidak membandingkan dengan algoritme lain.
3. Hasil klasifikasi sentimen hanya dibagi menjadi tiga kelas yaitu positif, netral, dan negatif.
4. *Tweet* yang merupakan data hanya *tweet* yang berbahasa Indonesia.
5. Jumlah data yang digunakan sebanyak 300 data.
6. Sistem yang dibuat hanya dapat menangani data yang seimbang setiap kelasnya.

1.6 Sistematika Pembahasan

Berikut sistematika pembahasan dalam penelitian ini:

BAB I PENDAHULUAN

Bab ini menjelaskan mengenai latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan sesuai dengan aturan dalam peneliltian

BAB II LANDASAN KEPUSTAKAAN

Landasan Kepustakaan menjelaskan penelitian-penelitian sebelumnya yang serupa dengan penelitian dalam proposal ini, serta dasar-dasar teori yang akan di implementasikan dalam penelitian ini seperti *preprocessing, term weighting, Naïve Bayes, Term Based Random Sampling,* serta tabel *confusion matrix* sehingga dapat mendukung penelitian.

BAB III METODOLOGI

Pada bab ini dijelaskan tentang bagaimana menerapkan penelitian seperti untuk mengimplementasikan *Naïve Bayes* dengan pembuatan daftar *Stopword* untuk analisis sentimen pengguna Twitter terhadap kebijakan Kuliah Daring.

BAB IV PERANCANGAN

Bab ini menjelaskan bagaimana proses perancangan dalam sistem yang akan dibangun.

BAB V IMPLEMENTASI

Pada bab ini menjelaskan bagimana implementasi sistem yang sudah dirancang di bab sebelumnya.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan pengujian terhadap sistem yang sudah dibangun dan menganalisis hasil yang didapatkan untuk menemukan kesimpulan dari hasil pengujian.

BAB VII PENUTUP

Pada bab terakhir ini menjelaskan tentang bagaimana kesimpulan dari penelitian ini dan saran untuk penelitian berikutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan dijelaskan bagaimana penelitian-penelitian sebelumnya yang serupa dengan penelitian yang sedang diajukan, serta dasar-dasar teori yang akan diimplementasikan dalam penelitian ini seperti *preprocessing, term weighting, Naïve Bayes, Term Based Random Sampling,* serta tabel *confusion matrix* yang dapat mendukung penelitian.

2.1 Kajian Pustaka

Pada bagian ini akan dibahasnya mengenai penelitian atau kajian pustaka yang digunakan dalam penelitian ini berasal dari beberapa penelitian yang sudah dilakukan sebelumnya dan memiliki keterkaitan dengan judul skripsi Pembentukan Daftar Stopword menggunakan Term Based Random Sampling pada Analisis Sentimen dengan metode Naïve Bayes (Studi Kasus: Kuliah Daring di Masa Pandemi).

Contoh salah satu penelitian terkait judul skripsi yang telah disebutkan adalah penelitian yang dilakukan oleh (Septian, et al., 2019) yaitu mengenai analisis sentimen pengguna Twitter terhadap polemik persepakbolaan Indonesia menggunakan pembobotan *TF-IDF* dan metode klasifikasi *K-Nearest Neighbor*. Penelitian ini menggunakan kamus kata tidak baku yang dibuat oleh peneliti secara manual yang nantinya akan digunakan sebagai normalisasi kata. Hasil pengujian yang didapatkan dari pengujian silang sebanyak 10 kali dan mendapatkan hasil akurasi optimal pada nilai k-23 sejumlah 79.99%.

Selain *K-Nearest Neighbor*, salah satu metode klasifikasi umum yang digunakan adalah *Naïve Bayes*. Pada penelitian sebelumnya oleh (Devita, et al., 2018) Kinerja metode *Naïve Bayes* dibandingkan dengan *K-Nearest Neighbor* untuk Klasifikasi Teks Artikel berbahasa Indonesia. Hasil yang didapatkan menunjukan metode *Naïve Bayes* memiliki kinerja yang lebih baik dengan tingkat akurasi 70% sedangkan metode *K-Nearest Neighbor* memiliki tingkat akurasi yuang cukup rendah yaitu 40%.

Contoh selanjutnya adalah pada penelitian yang dilakukan oleh (Rahman, et al., 2017) dimana dalam penelitiannya, metode *Multinomial Naïve Bayes* digunakan untuk Klasifikasi Berita Online dengan menggunakan *feature selection Document Frequency Thresholding* dan menggunakan *TF-IDF* untuk pembobotan *term* dan menghasilkan akurasi tertinggi pada saat menggunakan *TF-IDF* 86,62%.

Tak hanya seleksi fitur, *stopword* merupakan salah satu tahapan penting dari *preprocessing*, dalam tahap *preprocessing* perlu adanya suatu mekanisme sistem daftar *stopword* dinamik yang dapat menghasilkan daftar *stopword* yang sesuai dengan yang diperlukan sesuai saran dari penelitian sebelumnya (Rahutomo & Ririd, 2018). Contoh pembuatan *stopword* dinamik ada pada

penelitian selanjutnya yang dilakukan oleh (Dila Purnama Sari, et al., 2020) dimana dalam penelitiannya dilakukan pembentukan daftar *Stopword* menggunakan *Zipf Law* dan Pembobotan *Augmented TF-Probability IDF* pada klasifikasi Dokumen Ulasan Produk yang menggunakan metode *Support Vector Machine* dan *Polynomial Kernel* untuk memperoleh hasil klasifikasi. Daftar *Stopword* yang dibentuk secara dinamis dengan menggunakan metode *Zipf Law* dan pembobotan kata memiliki pengaruh terhadap hasil akurasi klasifikasi. Akurasi terbaik didapatkan pada saat persentase 15% untul daftar *stopword* yaitu dengan nilai *precision* 0,73, *recall* 0,7 dan *f-measure* 0,64

Lalu dilanjutkan penelitian oleh (Sa'rony, et al., 2019) analisis sentimen positif dan negatif yang dilakukan menggunakan *Multinomial Naïve Bayes* yang menggunakan *Raw Term Frequency* serta pembuatan *stopword* menggunakan *Term Based Random Sampling* dan berhasil mendapatkan *macroaverage* terbaik pada klasifikasi dengan *stoplist* 20 persen dengan *macroaverage* akurasi sebesar 0,94 *macroaverage precision* sebesar 0,945, *macroaverage recall* sebesar 0,94, dan *macroaverage f-measure* sebesar 0,938.

Contoh selanjutnya adalah pada penelitian (Imtiyazi, et al., 2015) dimana dilakukan perbandingan terhadap penggunaan *Multinomial Naïve Bayes* dengan *TF-IDF* dan dibandingkan terhadap *Multinomial Naïve Bayes* dengan *TF-Improved Gini*. Hasil yang didapatkan penggunaan *TF-IDF* memiliki kinerja lebih baik dibandingkan dengan *TF-iGini*.

Dari penelitian yang sudah disebutkan diatas, belum ada di antaranya yang melakukan Analisis Sentimen yang dibagi menjadi 3 kelas klasifikasi yaitu negatif, netral, dan positif yang menggunakan *Multinomial Naïve Bayes* sebagai metodenya serta pembentukan *Stopword* menggunakan Algoritme *Term Based Random Sampling* yang menggunakan *TF-IDF* sebagai pembobotan katanya.

2.1.1 *New Normal*

Dengan adanya pandemi Covid-19 ini, pemerintah Indonesia mengadakan sistem *New Normal* yakni dengan tujun untuk mempercepat penangan COVID-19 (Putsanra, 2020).

2.1.2 Kuliah Daring

Kuliah daring merupakan salah satu dari efek kebijakan sistem *New Normal* yang terjadi karena pandemi Covid-19 ini. Kuliah daring adalah metode pembelajaran yang dilakukan secara daring (*online*) dengan menggunakan berbagai fasilitas seperti platform *Zoom*, *Google Meet, Google Classroom*, situs pembelajaran universitas, dan lain- lain. Dengan adanya fasilitas- fasilitas tersebut, mahasiswa dan dosen tetap dapat berinteraksi satu sama lain layaknya kuliah secara tatap muka atau *offline* (Tania, 2020).

2.1.3 Twitter

Twitter merupakan sosial media besutan Amerika Serikat yang diluncurkan pada tahun 2006. Twitter merupakan salah satu contoh dari sosial media yang banyak digunakan oleh masyarakat Indonesia yang digunakan sebagai sarana pertukaran informasi di dunia digital. Dalam pengunaannya, Twitter memberi istilah kepada pertukaran informasi tersebut dengan nama *Tweets,* yang mana *Tweets* adalah suatu teks atau kata yang dibatasi panjang nya hingga 280 karakter yang nanti akan di-*posting* dalam *platform* Twitter tersebut.

2.2 Teks *Pre-processing*

Teks pre-processing merupakan langkah awal yang dilakukan dalam analisis sentimen untuk menyiapkan data yang berupa teks agar mudah untuk diproses nantinya (Gaddam, 2019). Teks pre-processing ini meliputi, *case folding*, *cleaning*, *tokenizing*, *stopword* removal dan *stemming*.

2.2.1 Case folding

Case folding adalah suatu tahapan untuk menyeragamkan kalimat menjadi huruf kecil atau *lowercase* semua. Contohnya, jika ada kalimat “Saya suka bermain Komputer” menjadi “saya suka bermain komputer”.

2.2.2 Cleaning

Cleaning adalah suatu tahapan pembersihan kalimat dari simbol-simbol, tanda baca, maupun angka. Contohnya, jika ada kalimat “Selamat pagi Adis, semoga harimu menyenangkan!” menjadi “Selamat pagi Adis semoga harimu menyenangkan”.

2.2.3 Tokenizing

Tokenizing adalah suatu tahapan untuk memisahkan antar kata dari suatu kalimat sehingga kata-kata tersebut menjadi satu tidak tergabung dengan kata-kata lainnya (Gaddam, 2019). Contohnya, jika ada kalimat “saya sedang bermain gitar” menjadi [‘saya’, ‘sedang’, ‘bermain’, ‘gitar’].

2.2.4 *Stopword Removal*

Stopword Removal adalah suatu tahapan untuk menghilangkan kata-kata yang kurang relevan berdasarkan kamus *stopword* yang digunakan (Gaddam, 2019). Kamus *Stopword* yang digunakan dalam penelitian ini adalah kamus *stopword* yang bersifat dinamis yang akan dibuat sesuai dengan kebutuhan sistem.

2.2.5 Stemming

Stemming adalah suatu tahapan untuk mencari kata dasar dari suatu kata (Gaddam, 2019). Contohnya jika ada kata “bermain” menjadi main.

2.3 *Term Based Random Sampling*

Term Based Random Sampling adalah suatu metode yang dapat digunakan untuk menghasilkan daftar *stopword* secara otomatis berdasarkan seberapa informatif kata tertentu (Lo, et al., 2005). Kita dapat mengetahui apakah kata tersebut *stopword* atau bukan dengan melihat kepentingannya, semakin tidak penting kata tersebut, maka lebih tinggi pula kata tersebut kemungkinan menjadi *stopword*. Untuk mencari nilai kepentingan dari suatu *term* dapat dilakukan dengan perhitungan dengan rumus dari teori *Kullback-Leibler.* Dengan rumus tersebut kita dapat memberi bobot dari suatu *term* pada dokumen sampel. Berikut rumus dari *Kullback-Leibler* direpresentasikan dalam Persamaan 2.1.

$$w(t) = P_x \,.\, log_2 \left(\frac{P_x}{P_c}\right) \quad (2.1)$$

Yang dimana P_x dipresentasikan dalam Persamaan 2.2 dan P_c dipresentasikan dalam Persamaan 2.3 adalah sebagai berikut.

$$P_x = \frac{tf_x}{l_x} \quad (2.2)$$

$$P_c = \frac{F}{token_c} \quad (2.3)$$

Keterangan :

$w(t)$: bobot *term* t pada dokumen sampel

tf_x : frekuensi kueri *term* dalam dokumen sampel

l_x : jumlah dari panjang dokumen sampel

F : frekuensi kueri *term* dari keseluruhan dokumen

$token_c$: total token dari keseluruhan dokumen

Dalam perhitungannya dilakukan pemilihan acak *term* dari keseluruhan *term*, lalu kemudian ambil dokumen yang mengandung *term* tersebut dan cari semua *term* dalam dokumen tersebut. Setiap *term* dalam dokumen tersebut akan dilakukan perhitungan bobot nya menggunakan *Kullback-Leibler*. Lalu setelah perhitungan bobotnya diambil sejumlah *X term* yang diurutkan dari bobot terendah yang dimana *X* adalah parameter yang dapat diubah-ubah nantinya. Dalam prosesnya pemilihan *term* acak ini dilakukan sebanyak *Y* kali dimana *Y* adalah sebuah parameter yang dapat diisi secara manual untuk mendapatkan hasil yang terbaik. Setelah melakukan proses yang dijelaskan sebelumnya dihitung rata-rata keseluruhan bobot yang didapat oleh *term* lalu

diambil sejumlah *L* dimana *L* adalah parameter yang dapat diubah nantinya. *L* adalah parameter yang menentukan berapa jumlah daftar *stopword* yang ingin digunakan.

2.4 *Term Frequency* – Inverse Document *Frequency* (*TF-IDF*)

Pada tahap ini dilakukan pembobotan kata yang mempresentasikan kata-kata tersebut untuk dilakukan perhitungan nantinya. Salah satu metode dalam *term weighting* yang sering digunakan adalah *Term Frequency – Inverse Document Frequency* ($tf.\ idf$) (Jones, 2004). Metode *TF-IDF* adalah penggabungan dua metode untuk melakukan pembobotan kata. TF atau *Term Frequency* adalah frekuensi kemunculan *term* pada suatu dokumen dan IDF atau *Inverse Document Frequency* adalah perhitungan *inverse* terhadap frekuensi dokumen yang mengandung kata tersebut (Prabowo, et al., 2016). Berikut rumus yang digunakan untuk perhitungan *TF-IDF* direpresentasikan pada Persamaan 2.4 dan Persamaan 2.5.

Berikut perhitungan nilai log tf :

$$tf_{\mathrm{t,\,d}} = 1 + \log\left(f_{\mathrm{t,\,d}}\right) \qquad (2.4)$$

Berikut perhitungan nilai idf :

$$idf(t) = \frac{\log(N)}{df_t} \qquad (2.5)$$

Keterangan :

$tf_{\mathrm{t,\,d}}$: frekuensi *term* pada dokumen *d*

$idf(t)$: nilai *Inverse Document Frequency* suatu *term t*

N : total dokumen

$df(t)$: nilai *Document Frequency* suatu *term t*

2.5 Algoritme *Naïve Bayes*

Algoritme *Naïve Bayes* yaitu algoritme klasifikasi *supervised* yang berbasis dengan teorema *Bayes* dengan asumsi independensi tiap fitur (Sawla, 2018). Algoritme ini menggunakan metode probabilistik dan statistik.

Algoritme ini mencari probabilitas tertinggi untuk proses klasifikasi. Perhitungan Algoritme *Naïve Bayes* direpresentasikan pada Persamaan 2.6 dan untuk perhitungan prior direpresentasikan pada Persamaan 2.7.

$$P(c|d) = P(c) * P(d|c) \qquad (2.6)$$

$$P(c) = \frac{N_c}{N} \quad (2.7)$$

Keterangan :

$P(c|d)$: *Posterior* atau Probabilitas kelas c diberikan dokumen d

$P(c)$: *Prior* atau Probabilitas awal muncul kategori c

$P(d|c)$: *Likelihood*

N_c : Jumlah dokumen kelas *c*

N : Jumlah seluruh dokumen

Dalam perhitungan *likelihood* atau *conditional probability* salah satu metodenya adalah menggunakan *Multinomial*. Perhitungan *conditional probability* dengan *multinomial* direpresentasikan pada Persamaan 2.8

$$P(w|c) = \frac{count(w,c) + 1}{count(c) + |V|} \quad (2.8)$$

Keterangan :

$P(w|c)$: Likelihood w dalam kelas c

$count(w,c)$: Jumlah kemunculan kata w pada kategori c

$count(c)$: Jumlah semua total kemunculan kata pada kategori c

$|V|$: Jumlah *term* unik atau fitur

Namun dalam penelitian kali ini digunakan *TF-IDF* sebagai pembobotan sehingga perhitungan *likelihood* atau *conditional probability* direpresentasikan pada Persamaan 2.9 (Rahman, et al., 2017).

$$P(w|c) = \frac{W_{ct} + 1}{(\Sigma_{\mathrm{W}\prime\in\mathrm{V}}\mathrm{W}'_{\mathrm{ct}}) + B'} \quad (2.9)$$

Keterangan :

$P(w|c)$: *Likelihood* w dalam kelas c

W_{ct} : Nilai pembobotan (W) *TF-IDF* dari *term t* di kategori *c*

$\Sigma_{\mathrm{W}\prime\in\mathrm{V}}\mathrm{W}'_{\mathrm{ct}}$: Jumlah bobot *TF-IDF* seluruh *term* pada kelas *c*

B' : Jumlah *IDF term* pada seluruh dokumen.

2.6 *Confusion Matrix*

Confusion Matrix adalah pengukur performa dari klasifikasi pembelajaran mesin (*Machine Learning*) (Narkhede, 2018). *Confusion Matrix* berisikan tabel untuk menampilkan hasil evaluasi yang didalamnya terdapat 2 kolom yaitu kelas hasil prediksi dan kelas sebenarnya.

Tabel 2.1 *Confusion Matrix*

		Predicted	
		Negatif	Positif
Actual	Negatif	TN	FP
	Positif	FN	TP

Keterangan :

- *True Negative* (TN) : jumlah dokumen yang *predicted* negatif dan *actual* negatif
- *False Positive* (FP) : jumlah dokumen yang *predicted* positif namun *actual* negatif
- *False Negative* (FN) : jumlah dokumen yang *predicted* negatif namun *actual* positif
- *True Postive* (TP) : jumlah dokumen yang *predicted* positif dan *actual* positif.

Fungsi dari *confusion matrix* untuk mempermudah evaluasi hasil klasifikasi untuk mencari *accuracy*, *precision*, *recall* dan *f-measure*. Berikut rumus *accuracy* direpresentasikan pada Persamaan 2.10, *recall* direpresentasikan pada Persamaan 2.11, *precision* direpresentasikan pada Persamaan 2.12, *f-measure* direpresentasikan pada Persamaan 2.13.

- *Accuracy* : kesesuaian nilai prediksi dengan nilai aktual

$$accuracy = \frac{TN + TP}{TN + FP + FN + TP} \quad (2.10)$$

- *Recall* : jumlah banyak atau sedikitnya kesesuaian informasi yang didapatkan berdasarkan sudut pandang kelas atau label yang digunakan

$$recall = \frac{TP}{TP + FN} \quad (2.11)$$

- *Precision* : tingkat ketepatan antara informasi yang diminta

$$precision = \frac{TP}{TP + FP} \quad (2.12)$$

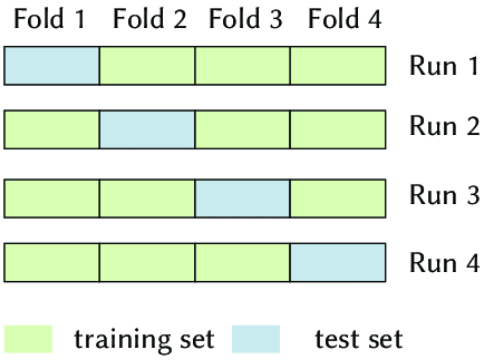
- *F-measure* : bobot harmonic mean pada *recall* dan *precision*

$$f - measure = \frac{2 * precision * recall}{precision + recall} \quad (2.13)$$

2.7 *K-Fold Cross Validation*

K-Fold Cross Validation adalah suatu metode yang berfungsi untuk membagi data sebanyak *K* dengan ukuran yang sama atau hampir sama rata.

Pada implementasinya pengujian *K-Fold* ini dilakukan dengan iterasi sebanyak K dimana pada setiap iterasinya data dibagi menjadi 2 tipe yaitu data latih dan data uji (Singh & Shukla, 2016). Berikut contoh illustrasi dari *K-Fold Cross Validation* yang ditunjukkan pada Gambar 2.1.



Gambar 2.1 Ilustrasi *K-Fold Cross Validation*

Sumber : https://www.researchgate.net/figure/The-technique-of-KFold-cross-validation-illustrated-here-for-the-case-K-4-involves_fig10_278826818 (2015)

Berdasarkan Gambar 2.1 ditunjukkan bahwa tiap iterasi dibagi menjadi 2 tipe data yaitu yang berwarna hijau adalah data latih dan yang berwarna biru adalah data uji. Untuk menghitung nilai evaluasi akhir maka dihitung rata-rata dari evaluasi tiap iterasi (Neale, et al., 2019).

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan metodologi yang digunakan pada penelitian ini. Metodologi yang digunakan berupa tipe penelitian, strategi penelitian, subjek penelitian, lokasi penelitian, teknik pengumpulan data, peralatan pendukung, implementasi algoritme.

3.1 Tipe Penelitian

Tipe penelitian yang dilakukan adalah bersifat non-implementatif dengan menggunakan pendekatan analitik. Penelitian bertipe non-implementatif adalah penelitian yang menguji hubungan terhadap suatu kejadian yang kemudian akan di analisis. Sedangkan pendekatan analitik memiliki fungsi untuk menjelaskan hubungan suatu kejadian dengan suatu objek penelitian yang sedang diteliti.

3.2 Strategi Penelitian

Strategi penelitian ini menggunakan studi kasus analisis sentimen masyarakat terhadap kuliah daring yang didapat dari Twitter. Data tersebut dilabeli manual oleh pakar lalu dibagi menjadi data latih dan data uji. Studi eksperimen berfokus kepada pengujian pada parameter *X, Y*, dan *L* pada *Term Based Random Sampling*.

3.3 Subjek Penelitian

Subjek penelitian yang digunakan pada penelitian ini adalah pengguna Twitter yang membahas mengenai kuliah daring.

3.4 Peralatan Pendukung

Peralatan pendukung yang digunakan pada penelitian ini adalah:

Tabel 3.1 Spesifikasi Hardware

Spesifikasi	Keterangan
Laptop	Dell XPS 15 9575
CPU	Core i7-8750G
GPU	NVIDIA GeForce GTX 1050 (4GB GDDR5)
RAM	16 GB
Tipe Memori	DDR4
SSD	512GB SSD PCIe NVMe

Tabel 3.2 Spesifikasi Software

Jenis	Keterangan
Operating System	MacOS Catalina 10.15.4
Bahasa Pemrograman	Python 3.7.7
IDE	Visual Studio Code
Library	Sastrawi, Pandas, Numpy, Re, Math

3.5 Lokasi Penelitian

Lokasi penelitian ini bertempat di Laboratorium Komputasi Cerdas, Fakultas Ilmu Komputer Universitas Brawijaya.

3.6 Teknik Pengumpulan Data

Teknik yang digunakan untuk pengumpulan data pada penelitian ini berasal dari Pengguna Twitter. Data diambil menggunakan *library Twint* yang berfungsi sebagai *data scrapper* Twittter untuk *Python*. Kata kunci yang digunakan pada saat pengumpulan data adalah "Kuliah Daring" dan "Kuliah Online". Pengumpulan data dilakukan dalam 7 bulan terhitung sejak April 2020 hingga Oktober 2020. Data yang dikumpulkan dilakukan proses normalisasi secara manual terlebih dahulu, kata yang dinormalisasi seperti berupa kata singkatan, kata tidak baku, dan kata-kata yang memiliki kesalahan penulisan.

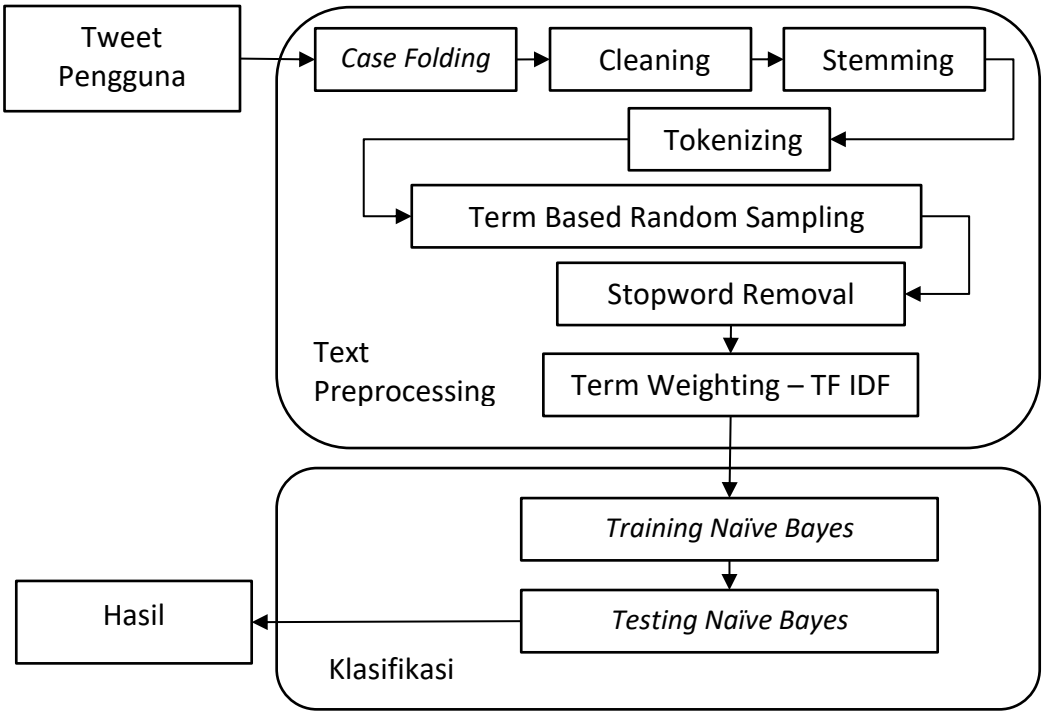
3.7 Data Penelitian

Pada penelitian kali ini, data yang digunakan adalah *tweet* berbahasa Indonesia. Total dokumen yang akan diambil dari Twitter adalah 300 dokumen dimana dari 300 dokumen akan dibagi menjadi 240 data latih, dan 60 data uji. Proses klasifikasi akan dibagi menjadi 3 yaitu positif, netral dan negatif.

3.8 Teknik Analisis Data

Teknik Analisis Data pada penelitian ini ditujukan untuk mengetahui kinerja dari sistem yang telah dibuat sesuai algoritme yang diajukan oleh peneliti. Tingkat kinerja sistem diperoleh dengan menggunakan *Confusion Matrix* dan nantinya hasil yang diterima akan dimasukan ke dalam tabel *Confusion Matrix* dan dicari nilai *precision, recall, accuracy, dan f-measure* pada tiap iterasi *fold* dalam *K-fold cross validation*.

3.9 Perancangan Algoritme



Gambar 3.1 Perancangan Algoritme

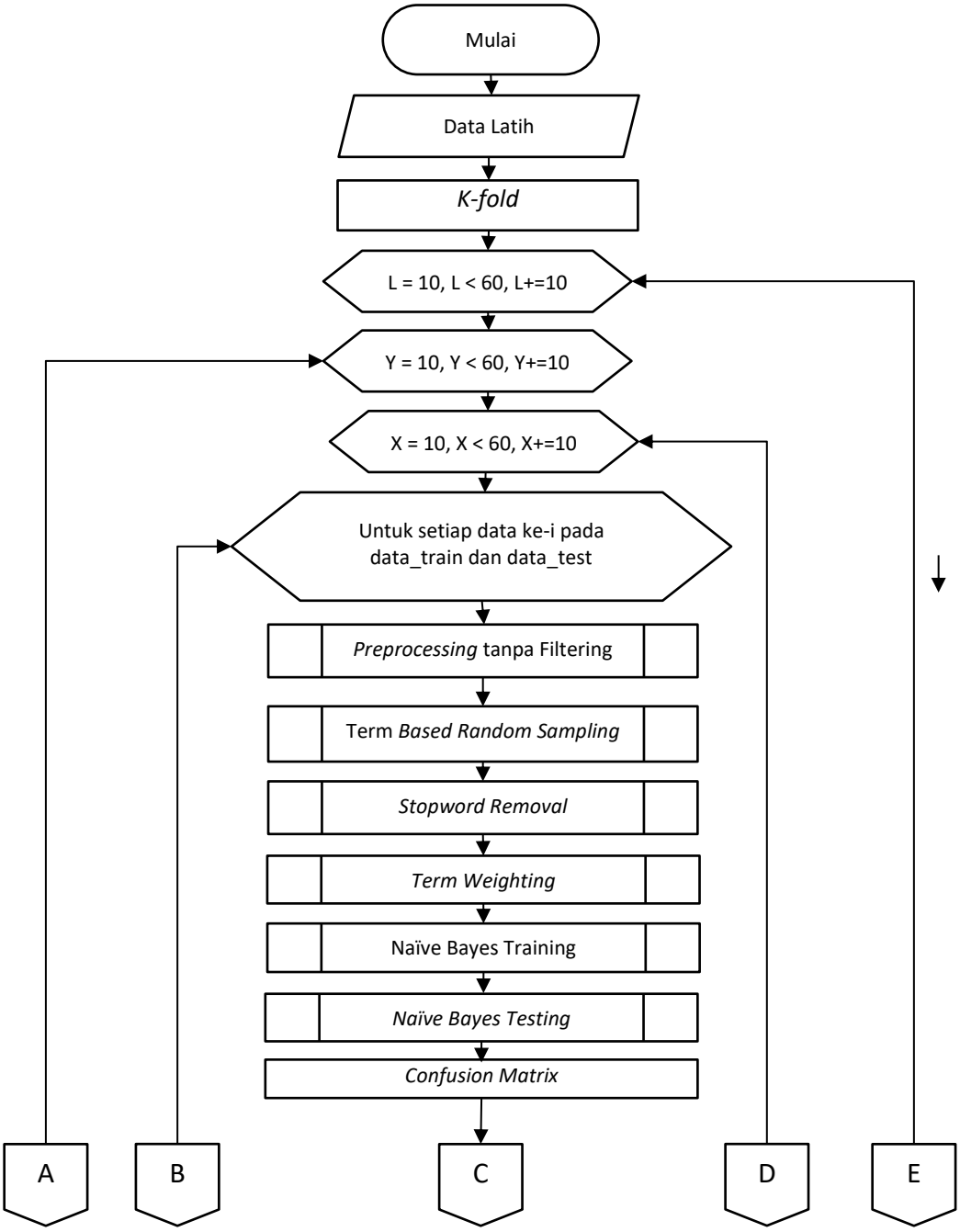
Perancangan Algoritme dapat dilihat pada Gambar 3.1. Tahapan ini diawali dengan melakukan pembuatan daftar *stopword* yang prosesnya diawali dengan *preprocessing* yang meliputi *case folding, cleaning, tokenizing*, dan *stemming*. *Preprocessing* ini bertujuan untuk merubah data latih berbentuk kumpulan dokumen menjadi term untuk dilakukan perhitungan algoritme *Term Based Random Sampling*. Setelah daftar *stopword* hasil dari algoritme *Term Based Random Sampling* didapatkan, selanjutnya data tersebut akan melalui tahap *stopword removal* atau penghapusan kata *stopword* dengan menggunakan daftar *stopword* yang telah dibuat sebelumnya. Setelah didapatkan daftar *term*, langkah selanjutnya adalah proses pembobotan kata dengan menggunakan $tf.idf$ untuk merubah kata tersebut menjadi suatu nilai yang nantinya dapat diproses oleh sistem untuk dilatih dan diklasifikasi menggunakan metode Multinomial Naïve Bayes.

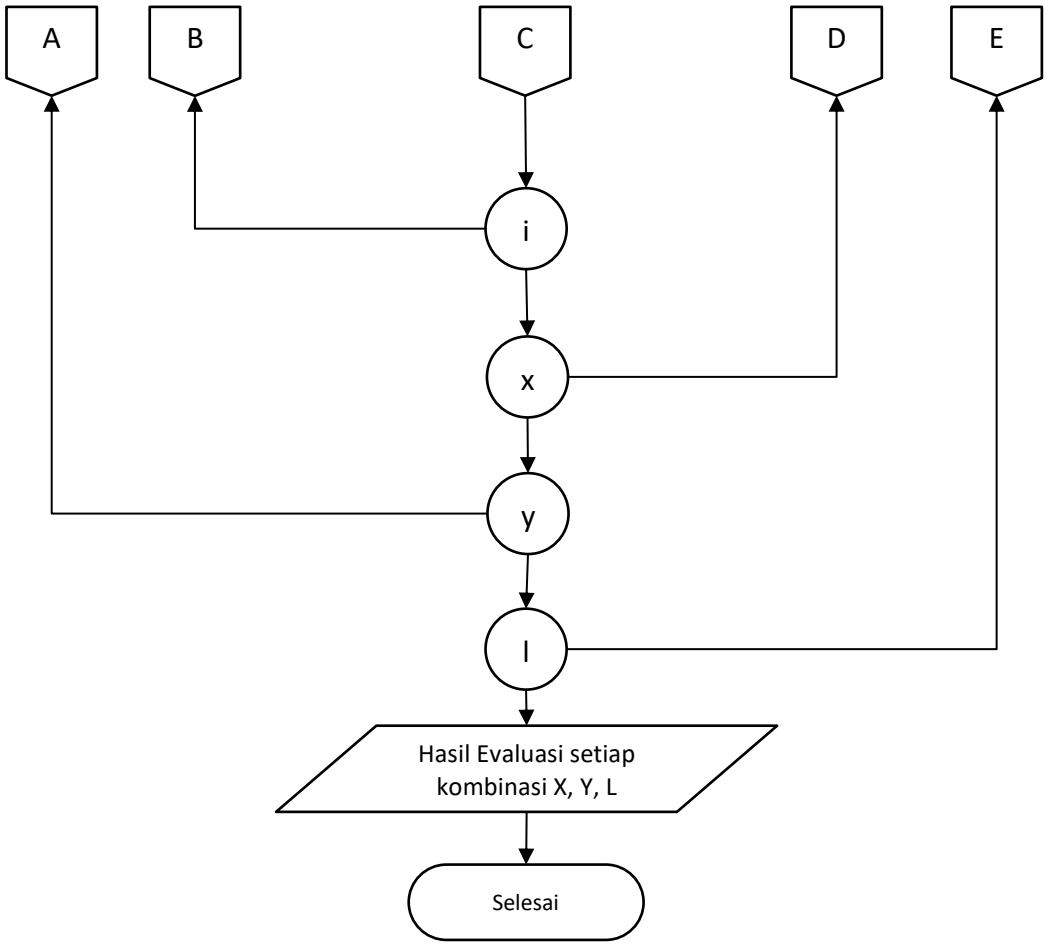
BAB 4 PERANCANGAN

Pada bab ini akan dijelaskan perancangan dengan diagram alir dari metode-metode yang digunakan pada penelitian ini serta manualisasi sistem klasifikasi dengan *Naïve Bayes* serta *Term Based Random Sampling* sebagai metode pembentuk daftar *stopword*.

4.1 Diagram Alir Sistem

Pada diagram ini akan dijelaskan bagaimana tahapan-tahapan dari sistem. Tahapan-tahapan tersebut dijelaskan pada Gambar 4.1.





Gambar 4.1 Diagram Alir Sistem

Berdasarkan pada Gambar 4.1 tahapan diawali dengan dengan data latih sebagai masukan dan dilanjutkan ke dalam *K-Fold* untuk dilakukan pembagian data latih dan data uji setiap foldnya. Setelah itu dilakukan perulangan L, Y, X dimana nilai masing-masing variabel L,Y,X akan berubah menjadi 10, 20, 30, 40, dan 50 setiap perulangannya. Selanjutnya akan kembali melakukan perulangan sebanyak jumlah *k-fold* pada data_train dan data_test yang dihasilkan dari proses *k-fold* sebelumnya.

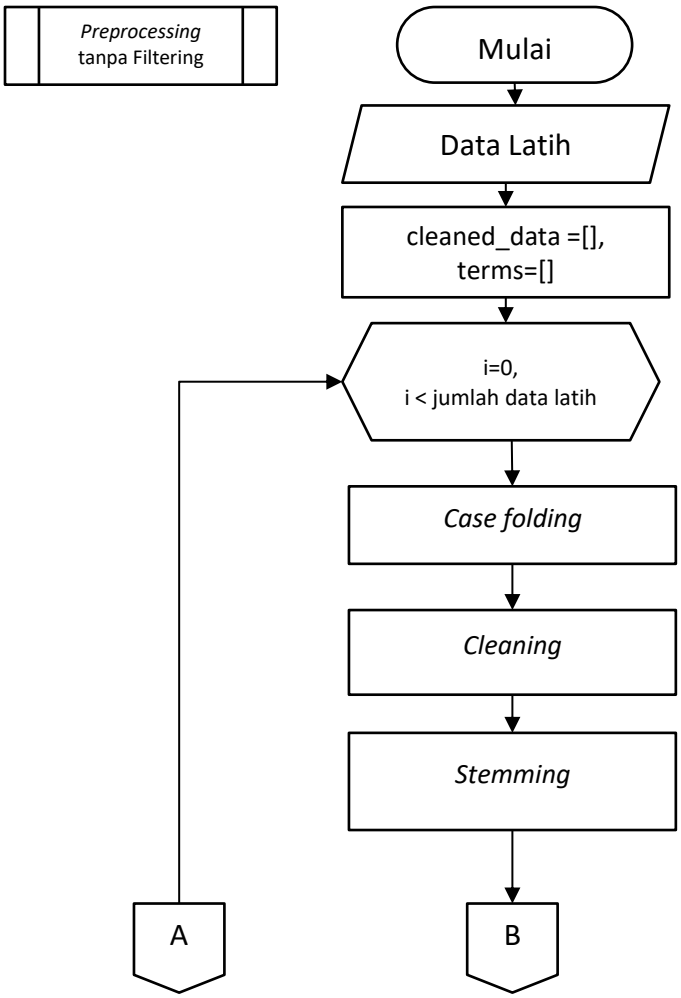
Setelah itu akan dilanjutkan dengan *preprocessing* data latih namun tidak menggunakan tahap *filtering*. Selanjutnya dilanjutkan oleh proses pembuatan daftar *stopword* menggunakan *Term Based Random Sampling* yang memiliki hasil berupa daftar *stopword*. Ketika daftar stopword sudah didapatkan maka langkah selanjutnya adalah penghapusan kata stopword yang akan dijalankan pada tahapan proses *Stopword Removal* sehingga setelah melalui proses tersebut maka data latih sudah bersih dari kata-kata stopword.

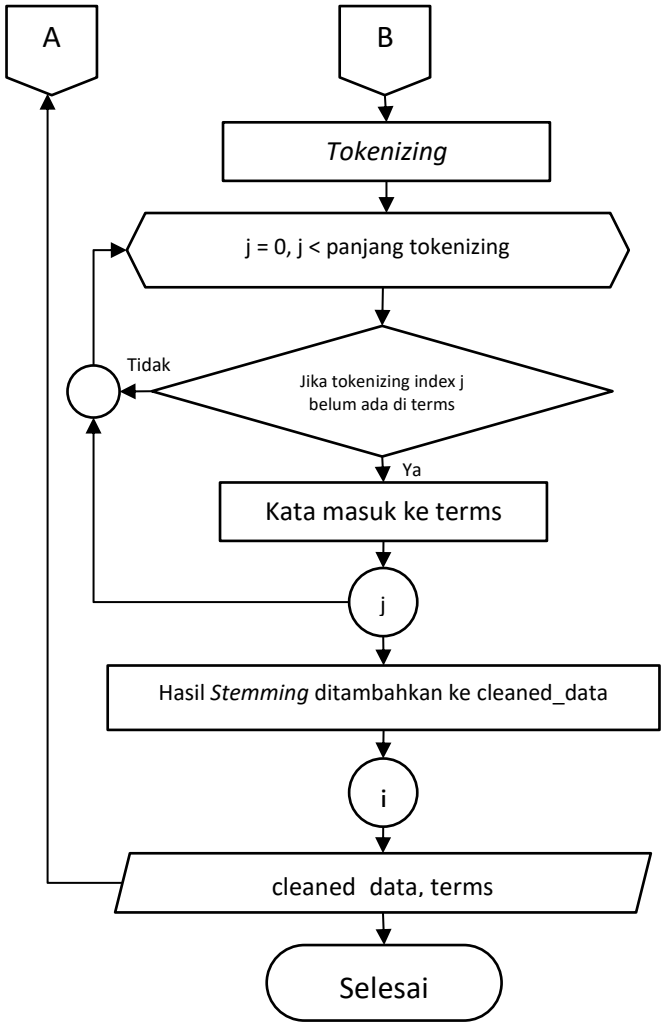
Setelah term didapatkan melalui tahapan sebelumnya, tahapan selanjutnya adalah proses *Term Weighting* menggunakan metode *term Frequency – inverse document Frequency*. Setelah bobot didapatkan akan dilanjutkan proses pelatihan *Naïve Bayes* yang akan menghasilkan *likelihood* serta *prior*. Pada tahapan ini akan menghasilkan *term*, *likelihood*, *prior* yang akan

digunakan pada proses berikutnya, yakni *Naïve Bayes Training dan Naïve Bayes Testing*. Setelah pengujian pada proses Naïve Bayes Testing dilakukan maka akan dilanjutkan untuk melakukan evaluasi dengan confusion matrix. Sehingga setelah melalui semua tahapan tersebut, sistem akan mengembalikan hasil evaluasi setiap kombinasi X, Y, dan L yang masing-masing dilakukan sebanyak 10-fold.

4.1.1 Diagram Alir *Preprocessing* tanpa *Filtering*

Pada tahapan *Preprocessing* tanpa *Filtering* ini terdapat beberapa tahapan yaitu *case folding, cleaning, tokenizing, stemming. Preprocessing* ini memiliki perbedaan dengan *preprocessing* pada umumnya karena tidak adanya *filtering* karena tujuan tahapan ini adalah proses menghasilkan suatu daftar *stopword.* Tahapan ini akan dijelaskan pada Gambar 4.2.

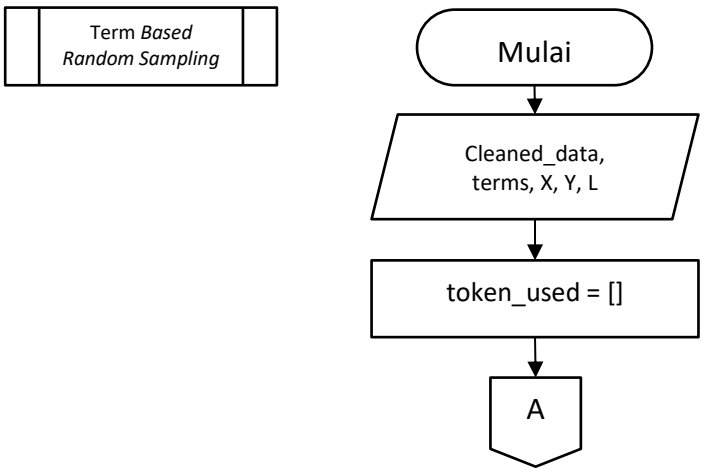


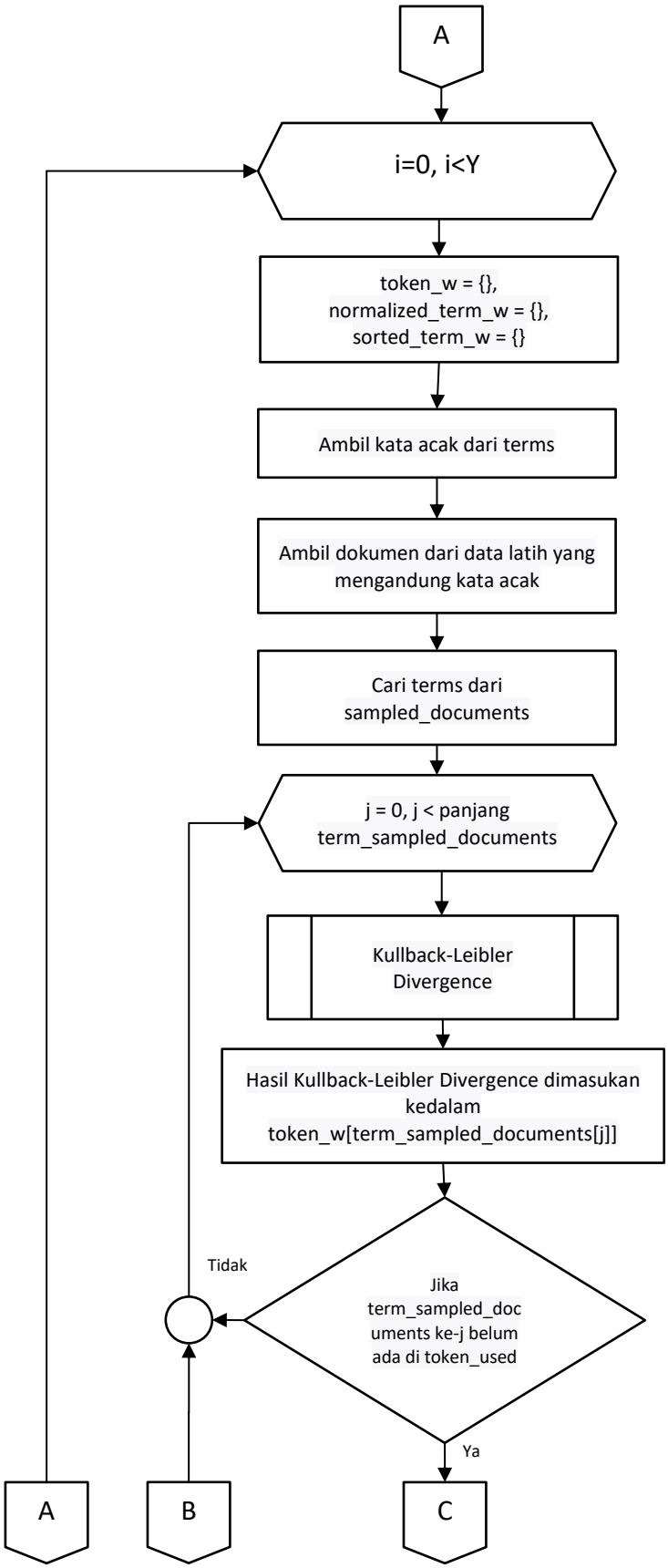


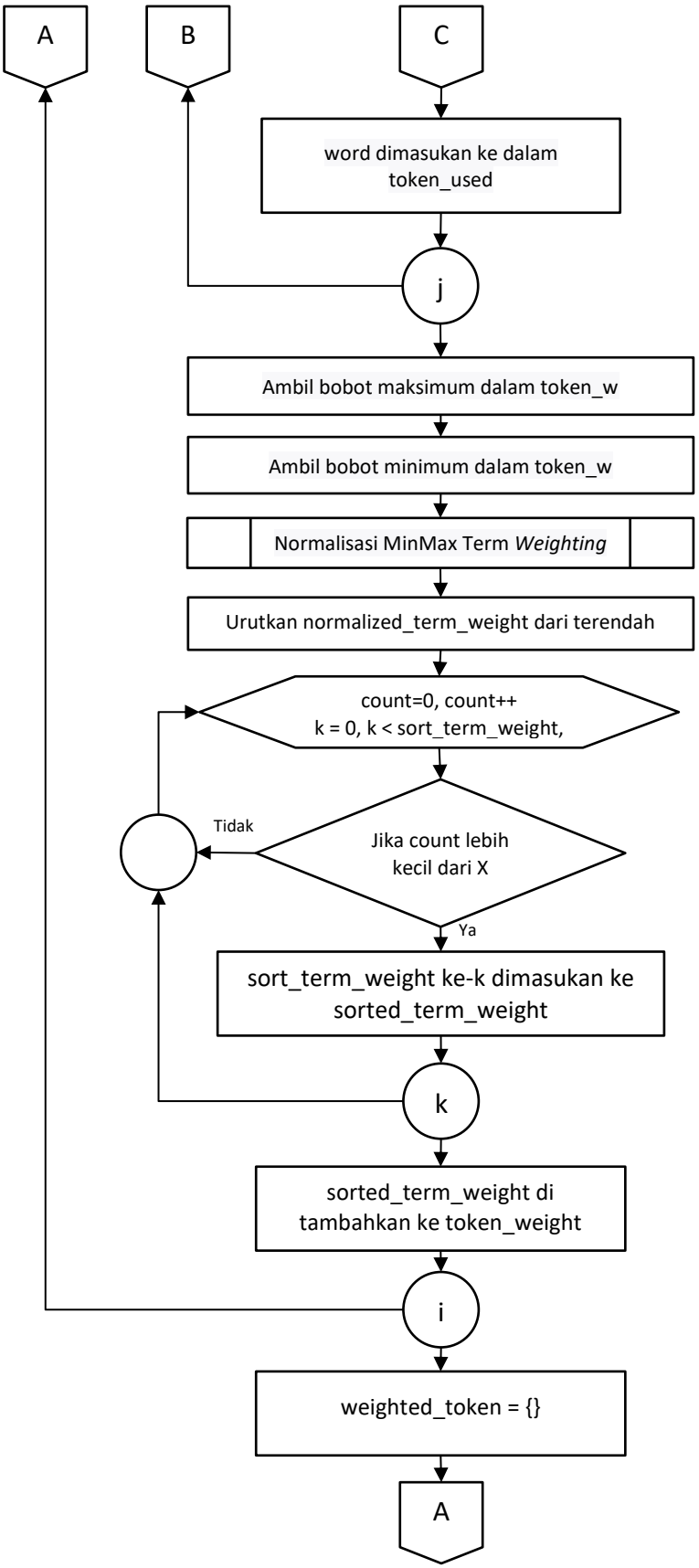
Gambar 4.2 Diagram Alir *Preprocessing* tanpa *Filtering*

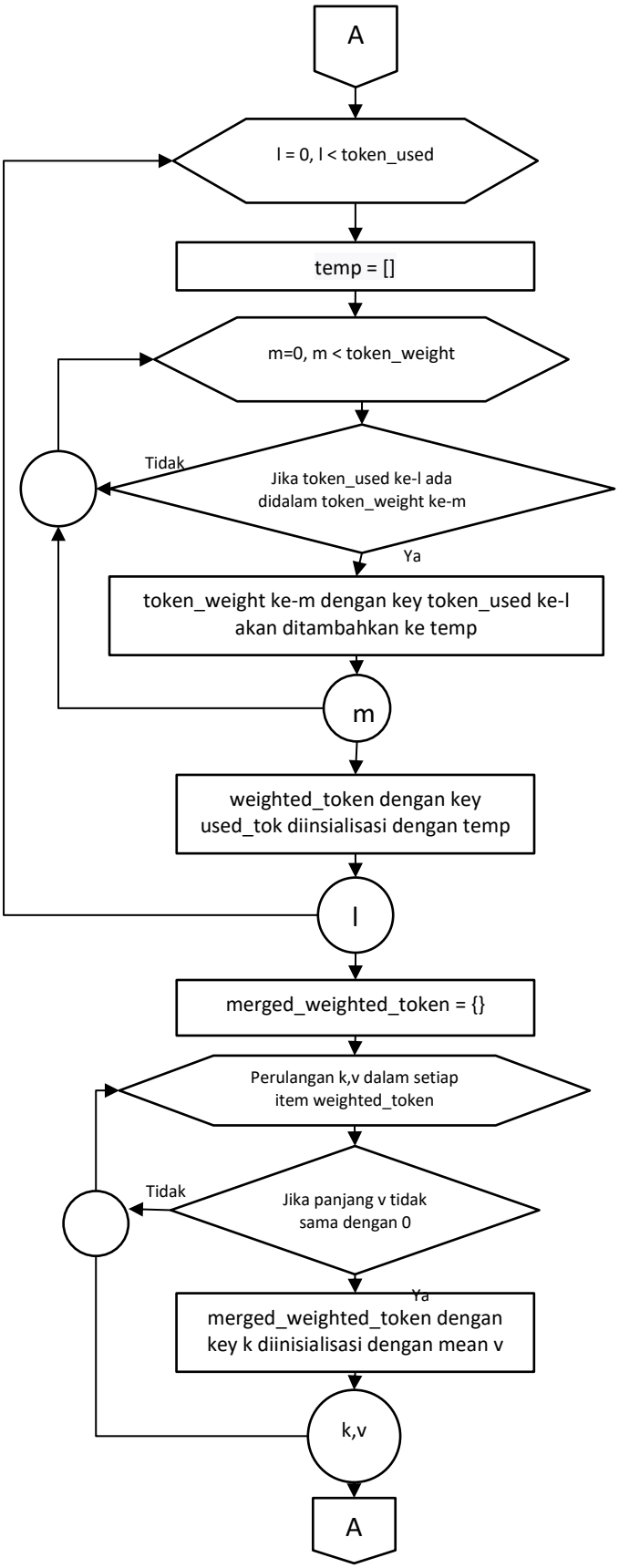
4.1.2 Diagram Alir *Term Based Random Sampling*

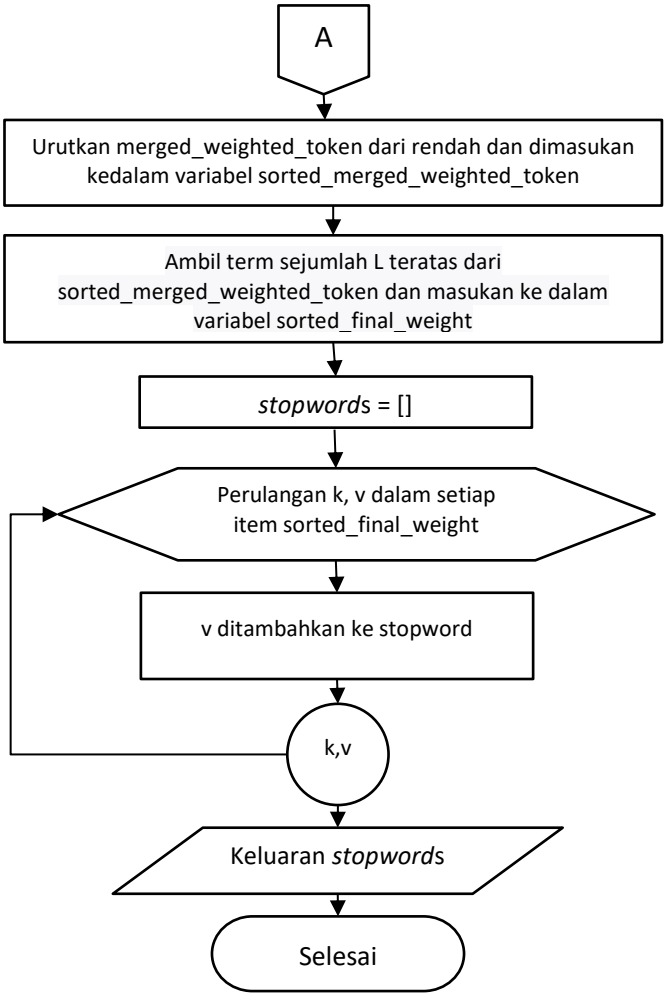
Pada tahapan *Term Based Random Sampling* ini terdapat beberapa tahapan-tahapan untuk mendapatkan *stopword* berdasarkan dokumen tertentu. Tahapan ini akan dijelaskan pada Gambar 4.3.




A
i=0, i<Y
token_w = {},
normalized_term_w = {},
sorted_term_w = {}
Ambil kata acak dari terms
Ambil dokumen dari data latih yang mengandung kata acak
Cari terms dari sampled_documents
j = 0, j < panjang term_sampled_documents
Kullback-Leibler Divergence
Hasil Kullback-Leibler Divergence dimasukan kedalam token_w[term_sampled_documents[j]]
Jika term_sampled_documents ke-j belum ada di token_used
Tidak
Ya
A
B
C


A
B
C
word dimasukan ke dalam token_used
j
Ambil bobot maksimum dalam token_w
Ambil bobot minimum dalam token_w
Normalisasi MinMax Term *Weighting*
Urutkan normalized_term_weight dari terendah
count=0, count++
k = 0, k < sort_term_weight,
Tidak
Jika count lebih kecil dari X
Ya
sort_term_weight ke-k dimasukan ke sorted_term_weight
k
sorted_term_weight di tambahkan ke token_weight
i
weighted_token = {}
A


A
l = 0, l < token_used
temp = []
m=0, m < token_weight
Tidak
Jika token_used ke-l ada
didalam token_weight ke-m
Ya
token_weight ke-m dengan key token_used ke-l
akan ditambahkan ke temp
m
weighted_token dengan key
used_tok diinsialisasi dengan temp
l
merged_weighted_token = {}
Perulangan k,v dalam setiap
item weighted_token
Tidak
Jika panjang v tidak
sama dengan 0
Ya
merged_weighted_token dengan
key k diinisialisasi dengan mean v
k,v
A

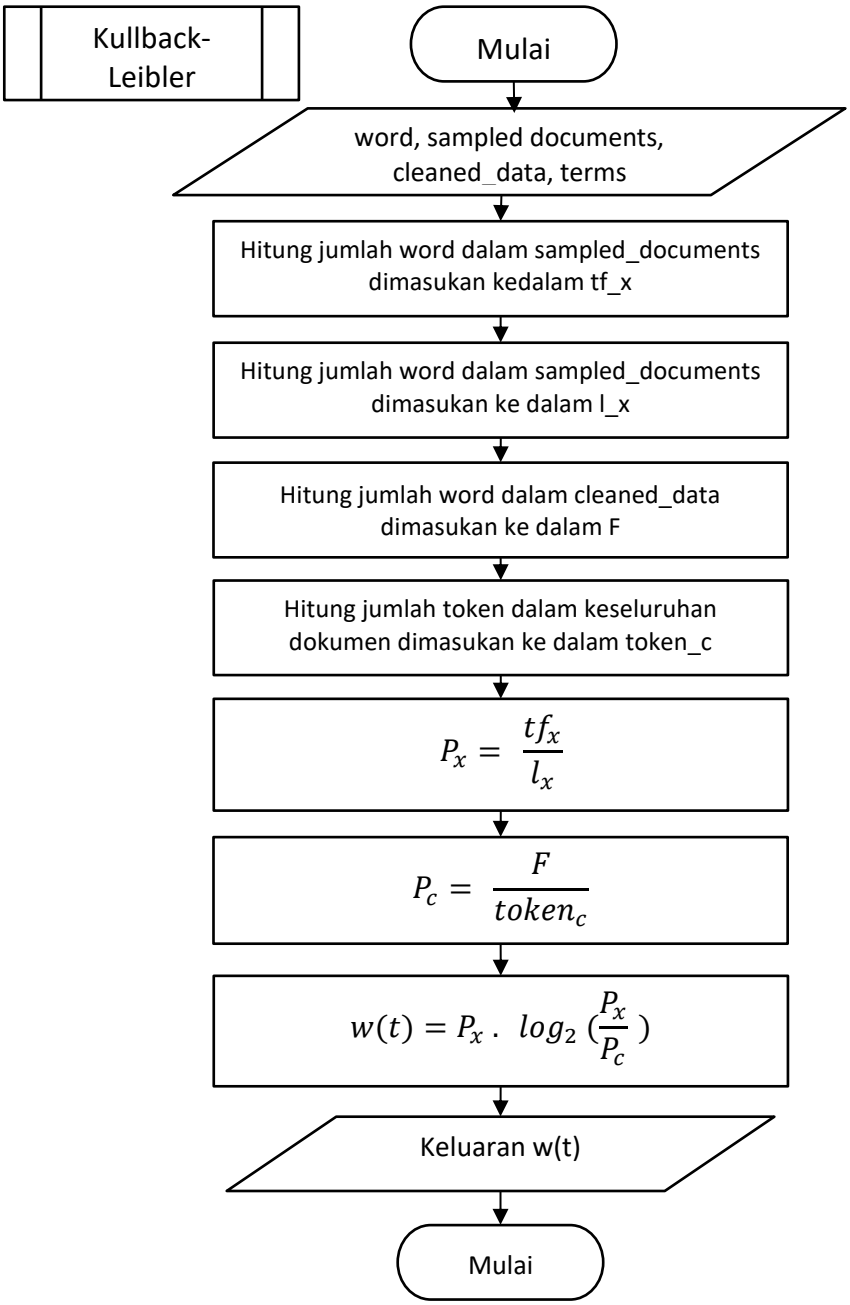


Gambar 4.3 Diagram Alir *Term Based Random Sampling*

Pada tahapan *Term Based Random Sampling* ini terdapat beberapa tahapan yaitu diawali dengan pilih *term* acak dari keseluruhan *term*, ambil dokumen yang mengandung dokumen tersebut, hitung bobot tiap *term* menggunakan *Kullback-Leibler*, normalisasi bobot dengan MinMax, ambil sejumlah X *term* yang diurutkan dari bobot terendah, lakukan proses sebelumnya sebanyak Y kali, hitung rata-rata keseluruhan bobot tiap *term*, dan yang terakhir ambil sejumlah L *term*.

4.1.2.1 Diagram Alir *Kullback-Leibler Divergence*

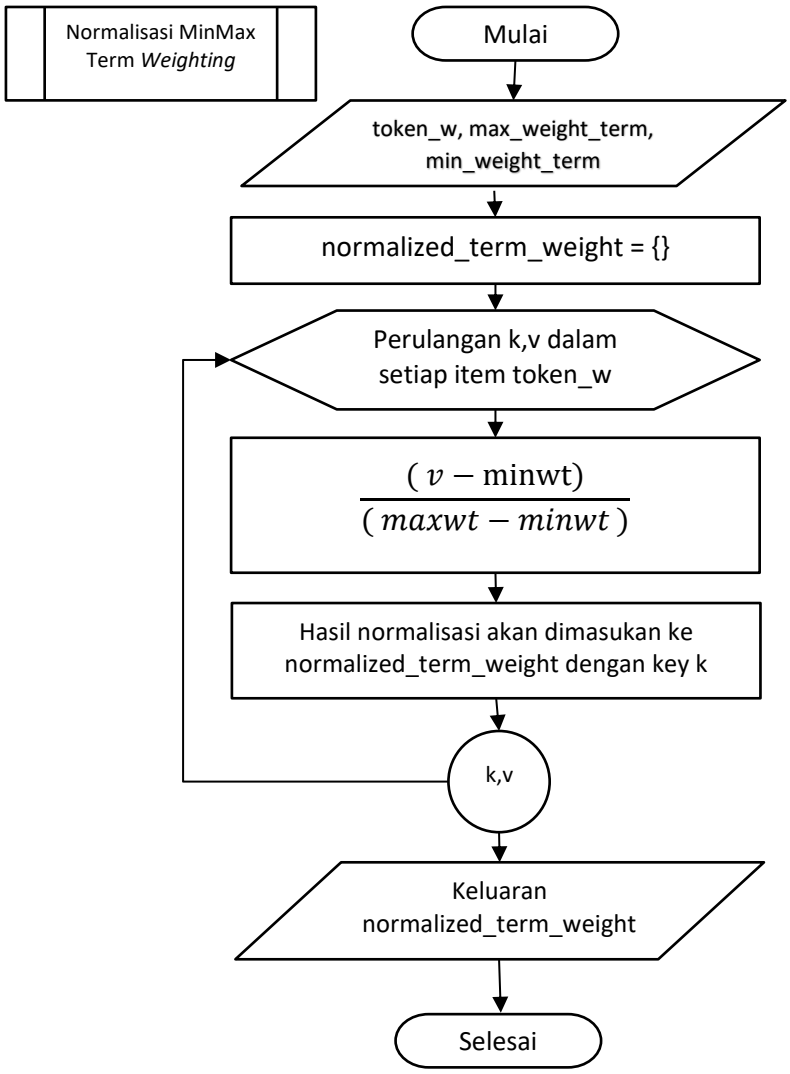
Pada tahapan *Kullback-Leibler Divergence* ini terdapat perhitungan pemberian bobot *term* untuk mendapatkan *stopword* berdasarkan dokumen tertentu. Tahapan ini akan dijelaskan pada Gambar 4.4.



Gambar 4.4 Diagram Alir *Kullback-Leibler Divergence*

4.1.2.2 Diagram Alir Normalisasi *MinMax Term Weighting*

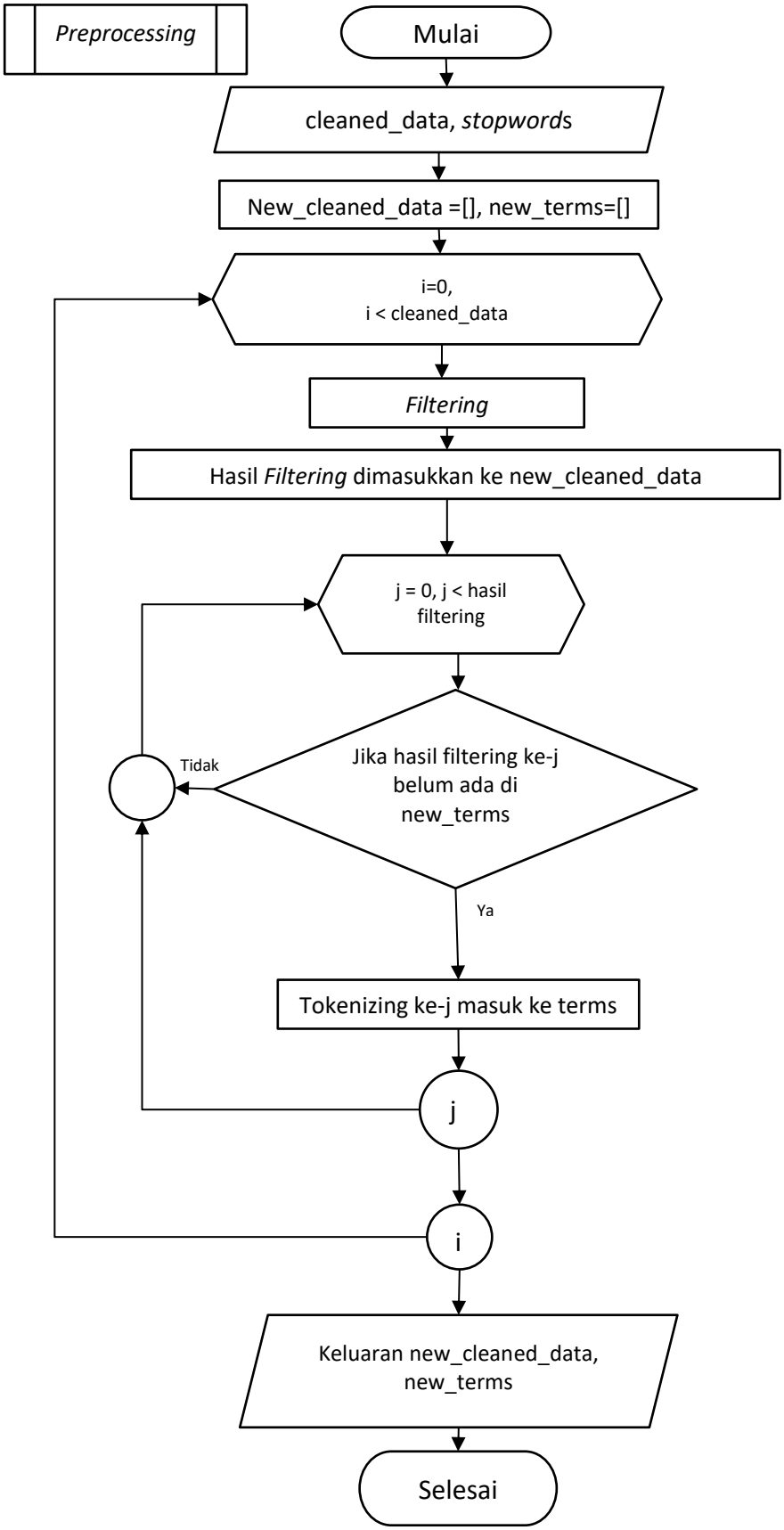
Pada tahapan ini terdapat perhitungan normalisasi bobot *term* dengan MinMax agar bobot dalam angka 0 hingga 1. Tahapan ini akan dijelaskan pada Gambar 4.5.



Gambar 4.5 Diagram Alir Normalisasi MinMax Term Weighting

4.1.3 Diagram Alir *Stopword Removal*

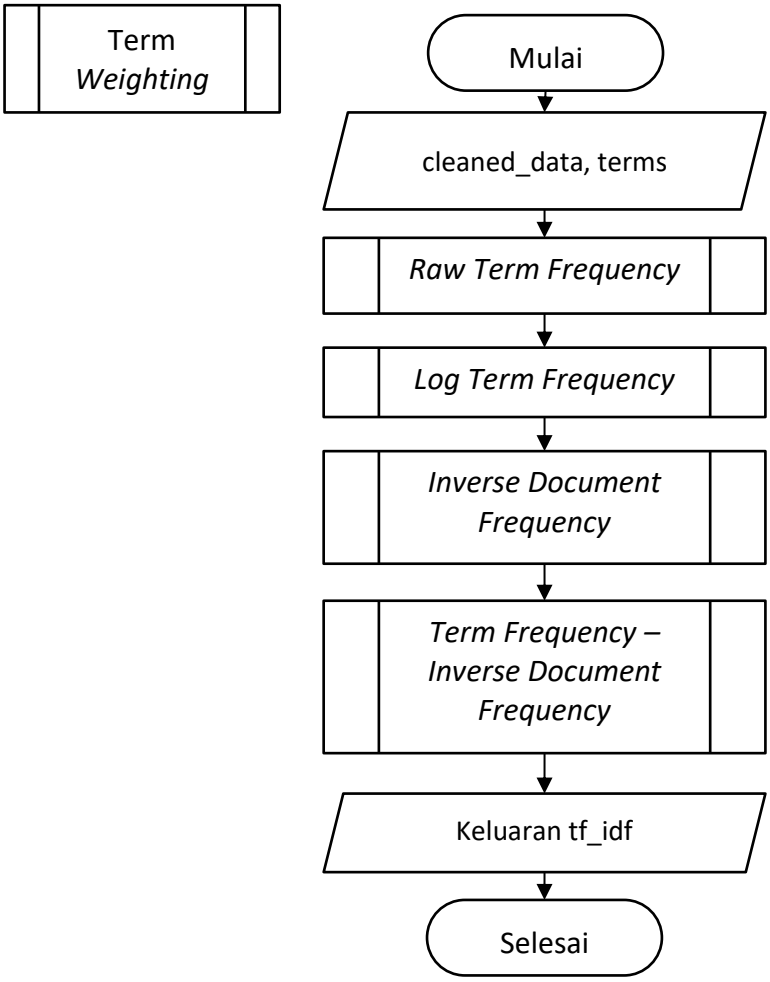
Tahapan *stopword removal* ini bertujuan untuk menghapuskan kata stopword pada data latih dengan daftar stopword yang telah dibentuk sebelumnya. Tahapan ini akan dijelaskan pada Gambar 4.6.



Gambar 4.6 Diagram Alir *Stopword Removal*

4.1.4 Diagram Alir *Term Weighting*

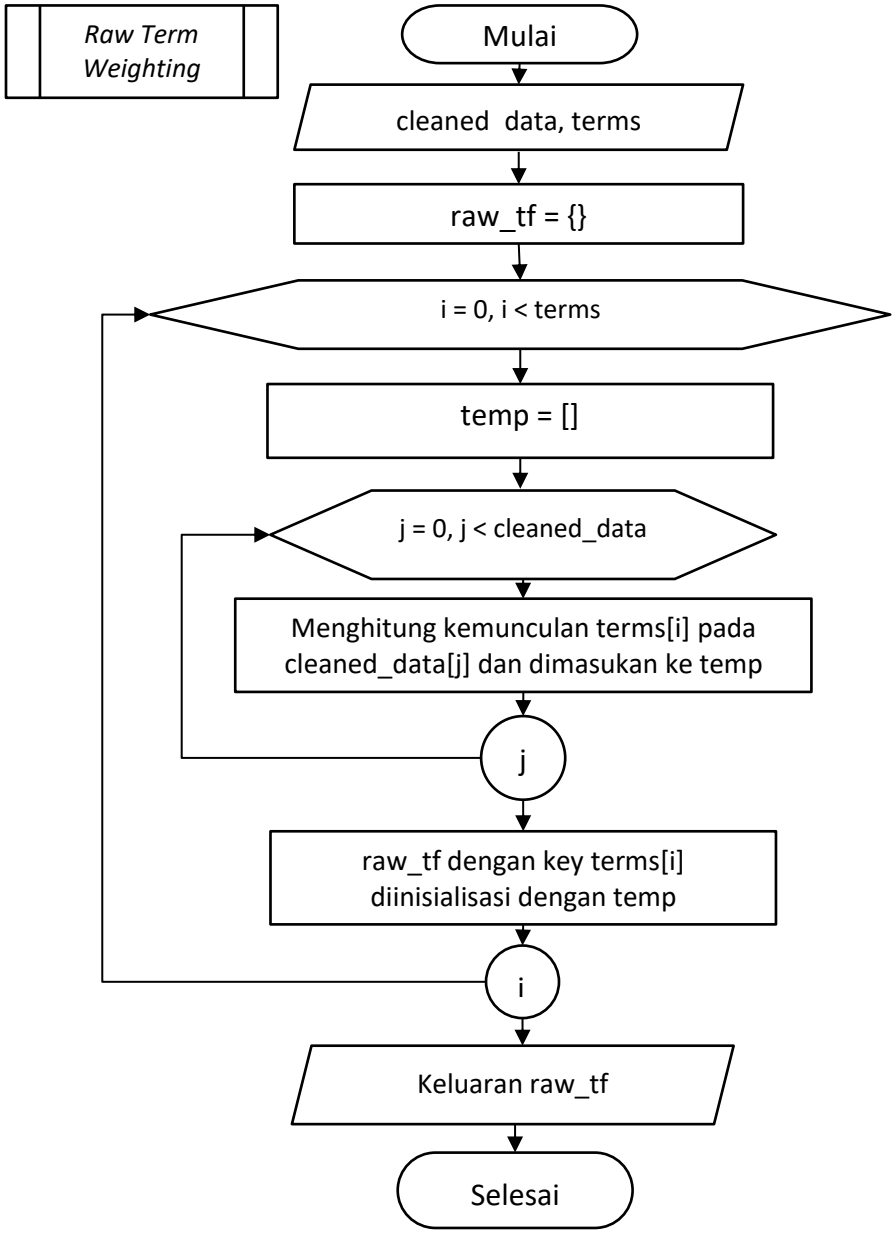
Dalam *Term Weighting* terdapat beberapa tahapan yaitu diawali dengan menghitung *raw term frequency*, *log term frequency, inverse document frequency,* dan *term frequency – inverse document frequency*. Tahapan ini akan dijelaskan pada Gambar 4.7.



Gambar 4.7 Diagram Alir *Term Weighting*

4.1.4.1 *Diagram Alir Raw Term Weighting*

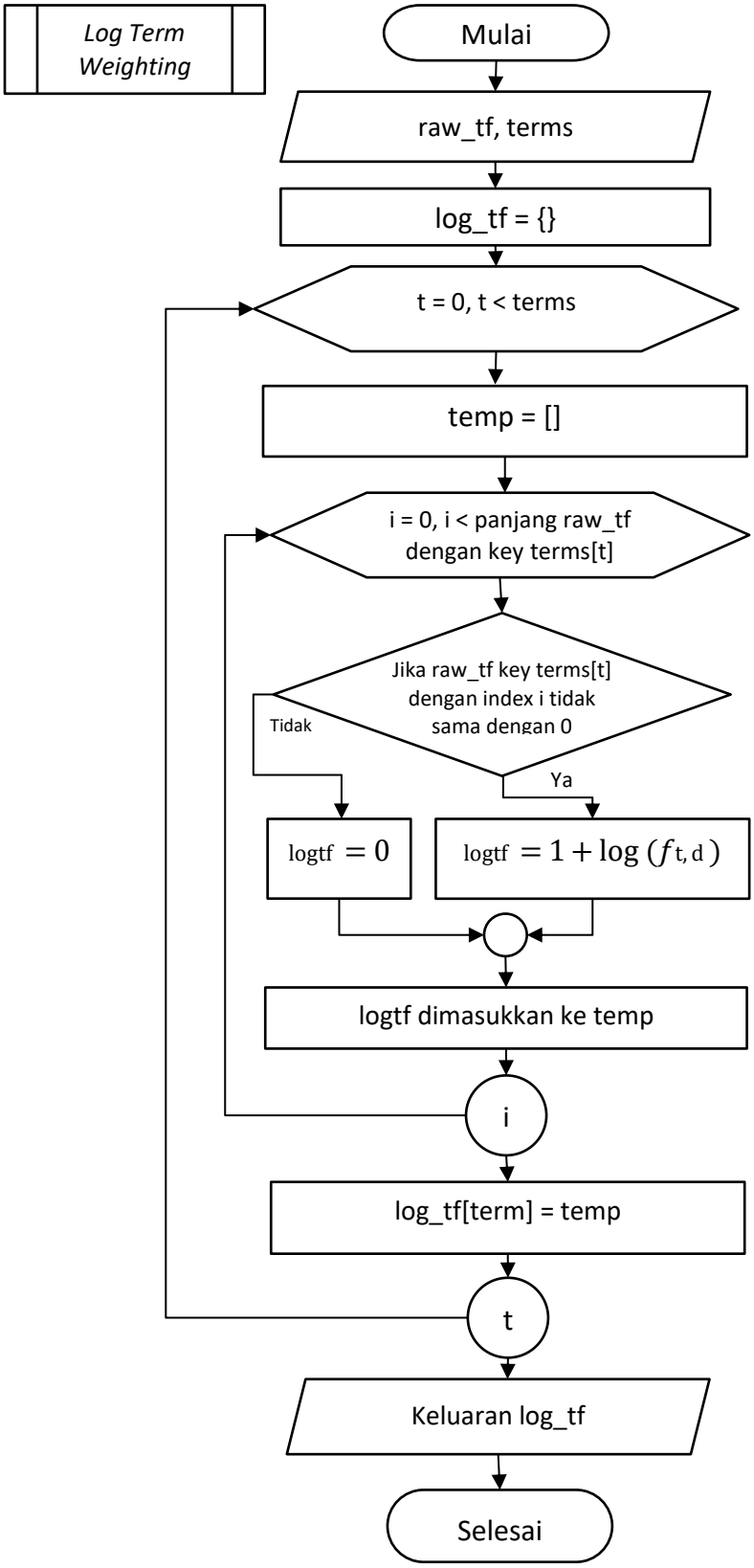
Tahapan *Raw Term Weighting* ini bertujuan untuk menghitung frekuensi setiap *term* yang terdapat dalam dokumen. Tahapan ini akan dijelaskan pada Gambar 4.8.



Gambar 4.8 Diagram Alir Raw Term Weighting

4.1.4.2 Diagram Alir *Log Term Weighting*

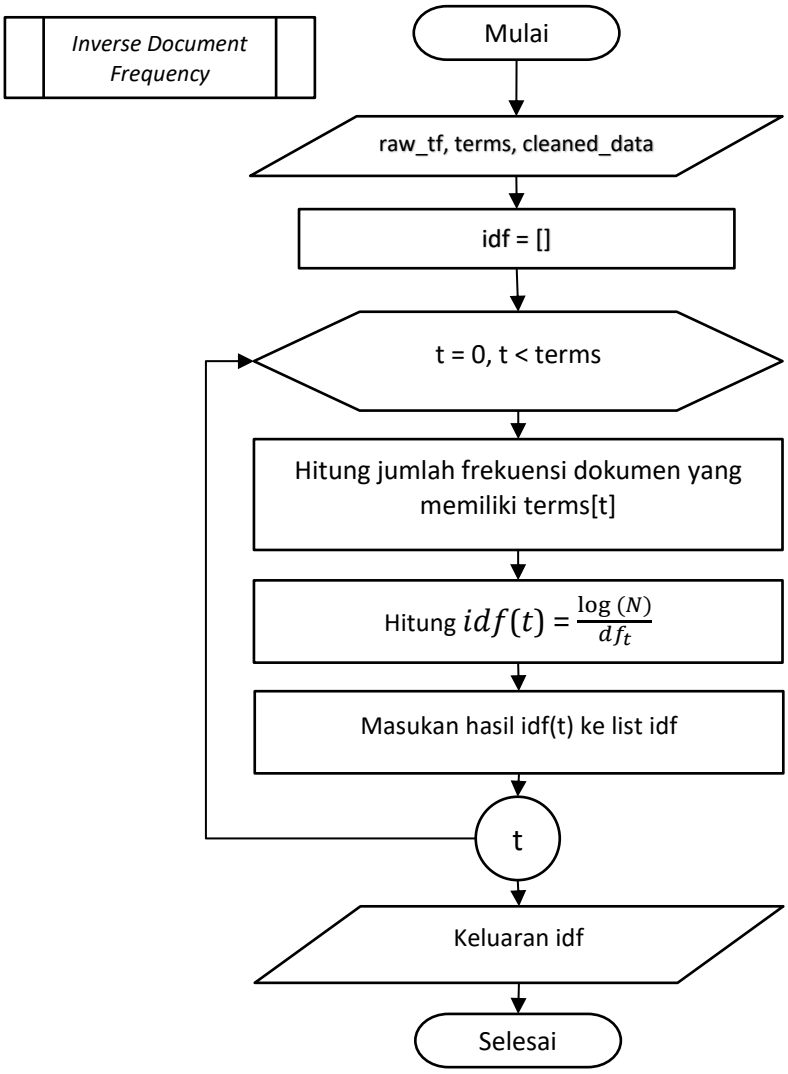
Tahapan *Log Term Weighting* ini bertujuan untuk menghitung frekuensi setiap *term* yang terdapat dalam dokumen lalu di logaritma. Tahapan ini akan dijelaskan pada Gambar 4.9.



Gambar 4.9 Diagram Alir Log Term Weighting

4.1.4.3 Diagram Alir *Inverse Document Frequency*

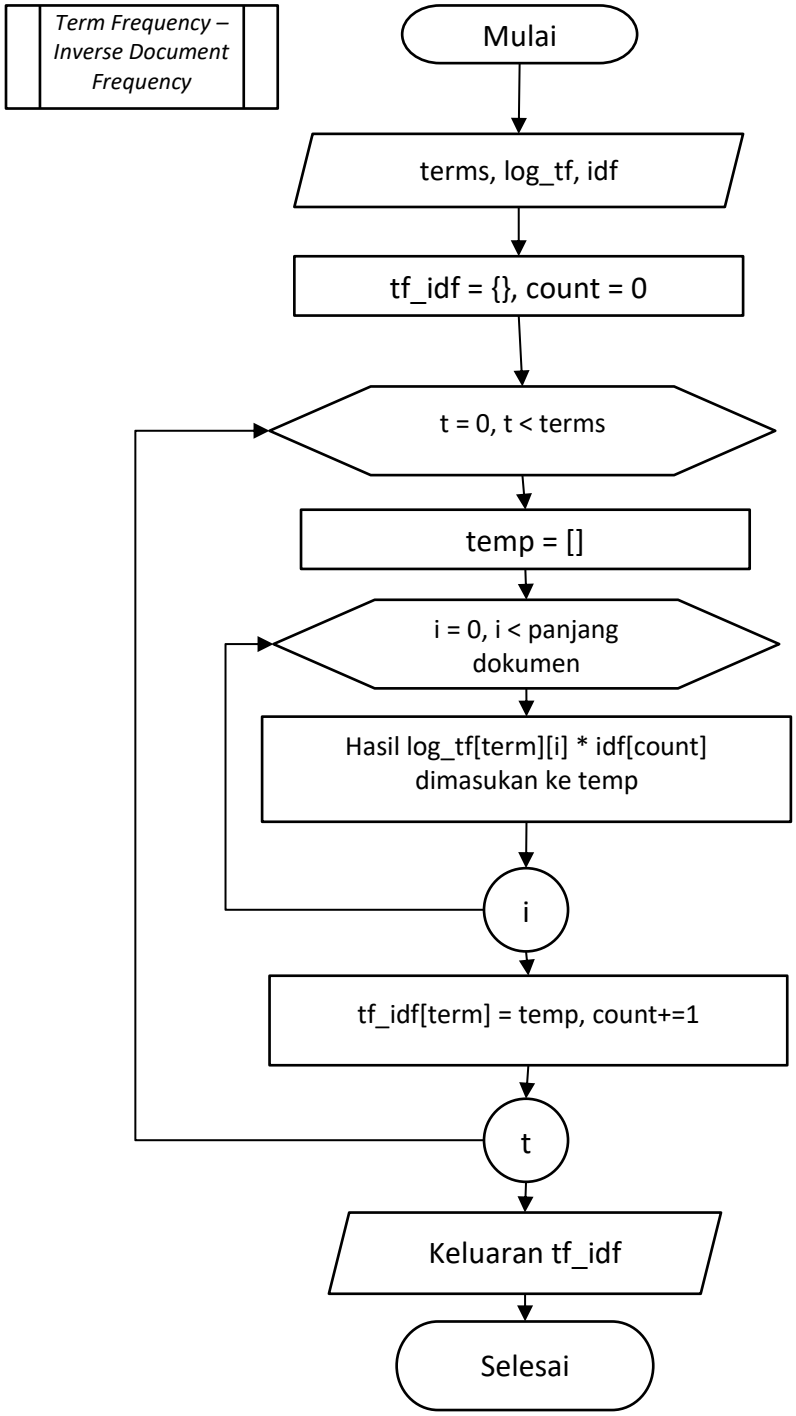
Tahapan *Inverse Document Frequency* ini bertujuan untuk perhitungan inverse terhadap frekuensi dokumen yang mengandung kata tersebut. Tahapan ini akan dijelaskan pada Gambar 4.10.



Gambar 4.10 Diagram Alir Inverse Document Frequency

4.1.4.4 Diagram Alir *Term Frequency - Inverse Document Frequency*

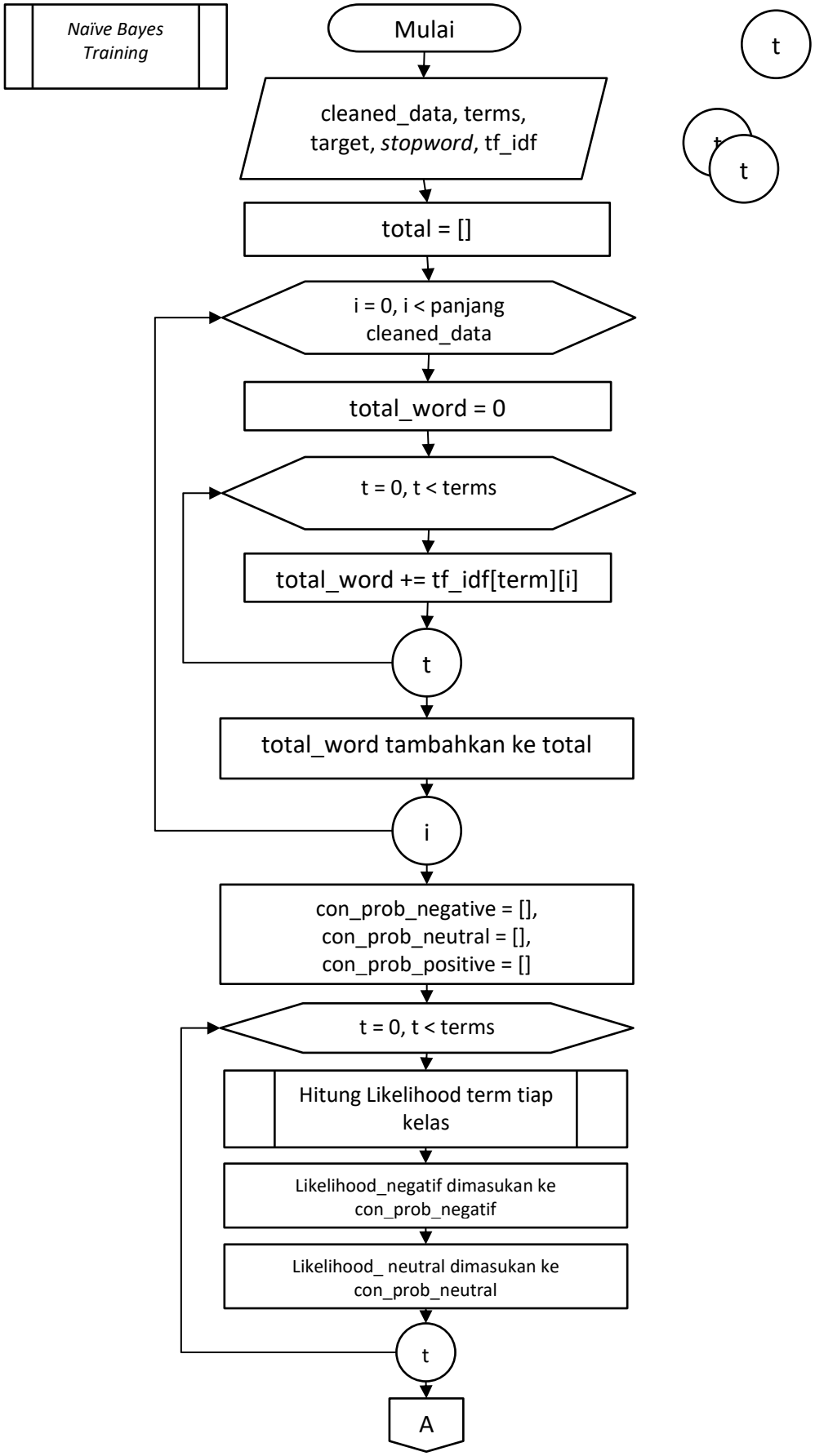
Tahapan *Term Frequency - Inverse Document Frequency* ini bertujuan untuk mengkalikan *log term Frequency* dengan *inverse document Frequency*. Tahapan ini akan dijelaskan pada Gambar 4.11.

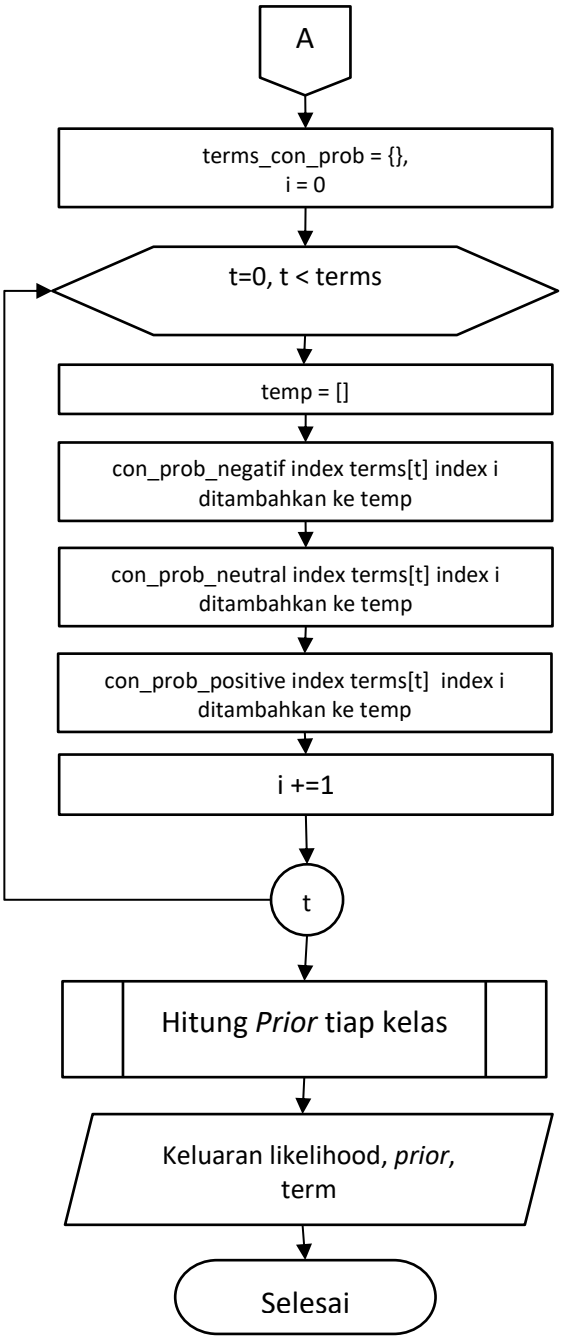


Gambar 4.11 Diagram Alir *Term Frequency - Inverse Document Frequency*

4.1.5 Diagram Alir *Naïve Bayes Training*

Pada tahapan *Naïve Bayes* Training ini terdapat beberapa tahapan yaitu mencari likelihood setiap kelas serta mencari *prior* tiap kelasnya. Tahapan ini akan dijelaskan pada Gambar 4.12.

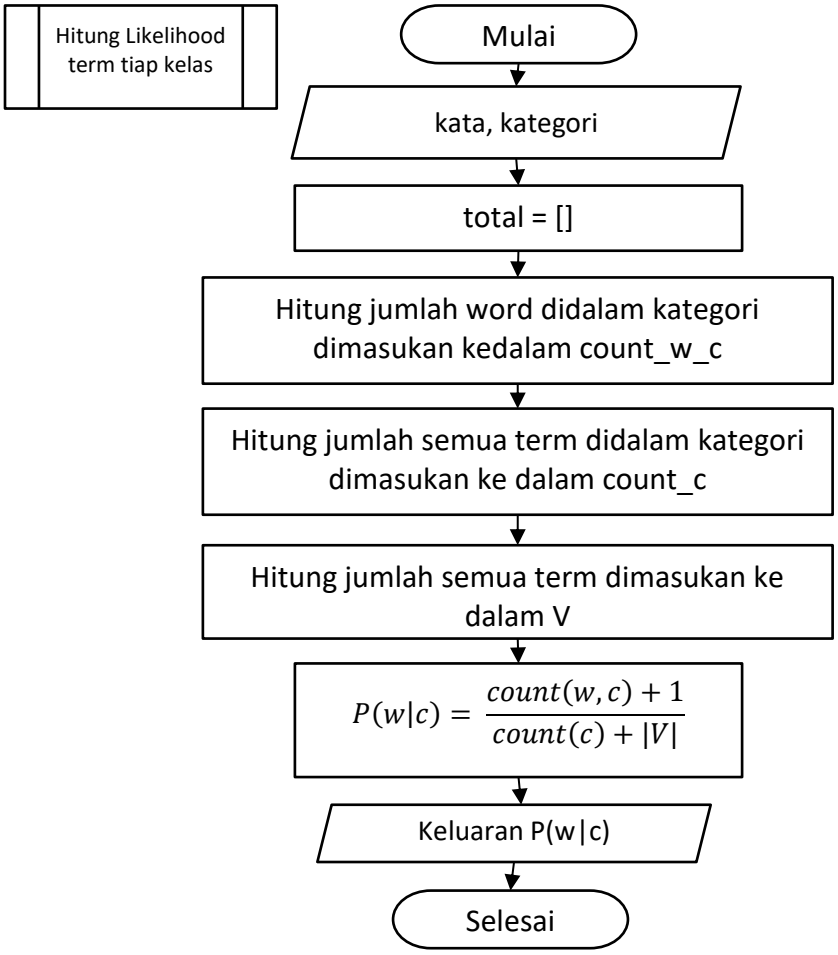

Naïve Bayes Training
Mulai
cleaned_data, terms, target, stopword, tf_idf
total = []
i = 0, i < panjang cleaned_data
total_word = 0
t = 0, t < terms
total_word += tf_idf[term][i]
t
total_word tambahkan ke total
i
con_prob_negative = [], con_prob_neutral = [], con_prob_positive = []
t = 0, t < terms
Hitung Likelihood term tiap kelas
Likelihood_negatif dimasukan ke con_prob_negatif
Likelihood_ neutral dimasukan ke con_prob_neutral
t
A
t
t
t



Gambar 4.12 Diagram Alir Naive Bayes Training

4.1.5.1 Diagram Alir Hitung *Likelihood term* tiap kelas

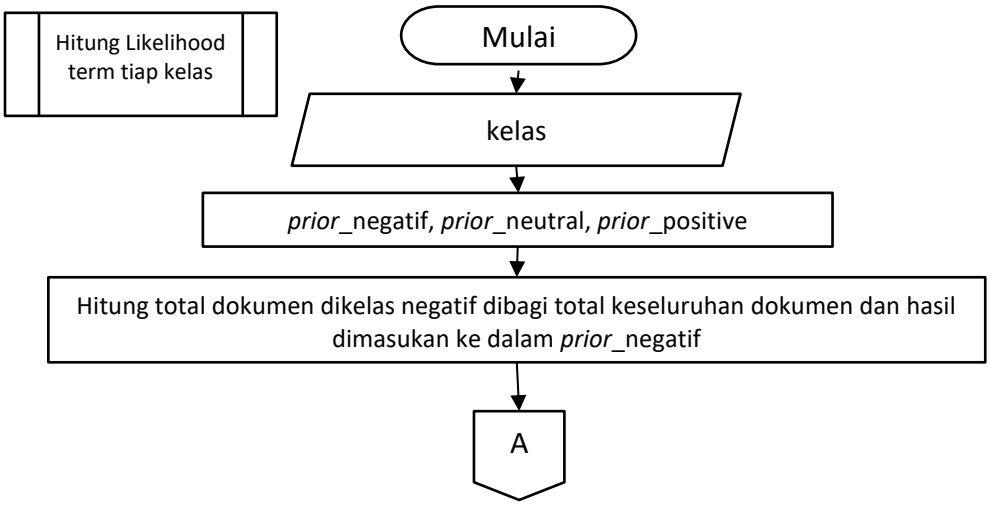
Pada tahapan Hitung *Likelihood term* tiap Kelas ini yaitu mencari likelihood *term* tertentu setiap kelasnya. Tahapan ini akan dijelaskan pada Gambar 4.13.

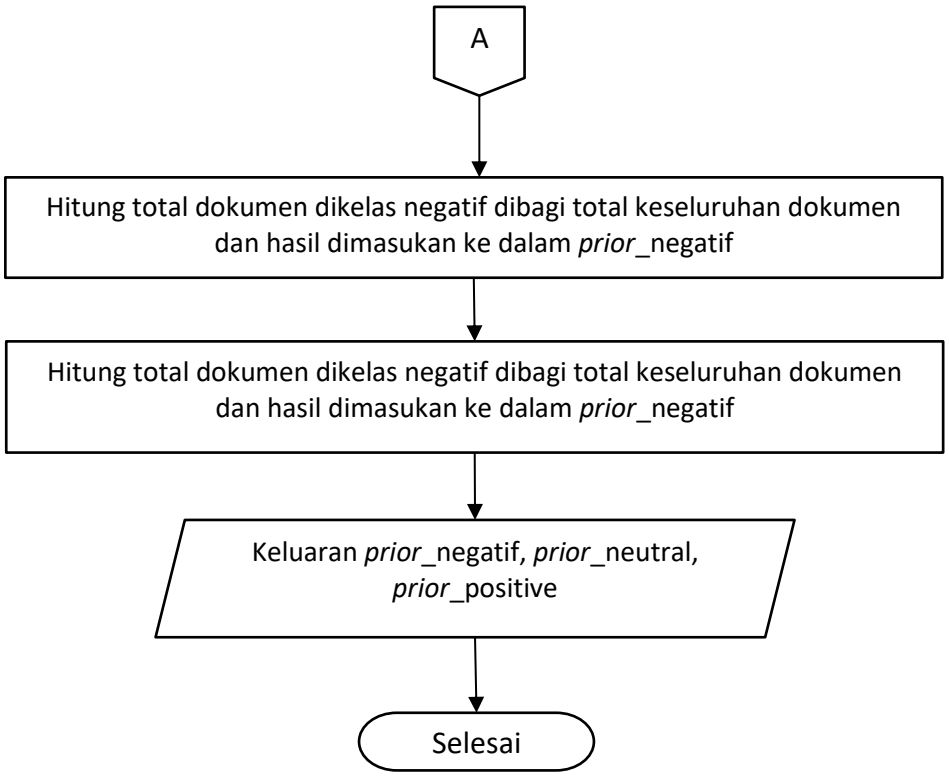


Gambar 4.13 Diagram Alir Hitung Likelihood *term* tiap kelas

4.1.5.2 Diagram Alir Hitung *Prior* tiap kelas

Pada tahapan Hitung *Prior* tiap Kelas ini yaitu mencari *prior* setiap kelasnya. Tahapan ini akan dijelaskan pada Gambar 4.14.

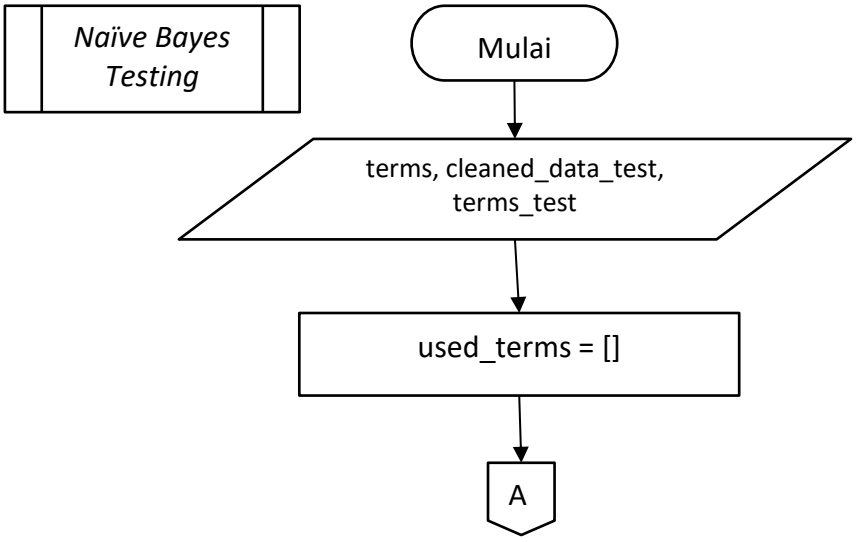


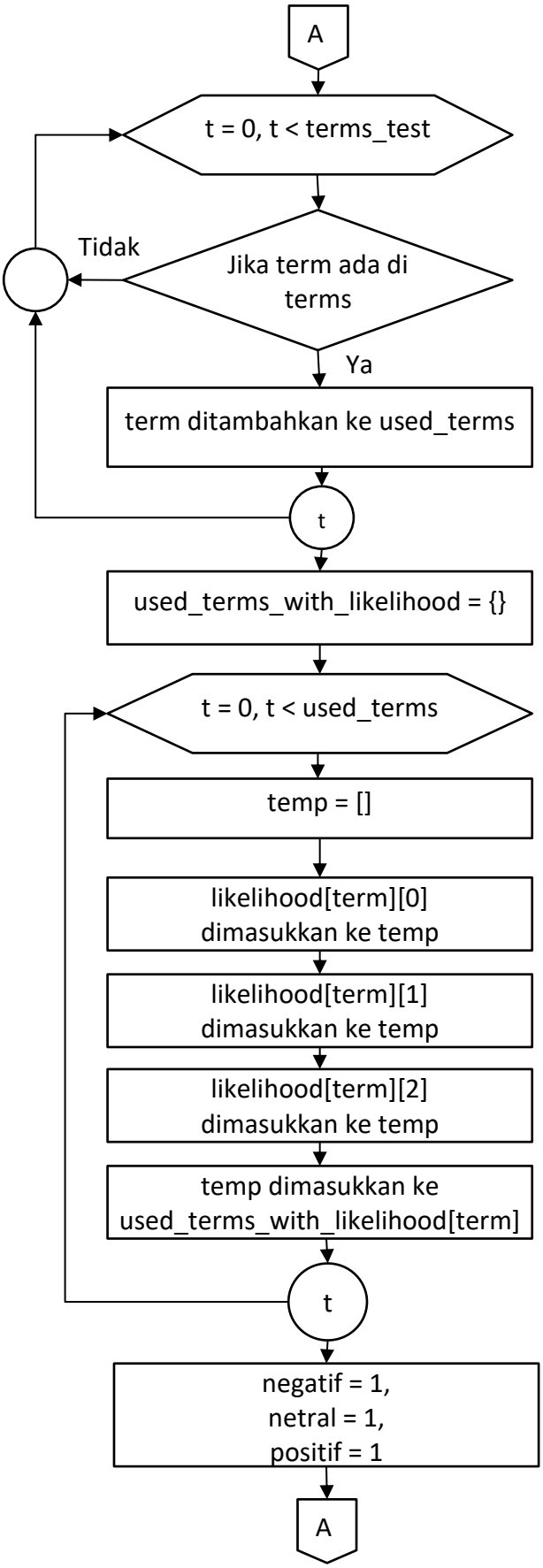


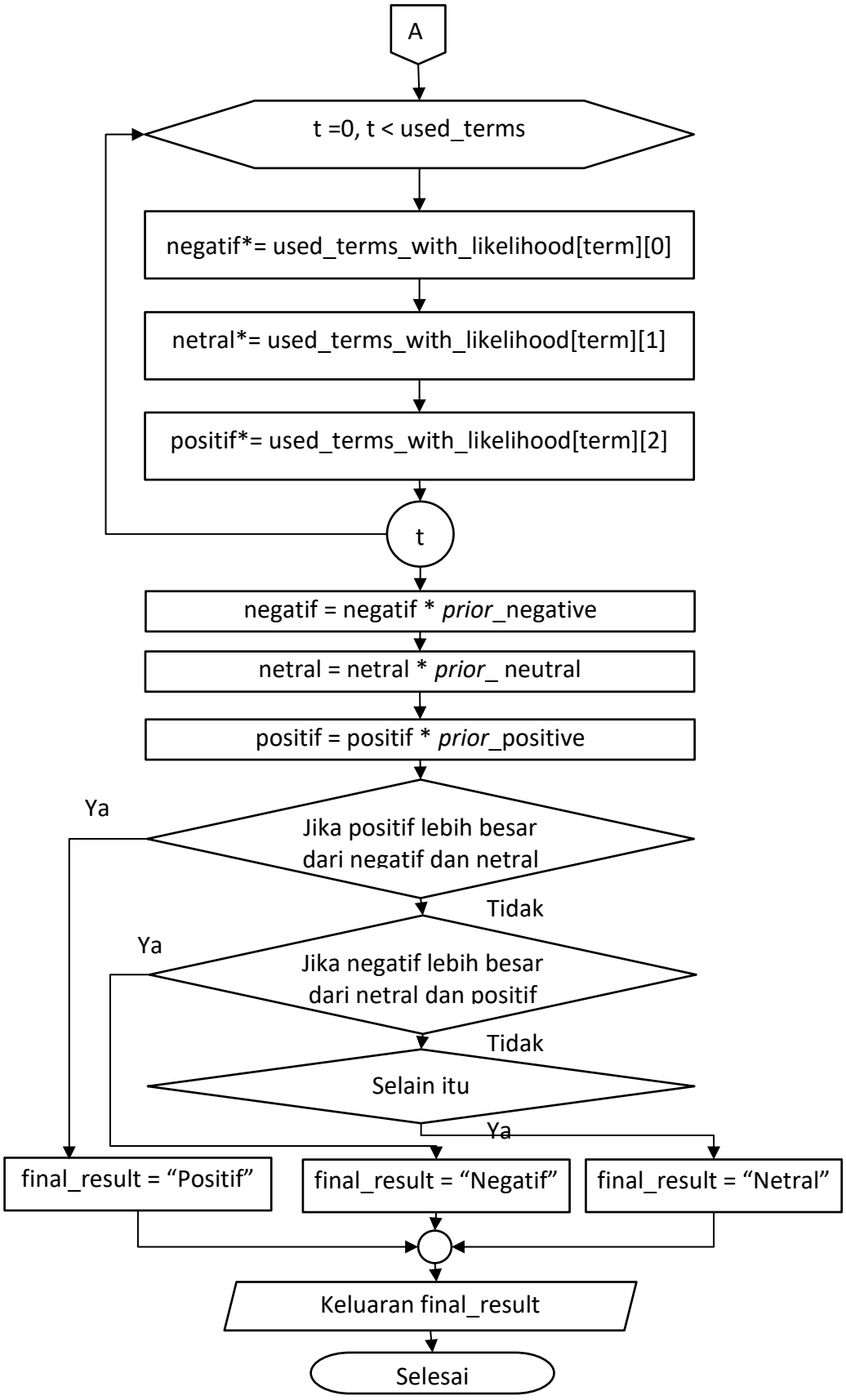
Gambar 4.14 Diagram Alir Hitung *Prior* tiap kelas

4.1.6 Diagram Alir *Naïve Bayes Testing*

Pada tahapan *Naïve Bayes* Testing ini berfungsi untuk menghitung *posterior* setiap kelasnya dari data uji. Tahapan ini akan dijelaskan pada Gambar 4.15.




A
t = 0, t < terms_test
Tidak
Jika term ada di terms
Ya
term ditambahkan ke used_terms
t
used_terms_with_likelihood = {}
t = 0, t < used_terms
temp = []
likelihood[term][0] dimasukkan ke temp
likelihood[term][1] dimasukkan ke temp
likelihood[term][2] dimasukkan ke temp
temp dimasukkan ke used_terms_with_likelihood[term]
t
negatif = 1,
netral = 1,
positif = 1
A



Gambar 4.15 Diagram Alir Naive Bayes Testing

4.2 Manualisasi

Pada perhitungan manual ini akan diawali dengan persiapan data dan tahapan manualisasi akan dibagi menjadi 3 tahapan yaitu pembuatan daftar *stopword*, pelatihan, dan pengujian.

4.2.1 Persiapan Data

Data yang digunakan berupa tweet dari pengguna Twitter yang memiliki kuliah daring atau kuliah online sebagai kata kuncinya. Dalam proses perhitungan manualisasi ini akan digunakan 9 dokumen data latih dan 1 dokumen sebagai data uji. Berikut adalah sampel data yang digunakan yang ditunjukkan pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Data Latih

No	Tweet	Kelas
1.	Aku selama kuliah online benar-benar tidak belajar sama sekali. Ujian selalu tidak jujur, tugas tinggal memindahkan dari internet, dosen hanya memberi tugas, tidak pernah ada penjelasan materi. Ditambah semester 5 mau tetap daring, mau jadi apa Aku	Negatif
2.	Rasanya mau berhenti kuliah saja kalau daring begini, seperti bayar cuma cuma, materi dikasih secara online, disuruh baca sendiri tanpa ada yang menjelaskan, berasa otodidak :"	Negatif
3.	Maaf, aku kuliah daring semakin malas. Kelas online saja ketiduran. Baik darimananya coba? Nilai sempurna bukan karena kita yang cerdas tapi karena dosennya yang kasihan sama kitanya. Tapi secara pemahaman, kosong sekali otak ini. Terima kasih	Negatif
4.	Sejujurnya aku oke-oke saja dengan kuliah daring. Cuma ya itu, kangen sama suasana kelas. Kalau corona sudah selesai, perpaduan offline-online sepertinya asik....	Netral
5.	Ada yang mempeributkan masalah kuliah online/daring, sebagian ada yang menyalahkan dosen ada juga yang menyalahkan diri sendiri. Mau kuliah online atau tidak semua tergantung pribadi masing-masing dalam memahami materi yang dikasih dosen:)	Netral
6.	Pak ini gimana anak sekolahan offline untuk beberapa zona, tapi kenapa mahasiswa tetap melaksanakan kuliah secara online / daring, justru mahasiswa lebih bisa beradaptasi dengan new normal dibandingkan dengan anak-anak yang masih sangat rentan, mohon dikaji lagi pak	Netral

Tabel 4.1 Data Latih (lanjutan)

No	Tweet	Kelas
7.	Saya berdoa kuliah tetap daring saja, kampus mau offline padahal tempat masih zona merah dan kerabat aku yang kerjanya dokter saja suka bilang lagi kerja keras karena pasien corona. Lebih nyaman online, tetap dirumah adalah jalanku	Positif
8.	Nilai positif saja yang diambil buang yang negatif. Positifnya (mungkin) ada beberapa mahasiswa yang tidak berani bertanya di kelas jadi lebih aktif bertanya di kuliah online (daring)	Positif
9.	Benar juga ya lama lama kuliah online jadi new normal sampai masa pandemi ini selesai juga bisa jadi online, kalau bisa daring kenapa harus kuliah offline ðÿ˜œ	Positif

Tabel 4.2 Data Uji

No	Tweet	Kelas
1.	Apa saya saja yang merasa kalau selama kuliah daring nyaman banget sampai saya tidak ingin masuk kuliah karena takut panik	?
2.	Aku merasa lebih leluasa dengan kuliah daring, tidak capek harus siap-siap berangkat. Hanya tinggal makan, beres didepan komputer sudah siap nyimak. Buat materi, selama online emang tidak pernah mengandalkan dosen atau teman. Jadi lebih banyak waktu buat searching sama buka text book.	?
3.	Jujur tidak ada senang-senangnya kuliah daring. Aku butuh praktik lapangan. Apalagi semester depan magang. Apa magang online juga? Bisa stres gara-gara banyak deadline	?
4.	Tatap langsung aja kadang tidak paham, apalagi kuliah daring, belum lagi jaringan lambat ditambah beberapa dosen yang jarang memberi kuliah online, atau cuma memberi tugas saja... Fix kampus ku belum siap menerapkan kuliah daring! pic.twitter.com/UHdReyLgh8	?
5.	Orang lain pada ribut sama keadaan kosan yang sudah ditinggal berbulan-bulan terus ribut gimana caranya balik ke kosan. Aku anteng-anteng saja jadi penghuni kos dari awal pemerintah nyuruh dirumah saja dan kuliah jadi daring	?

4.2.2 Manualisasi Pembuatan Daftar *Stopword*

Pada tahapan ini dilakukan manualisasi pembuatan daftar *stopword* dengan menggunakan *Term Based Random Sampling.*

Berikut adalah data yang digunakan untuk pembuatan *stopword* yang sudah melalui proses *case folding*, *cleaning*, tokenisasi yang dapat dilihat pada Tabel 4.3.

Tabel 4.3 Data Manualisasi Pembuatan *Stopword* yang sudah di *Preprocessing*

No	Tweet
1.	aku lama kuliah online benar benar tidak ajar sama sekali uji selalu tidak jujur tugas tinggal pindah dari internet dosen hanya beri tugas tidak pernah ada jelas materi tambah semester mau tetap daring mau jadi apa aku
2.	rasa mau henti kuliah saja kalau daring begini seperti bayar cuma cuma materi kasih cara online suruh baca sendiri tanpa ada yang jelas asa otodidak
3.	maaf aku kuliah daring makin malas kelas online saja tidur baik darimananya coba nilai sempurna bukan karena kita yang cerdas tapi karena dosen yang kasihan sama kita tapi cara paham kosong sekali otak ini terima kasih
4.	jujur aku oke oke saja dengan kuliah daring cuma ya itu kangen sama suasana kelas kalau corona sudah selesai padu offline online seperti asik
5.	ada yang ribut masalah kuliah online daring bagi ada yang salah dosen ada juga yang salah diri sendiri mau kuliah online atau tidak semua gantung pribadi masing masing dalam paham materi yang kasih dosen
6.	pak ini gimana anak sekolah offline untuk beberapa zona tapi kenapa mahasiswa tetap laksana kuliah cara online daring justru mahasiswa lebih bisa adaptasi dengan new normal banding dengan anak anak yang masih sangat rentan mohon kaji lagi pak
7.	saya doa kuliah tetap daring saja kampus mau offline padahal tempat masih zona merah dan kerabat aku yang kerja dokter saja suka bilang lagi kerja keras karena pasien corona lebih nyaman online tetap rumah adalah jalan
8.	nilai positif saja yang ambil buang yang negatif positif mungkin ada beberapa mahasiswa yang tidak berani tanya di kelas jadi lebih aktif tanya di kuliah online daring
9.	benar juga ya lama lama kuliah online jadi new normal sampai masa pandemi ini selesai juga bisa jadi online kalau bisa daring kenapa harus kuliah offline

Setelah melalui proses *preprocessing*, akan didapatkan sejumlah *term* yang dapat dilihat di Tabel 4.4.

Tabel 4.4 *Term* Manualisasi Pembuatan Daftar *Stopword*

Term
'aku', 'lama', 'kuliah', 'online', 'benar', 'tidak', 'ajar', 'sama', 'sekali', 'uji', 'selalu', 'jujur', 'tugas', 'tinggal', 'pindah', 'dari', 'internet', 'dosen', 'hanya', 'beri',

Tabel 4.5 *Term* Manualisasi Pembuatan Daftar *Stopword* (lanjutan)

Term
'pernah', 'ada', 'jelas', 'materi', 'tambah', 'semester', 'mau', 'tetap', 'daring', 'jadi', 'apa', 'rasa', 'henti', 'saja', 'kalau', 'begini', 'seperti', 'bayar', 'cuma', 'kasih', 'cara', 'suruh', 'baca', 'sendiri', 'tanpa', 'yang', 'asa', 'otodidak', 'maaf', 'makin', 'malas', 'kelas', 'tidur', 'baik', 'darimananya', 'coba', 'nilai', 'sempurna', 'bukan', 'karena', 'kita', 'cerdas', 'tapi', 'kasihan', 'paham', 'kosong', 'otak', 'ini', 'terima', 'oke', 'dengan', 'ya', 'itu', 'kangen', 'suasana', 'corona', 'sudah', 'selesai', 'padu', 'offline', 'asik', 'ribut', 'masalah', 'bagi', 'salah', 'juga', 'diri', 'atau', 'semua', 'gantung', 'pribadi', 'masing', 'dalam', 'pak', 'gimana', 'anak', 'sekolah', 'untuk', 'beberapa', 'zona', 'kenapa', 'mahasiswa', 'laksana', 'justru', 'lebih', 'bisa', 'adaptasi', 'new', 'normal', 'banding', 'masih', 'sangat', 'rentan', 'mohon', 'kaji', 'lagi', 'saya', 'doa', 'kampus', 'padahal', 'tempat', 'merah', 'dan', 'kerabat', 'kerja', 'dokter', 'suka', 'bilang', 'keras', 'pasien', 'nyaman', 'rumah', 'adalah', 'jalan', 'positif', 'ambil', 'buang', 'negatif', 'mungkin', 'berani', 'tanya', 'di', 'aktif', 'sampai', 'masa', 'pandemi', 'harus'

Setelah *term* didapatkan langkah selanjutnya adalah proses pembuatan *stopword* dengan *Term Based Random Sampling*. Dalam algoritme *Term Based Random Sampling* ini memiliki beberapa parameter yang harus ditentukan. X sebagai jumlah angka yang diambil dari urutan tertinggi dari tiap perulangan, Y adalah jumlah perulangan pemilihan kata acak, dan L adalah jumlah *stopword* yang ingin dibuat.

Dalam proses manualisasi ini akan digunakan Y = 50, X = 30, dan L akan diambil 20 persen dari keseluruhan. Adapun langkah-langkahnya akan dijelaskan sebagai berikut:

1. Pilih *term* acak dari keseluruhan *term*

$$w_{random} = "materi"$$

2. Ambil dokumen yang mengandung *term* tersebut dan dokumen tersebut akan menjadi dokumen sampel. Dokumen sampel yang diambil akan ditampilkan pada Tabel 4.5.

Tabel 4.5 Dokumen Sampel Manualisasi Pembuatan Daftar *Stopword*

No	Tweet
1.	aku lama kuliah online benar benar tidak ajar sama sekali uji selalu tidak jujur tugas tinggal pindah dari internet dosen hanya beri tugas tidak pernah ada jelas materi tambah semester mau tetap daring mau jadi apa aku
2.	rasa mau henti kuliah saja kalau daring begini seperti bayar cuma cuma materi kasih cara online suruh baca sendiri tanpa ada yang jelas asa otodidak

Tabel 4.5 Dokumen Sampel Manualisasi Pembuatan Daftar *Stopword*

No	Tweet
3.	ada yang ribut masalah kuliah online daring bagi ada yang salah dosen ada juga yang salah diri sendiri mau kuliah online atau tidak semua gantung pribadi masing masing dalam paham materi yang kasih dosen

1. Cari *term* dari dokumen sampel atau dokumen yang diambil. *Term* yang diambil dari dokumen sampel akan ditampilkan pada Tabel 4.6.

Tabel 4.6 *Term* Dokumen Sampel Manualisasi Pembuatan Daftar *Stopword*

Term dari dokumen sampel
'aku', 'lama', 'kuliah', 'online', 'benar', 'tidak', 'ajar', 'sama', 'sekali', 'uji', 'selalu', 'jujur', 'tugas', 'tinggal', 'pindah', 'dari', 'internet', 'dosen', 'hanya', 'beri', 'pernah', 'ada', 'jelas', 'materi', 'tambah', 'semester', 'mau', 'tetap', 'daring', 'jadi', 'apa', 'rasa', 'henti', 'saja', 'kalau', 'begini', 'seperti', 'bayar', 'cuma', 'kasih', 'cara', 'suruh', 'baca', 'sendiri', 'tanpa', 'yang', 'asa', 'otodidak', 'ribut', 'masalah', 'bagi', 'salah', 'juga', 'diri', 'atau', 'semua', 'gantung', 'pribadi', 'masing', 'dalam', 'paham'

2. Hitung bobot tiap *term* menggunakan *Kullback-Leibler* menggunakan Persamaan 2.1, 2.2 dan 2.3
   Berikut adalah beberapa contoh kata dalam perhitungan *Kullback-Leibler*.
   1. Kata “aku”

$$P_x(\text{aku}) = \frac{tf_x}{l_x} = \frac{2.0}{96.0} = 0.02083$$

$$P_c(\text{aku}) = \frac{F}{token_c} = \frac{5.0}{147.0} = 0.03401$$

$$w(aku) = P_x(aku)\ .\ log_2\left(\frac{P_x(aku)}{P_c(aku)}\right)$$

$$w(aku) = 0.02083\ .\ log_2\left(\frac{0.02083}{0.03401}\right)$$

$$w(aku) = -0.01473$$

   Setelah perhitungan diatas, kata “aku” mendapatkan nilai bobot sebesar -0.01473.

   2. Kata “lama”

$$P_x(\text{lama}) = \frac{tf_x}{l_x} = \frac{1.0}{96.0} = 0.01041$$

$$P_c(\text{lama}) = \frac{F}{token_c} = \frac{3.0}{147.0} = 0.02040$$

$$w(\text{lama}) = P_x(aku)\ .\ log_2\left(\frac{P_x(\text{lama})}{P_c(\text{lama})}\right)$$

$$w(\text{lama}) = 0.01041\ .log_2\left(\frac{0.01041}{0.02040}\right)$$

$$w(\text{lama}) = -0.01010$$

Setelah perhitungan diatas, kata "lama" mendapatkan nilai bobot sebesar -0.01010.

3. Kata "kuliah"

$$P_x(\text{kuliah}) = \frac{tf_x}{l_x} = \frac{4.0}{96.0} = 0.04166$$

$$P_c(\text{kuliah}) = \frac{F}{token_c} = \frac{11.0}{147.0} = 0.07482$$

$$w(\text{kuliah}) = P_x(\text{kuliah})\ .\ log_2\left(\frac{P_x(\text{kuliah})}{P_c(\text{kuliah})}\right)$$

$$w(\text{kuliah}) = 0.04166\ .log_2\left(\frac{0.04166}{0.07482}\right)$$

$$w(\text{kuliah}) = -0.03519$$

Setelah perhitungan diatas, kata "kuliah" mendapatkan nilai bobot sebesar -0.03519.

Dan dalam proses iterasi pertama ini, didapatkan nilai sebagai berikut yang akan ditampilkan dalam Tabel 4.7.

Tabel 4.7 Hasil *Kullback-Leibler* Manualisasi

Term	**Bobot *Kullback-Leibler***
aku	-0.014734
lama	-0.010107
kuliah	-0.035197
online	-0.035197
benar	0.000620
tidak	0.012199
ajar	0.006403
sama	-0.010107
sekali	-0.004013
uji	0.006403
selalu	0.006403
jujur	-0.004013
tugas	0.012806

Tabel 4.7 Hasil *Kullback-Leibler* Manualisasi (lanjutan)

Term	**Bobot *Kullback-Leibler***
tinggal	0.006403
pindah	0.006403
dari	0.006403
internet	0.006403
dosen	0.006240
hanya	0.006403
beri	0.006403
pernah	0.006403
ada	0.018316
jelas	0.012806
materi	0.019210
tambah	0.006403
semester	0.006403
mau	0.012199
tetap	-0.014430
daring	-0.030320
jadi	-0.014430
apa	0.006403
rasa	0.006403
henti	0.006403
saja	-0.020523
kalau	-0.010107
begini	0.006403
seperti	-0.004013
bayar	0.006403
cuma	0.000620
kasih	0.000620
cara	-0.010107
suruh	0.006403
baca	0.006403
sendiri	0.012806
tanpa	0.006403
yang	-0.033767
asa	0.006403
otodidak	0.006403
ribut	0.006403
masalah	0.006403
bagi	0.006403
salah	0.012806
juga	-0.010107
diri	0.006403

Tabel 4.7 Hasil *Kullback-Leibler* Manualisasi (lanjutan)

Term	**Bobot *Kullback-Leibler***
atau	0.006403
semua	0.006403
gantung	0.006403
pribadi	0.006403
masing	0.012806
dalam	0.006403
paham	-0.004013

3. Normalisasi bobot menggunakan *MinMax* agar didalam *range* 0 hingga 1.

Dalam perhitungan normalisasi *MinMax*, diperlukan untuk mencari nilai maksimum dan nilai minimum dari bobot yang sudah kita hitung sebelumnya yaitu:

$$\min_{\text{weight}} = -0.03519$$

$$\max_{\text{weight}} = 0.01920$$

Setelah minimum dan maksimum didapatkan, proses normalisasi dapat dilakukan dengan cara sebagai berikut.

$$X_{normalized} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Berikut adalah contoh perhitungan normalisasi MinMax menggunakan data sebelumnya.

1. Bobot “aku” = -0.01473

$$normalized_{weight}[aku] = \frac{-0.01473 - (-0.03519)}{0.01920 - (-0.03519)}$$

$$normalized_{weight}[aku] = 0.37611$$

Sehingga bobot “aku” yang sebelumnya adalah -0.01473 telah dinormalisasi menjadi 0.37611.

2. Bobot “lama” = -0.01010

$$normalized_{weight}[\text{lama}] = \frac{-0.01010 - (-0.03519)}{0.01920 - (-0.03519)}$$

$$normalized_{weight}[\text{lama}] = 0.46115$$

Sehingga bobot “lama” yang sebelumnya adalah -0.01010 telah dinormalisasi menjadi 0.46115.

3. Bobot “kuliah” = -0.03519

$$normalized_{weight}[\text{kuliah}] = \frac{-0.03519 - (-0.03519)}{0.01920 - (-0.03519)}$$

$$normalized_{weight}[\text{kuliah}] = 0.0$$

Sehingga bobot "kuliah" yang sebelumnya adalah -0.03519 telah dinormalisasi menjadi 0.0.

Sehingga setelah melalui proses normalisasi, didapatkan nilai bobot sebagai berikut yang akan ditampilkan dalam Tabel 4.8.

Tabel 4.8 Hasil Normalisasi *Kullback-Leibler* Manualisasi

Term	**Bobot *Kullback-Leibler***
aku	0.376114
lama	0.461158
kuliah	0.000000
online	0.000000
benar	0.658313
tidak	0.871147
ajar	0.764615
sama	0.461158
sekali	0.573155
uji	0.764615
selalu	0.764615
jujur	0.573155
tugas	0.882308
tinggal	0.764615
pindah	0.764615
dari	0.764615
internet	0.764615
dosen	0.761610
hanya	0.764615
beri	0.764615
pernah	0.764615
ada	0.983582
jelas	0.882308
materi	1.000000
tambah	0.764615
semester	0.764615
mau	0.871147
tetap	0.381695
daring	0.089628
jadi	0.381695
apa	0.764615
rasa	0.764615
henti	0.764615
saja	0.269697
kalau	0.461158

Tabel 4.8 Hasil Normalisasi *Kullback-Leibler* Manualisasi (lanjutan)

Term	**Bobot *Kullback-Leibler***
begini	0.764615
seperti	0.573155
bayar	0.764615
cuma	0.658313
kasih	0.658313
cara	0.461158
suruh	0.764615
baca	0.764615
sendiri	0.882308
tanpa	0.764615
yang	0.026281
asa	0.764615
otodidak	0.764615
ribut	0.764615
masalah	0.764615
bagi	0.764615
salah	0.882308
juga	0.461158
diri	0.764615
atau	0.764615
semua	0.764615
gantung	0.764615
pribadi	0.764615
masing	0.882308
dalam	0.764615
paham	0.573155

4. Ambil sejumlah X *term* (dimana X adalah parameter) yang diurutkan dari bobot terendah. Dalam manualisasi nilai X yang digunakan adalah 30. Berikut adalah 30 data terendah yang ditampilkan dalam Tabel 4.9.

Tabel 4.9 Hasil 30 Bobot Terendah

Term	**Bobot *Kullback-Leibler***
kuliah	0.000000
online	0.000000
yang	0.026281
daring	0.089628
saja	0.269697
aku	0.376114
tetap	0.381695
jadi	0.381695
lama	0.461158

Tabel 4.9 Hasil 30 Bobot Terendah (lanjutan)

Term	**Bobot *Kullback-Leibler***
sama	0.461158
kalau	0.461158
cara	0.461158
juga	0.461158
sekali	0.573155
jujur	0.573155
seperti	0.573155
paham	0.573155
benar	0.658313
cuma	0.658313
kasih	0.658313
dosen	0.761610
ajar	0.764615
uji	0.764615
selalu	0.764615
tinggal	0.764615
pindah	0.764615
dari	0.764615
internet	0.764615
hanya	0.764615
beri	0.764615

Setelah bobot normalisasi didapatkan, simpan bobot tersebut dan kumpulkan dalam suatu *dictionary* dengan kata kunci *term* tersebut untuk dicari rata-ratanya ditahapan berikutnya.

5. Lakukan proses 1 hingga 7 sebanyak Y kali (dimana Y adalah parameter). Dimana dalam proses manualisasi ini Y adalah 50. Sehingga nanti tiap masing-masing *term* memiliki sejumlah bobot berbeda-beda yang didapatkan tiap perulangan sebanyak Y kali. Berikut adalah contoh sampel *term* yang memiliki beberapa sampel bobot yang didapatkan di setiap perulangannya disaat perulangan sudah selesai yang akan ditampilkan pada Tabel 4.10.

Tabel 4.10 Sampel Hasil Keseluruhan Bobot Tiap Iterasi

Term	**Kumpulan Bobot**
aku	0.37611, 0.22853, 0.51485, 0.20722
lama	0.46115, 0.61264
kuliah	0.00000, 0.02525, 0.11938

6. Setelah itu hitung rata-rata keseluruhan bobot yang didapatkan setiap *term*
    1. Kata “aku”

$$w_{mean}(aku) = \frac{0.37611 + 0.22853 + 0.51485 + 0.20722}{4}$$
$$w_{mean}(aku) = 0.33167$$

2. Kata “lama”

$$w_{mean}(lama) = \frac{0.46115 + 0.61264}{2}$$
$$w_{mean}(lama) = 0.53689$$

3. Kata “kuliah”

$$w_{mean}(kuliah) = \frac{0.00000 + 0.02525 + 0.11938}{3}$$
$$w_{mean}(kuliah) = 0.04821$$

Setelah tiap *term* dicari rata-ratanya, maka dihasilkan hasil akhir bobot setiap *term* yang akan ditampilkan pada Tabel 4.11.

Tabel 4.11 Hasil Rata-rata Keseluruhan Bobot

Term	**Bobot Rata-rata**
aku	0.327350
lama	0.493203
kuliah	0.024470
online	0.024470
benar	0.656800
tidak	0.391615
ajar	0.688190
sama	0.424298
sekali	0.527057
uji	0.688190
selalu	0.688190
jujur	0.520274
tinggal	0.688190
pindah	0.688190
dari	0.688190
internet	0.688190
dosen	0.516360
hanya	0.688190
beri	0.680499
pernah	0.634799
ada	0.453460
jelas	0.541732
materi	0.409377
tambah	0.634799
semester	0.634799
mau	0.299504

Tabel 4.11 Hasil Rata-rata Keseluruhan Bobot (lanjutan)

Term	**Bobot Rata-rata**
tetap	0.371090
daring	0.070763
jadi	0.399814
apa	0.634799
rasa	0.728583
henti	0.728583
saja	0.286343
kalau	0.480921
begini	0.728583
seperti	0.542063
bayar	0.728583
cuma	0.659504
kasih	0.449530
cara	0.441262
suruh	0.728583
baca	0.728583
sendiri	0.476301
tanpa	0.728583
yang	0.171075
asa	0.721197
otodidak	0.721197
ribut	0.600122
masalah	0.600122
bagi	0.600122
salah	1.000.000
juga	0.405606
diri	0.600122
atau	0.600122
semua	0.600122
gantung	0.600122
pribadi	0.600122
masing	1.000.000
dalam	0.600122
paham	0.505293
maaf	0.630228
makin	0.630228
malas	0.630228
kelas	0.398823
tidur	0.630228
baik	0.630228

Tabel 4.11 Hasil Rata-rata Keseluruhan Bobot (lanjutan)

Term	Bobot Rata-rata
darimananya	0.630228
coba	0.630228
nilai	0.477369
sempurna	0.630228
bukan	0.630228
karena	0.539828
cerdas	0.630228
tapi	0.477876
kasihan	0.630228
kosong	0.630228
otak	0.630228
ini	0.405482
terima	0.630228
oke	1.000.000
dengan	0.522447
ya	0.534539
itu	0.629438
kangen	0.629438
suasana	0.629438
corona	0.482842
sudah	0.629438
selesai	0.534539
padu	0.629438
offline	0.379735
asik	0.629438
saya	0.652592
doa	0.638495
kampus	0.638495
padahal	0.638495
tempat	0.638495
masih	0.493741
zona	0.448584
merah	0.638495
dan	0.638495
kerabat	0.638495
dokter	0.638495
suka	0.638495
bilang	0.638495
lagi	0.448584
keras	0.638495
pasien	0.638495

Tabel 4.11 Hasil Rata-rata Keseluruhan Bobot (lanjutan)

Term	Bobot Rata-rata
lebih	0.427917
nyaman	0.638495
rumah	0.638495
adalah	0.638495
positif	1.000.000
ambil	0.585926
buang	0.585926
negatif	0.585926
mungkin	0.585926
beberapa	0.410118
mahasiswa	0.536879
berani	0.585926
tanya	1.000.000
di	1.000.000
aktif	0.585926
new	0.401453
normal	0.401453
sampai	0.620283
masa	0.641505
pandemi	0.641505
bisa	0.318457
kenapa	0.401453
harus	0.641505
gimana	0.553001
anak	0.818182
sekolah	0.553001
untuk	0.553001
laksana	0.553001
justru	0.553001
adaptasi	0.553001
banding	0.553001
sangat	0.553001
rentan	0.553001
mohon	0.502798
kaji	0.502798

7. Urutkan bobot secara meningkat dari bobot terendah hingga bobot tertinggi. Hasil rata-rata bobot yang sudah diurutkan dapat dilihat di Tabel 4.12.

Tabel 4.12 Hasil Rata-rata Bobot yang sudah diurutkan

Term	**Bobot Rata-rata**
kuliah	0.024470
online	0.024470
daring	0.070763
yang	0.171075
saja	0.286343
mau	0.299504
bisa	0.318457
aku	0.327350
tetap	0.371090
offline	0.379735
tidak	0.391615
kelas	0.398823
jadi	0.399814
new	0.401453
normal	0.401453
kenapa	0.401453
ini	0.405482
juga	0.405606
materi	0.409377
beberapa	0.410118
sama	0.424298
lebih	0.427917
cara	0.441262
zona	0.448584
lagi	0.448584
kasih	0.449530
ada	0.453460
sendiri	0.476301
nilai	0.477369
tapi	0.477876
kalau	0.480921
corona	0.482842
lama	0.493203
masih	0.493741
mohon	0.502798
kaji	0.502798
paham	0.505293
dosen	0.516360
jujur	0.520274
dengan	0.522447
sekali	0.527057

Tabel 4.12 Hasil Rata-rata Bobot yang sudah diurutkan (lanjutan)

Term	Bobot Rata-rata
ya	0.534539
selesai	0.534539
mahasiswa	0.536879
karena	0.539828
jelas	0.541732
seperti	0.542063
gimana	0.553001
sekolah	0.553001
untuk	0.553001
laksana	0.553001
justru	0.553001
adaptasi	0.553001
banding	0.553001
sangat	0.553001
rentan	0.553001
ambil	0.585926
buang	0.585926
negatif	0.585926
mungkin	0.585926
berani	0.585926
aktif	0.585926
ribut	0.600122
masalah	0.600122
bagi	0.600122
diri	0.600122
atau	0.600122
semua	0.600122
gantung	0.600122
pribadi	0.600122
dalam	0.600122
sampai	0.620283
itu	0.629438
kangen	0.629438
suasana	0.629438
sudah	0.629438
padu	0.629438
asik	0.629438
maaf	0.630228
makin	0.630228
malas	0.630228
tidur	0.630228

Tabel 4.12 Hasil Rata-rata Bobot yang sudah diurutkan (lanjutan)

Term	Bobot Rata-rata
baik	0.630228
darimananya	0.630228
coba	0.630228
sempurna	0.630228
bukan	0.630228
cerdas	0.630228
kasihan	0.630228
kosong	0.630228
otak	0.630228
terima	0.630228
pernah	0.634799
tambah	0.634799
semester	0.634799
apa	0.634799
doa	0.638495
kampus	0.638495
padahal	0.638495
tempat	0.638495
merah	0.638495
dan	0.638495
kerabat	0.638495
dokter	0.638495
suka	0.638495
bilang	0.638495
keras	0.638495
pasien	0.638495
nyaman	0.638495
rumah	0.638495
adalah	0.638495
masa	0.641505
pandemi	0.641505
harus	0.641505
saya	0.652592
benar	0.656800
cuma	0.659504
beri	0.680499
ajar	0.688190
uji	0.688190
selalu	0.688190
tinggal	0.688190
pindah	0.688190

Tabel 4.12 Hasil Rata-rata Bobot yang sudah diurutkan (lanjutan)

Term	Bobot Rata-rata
dari	0.688190
internet	0.688190
hanya	0.688190
asa	0.721197
otodidak	0.721197
rasa	0.728583
henti	0.728583
begini	0.728583
bayar	0.728583
suruh	0.728583
baca	0.728583
tanpa	0.728583
anak	0.818182
salah	1.000000
masing	1.000000
oke	1.000000
positif	1.000000
tanya	1.000000
di	1.000000

1. Ambil sejumlah L *term* (dimana L adalah parameter yang menentukan jumlah *stopword* yang ingin digunakan). Dalam manualisasi ini kita akan mencoba untuk menggunakan varian parameter L dengan nilai 20 persen.

   Berikut adalah daftar *stopword* jika parameter L yang digunakan adalah 20 persen yang akan ditampilkan pada Tabel 4.13.

Tabel 4.13 Daftar *Stopword* 20 persen

Term	Bobot Rata-rata
kuliah	0.024470
online	0.024470
daring	0.070763
yang	0.171075
saja	0.286343
mau	0.299504
bisa	0.318457
aku	0.327350
tetap	0.371090
offline	0.379735
tidak	0.391615
kelas	0.398823

Tabel 4.13 Daftar *Stopword* 20 persen (lanjutan)

Term	**Bobot Rata-rata**
jadi	0.399814
new	0.401453
normal	0.401453
kenapa	0.401453
ini	0.405482
juga	0.405606
materi	0.409377
beberapa	0.410118
sama	0.424298
lebih	0.427917
cara	0.441262
zona	0.448584
lagi	0.448584
kasih	0.449530
ada	0.453460
sendiri	0.476301

4.2.3 *Preprocessing*

Tahapan *preprocessing* ini adalah tahapan untuk menyiapkan data yang akan digunakan dengan mengubah data yang tidak terstruktur menjadi suatu data yang terstruktur agar dapat diolah oleh sistem. Tahapan *preprocessing* ini terdiri dari tahapan, *case folding*, *cleaning*, *stemming*, *tokenizing*.

4.2.3.1 *Case folding*

Pada tahapan ini terjadi perubahan huruf kapital didalam dokumen menjadi huruf kecil. Tahapan ini dapat dilihat di Tabel 4.14 dan 4.15

Tabel 4.14 Manualisasi *Case folding* Data Latih

No	Tweet	Kelas
1.	aku selama kuliah online benar-benar tidak belajar sama sekali. ujian selalu tidak jujur, tugas tinggal memindahkan dari internet, dosen hanya memberi tugas, tidak pernah ada penjelasan materi. ditambah semester 5 mau tetap daring, mau jadi apa aku	Negatif
2.	rasanya mau berhenti kuliah saja kalau daring begini, seperti bayar cuma cuma, materi dikasih secara online, disuruh baca sendiri tanpa ada yang menjelaskan, berasa otodidak :"	Negatif

Tabel 4.14 Manualisasi *Case folding* Data Latih (lanjutan)

No	Tweet	Kelas
3.	maaf, aku kuliah daring semakin malas. kelas online saja ketiduran. baik darimananya coba? nilai sempurna bukan karena kita yang cerdas tapi karena dosennya yang kasihan sama kitanya. tapi secara pemahaman, kosong sekali otak ini. terima kasih	Negatif
4.	sejujurnya aku oke-oke saja dengan kuliah daring. cuma ya itu, kangen sama suasana kelas. kalau corona sudah selesai, perpaduan offline-online sepertinya asik....	Netral
5.	ada yang mempeributkan masalah kuliah online/daring, sebagian ada yang menyalahkan dosen ada juga yang menyalahkan diri sendiri. mau kuliah online atau tidak semua tergantung pribadi masing-masing dalam memahami materi yang dikasih dosen:)	Netral
6.	pak ini gimana anak sekolahan offline untuk beberapa zona, tapi kenapa mahasiswa tetap melaksanakan kuliah secara online / daring, justru mahasiswa lebih bisa beradaptasi dengan new normal dibandingkan dengan anak-anak yang masih sangat rentan, mohon dikaji lagi pak	Netral
7.	saya berdoa kuliah tetap daring saja, kampus mau offline padahal tempat masih zona merah dan kerabat aku yang kerjanya dokter saja suka bilang lagi kerja keras karena pasien corona. lebih nyaman online, tetap dirumah adalah jalanku	Positif
8.	nilai positif saja yang diambil buang yang negatif. positifnya (mungkin) ada beberapa mahasiswa yang tidak berani bertanya di kelas jadi lebih aktif bertanya di kuliah online (daring)	Positif
9.	benar juga ya lama lama kuliah online jadi new normal sampai masa pandemi ini selesai juga bisa jadi online, kalau bisa daring kenapa harus kuliah offline ðÿ˜œ	Positif

Tabel 4.15 Manualisasi *Case folding* Data Uji

No	Tweet	Kelas
1.	apa saya saja yang merasa kalau selama kuliah daring nyaman banget sampai saya tidak ingin masuk kuliah karena takut panik	?
2.	aku merasa lebih leluasa dengan kuliah daring, tidak capek harus siap-siap berangkat. hanya tinggal makan, beres didepan komputer sudah siap nyimak. buat materi, selama online emang tidak pernah mengandalkan dosen atau temen. jadi lebih banyak waktu buat searching sama buka textbook	?

Tabel 4.15 Manualisasi *Case folding* Data Uji (lanjutan)

No	Tweet	Kelas
3.	jujur tidak ada senang-senangnya kuliah daring. aku butuh praktik lapangan. apalagi semester depan magang. apa magang online juga? bisa stres gara-gara banyak deadline	?
4.	tatap langsung aja kadang tidak paham, apalagi kuliah daring, belum lagi jaringan lambat ditambah beberapa dosen yang jarang memberi kuliah online, atau cuma memberi tugas saja... fix kampus ku belum siap menerapkan kuliah daring! pic.twitter.com/uhdreylgh8	?
5.	orang lain pada ribut sama keadaan kosan yang sudah ditinggal berbulan-bulan terus ribut gimana caranya balik ke kosan. aku anteng-anteng saja jadi penghuni kos dari awal pemerintah nyuruh dirumah saja dan kuliah jadi daring	?

4.2.3.2 *Cleaning*

Pada tahapan ini terjadi penghapusan simbol dan non karakter dalam dokumen. Tahapan ini dapat dilihat di Tabel 4.16 dan 4.17

Tabel 4.16 Manualisasi *Cleaning* Data Latih

No	Tweet	Kelas
1.	aku selama kuliah online benar benar tidak belajar sama sekali ujian selalu tidak jujur tugas tinggal memindahkan dari internet dosen hanya memberi tugas tidak pernah ada penjelasan materi ditambah semester mau tetap daring mau jadi apa aku	Negatif
2.	rasanya mau berhenti kuliah saja kalau daring begini seperti bayar cuma cuma materi dikasih secara online disuruh baca sendiri tanpa ada yang menjelaskan berasa otodidak	Negatif
3.	maaf aku kuliah daring semakin malas kelas online saja ketiduran baik darimananya coba nilai sempurna bukan karena kita yang cerdas tapi karena dosennya yang kasihan sama kitanya tapi secara pemahaman kosong sekali otak ini terima kasih	Negatif
4.	sejujurnya aku oke oke saja dengan kuliah daring cuma ya itu kangen sama suasana kelas kalau corona sudah selesai perpaduan offline online sepertinya asik	Netral
5.	ada yang mempeributkan masalah kuliah online daring sebagian ada yang menyalahkan dosen ada juga yang menyalahkan diri sendiri mau kuliah online atau tidak semua tergantung pribadi masing masing dalam memahami materi yang dikasih dosen	Netral

Tabel 4.16 Manualisasi *Cleaning* Data Latih (lanjutan)

No	Tweet	Kelas
6.	pak ini gimana anak sekolahan offline untuk beberapa zona tapi kenapa mahasiswa tetap melaksanakan kuliah secara online daring justru mahasiswa lebih bisa beradaptasi dengan new normal dibandingkan dengan anak anak yang masih sangat rentan mohon dikaji lagi pak	Netral
7.	saya berdoa kuliah tetap daring saja kampus mau offline padahal tempat masih zona merah dan kerabat aku yang kerjanya dokter saja suka bilang lagi kerja keras karena pasien corona lebih nyaman online tetap dirumah adalah jalanku	Positif
8.	nilai positif saja yang diambil buang yang negatif positifnya mungkin ada beberapa mahasiswa yang tidak berani bertanya di kelas jadi lebih aktif bertanya di kuliah online daring	Positif
9.	benar juga ya lama lama kuliah online jadi new normal sampai masa pandemi ini selesai juga bisa jadi online kalau bisa daring kenapa harus kuliah offline	Positif

Tabel 4.17 Manualisasi *Cleaning* Data Uji

No	Tweet	Kelas
1.	apa saya saja yang merasa kalau selama kuliah daring nyaman banget sampai saya tidak ingin masuk kuliah karena takut panik	?
2.	aku merasa lebih leluasa dengan kuliah daring tidak capek harus siap siap berangkat hanya tinggal makan beres didepan komputer sudah siap nyimak buat materi selama online emang tidak pernah mengandalkan dosen atau temen jadi lebih banyak waktu buat searching sama buka textbook	?
3.	jujur tidak ada senang senangnya kuliah daring aku butuh praktik lapangan apalagi semester depan magang apa magang online juga bisa stres gara gara banyak deadline	?
4.	tatap langsung aja kadang tidak paham apalagi kuliah daring belum lagi jaringan lambat ditambah beberapa dosen yang jarang memberi kuliah online atau cuma memberi tugas saja fix kampus ku belum siap menerapkan kuliah daring pic twitter com uhdreylgh	?
5.	orang lain pada ribut sama keadaan kosan yang sudah ditinggal berbulan bulan terus ribut gimana caranya balik ke kosan aku anteng anteng saja jadi penghuni kos dari awal pemerintah nyuruh dirumah saja dan kuliah jadi daring	?

4.2.3.3 *Stemming*

Pada tahapan ini terjadi perubahan kata menjadi kata dasar. Tahapan ini dibantu oleh *library* Sastrawi untuk proses *stemming*. Tahapan ini dapat dilihat di Tabel 4.18 dan 4.19

Tabel 4.18 Manualisasi *Stemming* Data Latih

No	Tweet	Kelas
1.	aku lama kuliah online benar benar tidak ajar sama sekali uji selalu tidak jujur tugas tinggal pindah dari internet dosen hanya beri tugas tidak pernah ada jelas materi tambah semester mau tetap daring mau jadi apa aku	Negatif
2.	rasa mau henti kuliah saja kalau daring begini seperti bayar cuma cuma materi kasih cara online suruh baca sendiri tanpa ada yang jelas asa otodidak	Negatif
3.	maaf aku kuliah daring makin malas kelas online saja tidur baik darimananya coba nilai sempurna bukan karena kita yang cerdas tapi karena dosen yang kasihan sama kita tapi cara paham kosong sekali otak ini terima kasih	Negatif
4.	jujur aku oke oke saja dengan kuliah daring cuma ya itu kangen sama suasana kelas kalau corona sudah selesai padu offline online seperti asik	Netral
5.	ada yang ribut masalah kuliah online daring bagi ada yang salah dosen ada juga yang salah diri sendiri mau kuliah online atau tidak semua gantung pribadi masing masing dalam paham materi yang kasih dosen	Netral
6.	pak ini gimana anak sekolahan offline untuk beberapa zona tapi kenapa mahasiswa tetap melaksanakan kuliah secara online daring justru mahasiswa lebih bisa beradaptasi dengan new normal dibandingkan dengan anak anak yang masih sangat rentan mohon dikaji lagi pak	Netral
7.	saya doa kuliah tetap daring saja kampus mau offline padahal tempat masih zona merah dan kerabat aku yang kerja dokter saja suka bilang lagi kerja keras karena pasien corona lebih nyaman online tetap rumah adalah jalan	Positif
8.	nilai positif saja yang ambil buang yang negatif positif mungkin ada beberapa mahasiswa yang tidak berani tanya di kelas jadi lebih aktif tanya di kuliah online daring	Positif
9.	benar juga ya lama lama kuliah online jadi new normal sampai masa pandemi ini selesai juga bisa jadi online kalau bisa daring kenapa harus kuliah offline	Positif

Tabel 4.19 Manualisasi *Stemming* Data Uji

No	Tweet	Kelas
1.	apa saya saja yang rasa kalau lama kuliah daring nyaman banget sampai saya tidak ingin masuk kuliah karena takut panik	?
2.	aku rasa lebih leluasa dengan kuliah daring tidak capek harus siap siap berangkat hanya tinggal makan beres depan komputer sudah siap nyimak buat materi lama online emang tidak pernah andal dosen atau temen jadi lebih banyak waktu buat searching sama buka textbook	?
3.	jujur tidak ada senang senang kuliah daring aku butuh praktik lapang apalagi semester depan magang apa magang online juga bisa stres gara gara banyak deadline	?
4.	tatap langsung aja kadang tidak paham apalagi kuliah daring belum lagi jaring lambat tambah beberapa dosen yang jarang beri kuliah online atau cuma beri tugas saja fix kampus ku belum siap terap kuliah daring pic twitter com uhdreylgh	?
5.	orang lain pada ribut sama ada kosan yang sudah tinggal bulan bulan terus ribut gimana cara balik ke kosan aku anteng anteng saja jadi huni kos dari awal perintah nyuruh rumah saja dan kuliah jadi daring	?

4.2.3.4 Tokenisasi

Pada tahapan ini terjadi perubahan kalimat menjadi kesatuan kata yang terpisah. Tahapan ini dapat dilihat di Tabel 4.20 dan 4.21

Tabel 4.20 Manualisasi Tokenisasi Data Latih

No	Tweet	Kelas
1.	['aku', 'lama', 'kuliah', 'online', 'benar', 'benar', 'tidak', 'ajar', 'sama', 'sekali', 'uji', 'selalu', 'tidak', 'jujur', 'tugas', 'tinggal', 'pindah', 'dari', 'internet', 'dosen', 'hanya', 'beri', 'tugas', 'tidak', 'pernah', 'ada', 'jelas', 'materi', 'tambah', 'semester', 'mau', 'tetap', 'daring', 'mau', 'jadi', 'apa', 'aku']	Negatif
2.	['rasa', 'mau', 'henti', 'kuliah', 'saja', 'kalau', 'daring', 'begini', 'seperti', 'bayar', 'cuma', 'cuma', 'materi', 'kasih', 'cara', 'online', 'suruh', 'baca', 'sendiri', 'tanpa', 'ada', 'yang', 'jelas', 'asa', 'otodidak']	Negatif
3.	['maaf', 'aku', 'kuliah', 'daring', 'makin', 'malas', 'kelas', 'online', 'saja', 'tidur', 'baik', 'darimananya', 'coba', 'nilai', 'sempurna', 'bukan', 'karena', 'kita', 'yang', 'cerdas', 'tapi', 'karena', 'dosen', 'yang', 'kasihan', 'sama', 'kita', 'tapi', 'cara', 'paham', 'kosong', 'sekali', 'otak', 'ini', 'terima', 'kasih']	Negatif

Tabel 4.20 Manualisasi Tokenisasi Data Latih (lanjutan)

No	Tweet	Kelas
4.	['jujur', 'aku', 'oke', 'oke', 'saja', 'dengan', 'kuliah', 'daring', 'cuma', 'ya', 'itu', 'kangen', 'sama', 'suasana', 'kelas', 'kalau', 'corona', 'sudah', 'selesai', 'padu', 'offline', 'online', 'seperti', 'asik']	Netral
5.	['ada', 'yang', 'ribut', 'masalah', 'kuliah', 'online', 'daring', 'bagi', 'ada', 'yang', 'salah', 'dosen', 'ada', 'juga', 'yang', 'salah', 'diri', 'sendiri', 'mau', 'kuliah', 'online', 'atau', 'tidak', 'semua', 'gantung', 'pribadi', 'masing', 'masing', 'dalam', 'paham', 'materi', 'yang', 'kasih', 'dosen']	Netral
6.	['pak', 'ini', 'gimana', 'anak', 'sekolah', 'offline', 'untuk', 'beberapa', 'zona', 'tapi', 'kenapa', 'mahasiswa', 'tetap', 'laksana', 'kuliah', 'cara', 'online', 'daring', 'justru', 'mahasiswa', 'lebih', 'bisa', 'adaptasi', 'dengan', 'new', 'normal', 'banding', 'dengan', 'anak', 'anak', 'yang', 'masih', 'sangat', 'rentan', 'mohon', 'kaji', 'lagi', 'pak']	Netral
7.	['saya', 'doa', 'kuliah', 'tetap', 'daring', 'saja', 'kampus', 'mau', 'offline', 'padahal', 'tempat', 'masih', 'zona', 'merah', 'dan', 'kerabat', 'aku', 'yang', 'kerja', 'dokter', 'saja', 'suka', 'bilang', 'lagi', 'kerja', 'keras', 'karena', 'pasien', 'corona', 'lebih', 'nyaman', 'online', 'tetap', 'rumah', 'adalah', 'jalan']	Positif
8.	['nilai', 'positif', 'saja', 'yang', 'ambil', 'buang', 'yang', 'negatif', 'positif', 'mungkin', 'ada', 'beberapa', 'mahasiswa', 'yang', 'tidak', 'berani', 'tanya', 'di', 'kelas', 'jadi', 'lebih', 'aktif', 'tanya', 'di', 'kuliah', 'online', 'daring']	Positif
9.	['benar', 'juga', 'ya', 'lama', 'lama', 'kuliah', 'online', 'jadi', 'new', 'normal', 'sampai', 'masa', 'pandemi', 'ini', 'selesai', 'juga', 'bisa', 'jadi', 'online', 'kalau', 'bisa', 'daring', 'kenapa', 'harus', 'kuliah', 'offline']	Positif

Tabel 4.21 Manualisasi Tokenisasi Data Uji

No	Tweet	Kelas
1.	['apa', 'saya', 'saja', 'yang', 'rasa', 'kalau', 'lama', 'kuliah', 'daring', 'nyaman', 'banget', 'sampai', 'saya', 'tidak', 'ingin', 'masuk', 'kuliah', 'karena', 'takut', 'panik']	?
2.	['aku', 'rasa', 'lebih', 'leluasa', 'dengan', 'kuliah', 'daring', 'tidak', 'capek', 'harus', 'siap', 'siap', 'berangkat', 'hanya', 'tinggal', 'makan', 'beres', 'depan', 'komputer', 'sudah', 'siap', 'nyimak', 'buat', 'materi', 'lama', 'online', 'emang', 'tidak', 'pernah', 'andal', 'dosen', 'atau', 'temen', 'jadi', 'lebih', 'banyak', 'waktu', 'buat', 'searching', 'sama', 'buka', 'textbook']	?

Tabel 4.21 Manualisasi Tokenisasi Data Uji (lanjutan)

No	Tweet	Kelas
3.	['jujur', 'tidak', 'ada', 'senang', 'senang', 'kuliah', 'daring', 'aku', 'butuh', 'praktik', 'lapang', 'apalagi', 'semester', 'depan', 'magang', 'apa', 'magang', 'online', 'juga', 'bisa', 'stres', 'gara', 'gara', 'banyak', 'deadline']	?
4.	['tatap', 'langsung', 'aja', 'kadang', 'tidak', 'paham', 'apalagi', 'kuliah', 'daring', 'belum', 'lagi', 'jaring', 'lambat', 'tambah', 'beberapa', 'dosen', 'yang', 'jarang', 'beri', 'kuliah', 'online', 'atau', 'cuma', 'beri', 'tugas', 'saja', 'fix', 'kampus', 'ku', 'belum', 'siap', 'terap', 'kuliah', 'daring', 'pic', 'twitter', 'com', 'uhdreylgh']	?
5.	['orang', 'lain', 'pada', 'ribut', 'sama', 'ada', 'kosan', 'yang', 'sudah', 'tinggal', 'bulan', 'bulan', 'terus', 'ribut', 'gimana', 'cara', 'balik', 'ke', 'kosan', 'aku', 'anteng', 'anteng', 'saja', 'jadi', 'huni', 'kos', 'dari', 'awal', 'perintah', 'nyuruh', 'rumah', 'saja', 'dan', 'kuliah', 'jadi', 'daring']	?

4.2.3.5 *Filtering*

Pada tahapan ini terjadi penghapusan kata yang terdapat dialam daftar *stopword*. *Stopword* yang digunakan pada tahapan ini adalah *stopword* yang dibuat dengan algoritme *Term Based Random Sampling* dengan Y senilai 50, X senilai 30 dan L senilai 20 persen. Berikut adalah hasil *filtering* yang ditampilkan pada Tabel 4.22 hingga Tabel 4.24.

Tabel 4.22 Manualisasi *Filtering* 20 Persen Data Latih

No	Tweet	Kelas
1.	['lama', 'benar', 'benar', 'ajar', 'sekali', 'uji', 'selalu', 'jujur', 'tugas', 'tinggal', 'pindah', 'dari', 'internet', 'dosen', 'hanya', 'beri', 'tugas', 'pernah', 'jelas', 'tambah', 'semester', 'apa']	Negatif
2.	['rasa', 'henti', 'kalau', 'begini', 'seperti', 'bayar', 'cuma', 'cuma', 'suruh', 'baca', 'tanpa', 'jelas', 'asa', 'otodidak']	Negatif
3.	['maaf', 'makin', 'malas', 'tidur', 'baik', 'darimananya', 'coba', 'nilai', 'sempurna', 'bukan', 'karena', 'kita', 'cerdas', 'tapi', 'karena', 'dosen', 'kasihan', 'kita', 'tapi', 'paham', 'kosong', 'sekali', 'otak', 'terima']	Negatif
4.	['jujur', 'oke', 'oke', 'dengan', 'cuma', 'ya', 'itu', 'kangen', 'suasana', 'kalau', 'corona', 'sudah', 'selesai', 'padu', 'seperti', 'asik']	Netral
5.	['ribut', 'masalah', 'bagi', 'salah', 'dosen', 'salah', 'diri', 'atau', 'semua', 'gantung', 'pribadi', 'masing', 'masing', 'dalam', 'paham', 'dosen']	Netral

Tabel 4.22 Manualisasi *Filtering* 20 Persen Data Latih (lanjutan)

No	Tweet	Kelas
6.	['pak', 'gimana', 'anak', 'sekolah', 'untuk', 'tapi', 'mahasiswa', 'laksana', 'justru', 'mahasiswa', 'adaptasi', 'dengan', 'banding', 'dengan', 'anak', 'anak', 'masih', 'sangat', 'rentan', 'mohon', 'kaji', 'pak']	Netral
7.	['saya', 'doa', 'kampus', 'padahal', 'tempat', 'masih', 'merah', 'dan', 'kerabat', 'kerja', 'dokter', 'suka', 'bilang', 'kerja', 'keras', 'karena', 'pasien', 'corona', 'nyaman', 'rumah', 'adalah', 'jalan']	Positif
8.	['nilai', 'positif', 'ambil', 'buang', 'negatif', 'positif', 'mungkin', 'mahasiswa', 'berani', 'tanya', 'di', 'aktif', 'tanya', 'di']	Positif
9.	['benar', 'ya', 'lama', 'lama', 'sampai', 'masa', 'pandemi', 'selesai', 'kalau', 'harus']	Positif

Tabel 4.23 Manualisasi *Filtering* 20 Persen Data Uji

No	Tweet	Kelas
1.	['apa', 'saya', 'rasa', 'kalau', 'lama', 'nyaman', 'banget', 'sampai', 'saya', 'ingin', 'masuk', 'karena', 'takut', 'panik']	?
2.	['rasa', 'leluasa', 'dengan', 'capek', 'harus', 'siap', 'siap', 'berangkat', 'hanya', 'tinggal', 'makan', 'beres', 'depan', 'komputer', 'sudah', 'siap', 'nyimak', 'buat', 'lama', 'emang', 'pernah', 'andal', 'dosen', 'atau', 'temen', 'banyak', 'waktu', 'buat', 'searching', 'buka', 'textbook']	?
3.	['jujur', 'senang', 'senang', 'butuh', 'praktik', 'lapang', 'apalagi', 'semester', 'depan', 'magang', 'apa', 'magang', 'stres', 'gara', 'gara', 'banyak', 'deadline']	?
4.	['tatap', 'langsung', 'aja', 'kadang', 'paham', 'apalagi', 'belum', 'jaring', 'lambat', 'tambah', 'dosen', 'jarang', 'beri', 'atau', 'cuma', 'beri', 'tugas', 'fix', 'kampus', 'ku', 'belum', 'siap', 'terap', 'pic', 'twitter', 'com', 'uhdreylgh']	?
5.	['orang', 'lain', 'pada', 'ribut', 'kosan', 'sudah', 'tinggal', 'bulan', 'bulan', 'terus', 'ribut', 'gimana', 'balik', 'ke', 'kosan', 'anteng', 'anteng', 'huni', 'kos', 'dari', 'awal', 'perintah', 'nyuruh', 'rumah', 'dan']	?

Sehingga didapatkan *term* data latih sebagai berikut.

Tabel 4.24 Manualisasi Daftar *Term*

Term
['lama', 'benar', 'ajar', 'sekali', 'uji', 'selalu', 'jujur', 'tugas', 'tinggal', 'pindah', 'dari', 'internet', 'dosen', 'hanya', 'beri', 'pernah', 'jelas', 'tambah', 'semester', 'apa',

Tabel 4.24 Manualisasi Daftar *Term* (lanjutan)

Term
'rasa', 'henti', 'kalau', 'begini', 'seperti', 'bayar', 'cuma', 'suruh', 'baca', 'tanpa', 'asa', 'otodidak', 'maaf', 'makin', 'malas', 'tidur', 'baik', 'darimananya', 'coba', 'nilai', 'sempurna', 'bukan', 'karena', 'kita', 'cerdas', 'tapi', 'kasihan', 'paham', 'kosong', 'otak', 'terima', 'oke', 'dengan', 'ya', 'itu', 'kangen', 'suasana', 'corona', 'sudah', 'selesai', 'padu', 'asik', 'ribut', 'masalah', 'bagi', 'salah', 'diri', 'atau', 'semua', 'gantung', 'pribadi', 'masing', 'dalam', 'pak', 'gimana', 'anak', 'sekolah', 'untuk', 'mahasiswa', 'laksana', 'justru', 'adaptasi', 'banding', 'masih', 'sangat', 'rentan', 'mohon', 'kaji', 'saya', 'doa', 'kampus', 'padahal', 'tempat', 'merah', 'dan', 'kerabat', 'kerja', 'dokter', 'suka', 'bilang', 'keras', 'pasien', 'nyaman', 'rumah', 'adalah', 'jalan', 'positif', 'ambil', 'buang', 'negatif', 'mungkin', 'berani', 'tanya', 'di', 'aktif', 'sampai', 'masa', 'pandemi', 'harus']

4.2.4 *Term Weighting*

Tahapan *term weighting* ini adalah tahapan untuk memberi bobot setiap *term* sesuai dengan ciri dari masing-masing *term* tersebut. Adapun tahap-tahap *term weighting* yang digunakan adalah *Raw term Frequency, log term Frequency, inverse document Frequency,* dan *term Frequency-inverse document (TF-IDF).*

4.2.4.1 *Raw Term Frequency Weighting*

Pada *Raw term Frequency weighting,* setiap *term* diberi bobot berdasarkan frekuensi kemunculan tersebut dalam suatu dokumen. Berikut adalah hasil *Raw term Frequency* yang dapat dilihat dalam Tabel 4.25.

Tabel 4.25 Manualisasi *Raw Term Frequency Weighting*

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
lama	1	0	0	0	0	0	0	0	2
benar	2	0	0	0	0	0	0	0	1
ajar	1	0	0	0	0	0	0	0	0
sekali	1	0	1	0	0	0	0	0	0
uji	1	0	0	0	0	0	0	0	0
selalu	1	0	0	0	0	0	0	0	0
jujur	1	0	0	1	0	0	0	0	0
tugas	2	0	0	0	0	0	0	0	0
tinggal	1	0	0	0	0	0	0	0	0
pindah	1	0	0	0	0	0	0	0	0
dari	1	0	0	0	0	0	0	0	0

Tabel 4.25 Manualisasi *Raw Term Frequency Weighting* (lanjutan)

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
internet	1	0	0	0	0	0	0	0	0
dosen	1	0	1	0	2	0	0	0	0
hanya	1	0	0	0	0	0	0	0	0
beri	1	0	0	0	0	0	0	0	0
pernah	1	0	0	0	0	0	0	0	0
jelas	1	1	0	0	0	0	0	0	0
tambah	1	0	0	0	0	0	0	0	0
semester	1	0	0	0	0	0	0	0	0
apa	1	0	0	0	0	0	0	0	0
rasa	0	1	0	0	0	0	0	0	0
henti	0	1	0	0	0	0	0	0	0
kalau	0	1	0	1	0	0	0	0	1
begini	0	1	0	0	0	0	0	0	0
seperti	0	1	0	1	0	0	0	0	0
bayar	0	1	0	0	0	0	0	0	0
cuma	0	2	0	1	0	0	0	0	0
suruh	0	1	0	0	0	0	0	0	0
baca	0	1	0	0	0	0	0	0	0
tanpa	0	1	0	0	0	0	0	0	0
asa	0	1	0	0	0	0	0	0	0
otodidak	0	1	0	0	0	0	0	0	0
maaf	0	0	1	0	0	0	0	0	0
makin	0	0	1	0	0	0	0	0	0
malas	0	0	1	0	0	0	0	0	0
tidur	0	0	1	0	0	0	0	0	0
baik	0	0	1	0	0	0	0	0	0
darimananya	0	0	1	0	0	0	0	0	0
coba	0	0	1	0	0	0	0	0	0
nilai	0	0	1	0	0	0	0	1	0

Tabel 4.25 Manualisasi *Raw Term Frequency Weighting* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
sempurna	0	0	1	0	0	0	0	0	0
bukan	0	0	1	0	0	0	0	0	0
karena	0	0	2	0	0	0	1	0	0
kita	0	0	2	0	0	0	0	0	0
cerdas	0	0	1	0	0	0	0	0	0
tapi	0	0	2	0	0	1	0	0	0
kasihan	0	0	1	0	0	0	0	0	0
paham	0	0	1	0	1	0	0	0	0
kosong	0	0	1	0	0	0	0	0	0
otak	0	0	1	0	0	0	0	0	0
terima	0	0	1	0	0	0	0	0	0
oke	0	0	0	2	0	0	0	0	0
dengan	0	0	0	1	0	2	0	0	0
ya	0	0	0	1	0	0	0	0	1
itu	0	0	0	1	0	0	0	0	0
kangen	0	0	0	1	0	0	0	0	0
suasana	0	0	0	1	0	0	0	0	0
corona	0	0	0	1	0	0	1	0	0
sudah	0	0	0	1	0	0	0	0	0
selesai	0	0	0	1	0	0	0	0	1
padu	0	0	0	1	0	0	0	0	0
asik	0	0	0	1	0	0	0	0	0
ribut	0	0	0	0	1	0	0	0	0
masalah	0	0	0	0	1	0	0	0	0
bagi	0	0	0	0	1	0	0	0	0
salah	0	0	0	0	2	0	0	0	0
diri	0	0	0	0	1	0	0	0	0
atau	0	0	0	0	1	0	0	0	0
semua	0	0	0	0	1	0	0	0	0

Tabel 4.25 Manualisasi *Raw Term Frequency Weighting* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
gantung	0	0	0	0	1	0	0	0	0
pribadi	0	0	0	0	1	0	0	0	0
masing	0	0	0	0	2	0	0	0	0
dalam	0	0	0	0	1	0	0	0	0
pak	0	0	0	0	0	2	0	0	0
gimana	0	0	0	0	0	1	0	0	0
anak	0	0	0	0	0	3	0	0	0
sekolah	0	0	0	0	0	1	0	0	0
untuk	0	0	0	0	0	1	0	0	0
mahasiswa	0	0	0	0	0	2	0	1	0
laksana	0	0	0	0	0	1	0	0	0
justru	0	0	0	0	0	1	0	0	0
adaptasi	0	0	0	0	0	1	0	0	0
banding	0	0	0	0	0	1	0	0	0
masih	0	0	0	0	0	1	1	0	0
sangat	0	0	0	0	0	1	0	0	0
rentan	0	0	0	0	0	1	0	0	0
mohon	0	0	0	0	0	1	0	0	0
kaji	0	0	0	0	0	1	0	0	0
saya	0	0	0	0	0	0	1	0	0
doa	0	0	0	0	0	0	1	0	0
kampus	0	0	0	0	0	0	1	0	0
padahal	0	0	0	0	0	0	1	0	0
tempat	0	0	0	0	0	0	1	0	0
merah	0	0	0	0	0	0	1	0	0
dan	0	0	0	0	0	0	1	0	0
kerabat	0	0	0	0	0	0	1	0	0
kerja	0	0	0	0	0	0	2	0	0
dokter	0	0	0	0	0	0	1	0	0

Tabel 4.25 Manualisasi *Raw Term Frequency Weighting* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
suka	0	0	0	0	0	0	1	0	0
bilang	0	0	0	0	0	0	1	0	0
keras	0	0	0	0	0	0	1	0	0
pasien	0	0	0	0	0	0	1	0	0
nyaman	0	0	0	0	0	0	1	0	0
rumah	0	0	0	0	0	0	1	0	0
adalah	0	0	0	0	0	0	1	0	0
jalan	0	0	0	0	0	0	1	0	0
positif	0	0	0	0	0	0	0	2	0
ambil	0	0	0	0	0	0	0	1	0
buang	0	0	0	0	0	0	0	1	0
negatif	0	0	0	0	0	0	0	1	0
mungkin	0	0	0	0	0	0	0	1	0
berani	0	0	0	0	0	0	0	1	0
tanya	0	0	0	0	0	0	0	2	0
di	0	0	0	0	0	0	0	2	0
aktif	0	0	0	0	0	0	0	1	0
sampai	0	0	0	0	0	0	0	0	1
masa	0	0	0	0	0	0	0	0	1
pandemi	0	0	0	0	0	0	0	0	1
harus	0	0	0	0	0	0	0	0	1

4.2.4.2 *Log Term Frequency Weighting*

Pada *log term Frequency weighting,* setiap *term* yang sudah dilakukan proses *Raw term Frequency weighting* akan dihitung logaritmanya. Berikut adalah contoh perhitungan dari *log term Frequency weighting.*

Misalkan kita menggunakan kata “benar” dan berikut adalah hasil yang didapatkan setelah proses *Raw term Frequency weighting* pada Tabel 4.26.

Tabel 4.26 Sampel Hasil Proses *Raw term Frequency weighting*

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
benar	2	0	0	0	0	0	0	0	1

Berikut adalah rumus yang digunakan dalam perhitungan *log term Frequency weighting.*

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf, jika\ tf_{t,d} > 0 \\ 0, jika\ tf_{t,d} = 0 \end{cases}$$

Sehingga berikut adalah contoh perhitungan dari *term* "aku" di D1, D2, dan D3.

$$weight(aku)(d1) = 1 + \log_{10} 2 = 1.30102$$

$$weight(aku)(d2) = 0$$

$$weight(aku)(d3) = 0$$

Setelah melalui proses perhitungan tersebut, berikut adalah hasil *log term Frequency* yang dapat dilihat dalam Tabel 4.27.

Tabel 4.27 Manualisasi Log *Term Frequency Weighting*

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
lama	1.0	0	0	0	0	0	0	0	1.301
benar	1.301	0	0	0	0	0	0	0	1.0
ajar	1.0	0	0	0	0	0	0	0	0
sekali	1.0	0	1.0	0	0	0	0	0	0
uji	1.0	0	0	0	0	0	0	0	0
selalu	1.0	0	0	0	0	0	0	0	0
jujur	1.0	0	0	1.0	0	0	0	0	0
tugas	1.301	0	0	0	0	0	0	0	0
tinggal	1.0	0	0	0	0	0	0	0	0
pindah	1.0	0	0	0	0	0	0	0	0
dari	1.0	0	0	0	0	0	0	0	0
internet	1.0	0	0	0	0	0	0	0	0
dosen	1.0	0	1.0	0	1.301	0	0	0	0
hanya	1.0	0	0	0	0	0	0	0	0
beri	1.0	0	0	0	0	0	0	0	0
pernah	1.0	0	0	0	0	0	0	0	0
jelas	1.0	1.0	0	0	0	0	0	0	0
tambah	1.0	0	0	0	0	0	0	0	0
semester	1.0	0	0	0	0	0	0	0	0
apa	1.0	0	0	0	0	0	0	0	0
rasa	0	1.0	0	0	0	0	0	0	0

Tabel 4.27 Manualisasi Log *Term Frequency Weighting* (lanjutan)

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
henti	0	1.0	0	0	0	0	0	0	0
kalau	0	1.0	0	1.0	0	0	0	0	1.0
begini	0	1.0	0	0	0	0	0	0	0
seperti	0	1.0	0	1.0	0	0	0	0	0
bayar	0	1.0	0	0	0	0	0	0	0
cuma	0	1.301	0	1.0	0	0	0	0	0
suruh	0	1.0	0	0	0	0	0	0	0
baca	0	1.0	0	0	0	0	0	0	0
tanpa	0	1.0	0	0	0	0	0	0	0
asa	0	1.0	0	0	0	0	0	0	0
otodidak	0	1.0	0	0	0	0	0	0	0
maaf	0	0	1.0	0	0	0	0	0	0
makin	0	0	1.0	0	0	0	0	0	0
malas	0	0	1.0	0	0	0	0	0	0
tidur	0	0	1.0	0	0	0	0	0	0
baik	0	0	1.0	0	0	0	0	0	0
darimana nya	0	0	1.0	0	0	0	0	0	0
coba	0	0	1.0	0	0	0	0	0	0
nilai	0	0	1.0	0	0	0	0	1.0	0
sempurna	0	0	1.0	0	0	0	0	0	0
bukan	0	0	1.0	0	0	0	0	0	0
karena	0	0	1.301	0	0	0	1.0	0	0
kita	0	0	1.301	0	0	0	0	0	0
cerdas	0	0	1.0	0	0	0	0	0	0
tapi	0	0	1.301	0	0	1.0	0	0	0
kasihan	0	0	1.0	0	0	0	0	0	0
paham	0	0	1.0	0	1.0	0	0	0	0
kosong	0	0	1.0	0	0	0	0	0	0
otak	0	0	1.0	0	0	0	0	0	0

Tabel 4.27 Manualisasi Log *Term Frequency Weighting* (lanjutan)

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
terima	0	0	1.0	0	0	0	0	0	0
oke	0	0	0	1.301	0	0	0	0	0
dengan	0	0	0	1.0	0	1.301	0	0	0
ya	0	0	0	1.0	0	0	0	0	1.0
itu	0	0	0	1.0	0	0	0	0	0
kangen	0	0	0	1.0	0	0	0	0	0
suasana	0	0	0	1.0	0	0	0	0	0
corona	0	0	0	1.0	0	0	1.0	0	0
sudah	0	0	0	1.0	0	0	0	0	0
selesai	0	0	0	1.0	0	0	0	0	1.0
padu	0	0	0	1.0	0	0	0	0	0
asik	0	0	0	1.0	0	0	0	0	0
ribut	0	0	0	0	1.0	0	0	0	0
masalah	0	0	0	0	1.0	0	0	0	0
bagi	0	0	0	0	1.0	0	0	0	0
salah	0	0	0	0	1.301	0	0	0	0
diri	0	0	0	0	1.0	0	0	0	0
atau	0	0	0	0	1.0	0	0	0	0
semua	0	0	0	0	1.0	0	0	0	0
gantung	0	0	0	0	1.0	0	0	0	0
pribadi	0	0	0	0	1.0	0	0	0	0
masing	0	0	0	0	1.301	0	0	0	0
dalam	0	0	0	0	1.0	0	0	0	0
pak	0	0	0	0	0	1.301	0	0	0
gimana	0	0	0	0	0	1.0	0	0	0
anak	0	0	0	0	0	1.477	0	0	0
sekolah	0	0	0	0	0	1.0	0	0	0
untuk	0	0	0	0	0	1.0	0	0	0
mahasiswa	0	0	0	0	0	1.301	0	1.0	0
laksana	0	0	0	0	0	1.0	0	0	0

Tabel 4.27 Manualisasi Log *Term Frequency Weighting* (lanjutan)

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
justru	0	0	0	0	0	1.0	0	0	0
adaptasi	0	0	0	0	0	1.0	0	0	0
banding	0	0	0	0	0	1.0	0	0	0
masih	0	0	0	0	0	1.0	1.0	0	0
sangat	0	0	0	0	0	1.0	0	0	0
rentan	0	0	0	0	0	1.0	0	0	0
mohon	0	0	0	0	0	1.0	0	0	0
kaji	0	0	0	0	0	1.0	0	0	0
saya	0	0	0	0	0	0	1.0	0	0
doa	0	0	0	0	0	0	1.0	0	0
kampus	0	0	0	0	0	0	1.0	0	0
padahal	0	0	0	0	0	0	1.0	0	0
tempat	0	0	0	0	0	0	1.0	0	0
merah	0	0	0	0	0	0	1.0	0	0
dan	0	0	0	0	0	0	1.0	0	0
kerabat	0	0	0	0	0	0	1.0	0	0
kerja	0	0	0	0	0	0	1.301	0	0
dokter	0	0	0	0	0	0	1.0	0	0
suka	0	0	0	0	0	0	1.0	0	0
bilang	0	0	0	0	0	0	1.0	0	0
keras	0	0	0	0	0	0	1.0	0	0
pasien	0	0	0	0	0	0	1.0	0	0
nyaman	0	0	0	0	0	0	1.0	0	0
rumah	0	0	0	0	0	0	1.0	0	0
adalah	0	0	0	0	0	0	1.0	0	0
jalan	0	0	0	0	0	0	1.0	0	0
positif	0	0	0	0	0	0	0	1.301	0
ambil	0	0	0	0	0	0	0	1.0	0
buang	0	0	0	0	0	0	0	1.0	0
negatif	0	0	0	0	0	0	0	1.0	0
mungkin	0	0	0	0	0	0	0	1.0	0

Tabel 4.27 Manualisasi Log *Term Frequency Weighting* (lanjutan)

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
berani	0	0	0	0	0	0	0	1.0	0
tanya	0	0	0	0	0	0	0	1.301	0
di	0	0	0	0	0	0	0	1.301	0
aktif	0	0	0	0	0	0	0	1.0	0
sampai	0	0	0	0	0	0	0	0	1.0
masa	0	0	0	0	0	0	0	0	1.0
pandemi	0	0	0	0	0	0	0	0	1.0
harus	0	0	0	0	0	0	0	0	1.0

4.2.4.3 *Inverse Document Frequency*

Pada *inverse document Frequency*, akan dilakukan proses perhitungan jumlah dokumen yang mengandung suatu *term* dan jumlah tersebut akan dilakukan proses *inverse.* Tahapan awal yang harus dilakukan adalah menghitung *document Frequency* atau frekuensi dokumen yang mengandung suatu *term*. Berikut adalah hasil perhitungan *document Frequency* yang ditampilkan pada Tabel 4.28.

Tabel 4.28 Manualisasi *Document Frequency*

Term	**DF**
lama	2
benar	2
ajar	1
sekali	2
uji	1
selalu	1
jujur	2
tugas	1
tinggal	1
pindah	1
dari	1
internet	1
dosen	3

Tabel 4.28 Manualisasi *Document Frequency* (lanjutan)

Term	**DF**
hanya	1
beri	1
pernah	1
jelas	2
tambah	1
semester	1
apa	1
rasa	1
henti	1
kalau	3
begini	1
seperti	2
bayar	1
cuma	2
suruh	1
baca	1
tanpa	1
asa	1
otodidak	1
maaf	1
makin	1
malas	1
tidur	1
baik	1
darimananya	1
coba	1
nilai	2
sempurna	1
bukan	1

Tabel 4.28 Manualisasi *Document Frequency* (lanjutan)

Term	**DF**
karena	2
kita	1
cerdas	1
tapi	2
kasihan	1
paham	2
kosong	1
otak	1
terima	1
oke	1
dengan	2
ya	2
itu	1
kangen	1
suasana	1
corona	2
sudah	1
selesai	2
padu	1
asik	1
ribut	1
masalah	1
bagi	1
salah	1
diri	1
atau	1
semua	1
gantung	1

Tabel 4.28 Manualisasi *Document Frequency* (lanjutan)

Term	**DF**
pribadi	1
masing	1
dalam	1
pak	1
gimana	1
anak	1
sekolah	1
untuk	1
mahasiswa	2
laksana	1
justru	1
adaptasi	1
banding	1
masih	2
sangat	1
rentan	1
mohon	1
kaji	1
saya	1
doa	1
kampus	1
padahal	1
tempat	1
merah	1
dan	1
kerabat	1
kerja	1
dokter	1

Tabel 4.28 Manualisasi *Document Frequency* (lanjutan)

Term	**DF**
suka	1
bilang	1
keras	1
pasien	1
nyaman	1
rumah	1
adalah	1
jalan	1
positif	1
ambil	1
buang	1
negatif	1
mungkin	1
berani	1
tanya	1
di	1
aktif	1
sampai	1
masa	1
pandemi	1
harus	1

Setelah *document Frequency* tiap *term*nya didapatkan, langkah selanjutnya adalah menghitung *inverse document Frequency* dengan Persamaan 2.5. Sehingga berikut adalah contoh perhitungan dari *term* "lama", "benar", "ajar".

$$idf(lama) = \log_{10}\left(\frac{9}{2}\right) = 0.6532$$

$$idf(benar) = \log_{10}\left(\frac{9}{2}\right) = 0.6532$$

$$idf(ajar) = \log_{10}(\frac{9}{1}) = 0.9542$$

Setelah melalui proses perhitungan tersebut, berikut adalah proses perhitungan *inverse document Frequency* keseluruhan *term* yang dapat dilihat dalam Tabel 4.29.

Tabel 4.29 Manualisasi *Inverse Document Frequency*

Term	**IDF**
lama	0.653
benar	0.653
ajar	0.954
sekali	0.653
uji	0.954
selalu	0.954
jujur	0.653
tugas	0.954
tinggal	0.954
pindah	0.954
dari	0.954
internet	0.954
dosen	0.477
hanya	0.954
beri	0.954
pernah	0.954
jelas	0.653
tambah	0.954
semester	0.954
apa	0.954
rasa	0.954
henti	0.954
kalau	0.477
begini	0.954
seperti	0.653
bayar	0.954
cuma	0.653

Tabel 4.29 Manualisasi *Inverse Document Frequency* (lanjutan)

Term	**IDF**
suruh	0.954
baca	0.954
tanpa	0.954
asa	0.954
otodidak	0.954
maaf	0.954
makin	0.954
malas	0.954
tidur	0.954
baik	0.954
darimananya	0.954
coba	0.954
nilai	0.653
sempurna	0.954
bukan	0.954
karena	0.653
kita	0.954
cerdas	0.954
tapi	0.653
kasihan	0.954
paham	0.653
kosong	0.954
otak	0.954
terima	0.954
oke	0.954
dengan	0.653
ya	0.653
itu	0.954
kangen	0.954

Tabel 4.29 Manualisasi *Inverse Document Frequency* (lanjutan)

Term	**IDF**
suasana	0.954
corona	0.653
sudah	0.954
selesai	0.653
padu	0.954
asik	0.954
ribut	0.954
masalah	0.954
bagi	0.954
salah	0.954
diri	0.954
atau	0.954
semua	0.954
gantung	0.954
pribadi	0.954
masing	0.954
dalam	0.954
pak	0.954
gimana	0.954
anak	0.954
sekolah	0.954
untuk	0.954
mahasiswa	0.653
laksana	0.954
justru	0.954
adaptasi	0.954
banding	0.954
masih	0.653

Tabel 4.29 Manualisasi *Inverse Document Frequency* (lanjutan)

Term	**IDF**
sangat	0.954
rentan	0.954
mohon	0.954
kaji	0.954
saya	0.954
doa	0.954
kampus	0.954
padahal	0.954
tempat	0.954
merah	0.954
dan	0.954
kerabat	0.954
kerja	0.954
dokter	0.954
suka	0.954
bilang	0.954
keras	0.954
pasien	0.954
nyaman	0.954
rumah	0.954
adalah	0.954
jalan	0.954
positif	0.954
ambil	0.954
buang	0.954
negatif	0.954
mungkin	0.954
berani	0.954
tanya	0.954

Tabel 4.29 Manualisasi *Inverse Document Frequency* (lanjutan)

Term	**IDF**
di	0.954
aktif	0.954
sampai	0.954
masa	0.954
pandemi	0.954
harus	0.954

4.2.4.4 *Term Frequency - Inverse Document Frequency (TF-IDF)*

Pada *term Frequency - inverse document Frequency*, akan dilakukan proses perhitungan perkalian antara *log term Frequency* dikali dengan *inverse document Frequency.* Berikut adalah hasil perhitungan perkalian antara *log term Frequency* dikali dengan *inverse document Frequency* yang ditampilkan pada Tabel 4.30*.*

Tabel 4.30 Manualisasi *Term Frequency* - Inverse Document *Frequency*

Term	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**D8**	**D9**
lama	0.653	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.850
benar	0.850	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.653
ajar	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
sekali	0.653	0.0	0.653	0.0	0.0	0.0	0.0	0.0	0.0
uji	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
selalu	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
jujur	0.653	0.0	0.0	0.653	0.0	0.0	0.0	0.0	0.0
tugas	1.241	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tinggal	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
pindah	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
dari	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
internet	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
dosen	0.477	0.0	0.477	0.0	0.621	0.0	0.0	0.0	0.0
hanya	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
beri	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
pernah	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabel 4.30 Manualisasi *Term Frequency - Inverse Document Frequency* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
Jelas	0.653	0.653	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tambah	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
semester	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
apa	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
rasa	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
henti	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
kalau	0.0	0.477	0.0	0.477	0.0	0.0	0.0	0.0	0.477
begini	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
seperti	0.0	0.653	0.0	0.653	0.0	0.0	0.0	0.0	0.0
bayar	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
cuma	0.0	0.850	0.0	0.653	0.0	0.0	0.0	0.0	0.0
suruh	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
baca	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
tanpa	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
asa	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
otodidak	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0	0.0
maaf	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
makin	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
malas	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
tidur	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
baik	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
darimana nya	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
coba	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
nilai	0.0	0.0	0.653	0.0	0.0	0.0	0.0	0.653	0.0
sempurna	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
bukan	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
karena	0.0	0.0	0.850	0.0	0.0	0.0	0.653	0.0	0.0
kita	0.0	0.0	1.241	0.0	0.0	0.0	0.0	0.0	0.0
cerdas	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
tapi	0.0	0.0	0.850	0.0	0.0	0.653	0.0	0.0	0.0

Tabel 4.30 Manualisasi *Term Frequency - Inverse Document Frequency* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
kasihan	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
paham	0.0	0.0	0.653	0.0	0.653	0.0	0.0	0.0	0.0
kosong	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
otak	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
terima	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0	0.0
oke	0.0	0.0	0.0	1.241	0.0	0.0	0.0	0.0	0.0
dengan	0.0	0.0	0.0	0.653	0.0	0.850	0.0	0.0	0.0
ya	0.0	0.0	0.0	0.653	0.0	0.0	0.0	0.0	0.653
itu	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0
kangen	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0
suasana	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0
corona	0.0	0.0	0.0	0.653	0.0	0.0	0.653	0.0	0.0
sudah	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0
selesai	0.0	0.0	0.0	0.653	0.0	0.0	0.0	0.0	0.653
padu	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0
asik	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0	0.0
ribut	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
masalah	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
bagi	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
salah	0.0	0.0	0.0	0.0	1.241	0.0	0.0	0.0	0.0
diri	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
atau	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
semua	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
gantung	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
pribadi	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
masing	0.0	0.0	0.0	0.0	1.241	0.0	0.0	0.0	0.0
dalam	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0	0.0
pak	0.0	0.0	0.0	0.0	0.0	1.241	0.0	0.0	0.0
gimana	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
anak	0.0	0.0	0.0	0.0	0.0	1.410	0.0	0.0	0.0

Tabel 4.30 Manualisasi *Term Frequency - Inverse Document Frequency* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
sekolah	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
untuk	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
mahasiswa	0.0	0.0	0.0	0.0	0.0	0.850	0.0	0.653	0.0
laksana	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
justru	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
adaptasi	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
banding	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
masih	0.0	0.0	0.0	0.0	0.0	0.653	0.653	0.0	0.0
sangat	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
rentan	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
mohon	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
kaji	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0	0.0
saya	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
doa	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
kampus	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
padahal	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
tempat	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
merah	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
dan	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
kerabat	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
kerja	0.0	0.0	0.0	0.0	0.0	0.0	1.241	0.0	0.0
dokter	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
suka	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
bilang	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
keras	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
pasien	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
nyaman	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
rumah	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
adalah	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0
jalan	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0	0.0

Tabel 4.30 Manualisasi *Term Frequency - Inverse Document Frequency* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9
positif	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.241	0.0
ambil	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0
buang	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0
negatif	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0
mungkin	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0
berani	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0
tanya	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.241	0.0
di	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.241	0.0
aktif	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954	0.0
sampai	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954
masa	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954
pandemi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954
harus	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.954

4.2.5 Manualisasi *Naïve Bayes* Training

Pada tahapan ini terjadi pelatihan *Naïve Bayes* Multinomial untuk mendapatkan nilai *prior* tiap kelasnya dan *likelihood* tiap kata pada tiap kelasnya. Pada tahap manualisasi pelatihan *Naïve Bayes* ini, peneliti menggunakan contoh hasil pembobotan dengan *stoplist* 20 persen.

Tahapan ini diawali dengan pencarian *prior* untuk tiap kelasnya. Perhitungan *prior* dapat menggunakan Persamaan 2.7. Sehingga dapat ditentukan *prior* dari tiap kelasnya adalah sebagai berikut:

$$P(negatif) = \frac{3}{9} = 0.333$$

$$P(netral) = \frac{3}{9} = 0.333$$

$$P(positif) = \frac{3}{9} = 0.333$$

Dalam Persamaan 2.9 tersebut kita memasukan bobot *term*, jumlah bobot tiap kelas, serta jumlah idf yang sudah didapatkan diperhitungan sebelumnya. Berikut adalah contoh perhitungan dari *term* "berani":

$$P(berani|negatif) = \frac{0.0 + 1}{47.54539 + 107.483}$$

$$P(berani|negatif) = 0.00645$$

$$P(berani|netral) = \frac{0.0 + 1}{40.51552 + 107.483}$$

$$P(berani|netral) = 0.00675$$

$$P(berani|positif) = \frac{0.95424 + 1}{37.28323 + 107.483}$$

$$P(berani|positif) = 0.01349$$

Setelah dilakukan dengan semua *term* dalam keseluruhan dokumen setiap kelasnya maka didapatkan likelihood yang akan ditampilkan pada Tabel 4.31.

Tabel 4.31 Manualisasi *Likelihood*

Term	**Negatif**	**Netral**	**Positif**
lama	0.010664	0.006757	0.012778
benar	0.011932	0.006757	0.011420
ajar	0.012606	0.006757	0.006908
sekali	0.014877	0.006757	0.006908
uji	0.012606	0.006757	0.006908
selalu	0.012606	0.006757	0.006908
jujur	0.010664	0.011170	0.006908
tugas	0.014459	0.006757	0.006908
tinggal	0.012606	0.006757	0.006908
pindah	0.012606	0.006757	0.006908
dari	0.012606	0.006757	0.006908
internet	0.012606	0.006757	0.006908
dosen	0.012606	0.010951	0.006908
hanya	0.012606	0.006757	0.006908
beri	0.012606	0.006757	0.006908
pernah	0.012606	0.006757	0.006908
jelas	0.014877	0.006757	0.006908
tambah	0.012606	0.006757	0.006908
semester	0.012606	0.006757	0.006908
apa	0.012606	0.006757	0.006908
rasa	0.012606	0.006757	0.006908
henti	0.012606	0.006757	0.006908

Tabel 4.31 Manualisasi *Likelihood* (lanjutan)

Term	**Negatif**	**Netral**	**Positif**
kalau	0.009528	0.009981	0.010203
begini	0.012606	0.006757	0.006908
seperti	0.010664	0.011170	0.006908
bayar	0.012606	0.006757	0.006908
cuma	0.011932	0.011170	0.006908
suruh	0.012606	0.006757	0.006908
baca	0.012606	0.006757	0.006908
tanpa	0.012606	0.006757	0.006908
asa	0.012606	0.006757	0.006908
otodidak	0.012606	0.006757	0.006908
maaf	0.012606	0.006757	0.006908
makin	0.012606	0.006757	0.006908
malas	0.012606	0.006757	0.006908
tidur	0.012606	0.006757	0.006908
baik	0.012606	0.006757	0.006908
darimananya	0.012606	0.006757	0.006908
coba	0.012606	0.006757	0.006908
nilai	0.010664	0.006757	0.011420
sempurna	0.012606	0.006757	0.006908
bukan	0.012606	0.006757	0.006908
karena	0.011932	0.006757	0.011420
kita	0.014459	0.006757	0.006908
cerdas	0.012606	0.006757	0.006908
tapi	0.011932	0.011170	0.006908
kasihan	0.012606	0.006757	0.006908
paham	0.010664	0.011170	0.006908
kosong	0.012606	0.006757	0.006908
otak	0.012606	0.006757	0.006908
terima	0.012606	0.006757	0.006908

Tabel 4.31 Manualisasi *Likelihood* (lanjutan)

Term	**Negatif**	**Netral**	**Positif**
oke	0.006450	0.015145	0.006908
dengan	0.006450	0.016913	0.006908
ya	0.006450	0.011170	0.011420
itu	0.006450	0.013204	0.006908
kangen	0.006450	0.013204	0.006908
suasana	0.006450	0.013204	0.006908
corona	0.006450	0.011170	0.011420
sudah	0.006450	0.013204	0.006908
selesai	0.006450	0.011170	0.011420
padu	0.006450	0.013204	0.006908
asik	0.006450	0.013204	0.006908
ribut	0.006450	0.013204	0.006908
masalah	0.006450	0.013204	0.006908
bagi	0.006450	0.013204	0.006908
salah	0.006450	0.015145	0.006908
diri	0.006450	0.013204	0.006908
atau	0.006450	0.013204	0.006908
semua	0.006450	0.013204	0.006908
gantung	0.006450	0.013204	0.006908
pribadi	0.006450	0.013204	0.006908
masing	0.006450	0.015145	0.006908
dalam	0.006450	0.013204	0.006908
pak	0.006450	0.015145	0.006908
gimana	0.006450	0.013204	0.006908
anak	0.006450	0.016281	0.006908
sekolah	0.006450	0.013204	0.006908
untuk	0.006450	0.013204	0.006908
mahasiswa	0.006450	0.012499	0.011420
laksana	0.006450	0.013204	0.006908

Tabel 4.31 Manualisasi *Likelihood* (lanjutan)

Term	**Negatif**	**Netral**	**Positif**
justru	0.006450	0.013204	0.006908
adaptasi	0.006450	0.013204	0.006908
banding	0.006450	0.013204	0.006908
masih	0.006450	0.011170	0.011420
sangat	0.006450	0.013204	0.006908
rentan	0.006450	0.013204	0.006908
mohon	0.006450	0.013204	0.006908
kaji	0.006450	0.013204	0.006908
saya	0.006450	0.006757	0.013499
doa	0.006450	0.006757	0.013499
kampus	0.006450	0.006757	0.013499
padahal	0.006450	0.006757	0.013499
tempat	0.006450	0.006757	0.013499
merah	0.006450	0.006757	0.013499
dan	0.006450	0.006757	0.013499
kerabat	0.006450	0.006757	0.013499
kerja	0.006450	0.006757	0.015484
dokter	0.006450	0.006757	0.013499
suka	0.006450	0.006757	0.013499
bilang	0.006450	0.006757	0.013499
keras	0.006450	0.006757	0.013499
pasien	0.006450	0.006757	0.013499
nyaman	0.006450	0.006757	0.013499
rumah	0.006450	0.006757	0.013499
adalah	0.006450	0.006757	0.013499
jalan	0.006450	0.006757	0.013499
positif	0.006450	0.006757	0.015484
ambil	0.006450	0.006757	0.013499
buang	0.006450	0.006757	0.013499

Tabel 4.31 Manualisasi *Likelihood* (lanjutan)

Term	**Negatif**	**Netral**	**Positif**
negatif	0.006450	0.006757	0.013499
mungkin	0.006450	0.006757	0.013499
berani	0.006450	0.006757	0.013499
tanya	0.006450	0.006757	0.015484
di	0.006450	0.006757	0.015484
aktif	0.006450	0.006757	0.013499
sampai	0.006450	0.006757	0.013499
masa	0.006450	0.006757	0.013499
pandemi	0.006450	0.006757	0.013499
harus	0.006450	0.006757	0.013499

4.2.6 Manualisasi *Naïve Bayes Testing*

Setelah *prior* dan *likelihood* didapatkan dalam proses *training.* Selanjutnya adalah tahapan testing yang dimana didalamnya terjadi perkalian nilai *prior* tiap kelasnya dan *likelihood* tiap kata pada tiap kelasnya. Pada tahap manualisasi pengujian *Naïve Bayes* ini, peneliti menggunakan contoh hasil pembobotan dengan *stoplist* 20 persen.

Dalam tahapan ini, akan dicari probabilitas tertinggi masing-masing kelas untuk proses klasifikasi yang dihitung menggunakan Persamaan 2.6.

Setelah dilakukan *preprocessing* pada pembahasan sebelumnya, berikut adalah hasil akhir dari tahap *preprocessing* dari data uji menggunakan *stopword* 20 persen yang ditampilkan pada Tabel 4.32.

Tabel 4.32 Hasil *Preprocessing* Data Uji

No	Tweet	Kelas
1.	['apa', 'rasa', 'lama', 'nyaman', 'banget', 'ingin', 'masuk', 'takut', 'panik']	?

Sehingga dapat dihitung *posterior* dari tiap kelasnya sebagai berikut:

$$P(negatif|d) = P(negatif) * P(apa|negatif) * P(rasa|negatif) * P(lama|negatif) * P(nyaman|negatif) * P(banget|negatif) * P(ingin|negatif) * P(masuk|negatif) * P(takut|negatif) * P(panik|negatif)$$

$$P(negatif|d) = 0.33333 * 0.01260 * 0.00645 * 0.01260 * 0.00952 * 0.0106 * 0.00645 * 0.00645 * 0.00645 * 0.01193$$

$$P(negatif|d) = 1.1117643466553663e-19$$

Sehingga didapatkan bahwa probabilitas data uji diklasifikasikan kelas negatif adalah $1.1117643466553663e-19$.

$$P(netral|d) = P(netral) * P(apa|netral) * P(rasa|netral) * P(lama|netral) * P(nyaman|netral) * P(banget|netral) * P(ingin|netral) * P(masuk|netral) * P(takut|netral) * P(panik|netral)$$

$$P(netral|d) = 0.33333 * 0.00675 * 0.00675 * 0.00675 * 0.00998 * 0.00675 * 0.00675 * 0.00675 * 0.00675 * 0.00675$$

$$P(netral|d) = 1.4453587316155e-20$$

Sehingga didapatkan bahwa probabilitas data uji diklasifikasikan kelas netral adalah $1.4453587316155e-20$.

$$P(positif|d) = P(positif) * P(apa|positif) * P(rasa|positif) * P(lama|positif) * P(nyaman|positif) * P(banget|positif) * P(ingin|positif) * P(masuk|positif) * P(takut|positif) * P(panik|positif)$$

$$P(positif|d) = 0.33333 * 0.00690 * 0.01349 * 0.00690 * 0.01020 * 0.01277 * 0.01349 * 0.01349 * 0.01349 * 0.01141$$

$$P(positif|d) = 7.864406782717774e-19$$

Sehingga didapatkan bahwa probabilitas data uji diklasifikasikan kelas positif adalah $7.864406782717774e-19$. Berikut adalah hasil *posterior* dari setiap kelas yang ditampilkan pada Tabel 4.33.

Tabel 4.33 Hasil Manualisasi Posterior setiap Kelas

Klasifikasi	***Posterior***
Negatif	1.1117643466553663e-19
Netral	1.4453587316155e-20
Positif	7.864406782717774e-19

Kelas Positif memiliki nilai *posterior* tertinggi dibanding dengan *posterior* kelas lainnya. Oleh karena itu data uji dapat diklasifikasikan sebagai kelas Positif.

Data uji lainnya mengikuti proses yang sama seperti perhitungan sebelumnya, sehingga didapatkan hasil klasifikasi serta *posterior*nya yang ditampilkan pada Tabel 4.34, Tabel 4.35, Tabel 4.36, Tabel 4.37, dan Tabel 4.38.

Tabel 4.34 Hasil Manualisasi Data Uji 1

Data Uji 1		
Apa saya saja yang merasa kalau selama kuliah daring nyaman banget sampai saya tidak ingin masuk kuliah karena takut panik		
Negatif	**Netral**	**Positif**
1.1117643466553663e-19	1.4453587316155e-20	7.864406782717774e-19
Actual: Positif		
Prediction: Positif		

Tabel 4.35 Hasil Manualisasi Data Uji 2

Data Uji 2		
Aku merasa lebih leluasa dengan kuliah daring, tidak capek harus siap-siap berangkat. Hanya tinggal makan, beres didepan komputer sudah siap nyimak. Buat materi, selama online emang tidak pernah mengandalkan dosen atau teman. Jadi lebih banyak waktu buat searching sama buka text book.		
Negatif	**Netral**	**Positif**
1.9587729876808433e-21	1.0243444918164579e-21	2.9806998411846044e-22
Actual: Positif		
Prediction: Negatif		

Tabel 4.36 Hasil Manualisasi Data Uji 3

Data Uji 3		
Jujur tidak ada senang-senangnya kuliah daring. Aku butuh praktik lapangan. Apalagi semester depan magang. Apa magang online juga? Bisa stres gara-gara banyak deadline		
Negatif	**Netral**	**Positif**
5.64845483313626e-07	1.699942956949124e-07	1.0986925237206374e-07
Actual: Negatif		
Prediction: Negatif		

Tabel 4.37 Hasil Manualisasi Data Uji 4

Data Uji 4		
Tatap langsung aja kadang tidak paham, apalagi kuliah daring, belum lagi jaringan lambat ditambah beberapa dosen yang jarang memberi kuliah online, atau cuma memberi tugas saja... Fix kampus ku belum siap menerapkan kuliah daring! pic.twitter.com/UHdReyLgh8		
Negatif	**Netral**	**Positif**
6.443055725811558e-19	8.470540171181903e-20	2.332649424457759e-20
Actual: Negatif		
Prediction: Negatif		

Tabel 4.38 Hasil Manualisasi Data Uji 5

Data Uji 5		
Orang lain pada ribut sama keadaan kosan yang sudah ditinggal berbulan-bulan terus ribut gimana caranya balik ke kosan. Aku anteng-anteng saja jadi penghuni kos dari awal pemerintah nyuruh dirumah saja dan kuliah jadi daring		
Negatif	**Netral**	**Positif**
3.815427230622958e-18	2.1121804600458974e-17	6.5992632732260386e-18
Actual: Netral		
Prediction: Netral		

4.2.7 Manualisasi Evaluasi *Confusion Matrix*

Pada tahapan ini akan dijelaskan hasil evaluasi yang didapatkan dari pengujian yang sudah dijelaskan sebelumnya. Berdasarkan pengujian sebelumnya berikut adalah hasil klasifikasi yang didapatkan yang akan ditampilkan pada Tabel 4.39.

Tabel 4.39 Manualisasi *Confusion Matrix*

	Predicted			
Actual		**Negatif**	**Netral**	**Positif**
	Negatif	2	0	0
	Netral	0	1	0
	Positif	1	0	1

Setelah tabel *confusion matrix* dibuat, langkah selanjutnya adalah menghitung *accuracy* keseluruhan dan *accuracy, precision, recall,* dan *f-measure* tiap kelasnya. Untuk melakukan perhitungan *accuracy, precision, recall,* dan *f-*

measure tiap kelasnya diperlukan pencarian TP, FN, FP, TN terlebih dahulu yang memiliki tiap-tiap istilah tersebut memiliki definisi sebagai berikut yang akan ditampilkan pada Tabel 4.40.

Tabel 4.40 Definisi TP, FN, FP, dan TN

Note	
TP	Jumlah benar kelas tersebut
FN	Jumlah baris kelas tersebut tanpa TP
FP	Jumlah kolom kelas tersebut tanpa TP
TN	Jumlah semua baris dan kolom kecuali baris dan kolom kelas tersebut

Sehingga dapat ditentukan TP, FN, FP, TN untuk setiap kelasnya sebagaimana yang akan ditampilkan pada Tabel 4.41.

Tabel 4.41 Hasil Manualisasi TP, FN, FP, dan TN setiap kelas

Negatif	**TP**	**FN**	**FP**	**TN**
	2	0	1	2
Netral	**TP**	**FN**	**FP**	**TN**
	1	0	0	4
Positif	**TP**	**FN**	**FP**	**TN**
	1	1	0	3

Setelah semua didapatkan, kita bisa langsung menghitung *accuracy, precision, recall,* serta *f-measure* tiap kelasnya. Berikut adalah perhitungan *accuracy*, *precision, recall, f-measure* untuk masing-masing kelasnya.

Berikut adalah perhitungan untuk pencarian *accuracy*.

$$accuracy_{negatif} = \frac{2+2}{2+1+0+2} = \frac{4}{5} = 0.8$$

$$accuracy_{netral} = \frac{4+1}{4+0+0+1} = \frac{5}{5} = 1$$

$$accuracy_{positif} = \frac{3+1}{3+0+1+1} = \frac{4}{5} = 0.8$$

$$accuracy_{M} = \frac{0.8+1+0.8}{3} = 0.867$$

Berikut adalah perhitungan untuk pencarian *precision*.

$$precision_{negatif} = \frac{2}{2+1} = 0.667$$

$$precision_{netral} = \frac{1}{1+0} = 1$$

$$precision_{positif} = \frac{1}{1+0} = 1$$

$$precision_M = \frac{0.667 + 1 + 1}{3} = 0.889$$

Berikut adalah perhitungan untuk menghitung *recall.*

$$recall_{negatif} = \frac{2}{2+0} = 1$$

$$recall_{netral} = \frac{1}{1+0} = 1$$

$$recall_{positif} = \frac{1}{1+1} = 0.5$$

$$recall_M = \frac{1 + 1 + 0.5}{3} = 0.833$$

Berikut adalah perhitungan untuk menghitung *f-measure*.

$$f - measure_M = \frac{2 * \ 0.889 \ * 0.833}{0.889 + 0.833} = 0.86$$

Setelah setiap melalui proses perhitungan tersebut, berikut adalah hasil evaluasi yang akan ditampilkan pada Tabel 4.42.

Tabel 4.42 Hasil Evaluasi Manualisasi

Accuracy	Precision	Recall	F-Measure
0.867	0.889	0.933	0.86

4.3 Perancangan Pengujian

Pada tahapan ini akan dijelaskan perancangan pengujian mengenai hasil klasifikasi sentimen yang terdiri dari kelas negatif, netral, dan positif dengan menggunakan metode *Naïve Bayes Multinomial* serta pembuatan daftar *stopword* dengan *Term Based Random Sampling*. Pada bagian ini ada 3 skenario perancangan pengujian yang akan dilakukan.

4.3.1 Perancangan Pengujian Kombinasi Terbaik Parameter X, Y, dan L terhadap Hasil Evaluasi Sistem menggunakan K-fold Cross Validation.

Dalam melakukan perancangan pengujian kombinasi terbaik parameter X, Y, dan L terhadap hasil evaluasi diperlukan tabel pengujian yang dapat dilihat dalam Tabel 4.43.

Tabel 4.43 Perancangan Pengujian Kombinasi Terbaik X, Y, L terhadap Hasil Evaluasi

Parameter			Hasil Evaluasi								
X	Y	L	K-fold	Accuracy	Precision	Recall	F-Measure	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F-Measure
10	10	10	1								
			2								
			3								
			4								
			5								
			6								
			7								
			8								
			9								
			10								
…	…	…									

4.3.2 Perancangan Pengujian Perbandingan *stopword Term Based Random Sampling* dengan tanpa *Stopword Removal* Akurasi Sistem.

Dalam melakukan perancangan pengujian perbandingan *stopword Term Based Random Sampling* dengan tanpa Stopword Removal dalam Akurasi Sistem diperlukan tabel pengujian yang dapat dilihat dalam Tabel 4.44.

Tabel 4.44 Perancangan Pengujian Perbandingan *stopword Term Based Random Sampling* dengan tanpa *Stopword Removal* dalam Akurasi Sistem

k-fold	*Stopword*	Accuracy	Precision	Recall	F-Measure
1	Tanpa *Stopword*				
	TBRS				
2	Tanpa *Stopword*				
	TBRS				
Dst.	…..				

4.3.3 Perancangan Pengujian Perbandingan Akurasi Pengunaan *Stopword* Tala dan *Stopword Term Based Random Sampling*.

Dalam melakukan perancangan pengujian terhadap Akurasi Pengunaan *Stopword* Tala dan *Stopword Term Based Random Sampling* diperlukan tabel pengujian yang dapat dilihat dalam Tabel 4.45.

Tabel 4.45 Perancangan Pengujian Perbandingan Akurasi Pengunaan *Stopword* Tala dan *Stopword Term Based Random Sampling*

k-fold	*Stopword*	Accuracy	Precision	Recall	F-Measure
1	Tala				
	TBRS				
2	Tala				
	TBRS				
Dst.	…..				

BAB 5 IMPLEMENTASI

Pada bab ini akan dijelaskan implementasi kode program berdasarkan yang sudah dirancang serta dilakukan penjelasan tiap baris pada kode program.

5.1 Implementasi *Preprocessing*

Pada bagian *preprocessing* ini akan meliputi proses *case folding, cleaning, stemming, stopword removal* (opsional), serta *tokenizing*. Tahapan *preprocessing* ini diawali dengan pemanggilan *library* yang diperlukan untuk prosesnya seperti pada Kode Program 5.1.

```
Import library
1  import re
2  from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
```

Kode Program 5.1 Import Library

Penjelasan Kode Program 5.1:

Baris 1 Digunakan untuk *import library re*

Baris 2 Digunakan untuk *import library StemmerFactory* dari Sastrawi

Setelah tahapan *import library*, tahapan selanjutnya adalah tahapan *setup library* yang ditunjukkan pada Kode Program 5.2.

```
Setup library
1  stemmerFactory = StemmerFactory()
2  self.stemmer = stemmerFactory.create_stemmer()
```

Kode Program 5.2 Setup Library

Penjelasan Kode Program 5.2:

Baris 1-2 Digunakan untuk pembuatan fungsi stemmer dari objek instansiasi kelas *StemmerFactory* yang memanggil fungsi create_stemmer

Setelah tahapan *setup library*, tahapan selanjutnya adalah tahapan *preprocessing* yang ditunjukkan pada Kode Program 5.3.

```
Algoritme Preprocessing
1   def preprocessing(self,data,stopwords=None):
2       for i in range(len(data)):
3           case_folding = data[i].lower()
4           remove_newline = case_folding.replace("\n"," ")
5           cleaning = re.sub(r'[^a-zA-Z]', " ",remove_newline)
6           if stopwords != None:
7               stemming = self.stemmer.stem(cleaning)
8               filtered_words = [word for word in stemming.split
9   () if word not in stopwords]
10              self.cleaned_data.append(" ".join(filtered_words)
11  )
12              tokenizing = [word for word in filtered_words if
13  word.isalpha()]
14          else:
15              stemming = self.stemmer.stem(cleaning)
```

```
16              self.cleaned_data.append(stemming)
17              tokenizing = [word for word in stemming.split() i
18  f word.isalpha()]
19          for word in tokenizing:
20              self.token.append(word)
21              if word not in self.terms:
22                  self.terms.append(word)
23      return self.cleaned_data, self.terms
```

Kode Program 5.3 *Preprocessing*

Penjelasan Kode Program 5.3:

Baris 1	Mendefinisikan *method preprocessing* dengan parameter *data,* dan *stopwords* yang memiliki *default value None*.
Baris 2	Melakukan proses perulangan I hingga sepanjang variabel data
Baris 3	Mengecilkan huruf didalam data index i dan disimpan dalam case_folding
Baris 4	Menghilangkan *newline* dari dalam kalimat dan menggantinya dengan spasi dan disimpan pada remove_newline
Baris 5	Menghilangkan *non-alphabet* dari dalam kalimat dan disimpan pada *cleaning*
Baris 6	Melakukan seleksi jika *stopwords* tidak sama dengan *None*
Baris 7	Melakukan *stemming* variabel *cleaning* dan disimpan ke dalam *stemming*
Baris 8	Melakukan proses penghapusan kata *stopwords* didalam variabel *stemming* dan disimpan dalam *filtered_words*
Baris 10	Menambahkan *filtered_words* yang sudah di *join* dengan spasi kedalam *cleaned_data.*
Baris 12	Melakukan tokenisasi kata didalam *filtered_words*
Baris 14	Selain itu
Baris 15	Melakukan *stemming* variabel *cleaning* dan disimpan ke dalam *stemming*
Baris 16	Menambahkan *stemming* kedalam *cleaned_data.*
Baris 17	Melakukan tokenisasi kata didalam *stemming*
Baris 19-22	Melakukan perulangan kata variabel *tokenizing* untuk memasukan setiap kata ke dalam *token*, serta melakukan seleksi jika kata belum ada di *terms* maka kata tersebut dimasukkan ke dalam variabel *terms*
Baris 23	Mengembalikan *self.cleaned_data*, dan *self.terms*

Setelah tahapan *preprocessing*, tahapan selanjutnya adalah tahapan *get token* yang ditunjukkan pada Kode Program 5.4.

```
Get Token
1   def get_token(self):
2       return self.token
```

Kode Program 5.4 *Get Token*

Penjelasan Kode Program 5.4:

Baris 1 Mendefinisikan *method get_token*.

Baris 2 Mengembalikan *self.token*

Setelah tahapan *get token*, tahapan selanjutnya adalah tahapan *remove stopword* yang digunakan untuk menghapus *stopword* jika tahap *preprocessing* awal dilakukan tanpa penghapusan *stopword*. Proses ini ditunjukkan pada Kode Program 5.5.

```
Remove Stopword
1   def remove_stopword(self,cleaned_data,stopwords):
2       new_cleaned_data = []
3       new_terms = []
4       for data in cleaned_data:
5           filtered_words = [word for word in data.split() if wo
6   rd not in stopwords]
7           new_cleaned_data.append(" ".join(filtered_words))
8           for word in filtered_words:
9               self.token.append(word)
10              if word not in new_terms:
11                  new_terms.append(word)
12      return new_cleaned_data, new_terms
```

Kode Program 5.5 Remove *Stopword*

Penjelasan Kode Program 5.5:

Baris 1 Mendefinisikan *method remove_stopword* dengan parameter *cleaned_data,* dan *stopwords*.

Baris 2-3 Mendefinisikan *new_cleaned_data* dan *new_terms* sebagai list

Baris 4 Melakukan perulangan data dalam *cleaned_data*

Baris 5 Melakukan proses penghapusan kata *stopwords* didalam variabel *data* dan disimpan dalam *filtered_words*

Baris 7 Menambahkan *filtered_words* yang sudah di *join* dengan spasi kedalam new*_cleaned_data.*

Baris 8-11 Melakukan perulangan kata variabel *filtered_words* untuk memasukan setiap kata ke dalam *token*, serta melakukan seleksi jika kata belum ada di *new_terms* maka kata tersebut dimasukkan ke dalam variabel *new_terms*

Baris 12 Mengembalikan *new_cleaned_data*, dan *new_terms*

5.2 Implementasi *Term Based Random Sampling*

Pada bagian ini *preprocessing* ini akan meliputi proses *case folding, cleaning, stemming, stopword removal* (opsional), serta *tokenizing*. Tahapan *preprocessing* ini diawali dengan pemanggilan *library* yang diperlukan untuk prosesnya seperti pada Kode Program 5.6.

```
Import library
1  import numpy as np
2  import re
3  from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
4  from random import randint
```

Kode Program 5.6 Import Library

Penjelasan Kode Program 5.6:

Baris 1 Digunakan untuk *import library numpy* sebagai *np*

Baris 2 Digunakan untuk *import library re*

Baris 3 Digunakan untuk *import library StemmerFactory* dari Sastrawi

Baris 4 Digunakan untuk *import library randint* dari *random*

Setelah tahapan *import library*, tahapan selanjutnya adalah tahapan *setup library* yang ditunjukkan pada Kode Program 5.7.

```
Setup library
1  def setup_library(self):
2      stemmerFactory = StemmerFactory()
3      self.stemmer = stemmerFactory.create_stemmer()
```

Kode Program 5.7 Setup Library

Penjelasan Kode Program 5.7:

Baris 1 Mendefinisikan *method setup_library*

Baris 2-3 Digunakan untuk pembuatan fungsi stemmer dari objek instansiasi kelas *StemmerFactory* yang memanggil fungsi create_stemmer

Setelah tahapan *setup library*, tahapan selanjutnya adalah tahapan persiapan *method* yang digunakan dalam proses pembuatan *stopwords.* Proses diawali dengan *method generate random words* yang ditunjukkan pada Kode Program 5.8.

```
Algoritme Generate Random Words
1  def generate_random_words(self,token):
2      return token[randint(0,len(token)-1)]
```

Kode Program 5.8 Generate *Random* Words

Penjelasan Kode Program 5.8:

Baris 1 Mendefinisikan *method generate_random_words* dengan parameter *token*

Baris 2 Mengembalikan token dengan index yang ditentukan secara

random dari 0 hingga panjang token dikurang 1

Setelah tahapan *generate random words*, tahapan selanjutnya adalah tahapan *get documents contains words* yang ditunjukkan pada Kode Program 5.9

```
Algoritme Get Documents Contains Words
1  def get_documents_contains_words(self,words,documents):
2      sampled_documents = []
3      for tweet in documents:
4          if words in tweet.split():
5              sampled_documents.append(tweet)
6      return sampled_documents
```

Kode Program 5.9 Get Documents Contains Words

Penjelasan Kode Program 5.9:

Baris 1 Mendefinisikan *method get_documents_contains_words* dengan parameter *words* dan *documents*

Baris 2 Mendefinisikan *sampled_documents* sebagai list

Baris 3 Melakukan perulangan tweet dalam documents

Baris 4 - 5 Jika *words* ada didalam *list tweet* maka *tweet* dimasukkan ke *sampled_documents*.

Baris 6 Mengembalikan *sampled_documents*

Setelah tahapan *get documents contains words*, tahapan selanjutnya adalah tahapan *count words* yang ditunjukkan pada Kode Program 5.10.

```
Algortime Count Words
1  def count_words(self,word, documents):
2      count = 0
3      for tweet in documents:
4          for w in tweet.split():
5              if word == w:
6                  count+=1
7      return float(count)
```

Kode Program 5.10 Count Words

Penjelasan Kode Program 5.10:

Baris 1 Mendefinisikan *method count_words* dengan parameter *word* dan *documents*

Baris 2 Mendefinisikan *count dan diinisialisasi dengan 0*

Baris 3 Melakukan perulangan *tweet* dalam *documents*

Baris 4 Melakukan perulangan w didalam *tweet* yang di *split*

Baris 5 - 6 Jika *word* sama dengan w maka *count* ditambah 1.

Baris 7 Mengembalikan *count* yang diubah menjadi *float*

Setelah tahapan *count words*, tahapan selanjutnya adalah tahapan *get sum of the length document* yang ditunjukkan pada Kode Program 5.11.

```
Algoritme Get Sum of the length document
1  def get_sum_of_the_length_document(self,documents):
2      sum = 0
3      for tweet in documents:
4          sum+=len(tweet.split())
5      return float(sum)
```

Kode Program 5.11 Get Sum of the length document

Penjelasan Kode Program 5.11:

Baris 1 Mendefinisikan *method get_sum_of_the_length_document* dengan parameter *documents*

Baris 2 Mendefinisikan *sum dan diinisialisasi dengan 0*

Baris 3 Melakukan perulangan *tweet* dalam *documents*

Baris 4 Variabel *sum* ditambahkan dengan panjang *tweet* yang di *split*

Baris 5 Mengembalikan *sum* yang diubah menjadi *float*

Setelah tahapan *get sum of the length document*, tahapan selanjutnya adalah tahapan *get term* yang ditunjukkan pada Kode Program 5.12.

```
Algoritme Get Term
1  def get_term(self,documents):
2      term = []
3      for tweet in documents:
4          for word in tweet.split():
5              if word not in term:
6                  term.append(word)
7      return term
```

Kode Program 5.12 Get *Term*

Penjelasan Kode Program 5.12:

Baris 1 Mendefinisikan *method get_term* dengan parameter *documents*

Baris 2 Mendefinisikan *term* sebagai *list*

Baris 3 Melakukan perulangan *tweet* dalam *documents*

Baris 4 Melakukan perulangan *word* dalam *tweet* yang di *split*

Baris 5 Jika *word* tidak ada di dalam *term*

Baris 6 *Word* dimasukkan kedalam *term*

Baris 7 Mengembalikan *term*

Setelah tahapan *get term*, tahapan selanjutnya adalah tahapan *get total token* yang ditunjukkan pada Kode Program 5.13

```
Algoritme Get Total Token
1  def get_total_token(self,token):
2      return float(len(token))
```

Kode Program 5.13 Get Total Token

Penjelasan Kode Program 5.13:

Baris 1 Mendefinisikan *method get_total_token* dengan parameter *token*

Baris 2 Mengembalikan panjang *token* dalam bentuk float

Setelah tahapan *get total token*, tahapan selanjutnya adalah tahapan perhitungan *Kullback-Leibler Divergence* yang ditunjukkan pada Kode Program 5.14.

```
Algoritme Hitung KL Div
1   def kl_div(self,word,sampled_documents):
2       tf_x = self.count_words(word,sampled_documents)
3       l_x = self.get_sum_of_the_length_document(sampled_document
4   s)
5       p_x = tf_x / l_x
6       F = self.count_words(word,self.cleaned_data)
7       token_c = self.get_total_token(self.terms)
8       p_c = F / token_c
9       w_t = p_x * np.log2(p_x/p_c)
10      return w_t
```

Kode Program 5.14 Hitung KL Div

Penjelasan Kode Program 5.14:

Baris 1 Mendefinisikan *method kl_div* dengan parameter *word* dan *documents*

Baris 2 Menghitung nilai *tf_x* dengan memanggil fungsi *count_words* dengan *parameter word* dan *sampled_documents.*

Baris 3 Menghitung nilai l_*x* dengan memanggil fungsi *get_sum_of_the_length_document* dengan *parameter sampled_documents.*

Baris 5 Menghitung nilai *p_x* dengan cara *tf_x* dibagi dengan *l_x*

Baris 6 Menghitung nilai *F* dengan memanggil *method count_words* dengan parameter *word* dan *self.cleaned_data*

Baris 7 Menghitung nilai *token_c* dengan memanggil *method get_total_token* dengan parameter *self.terms*

Baris 8 Menghitung nilai *p_c* dengan cara *F* dibagi dengan *token_c*

Baris 9 Menghitung w_t dengan cara *p_x* dikali dengan hasil log2 dari *p_x* dibagi dengan *p_c*

Baris 10 Mengembalikan bobot

Setelah tahapan perhitungan *Kullback-Leibler Divergence,* tahapan selanjutnya adalah tahapan perhitungan *create stopwords* yang memanggil *method-method* yang sudah dibuat sebelumnya yang ditunjukkan pada Kode Program 5.15.

```
Algoritme Create Stopwords
1   def create_stopwords(self,cleaned_data,terms):
2       self.cleaned_data = cleaned_data
```

```
    self.terms = terms

    for i in range(self.Y):
        w_random = self.generate_random_words(self.terms)
        sampled_documents = self.get_documents_contains_words(w_random, self.cleaned_data)
        term_sampled_documents = self.get_term(sampled_documents)
        token_w = {}
        for word in term_sampled_documents:
            token_w[word] = self.kl_div(word,sampled_documents)
            if word not in self.token_used:
                self.token_used.append(word)
        maximum = max(token_w, key=token_w.get)
        minimum = min(token_w, key=token_w.get)
        max_weight_term = token_w[maximum]
        min_weight_term = token_w[minimum]
        normalized_term_weight = {}
        for k,v in token_w.items():
            normalized_term_weight[k] = ( v - min_weight_term) / (max_weight_term - min_weight_term)
        sort_term_weight = sorted(normalized_term_weight.items(), key=lambda x: x[1])
        sorted_term_weight = {}
        count = 0
        for i in sort_term_weight:
            if count < self.X:
                sorted_term_weight[i[0]] = i[1]
            else:
                break
            count+=1
        self.token_weight.append(sorted_term_weight)

    weighted_token = {}
    for used_tok in self.token_used:
        temp = []
        for tok_w in self.token_weight:
            if used_tok in tok_w:
                temp.append(tok_w[used_tok])
        weighted_token[used_tok] = temp

    merged_weighted_token = {}
    for k,v in weighted_token.items():
        if len(v) != 0:
            merged_weighted_token[k] = np.mean(v)

    sorted_merged_weighted_token = sorted(merged_weighted_token.items(), key=lambda x: x[1])
    sorted_final_weight = {}
    count = 0
    l_value = int(len(sorted_merged_weighted_token) * ( self.L / 100))
    for i in sorted_merged_weighted_token:
        if count < l_value:
            sorted_final_weight[i[0]] = i[1]
        else:
            break
        count+=1
```

```
62
63      stopwords = []
64      for k, v in sorted_final_weight.items():
65          stopwords.append(k)
66
67      return stopwords
```

Kode Program 5.15 Create *Stopwords*

Penjelasan Kode Program 5.15:

Baris 1	Mendefinisikan *method create_stopwords* dengan parameter *cleaned_data* dan *terms*
Baris 2-3	Memasukan cleaned_data dan *term*s kedalam variabel kelas ini
Baris 5	Melakukan perulangan index i sepanjang Y
Baris 6	Memanggil fungsi generate_*random*_words dengan parameter *term*s lalu dimasukan ke w_*random*
Baris 7	Memanggil fungsi get_documents_contains_words dengan parameter w_*random* dan cleaned_data lalu dimasukan ke dalam sampled_documents
Baris 9	Memanggil fungsi get_*term* dengan parameter sampled_documents lalu dimasukan ke dalam *term*_sampled_documents.
Baris 11	Mendefinisikan token_w sebagai dict
Baris 12	Melakukan perulangan word didalam *term*_sampled_documents
Baris 13	Memanggil fungsi kl_div dengan parameter word dan sampled_documents lalu dimasukan ke dalam token_w dengan index word
Baris 15	Jika word tidak ada di dalam token_used maka word ditambahkan ke dalam list token_used
Baris 17-20	Mencari nilai bobot maximum dan minimum dari token_w yang di masukkan ke dalam max_weight_*term* dan min_weight_*term*
Baris 21	Mendefinisikan normalized_*term*_weight sebagai dict
Baris 22	Melakukan perulangan k dan v dari item token_w
Baris 23-24	Melakukan perhitungan normalisasi bobot dengan min max dengan cara v dikurangi min_weight_*term* lalu hasilnya dibagi dengan max_weight_*term* dikurangi min_weight_*term* lalu dimasukkan kedalam normalized_*term*_weight
Baris 25	Mengurutkan normalized_*term*_weight dan hasilnya dimasukkan ke dalam sort_*term*_weight
Baris 27	Mendefinisikan sorted_*term*_weight sebagai dict

Baris 28	Mendefinisikan count dan diinisialisasi dengan 0
Baris 29	Melakukan perulangan i dalam sort_*term*_weight
Baris 30 – 31	Jika count kurang dari self.X maka i index 1 dimasukkan kedalam sorted_*term*_weight index i index 0
Baris 32 – 33	Selain itu berhentikan loop
Baris 34	Count diincrement 1
Baris 35	Variabel sorted_*term*_weight dimasukkan ke dalam token_weight
Baris 37	Mendefinisikan weighted_token sebagai dict
Baris 38	Melakukan perulangan used_tok dalam token_used
Baris 39	Mendefinisikan temp sebagai list
Baris 40	Melakukan perulangan tok_w dalam token_weight
Baris 41-42	Jika used_tok ada didalam tok_w maka tok_w index used_tok akan dimasukkan kedalam temp
Baris 43	Temp dimasukkan kedalam weighted_token dengan index used_tok
Baris 45	Mendefinisikan merged_weighted_token sebagai dict
Baris 46	Melakukan perulangan k dan v dalam item weighted_token
Baris 47 – 48	Jika panjang v tidak sama dengan 0 maka v akan dicari rata-rata dan dimasukkan ke dalam merged_weighted_token dengan index k
Baris 50	Mengurutkan merged_weighted_token dan dimasukkan ke dalam sorted_merged_weighted_token.
Baris 52	Mendefinisikan sorted_final_weight sebagai dict
Baris 53	Mendefinisikan count dan diinisialisasi dengan 0
Baris 54	L_value diinisialisasi dengan panjang sorted_merged_weighted_token yang dikalikan dengan L dibagi 100
Baris 56	Melakukan perulangan i dalam sorted_merged_weighted_token
Baris 57 – 58	Jika count kurang dari l_value maka sorted_final_weight indeks i indeks 0 diinisialisasi dengan i indeks 1
Baris 59 – 60	Selain itu maka perulangan diberhentikan
Baris 61	Count diincrement 1
Baris 63	Mendefinisikan *stopword* sebagai list

Baris 64 - 65 Melakukan perulangan k dan v dalam item sorted_final_weight dan setiap perulangannya k ditambahkan ke dalam *stopword*s

Baris 67 Mengembalikan nilai *stopword*s

5.3 Implementasi *Term Weighting*

Pada bagian *term weighting* ini akan meliputi proses *Raw term Frequency, log term Frequency, inverse document Frequency,* dan *term Frequency-inverse document Frequency*. Tahapan *term weighitng* ini diawali dengan pemanggilan *library* yang diperlukan untuk prosesnya seperti pada Kode Program 5.17.

```
Import library
1  Import math
```

Kode Program 5.16 Import Library

Penjelasan Kode Program 5.16:

Baris 1 Digunakan untuk *import library math*

Setelah tahapan *import library*, tahapan selanjutnya adalah tahapan *count word* yang ditunjukkan pada Kode Program 5.17.

```
Count Word
1  def count_word(self,term, document):
2      count = 0
3      for word in document.split():
4          if term == word:
5              count+=1
6      return count
```

Kode Program 5.17 Count Word

Penjelasan Kode Program 5.17:

Baris 1 Mendefisinikan method count_word dengan parameter *term* dan document

Baris 2 Count diinisialisasikan dengan 0

Baris 3 Melakukan perulangan word dalam document yang di split

Baris 4 – 5 Jika *term* sama dengan word maka count ditambah 1

Baris 6 Mengembalikan nilai count

Setelah tahapan *count word*, tahapan selanjutnya adalah tahapan *count dft* yang ditunjukkan pada Kode Program 5.18.

```
Count DFT
1  def count_dft(self,numbers):
2      count = len(self.data)
3      for number in numbers:
4          if number == 0:
5              count-=1
6      return count
```

Kode Program 5.18 Count DFT

Penjelasan Kode Program 5.18:

Baris 1 Mendefisinikan method count_dft dengan parameter numbers

Baris 2 Count diinisialisasikan dengan panjang self.data

Baris 3 Melakukan perulangan number dalam numbers

Baris 4 – 5 Jika number sama dengan 0 maka count dikurang 1

Baris 6 Mengembalikan nilai count

Setelah tahapan *count dft*, tahapan selanjutnya adalah tahapan *Raw tf weighting* yang ditunjukkan pada Kode Program 5.19.

Get *Raw* TF *Weighting*

```
1  def get_Raw_tf_weighting(self):
2      for term in self.terms:
3          temp = []
4          for data in self.data:
5              temp.append(self.count_word(term, data))
6          self.Raw_tf[term] = temp
7      return self.Raw_tf
```

Kode Program 5.19 Get *Raw* TF *Weighting*

Penjelasan Kode Program 5.19:

Baris 1 Mendefinisikan method get_*Raw*_tf_*weighting*

Baris 2 Melakukan perulangan *term* dalam self.*term*s

Baris 3 *Term* didefinisikan sebagai list

Baris 4 Melakukan perulangan data dalam self.data

Baris 5 Variabel temp ditambahkan dari hasil pemanggilan method count_word dengan parameter *term* dan data

Baris 6 Nilai temp dimasukkan ke dalam self.*Raw*_tf index *term*

Baris 7 Mengembalikan nilai self.*Raw*_tf

Setelah tahapan *Raw tf weighting*, tahapan selanjutnya adalah tahapan *log tf weighting* yang ditunjukkan pada Kode Program 5.20.

Get Log TF *Weighting*

```
1  def get_log_tf_weighting(self):
2      self.get_Raw_tf_weighting()
3      for term in self.terms:
4          temp = []
5          for i in range(len(self.Raw_tf[term])):
6              tf = 0 if (self.Raw_tf[term])[i] == 0 else 1+math.
7  log((self.Raw_tf[term])[i],10)
8              temp.append(tf)
9          self.log_tf[term] = temp
10     return self.log_tf
```

Kode Program 5.20 Get Log TF *Weighting*

Penjelasan Kode Program 5.20:

Baris 1 Mendefinisikan method get_log_tf_*weighting*

Baris 2 Memanggil method get_*Raw*_tf_*weighting*

Baris 3 Melakukan perulangan *term* dalam self.*term*s

Baris 4 Mendefinisikan temp sebagai dict

Baris 5 Melakukan perulangan i hingga panjang *Raw*_tf index *term*

Baris 6 – 7 Menghitung nilai tf akan 0 jika *Raw*_tf index *term* index i sama dengan 0, selain itu nilai tf akan diisi dengan perhitungan 1 ditambah log 10 dari nilai *Raw*_tf *term* index i.

Baris 8 Nilai tf dimasukkan kedalam temp

Baris 9 Nilai temp dimasukkan kedalam self.log_tf

Baris 10 Mengembalikan nilai self.log_tf

Setelah tahapan log tf *weighting*, tahapan selanjutnya adalah tahapan *calculate idf* yang ditunjukkan pada Kode Program 5.21

Calculate IDF

```
1  def calculate_idf(self):
2      for term in self.terms:
3          df = self.count_dft(self.Raw_tf[term])
4          idf_value = math.log(len(self.data)/df,10)
5          self.idf.append(idf_value)
6      return self.idf
```

Kode Program 5.21 Calculate IDF

Penjelasan Kode Program 5.21:

Baris 1 Mendefinisikan method calculate_idf

Baris 2 Melakukan perulangan *term* dalam self.*term*s

Baris 3 Menghitung df dengan memanggil method count_dft dengan parameter self.*Raw*_tf index *term*

Baris 4 Menghitung idf value dengan menghitung log 10 dari hasil pembagian panjang self.data dan df

Baris 5 Nilai idf value dimasukkan kedalam self.idf

Baris 6 Mengembalikan nilai self.idf

Setelah tahapan calculate idf, tahapan selanjutnya adalah tahapan *get idf* yang ditunjukkan pada Kode Program 5.22

Get IDF

```
1  def get_idf(self):
2      return self.idf
```

Kode Program 5.22 Get IDF

Penjelasan Kode Program 5.22:

Baris 1 Mendefinisikan method get_idf

Baris 2 Mengembalikan nilai self.idf

Setelah tahapan get idf, tahapan selanjutnya adalah tahapan *get tf idf weighting* yang ditunjukkan pada Kode Program 5.23

```
Get TF IDF Weighting
1   def get_tf_idf_weighting(self):
2       self.get_log_tf_weighting()
3       self.calculate_idf()
4       count = 0
5       for term in self.terms:
6           temp = []
7           for i in range(len(self.data)):
8               tfidf_value = self.log_tf[term][i]*self.idf[count]
9               temp.append(tfidf_value)
10          self.tf_idf[term] = temp
11          count+=1
12      return self.tf_idf
```

Kode Program 5.23 Get TF IDF *Weighting*

Penjelasan Kode Program 5.23:

Baris 1 Mendefinisikan method get_tf_idf_*weighting*

Baris 2 Memanggil method get_log_tf_*weighting*

Baris 3 Memanggil method calculate_idf

Baris 4 Count diinisialisasikan dengan 0

Baris 5 Melakukan perulangan *term* dalam self.*term*s

Baris 6 Mendefinisikan *term* sebagai list

Baris 7 Melakukan perulangan i hingga sepanjang self.data

Baris 8 Menghitung tfidf value dengan perkalian self.log_tf index *term* index i dengan self.idf index count

Baris 9 Niai tfidf value dimasukkan kedalam temp

Baris 10 Nilai temp dimasukkan kedalam self.tf_idf index *term*

Baris 11 Nilai count ditambahkan 1

Baris 12 Mengembalikan nilai self.tf_idf

5.4 Implementasi *Naïve Bayes*

Pada bagian *Naïve Bayes* ini akan dibagi menjadi 2 bagian yaitu bagian training atau pelatihan dan bagian testing atau pengujian. Sebelum masuk ke dalam bagian training maupun testing. Tahapan *Naïve Bayes* ini diawali dengan beberapa fungsi yang diperlukan dalam perhitungan pelatihan maupun pengujian. Tahapan awal tahap ini diawali dengan pemanggilan *library* dan kelas yang diperlukan seperti yang ditunjukkan pada Kode Program 5.24.

```
Import library dan kelas
1   import re
2   from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
```

```
3 from preprocessing import Preprocessing
4 from weighting import Weighting
```

Kode Program 5.24 Import Library

Penjelasan Kode Program 5.24:

Baris 1 Digunakan untuk *import library math*

Baris 2 Digunakan untuk *import library StemmerFactory* dari Sastrawi

Baris 3 Digunakan untuk *import* kelas *Preprocessing* dari *file preprocessing*

Baris 4 Digunakan untuk *import* kelas *Weighting* dari *file weighting*

Setelah tahapan *import library* dan kelas, tahapan selanjutnya adalah tahapan *count word* yang ditunjukkan pada Kode Program 5.25.

```
Count Word
1 def count_word(self,term,document):
2     count = 0
3     for word in document.split():
4         if term == word:
5             count += 1
6     return count
```

Kode Program 5.25 Count Word

Penjelasan Kode Program 5.25:

Baris 1 Mendefinisikan method count_word dengan parameter *term* dan document

Baris 2 Count diinisialisasi dengan 0

Baris 3 Melakukan perulangan word dalam document yang di split

Baris 4 Jika *term* sama dengan word maka count ditambah 1

Baris 5 Mengembalikan nilai count

Setelah tahapan *count word* dan kelas, tahapan selanjutnya adalah tahapan *count specific word in category* yang ditunjukkan pada Kode Program 5.26.

```
Count Specific Word in Category
1 def count_specific_word_in_category(self,word,category):
2     wct = 0
3     indexDocument = 0
4     for wt in self.weighted_terms[word]:
5         if self.target[indexDocument]==category:
6             wct = wct + wt
7         indexDocument += 1
8     return wct
```

Kode Program 5.26 Count Specific Word in Category

Penjelasan Kode Program 5.26:

Baris 1 Mendefinisikan method count_specific_word_in_category dengan parameter word dan category

Baris 2 – 3 Wct dan indexDocument diinisialisasi dengan 0

Baris 4 Melakukan perulangan wt dalam self.weighted_*term*s index word

Baris 5 - 6 Jika self.target index indexDocument sama dengan category maka wct ditambah wt

Baris 7 Nilai indexDocument ditambah 1

Baris 8 Mengembalikan nilai wct

Setelah tahapan *count specific word in category*, tahapan selanjutnya adalah tahapan *count all word in category* yang ditunjukkan pada Kode Program 5.27.

Count All Word in Category

```
1  def count_all_word_in_category(self,category):
2      counter = 0
3      indexDocument = 0
4      for totalTiapDokumen in self.total:
5          if self.target[indexDocument]==category:
6              counter = counter + totalTiapDokumen
7          indexDocument += 1
8      return counter
```

Kode Program 5.27 Count All Word in Category

Penjelasan Kode Program 5.27:

Baris 1 Mendefinisikan method count_all_word_in_category dengan parameter category

Baris 2 – 3 Counter dan indexDocument diinisialisasi dengan 0

Baris 4 Melakukan perulangan totalTiapDokumen dalam self.total

Baris 5 - 6 Jika self.target index indexDocument sama dengan category maka counter ditambah totalTiapDokumen

Baris 7 Nilai indexDocument ditambah 1

Baris 8 Mengembalikan nilai counter

Setelah tahapan *count all word in category*, tahapan selanjutnya adalah tahapan *count all word in category* yang ditunjukkan pada Kode Program 5.28.

Get Total IDF

```
1  def get_total_idf(self):
2      idf_total = 0
3      for idf_item in self.idf:
4          idf_total+=idf_item
5      return idf_total
```

Kode Program 5.28 Get Total IDF

Penjelasan Kode Program 5.28:

Baris 1 Mendefinisikan method get_total_idf

Baris 2 Idf_total diinisialisasi dengan 0

Baris 3 Melakukan perulangan idf_item dalam self.idf

Baris 4 Idf_total ditambah dengan idf_iem

Baris 5 Mengembalikan idf_total

Setelah tahapan *get total idf,* tahapan selanjutnya adalah tahapan *calculate probability multinomial* yang ditunjukkan pada Kode Program 5.29.

Calcaulte Probability Multinomial

```
1 def calculate_probability_multinomial(self,word, category):
2     return (self.count_specific_word_in_category(word, category
3 ) + 1) / (self.count_all_word_in_category(category) + self.get_
4 total_idf())
```

Kode Program 5.29 Calculate Probability Multinomial

Penjelasan Kode Program 5.29:

Baris 1 Mendefinisikan method calculate_probability_multinomial dengan parameter word dan category

Baris 2 Mengembalikan nilai perhitungan dari pemanggilan fungsi count_specific_word_in_category dengan parameter word dan category ditambah 1 dan dibagi dengan hasil pertambahan dari pemanggilan fungsi count_all_word_in_category parameter category dengan get_total_idf

Setelah tahapan *calculate probability multinomial*, tahapan selanjutnya adalah tahapan *get total document* yang ditunjukkan pada Kode Program 5.30.

Get Total Document

```
1 def get_total_document(self):
2     return len(self.cleaned_data)
```

Kode Program 5.30 Get Total Document

Penjelasan Kode Program 5.30:

Baris 1 Mendefinisikan method get_total_document

Baris 2 Mengembalikan nilai panjang dari self.cleaned_data

Setelah tahapan *get total document*, tahapan selanjutnya adalah tahapan *get total document with specific category* yang ditunjukkan pada Kode Program 5.31.

Get Total Document With Specific Category

```
1 def get_total_document_with_specific_category(self,category):
2     return len([tgt for tgt in self.target if tgt == category])
```

Kode Program 5.31 Get Total Document With Specific Category

Penjelasan Kode Program 5.31:

Baris 1 Mendefinisikan method calculate_probability_multinomial dengan parameter word dan category

Baris 2 Mengembalikan nilai perhitungan dari pemanggilan fungsi

count_specific_word_in_category dengan parameter word dan category ditambah 1 dan dibagi dengan hasil pertambahan dari pemanggilan fungsi count_all_word_in_category parameter category dengan get_total_idf

Setelah tahapan *get total document with specific category*, tahapan selanjutnya adalah tahapan pelatihan dan pengujian.

5.4.1 Implementasi *Naïve Bayes* Training

Pada bagian *Naïve Bayes* Training ini akan dijelaskan tahapan-tahapan pelatihan *Naïve Bayes* yaitu mencari *prior* serta likelihood untuk tiap kelas menggunakan method-method yang sudah ditunjukkan sebelumnya. Tahapan ini akan ditunjukkan pada Kode Program 5.32.

Fit

```
def fit(self, cleaned_data, terms, target, stopwords, idf, wei
ght = None):
    self.cleaned_data = cleaned_data
    self.terms = terms
    self.target = target
    if weight == None:
        weighting = Weighting(self.cleaned_data, self.terms)
        self.weighted_terms = weighting.get_tf_idf_weighting()
    else:
        self.weighted_terms = weight
        self.idf = idf
    self.stopwords = stopwords

    for i in range(len(self.cleaned_data)):
        total_word = 0
        for term in self.terms:
            total_word += self.weighted_terms[term][i]
        self.total.append(total_word)

    for term in self.terms:
        self.con_prob_negative.append(self.calculate_probabili
ty_multinomial(term, 'Negatif'))
        self.con_prob_neutral.append(self.calculate_probabilit
y_multinomial(term, 'Netral'))
        self.con_prob_positive.append(self.calculate_probabili
ty_multinomial(term, 'Positif'))

    self.likelihood = {}
    indexKomentar = 0
    for term in self.terms:
        temp = []
        temp.append(self.con_prob_negative[indexKomentar])
        temp.append(self.con_prob_neutral[indexKomentar])
        temp.append(self.con_prob_positive[indexKomentar])
        self.likelihood[term] = temp
        indexKomentar += 1

    self.prior_negative = self.get_total_document_with_specifi
c_category(
        'Negatif') / self.get_total_document()
    self.prior_neutral = self.get_total_document_with_specific
```

```
42  _category(
43          'Netral') / self.get_total_document()
44      self.prior_positive = self.get_total_document_with_specifi
45  c_category(
46          'Positif') / self.get_total_document()
```

Kode Program 5.32 Naive *Bayes* Training

Penjelasan Kode Program 5.32:

Baris 1 – 2 Mendefinisikan method *fit* dengan parameter cleaned_data, *term*s, target, *stopword*s, idf, dan weight yang memiliki default value None

Baris 3 – 5 Memasukan cleaned_data, *term*s, dan target ke dalam variabel tersebut didalam kelas ini

Baris 6 - 8 Jika weight sama dengan None maka dilakukan proses perhitungan *weighting* dengan memanggil kelas *Weighting* dan memanggil fungsi get_tf_idf_*weighting* dan memasukkannya ke dalam variabel self.weighted_*term*s

Baris 9 - 11 Selain itu weight dimasukkan ke dalam self.weighted_*term*s dan idf dimasukkan ke dalam self.idf

Baris 12 *Stopword*s dimasukkan kedalam self.*stopword*s

Baris 14 Melakukan perulangan i hingga sepanjang cleaned_data

Baris 15 Total_word diinisialisasi dengan 0

Baris 16 Melakukan perulangan *term* dalam self.*term*s

Baris 17 Total_word ditambah dengan self.weighted_*term*s index *term* index i

Baris 18 Total_word dimasukkan kedalam self.total

Baris 20 Melakukan perulangan *term* dalam self.*term*s

Baris 21 - 26 Memanggil calculate_probability_multinomial dengan parameter *term* dan setiap kategorinya dan dimasukkan masing-masing kedalam self.con_prob_negative, self.con_prob_netral, self.con_prob_positive

Baris 28 Mendefinisikan likelihood sebagai dict

Baris 29 indexKomentar diinisialisasi dengan 0

Baris 30 Melakukan perulangan *term* dalam self.*term*s

Baris 31 Temp didefinisikan sebagai list

Baris 32 - 34 Temp dimasukkan self.con_prob_negative, self.con_prob_netral, self.con_prob_positive masing-masing index indexKomentar

Baris 35 - 36 Temp dimasukkan kedalam self.likelihood index *term* dan

indexKomentar ditambah 1

Baris 38 - 46 Menghitung *prior* setiap kelas dengan memanggil fungsi get_total_document_with_specific_category dengan parameter kelas yang dibagi dengan get_total_document

5.4.2 Implementasi *Naïve Bayes* Testing

Pada bagian *Naïve Bayes* Testing ini akan dijelaskan tahapan-tahapan pelatihan *Naïve Bayes* yaitu mencari *posterior* untuk tiap kelas menggunakan method-method yang sudah ditunjukkan sebelumnya. Tahapan ini akan ditunjukkan pada Kode Program 5.33.

Predict

```
1   def predict(self,data_test,expected_result):
2       self.used_terms = []
3       prepro = Preprocessing()
4       cleaned_data_test, terms_test = prepro.preprocessing([data
5   _test],self.stopwords)
6       terms_test = prepro.get_token()
7       for term in terms_test:
8           if term in self.terms:
9               self.used_terms.append(term)
10
11      for term in self.used_terms:
12          temp = []
13          temp.append(self.likelihood[term][0])
14          temp.append(self.likelihood[term][1])
15          temp.append(self.likelihood[term][2])
16          self.used_terms_with_likelihood[term] = temp
17
18      negatif = 1
19      netral = 1
20      positif = 1
21      for term in self.used_terms:
22          negatif *= self.used_terms_with_likelihood[term][0]
23          netral *= self.used_terms_with_likelihood[term][1]
24          positif *= self.used_terms_with_likelihood[term][2]
25
26      negatif = negatif * self.prior_negative
27      netral = netral * self.prior_neutral
28      positif = positif * self.prior_positive
29      finalResult = ""
30      if (positif > negatif and positif > netral):
31          finalResult = "Positif"
32      elif negatif > positif and negatif > netral:
33          finalResult = "Negatif"
34      elif netral > positif and netral > negatif:
35          finalResult = "Netral"
36
37      return finalResult
```

Kode Program 5.33 Naive *Bayes* Testing

Penjelasan Kode Program 5.33:

Baris 1	Mendefinisikan method *predict* dengan parameter data_test dan expected_result
Baris 2	Mendefinisikan self.used_*term*s sebagai list
Baris 3 - 5	Membuat objek dari kelas *Preprocessing* dan memanggil method *preprocessing* dengan parameter data_test dan self.*stopword*s dan dimasukkan ke dalam cleaned_data_test dan *term*s_test
Baris 6	*Term*s_test dimasukkan dengan get_token dari objek prepro
Baris 7	Melakukan perulangan *term* dalam self.*term*s
Baris 8 - 9	Jika *term* ada didalam self.*term*s maka *term* dimasukkan kedalam self.used_*term*
Baris 11	Melakukan perulangan *term* dalam self.used_*term*s
Baris 12	Mendefinisikan temp sebagai list
Baris 13	Temp dimasukkan self.likelihood index *term* index 0 hingga 2
Baris 16	Temp dimasukkan ke self.used_*term*s_with_likelihood index *term*
Baris 18 - 20	Negatif, netral, positif diinisialisasi dengan 0
Baris 21	Melakukan perulangan *term* dalam self.used_*term*s
Baris 22	Negatif dikali sama dengan self.used_*term*s_with_likelihood index *term* index 0
Baris 23	Netral dikali sama dengan self.used_*term*s_with_likelihood index *term* index 1
Baris 24	Positif dikali sama dengan self.used_*term*s_with_likelihood index *term* index 2
Baris 26-28	Negatif, netral, dan positif masing-masing dikali dengan *prior* yang sudah dihitung pada training
Baris 29	Mendefinisikan finalResult sebagai String
Baris 30	Jika positif lebih dari negatif dan netral maka finalResult diinisialisasi dengan Positif, selain itu jika negatif lebih dari positif dan netral maka finalResult diinisialisasi dengan Netral, dan jika netral lebih dari positif dan negatif maka finalResult diinisialisasi dengan Netral
Baris 37	Mengembalikan nilai finalResult

5.5 Implementasi K Fold

Pada bagian implementasi k-fold cross validation ini akan diawali dengan tahapan persiapan data yang diperlukan seperti yang ditunjukkan pada Kode Program 5.34.

Prepare Data

```
1   def _prepare_data(self):
2       self.data_negative = []
3       self.data_netral = []
4       self.data_positive = []
5
6       for i in range(len(self.data)):
7           if self.target[i] == self.NEGATIVE:
8               self.data_negative.append(self.data[i])
9           elif self.target[i] == self.NETRAL:
10              self.data_netral.append(self.data[i])
11          elif self.target[i] == self.POSITIVE:
12              self.data_positive.append(self.data[i])
13          else:
14              return None
```

Kode Program 5.34 Persiapan Data

Penjelasan Kode Program 5.34:

Baris 1 Mendefinisikan method prepare_data bersifat private

Baris 2 – 4 Mendefinisikan data_negative, data_netral, dan data_positive sebagai list

Baris 6 - 14 Melakukan perulangan i hingga sepanjang data dan dilakukan seleksi data berdasarkan target dan dimasukkan kedalam variabel terkait.

Setelah tahapan persiapan data, tahapan selanjutnya adalah tahapan *get data sequence* yang ditunjukkan pada Kode Program 5.35.

Get Data Sequence

```
1   def get_data_sequence(self):
2       data_test_size = len(self.data_negative) / self.fold
3       self.test_size = len(self.data_negative) * self.test_siz
4   e
5       data_test_start_index = 0
6       data_test_end_index = self.test_size
7
8       data_train = []
9       data_test = []
10      for i in range(self.fold):
11          data_neg_test = []
12          data_net_test = []
13          data_pos_test = []
14          data_neg_train = []
15          data_net_train = []
16          data_pos_train = []
17
18          check = False
19
20          for j in range(len(self.data_negative)):
```

```
            if j >= data_test_start_index and j < data_test_
end_index:
                temp = []
                temp.append(self.data_negative[j])
                temp.append(self.NEGATIVE)
                data_neg_test.append(temp)
                temp = []
                temp.append(self.data_netral[j])
                temp.append(self.NETRAL)
                data_net_test.append(temp)
                temp = []
                temp.append(self.data_positive[j])
                temp.append(self.POSITIVE)
                data_pos_test.append(temp)
                if j == 99 and data_test_start_index == 90:
                    for k in range(10):
                        temp = []
                        temp.append(self.data_negative[k])
                        temp.append(self.NEGATIVE)
                        data_neg_test.append(temp)
                        temp = []
                        temp.append(self.data_netral[k])
                        temp.append(self.NETRAL)
                        data_net_test.append(temp)
                        temp = []
                        temp.append(self.data_positive[k])
                        temp.append(self.POSITIVE)
                        data_pos_test.append(temp)
            else:
                if i!=9:
                    temp = []
                    temp.append(self.data_negative[j])
                    temp.append(self.NEGATIVE)
                    data_neg_train.append(temp)
                    temp = []
                    temp.append(self.data_netral[j])
                    temp.append(self.NETRAL)
                    data_net_train.append(temp)
                    temp = []
                    temp.append(self.data_positive[j])
                    temp.append(self.POSITIVE)
                    data_pos_train.append(temp)
                else:
                    if j > 9:
                        temp = []
                        temp.append(self.data_negative[j])
                        temp.append(self.NEGATIVE)
                        data_neg_train.append(temp)
                        temp = []
                        temp.append(self.data_netral[j])
                        temp.append(self.NETRAL)
                        data_net_train.append(temp)
                        temp = []
                        temp.append(self.data_positive[j])
                        temp.append(self.POSITIVE)
                        data_pos_train.append(temp)

        data_combine_test = data_neg_test + data_net_test +
data_pos_test
```

```
80          data_combine_train = data_neg_train + data_net_train
81   + data_pos_train
82
83          data_combine_test_tweet = [data[0] for data in data_
84  combine_test]
85          data_combine_test_target = [data[1] for data in data
86  _combine_test]
87
88          data_combine_train_tweet = [data[0] for data in data
89  _combine_train]
90          data_combine_train_target = [data[1]
91                                       for data in data_com
92  bine_train]
93
94          data_dict_test = {}
95          data_dict_train = {}
96          data_dict_test["tweet"] = data_combine_test_tweet
97          data_dict_test["target"] = data_combine_test_target
98
99          data_dict_train["tweet"] = data_combine_train_tweet
100         data_dict_train["target"] = data_combine_train_targe
101 t
102
103         data_train.append(data_dict_train)
104         data_test.append(data_dict_test)
105         data_test_start_index += data_test_size
106         data_test_end_index += data_test_size
107
108     return data_train, data_test
```

Kode Program 5.35 Get Data Sequence

Penjelasan Kode Program 5.35:

Baris 1	Mendefinisikan method get_data_sequence
Baris 2 – 6	Menghitung ukuran data test, dan menentukan index awal serta akhir index data test
Baris 8 – 9	Mendefinisikan data_train dan data_test sebagai list
Baris 10	Melakukan perulangan i hingga sepanjang fold
Baris 11 – 18	Mendefinisikan variabel data test dan data train tiap kelas, check didefinisikan false
Baris 20	Melakukan perulangan j hingga sepanjang data_negative
Baris 21 – 34	Melakukan seleksi jika index j berada didalam jangka start dan end index maka akan dimasukkan kedalam data_test masing-masing kelasnya
Baris 35 - 48	Jika index j sama dengan 99 dan start 90 melakukan perulangan k hingga 10 untuk memasukkan ke dalam data_test masing-masing kelasnya
Baris 49 - 76	Jika tidak, jika i tidak sama dengan 9 maka akan dimasukkan kedalam data_train masing-masing kelasnya, selain itu jika j lebih dari 9 maka akan dimasukkan kedalam data_train masing-

masing kelasnya.

Baris 78 - 81	Menggabungkan data_test tiap kelas, dan data_train tiap kelas
Baris 83 - 92	Memisahkan tweet dengan target dari data test dan data train yang sudah di combine
Baris 94 - 101	Membuat dictionary test dan train dengan kata key tweet untuk data_combine_test_tweet dan key target untuk data_combine_test_target dan dictionary train dengan kata key tweet untuk data_combine_train_tweet dan key target untuk data_combine_train_target
Baris 103 – 104	Dictionary train dan test dimasukkan kedalam data_train dan data_test
Baris 105 - 106	Start index dan end index ditambahkan data_test_size
Baris 108	Mengembalikan data_train dan data_test

5.6 Implementasi *Confusion Matrix*

Pada bagian implementasi *confusion matrix* ini akan diawali dengan tahapan *create confusion matrix* seperti yang ditunjukkan pada Kode Program 5.36.

Create Confusion Matrix

```
1   def create_confusion_matrix(self,actual,predicted):
2       self.cm = pd.DataFrame(np.zeros((3, 3), dtype=int), index
3   =['Actually Negatif', 'Actually Netral', 'Actually Positif'],
4    columns=[
5                               'Predicted Negatif', 'Predicted N
6   etral', 'Predicted Positif'])
7       for i in range(len(actual)):
8           if actual[i] == self.NEGATIVE:
9               if predicted[i] == self.NEGATIVE:
10                  self.cm.loc["Actually " + self.NEGATIVE,
11                              "Predicted " + self.NEGATIVE] +=
12  1
13              elif predicted[i] == self.NETRAL:
14                  self.cm.loc["Actually " + self.NEGATIVE,
15                              "Predicted " + self.NETRAL] += 1
16              elif predicted[i] == self.POSITIVE:
17                  self.cm.loc["Actually " + self.NEGATIVE,
18                              "Predicted " + self.POSITIVE] +=
19  1
20          elif actual[i] == self.NETRAL:
21              if predicted[i] == self.NEGATIVE:
22                  self.cm.loc["Actually " + self.NETRAL,
23                              "Predicted " + self.NEGATIVE] +=
24  1
25              elif predicted[i] == self.NETRAL:
26                  self.cm.loc["Actually " + self.NETRAL,
27                              "Predicted " + self.NETRAL] += 1
28              elif predicted[i] == self.POSITIVE:
29                  self.cm.loc["Actually " + self.NETRAL,
30                              "Predicted " + self.POSITIVE] +=
```

```
1
        elif actual[i] == self.POSITIVE:
            if predicted[i] == self.NEGATIVE:
                self.cm.loc["Actually " + self.POSITIVE,
                            "Predicted " + self.NEGATIVE] +=
1
            elif predicted[i] == self.NETRAL:
                self.cm.loc["Actually " + self.POSITIVE,
                            "Predicted " + self.NETRAL] += 1
            elif predicted[i] == self.POSITIVE:
                self.cm.loc["Actually " + self.POSITIVE,
                            "Predicted " + self.POSITIVE] +=
1
```

Kode Program 5.36 *Create Confusion Matrix*

Penjelasan Kode Program 5.36:

Baris 1 Mendefinisikan method create_*confusion_matrix* dengan parameter actual dan predicted

Baris 2 – 6 Membuat DataFrame *confusion matrix* dengan index sebagai actual dan kolom sebagai predicted tiap kelas dan diinisialisasi dengan 0

Baris 7 – 8 Melakukan perulangan i sepanjang actual, jika actual index i negatif

Baris 9 - 12 Jika predicted negatif maka dataframe dengan index actual negatif dan predicted negatif ditambah 1

Baris 13 – 15 Jika predicted netral maka dataframe dengan index actual negatif dan predicted netral ditambah 1

Baris 16 – 19 Jika predicted positif maka dataframe dengan index actual negatif dan predicted positif ditambah 1

Baris 20 Selain itu jika actual index i sama dengan netral

Baris 21 – 24 Jika predicted negatif maka dataframe dengan index actual netral dan predicted negatif ditambah 1

Baris 25 – 27 Jika predicted netral maka dataframe dengan index actual netral dan predicted netral ditambah 1

Baris 28 – 31 Jika predicted positif maka dataframe dengan index actual netral dan predicted positif ditambah 1

Baris 32 Selain itu jika actual index i sama dengan positif

Baris 33 – 36 Jika predicted negatif maka dataframe dengan index actual positif dan predicted negatif ditambah 1

Baris 37 – 39 Jika predicted netral maka dataframe dengan index actual positif dan predicted netral ditambah 1

Baris 40 - 43 Jika predicted positif maka dataframe dengan index actual positif dan predicted positif ditambah 1

Setelah tahapan *create confusion matrix*, tahapan selanjutnya adalah tahapan mencari tp, tn, fp, dan fn yang ditunjukkan pada Kode Program 5.37.

Find TP FN FP TN

```
1   def find_tp_fn_fp_tn(self):
2       self.tp_negatif = self.cm.loc["Actually " +
3                                     self.NEGATIVE, "Predicte
4   d " + self.NEGATIVE]
5       self.tp_netral = self.cm.loc["Actually " +
6                                     self.NETRAL, "Predicted
7   " + self.NETRAL]
8       self.tp_positif = self.cm.loc["Actually " +
9                                     self.POSITIVE, "Predicte
10  d " + self.POSITIVE]
11
12      temp = self.cm.copy()
13      temp.loc["Actually " + self.NEGATIVE, "Predicted " + sel
14  f.NEGATIVE] = 0
15      self.fn_negatif = sum(temp.loc["Actually " + self.NEGATI
16  VE, :])
17      self.fp_negatif = sum(temp.loc[:, "Predicted " + self.NE
18  GATIVE])
19
20      temp = self.cm.copy()
21      temp.loc["Actually " + self.NETRAL, "Predicted " + self.
22  NETRAL] = 0
23      self.fn_netral = sum(temp.loc["Actually " + self.NETRAL,
24   :])
25      self.fp_netral = sum(temp.loc[:, "Predicted " + self.NET
26  RAL])
27
28      temp = self.cm.copy()
29      temp.loc["Actually " + self.POSITIVE, "Predicted " + sel
30  f.POSITIVE] = 0
31      self.fn_positif = sum(temp.loc["Actually " + self.POSITI
32  VE, :])
33      self.fp_positif = sum(temp.loc[:, "Predicted " + self.PO
34  SITIVE])
35
36      temp = self.cm.copy()
37      temp = temp.drop("Actually " + self.NEGATIVE, axis = 0).
38  drop("Predicted " + self.NEGATIVE, axis = 1)
39      self.tn_negatif = sum(temp.sum())
40
41      temp = self.cm.copy()
42      temp = temp.drop("Actually " + self.NETRAL, axis = 0).dr
43  op("Predicted " + self.NETRAL, axis = 1)
44      self.tn_netral = sum(temp.sum())
45
46      temp = self.cm.copy()
47      temp = temp.drop("Actually " + self.POSITIVE, axis = 0).
48  drop("Predicted " + self.POSITIVE, axis = 1)
49      self.tn_positif = sum(temp.sum())
```

Kode Program 5.37 Find TP, FP, FN, TN

Penjelasan Kode Program 5.37:

Baris 1 Mendefinisikan method find_tp_fn_fp_tn

Baris 2 – 10 Menghitung tp tiap kelas dengan mengambil *confusion matrix* dengan index actually kelas tersebut dan predicted kelas tersebut

Baris 12 - 34 Menghitung fn dan fp tiap kelas dengan diawali mengubah nilai tp menjadi 0 lalu untutk fn dilakukan penjumlahan actually kelas tersebut dengan semua kolom, lalu untuk fp dilakukan penjumlahan semua index dengan hanya kolom predicted kelas tersebut.

Baris 36 - 49 Menghitung tn tiap kelas dengan diawali menghapus index actually kelas tersebut dan menghapus kolom predicted kelas tersebut lalu sisa *confusion matrix* yang ada dijumlahkan

Setelah tahapan mencari tp, fn, fp, dan tn, tahapan selanjutnya adalah tahapan mencari akurasi yang ditunjukkan pada Kode Program 5.38.

Get Accuracy

```
1  def get_accuracy(self):
2      accuracy_each_class = []
3      accuracy_each_class.append((self.tn_negatif + self.tp_neg
4  atif)/(self.tn_negatif + self.tp_negatif+ self.fn_negatif + s
5  elf.fp_negatif))
6      accuracy_each_class.append((self.tn_netral + self.tp_netr
7  al)/(self.tn_netral + self.tp_netral+ self.fn_netral + self.f
8  p_netral))
9      accuracy_each_class.append((self.tn_positif + self.tp_pos
10 itif)/(self.tn_positif + self.tp_positif+ self.fn_positif + s
11 elf.fp_positif))
12     return np.mean(accuracy_each_class)
```

Kode Program 5.38 *Get Accuracy Each Class*

Penjelasan Kode Program 5.38:

Baris 1 Mendefinisikan method get_accuracy

Baris 2 – 11 Menghitung hasil akurasi setiap kelas dengan cara menambahkan tn kelas tersebut dan tp kelas tersebut lalu hasilnya dibagi dengan hasil penjumlahan dari tp, fn, fp, dan tn kelas tersebut

Baris 12 Mengembalikan nilai rata-rata dari accuracy_each_class

Setelah tahapan mencari akurasi tiap kelas, tahapan selanjutnya adalah tahapan mencari *precision* yang ditunjukkan pada Kode Program 5.39.

Get Precision

```
1  def get_precision(self):
2      precision_each_class = []
3      precision_each_class.append((self.tp_negatif)/(self.tp_ne
4  gatif+ self.fp_negatif))
5      precision_each_class.append((self.tp_netral)/(self.tp_net
6  ral+ self.fp_netral))
7      precision_each_class.append((self.tp_positif)/(self.tp_po
8  sitif+ self.fp_positif))
9      return np.mean(precision_each_class)
```

Kode Program 5.39 *Get Precision*

Penjelasan Kode Program 5.39:

Baris 1 Mendefinisikan method get_precision

Baris 2 – 8 Menghitung hasil *precision* setiap kelas dengan cara tp kelas tersebut dibagi dengan hasil penjumlahan dari tp dan fp kelas tersebut

Baris 9 Mengembalikan nilai rata-rata dari precision_each_class

Setelah tahapan mencari *precision*, tahapan selanjutnya adalah tahapan mencari *recall* yang ditunjukkan pada Kode Program 5.40.

```
Get Recall
1  def get_recall_each_class(self):
2      recall_each_class = []
3      recall_each_class.append((self.tp_negatif)/(self.tp_negat
4  if+ self.fn_negatif))
5      recall_each_class.append((self.tp_netral)/(self.tp_netral
6  + self.fn_netral))
7      recall_each_class.append((self.tp_positif)/(self.tp_posit
8  if+ self.fn_positif))
9      return np.mean(recall_each_class)
```

Kode Program 5.40 *Get Recall*

Penjelasan Kode Program 5.40:

Baris 1 Mendefinisikan method get_recall

Baris 2 – 8 Menghitung hasil *recall* setiap kelas dengan cara tp kelas tersebut dibagi dengan hasil penjumlahan dari tp dan fn kelas tersebut

Baris 9 Mengembalikan nilai rata-rata dari recall_each_class

Setelah tahapan mencari *recall*, tahapan selanjutnya adalah tahapan mencari *fmeasure* yang ditunjukkan pada Kode Program 5.42.

```
Get F-Measure
1  def get_fmeasure(self):
2      precision = self.get_precision()
3      recall = self.get_recall()
4      fmeasure = (2 * precision * recall)/(precision+recall)
5      return fmeasure
```

Kode Program 5.41 *Get F-Measure*

Penjelasan Kode Program 5.42:

Baris 1 Mendefinisikan method get_fmeasure

Baris 2 – 3 Memanggil method get_precision dan get_recall dan memasukannya kedalam variabel

Baris 4 Menghitung fmeasure dengan cara hasil perkalian 2, precision, dan recall dibagi dengan hasil penjumlahan dari precision ditambah recall

Baris 5 Mengembalikan fmeasure

Setelah tahapan mencari *fmeasure*, tahapan selanjutnya adalah tahapan membuat method *score* yang berfungsi untuk memanggil method-method yang sudah dibaut sebelumnya yang ditunjukkan pada Kode Program 5.43.

```
Score
1  def score(self, actual, predicted):
2      self.actual = actual
3      self.create_confusion_matrix(actual,predicted)
4      self.find_tp_fn_fp_tn()
5      return self.get_accuracy(),self.get_precision(),self.get_
6  recall(),self.get_fmeasure()
```

Kode Program 5.42 *Score*

Penjelasan Kode Program 5.43:

Baris 1 Mendefinisikan method score dengan parameter actual dan predicted

Baris 2 Actual dimasukkn kedalam ke variabel actual dalam kelas

Baris 3 – 4 Memanggil method create_*confusion_matrix* dengan parameter actual dan predicted, dan memanggil method find_tp_tn_fp_tn

Baris 5 - 6 Mengembalikan nilai dengan memanggil method get_accuracy, get_precision, get_recall, get_fmeasure.

5.7 Implementasi Main

Pada bagian implementasi main ini akan dilakukan pemanggilan metode dan kelas-kelas yang sudah dibuat sebelumnya dan tahap ini akan diawali dengan *import library* dan kelas yang sudah dibuat sebelumnya yang ditunjukkan pada Kode Program 5.44.

```
Import library dan kelas
1  import pandas as pd
2  from tbrs import TermBasedRandomSampling
3  from preprocessing import Preprocessing
4  from naiveBayes import NBMultinomial
5  from weighting import Weighting
6  from kfold import KFold
7  from confusionmatrix import ConfusionMatrix
```

Kode Program 5.43 Import Library Main

Penjelasan Kode Program 5.44:

Baris 1 Mengimpor library pandas

Baris 2 Mengimpor kelas *TermBasedRandomSampling*

Baris 3 Mengimpor kelas *Preprocessing*

Baris 4 Mengimpor kelas NBMultinomial

Baris 5 Mengimpor kelas *Weighting*

Baris 6 Mengimpor kelas KFold

Baris 7 Mengimpor kelas *ConfusionMatrix*

Setelah tahapan mengimpor library dan kelas yang diperlukan, tahapan selanjutnya adalah tahapan utama yang memanggil kelas yang sudah diimpor sebelumnya yang ditunjukkan pada Kode Program 5.45.

Main

```
1   data = pd.read_excel(
2       r'C:\Users\PPATK\Desktop\Code  2\Code\Skripsi.xlsx',"Data
3   Coding")
4   data_tweet = data['Tweet']
5   data_target = data['Label']
6
7   kfold = KFold(data_tweet,data_target,10)
8   data_train, data_test = kfold.get_data_sequence()
9
10  x_array = []
11  y_array = []
12  l_array = []
13  kfold_per_combination = []
14  list_acc = []
15  list_prec = []
16  list_recall = []
17  list_fmeasure = []
18  fold_accuracy = []
19  fold_precision = []
20  fold_recall = []
21  fold_fmeasure = []
22
23  count=1
24  for l in range(10,60,10):
25      for y in range (10,60,10):
26          for x in range (10,60,10):
27              print("PERULANGAN " + str(count))
28              count+=1
29              print('X={}, Y={}, L={}'.format(x,y,l))
30              x_array.append(x)
31              y_array.append(y)
32              l_array.append(l)
33              for i in range(9):
34                  x_array.append(" ")
35                  y_array.append(" ")
36                  l_array.append(" ")
37
38              accuracy_total_accumulation = 0
39              precision_total_accumulation = 0
40              recall_total_accumulation = 0
41              fmeasure_total_accumulation = 0
42
43              for i in range(len(data_train)):
44                  kfold_per_combination.append(i+1)
45                  y_test = []
46                  y_pred = []
47
48                  prepro = Preprocessing()
49                  cleaned_data,            terms            =
50  prepro.preprocessing(data_train[i]["tweet"])
```

```
51
52                  tbrs  =  TermBasedRandomSampling(X=x,  Y=y,
53  L=l)
54                  stopwords                                    =
55  tbrs.create_stopwords(cleaned_data,terms)
56
57                  prepro2 = Preprocessing()
58                  new_cleaned_data,        new_terms           =
59  prepro2.remove_stopword(cleaned_data, stopwords)
60
61                  weight     =     Weighting(new_cleaned_data,
62  new_terms)
63                  tfidf = weight.get_tf_idf_weighting()
64                  idf = weight.get_idf()
65
66                  nb = NBMultinomial()
67
68  nb.fit(new_cleaned_data,new_terms,data_train[i]["target"],st
69  opwords,idf,tfidf)
70
71                  for j in range(len(data_test[i]["tweet"])):
72                      prediction                               =
73  nb.predict(data_test[i]["tweet"][j],data_test[i]["target"][j
74  ])
75                      y_test.append(data_test[i]["target"][j])
76                      y_pred.append(prediction)
77
78                  cm = ConfusionMatrix()
79                  accuracy,  precision,  recall,  fmeasure  =
80  cm.score(y_test, y_pred)
81                  list_acc.append(accuracy)
82                  list_prec.append(precision)
83                  list_recall.append(recall)
84                  list_fmeasure.append(fmeasure)
85
86                  accuracy_total_accumulation+=accuracy
87                  precision_total_accumulation+=precision
88                  recall_total_accumulation+=recall
89                  fmeasure_total_accumulation+=fmeasure
90
91              accuracy_total                                   =
92  float(accuracy_total_accumulation/len(data_train))
93              precision_total                                  =
94  float(precision_total_accumulation/len(data_train))
95              recall_total                                     =
96  float(recall_total_accumulation/len(data_train))
97              fmeasure_total                                   =
98  float(fmeasure_total_accumulation/len(data_train))
99              for i in range(len(data_train)):
100                 fold_accuracy.append(accuracy_total)
101                 fold_precision.append(precision_total)
102                 fold_recall.append(recall_total)
103                 fold_fmeasure.append(fmeasure_total)
104
105 df  =  pd.DataFrame({'X':x_array,'Y':y_array,'L':l_array,'K-
106 Fold':kfold_per_combination,'Accuracy':list_acc,'Precision':
107 list_prec,'Recall':list_recall,'F-
108 Measure':list_fmeasure,'Fold   Accuracy':fold_accuracy,'Fold
109 Precision':fold_precision,'Fold Recall':fold_recall,'Fold F-
```

```
110 Measure':fold_fmeasure})
111 print(df)
112 df.to_excel(r'output.xlsx', index = False, header=True)
```

Kode Program 5.44 *Main*

Penjelasan Kode Program 5.45:

Baris 1 – 3 Mengakses data Skripsi dengan sheet Data Coding

Baris 4 – 5 Mengambil kolom Tweet dan kolom Label dan dimasukkan kedalam data_tweet dan data_target

Baris 7 – 8 Membuat objek Kfold dengan parameter data_tweet, data_target, dan 10 sebagai jumlah kfold, memanggil fungsi get_data_sequence dan memasukkan hasilnya kedalam data_train, dan data_test

Baris 10 – 23 Mendefinisikan x_array, y_array, l_array, kfold_per_combination, list_acc, list_prec, list_recall ,list_fmeasure, fold_accuracy, fold_precision ,fold_recall, fold_fmeasure sebagai list dan count diinisialisasi dengan 1

Baris 24 – 26 Melakukan perulangan l dari *range* 10 hingga 60 dengan langkah 10, melakukan perulangan y dari *range* 10 hingga 60 dengan langkah 10, melakukan perulangan x dari *range* 10 hingga 60 dengan langkah 10.

Baris 27 – 32 Mencetak angka perulangan, count ditambah 1, mencetak kombinasi x y l yang digunakan, dan memasukan x, y, dan l ke dalam masing-masing list x_array, y_array, l_array

Baris 38 - 41 accuracy_total_accumulation, precision_total_accumulation, recall_total_accumulation, fmeasure_total_accumulation diinisialisasi dengan 0

Baris 43 Melakukan perulangan i hingga sepanjang data_train

Baris 44 – 46 Kfold_per_combination di masukkan nilai i + 1, dan mendefinisikan y_test, y_pred sebagai list

Baris 48 – 50 Membuat objek *Preprocessing* dan memanggil method *preprocessing* dengan parameter data_train index i index tweet dan nilai kembalian akan dimasukkan ke dalam cleaned_data dan *term*s

Baris 52 – 55 Membuat objek *TermBasedRandomSampling* dengan parameter x, y, dan l, dan memanggil method create_*stopword*s dengan parameter cleaned_data, *term*s dan hasilnya akan dimasukkan kedalam *stopword*s

Baris 57 – 59 Membuat objek *Preprocessing* ke 2 untuk menghapus *stopword* yang sudah dihasilkan sebelumnya dengan cara memanggil fungsi remove_*stopword* dengan cleaned_data dan *term*s sebagai parameter dan dimasukkan hasilnya ke dalam

	new_cleaned_data dan new_*term*s
Baris 61 – 64	Membuat objek *Weighting* dengan parameter new_cleaned_data dan new_*term*s dan memanggil method get_tf_idf_*weighting* dan get_idf yang masing-masing dimasukkan kedalam tfidf dan idf
Baris 66 – 69	Membuat objek NBMultinomial dan memanggil method fit dengan parameter new_cleaned_data, new_*term*s, data_train index i index target, *stopword*s, idf, dan tfidf
Baris 71	Melakukan perulangan j sepanjang data_test index i index tweet
Baris 72 – 74	Memanggil method predict dari kelas NBMultinomial dengan parameter data_test index i index tweet index ke j, data_test index i index target index ke j
Baris 75 – 76	Memasukkan target ke dalam y_test dan hasil prediksi ke y_pred
Baris 69 – 72	Membuat objek *ConfusionMatrix* dan memanggil method score dengan parameter y_test, y_pred dan hasilnya dimasukkan ke dalam accuracy, precision, recall, dan fmeasure
Baris 78 – 80	Hasil acccuracy dikalikan dengan 100 dan hasilnya dimasukkan kedalam accuracy_per_fold dan dijumlahkan dengan accuracy_total_accumulation
Baris 81 – 84	Accuracy dimasukkan ke dalam list_acc, precision dimasukkan ke dalam list_precision, recall dimasukkan ke dalam list_recall, fmeasure dimasukkan ke dalam list_fmeasure
Baris 86 – 89	accuracy_total_accumulation ditambah sama dengan accuracy, precision_total_accumulation ditambah sama dengan precision, recall_total_accumulation ditambah sama dengan recall, fmeasure_total_accumulation ditambah sama dengan fmeasure
Baris 91 – 103	Menghitung akumulasi accuracy, precision, recall, fmeasure total dengan cara dibagi dengan panjang data_train dan hasilnya dimasukkan ke dalam fold_accuracy, fold_precision, fold_recall, dan fold_fmeasure sebanyak panjang data_train
Baris 105 - 110	Membuat dataframe dan memasukkan variabel x_array, y_array, l_array, kfold_per_combination, list_acc, list_prec, list_recall, list_fmeasure, fold_ accuracy, fold_ precision, fold_recall, dan fold_fmeasure.
Baris 111 - 112	Mencetak dan menyimpan DataFrame dalam bentuk excel

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini akan dijelaskan mengenai pengujian serta analisis dari hasil pengujian yang telah dilakukan. Pengujian yang dilakukan antara lain adalah pengujian parameter X, Y, dan L, pengaruh *stopword* TBRS, serta perbandingannya dengan *stopword* Tala.

6.1 Pengujian dan Analisis Kombinasi Parameter X, Y, dan L terbaik terhadap Hasil Evaluasi Sistem menggunakan K-fold Cross Validation.

Dalam pengujian parameter X, Y, L ini ditujukan untuk mencari tahu kombinasi parameter X, Y dan L terbaik terhadap hasil akurasi pada sistem serta mencari tahu bagaimana pengaruh masing-masing parameter. Nilai parameter yang digunakan adalah kombinasi dari 10, 20, 30, 40, dan 50. Hasil pengujian akan ditampilkan berdasarkan kombinasi parameter yang memiliki 3 akurasi tertinggi dan 2 akurasi terendah yang akan ditampilkan pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Kombinasi Parameter X, Y, L terbaik terhadap Hasil Evaluasi

X	Y	L	K-Fold	Accuracy	Precision	Recall	F-Measure	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F-Measure
10	10	40	1	0,789	0,717	0,683	0,7	0,758	0,658	0,636	0,647
			2	0,667	0,495	0,5	0,498				
			3	0,678	0,53	0,517	0,523				
			4	0,722	0,589	0,583	0,586				
			5	0,767	0,639	0,65	0,645				
			6	0,819	0,757	0,725	0,741				
			7	0,729	0,594	0,593	0,594				
			8	0,797	0,702	0,696	0,699				
			9	0,8	0,768	0,7	0,732				
			10	0,811	0,791	0,717	0,752				
40	30	10	1	0,767	0,704	0,65	0,676	0,756	0,653	0,633	0,643
			2	0,644	0,463	0,467	0,465				
			3	0,678	0,516	0,517	0,517				
			4	0,711	0,572	0,567	0,569				
			5	0,744	0,612	0,617	0,615				
			6	0,808	0,735	0,709	0,721				
			7	0,763	0,649	0,644	0,646				
			8	0,819	0,734	0,73	0,732				
			9	0,8	0,745	0,7	0,722				
			10	0,822	0,801	0,733	0,766				

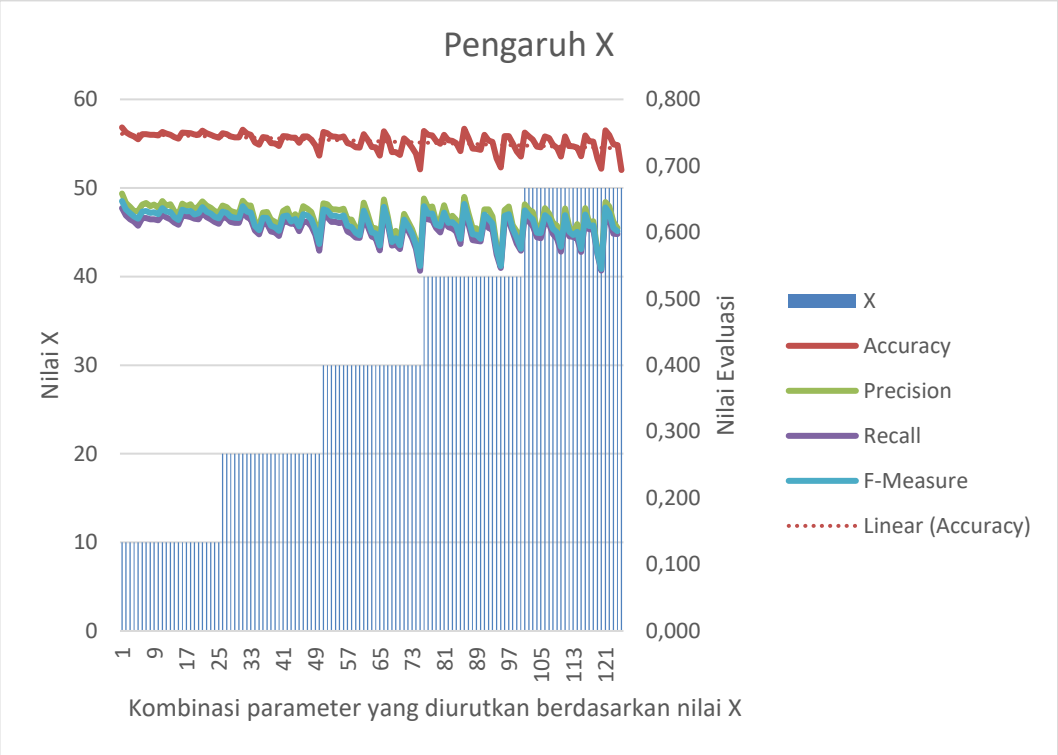
Tabel 6.1 Hasil Pengujian Kombinasi Parameter X, Y, L terbaik terhadap Hasil Evaluasi (lanjutan)

X	Y	L	K-Fold	Accuracy	Precision	Recall	F-Measure	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F-Measure
20	20	20	1	0,756	0,678	0,633	0,655	0,754	0,647	0,631	0,639
			2	0,667	0,496	0,5	0,498				
			3	0,667	0,509	0,5	0,505				
			4	0,733	0,608	0,6	0,604				
			5	0,767	0,639	0,65	0,645				
			6	0,831	0,762	0,744	0,753				
			7	0,74	0,609	0,608	0,609				
			8	0,785	0,683	0,679	0,681				
			9	0,789	0,697	0,683	0,69				
			10	0,811	0,791	0,717	0,752				
...	...	...	...	....	...	...	...	....	....	....	....
30	50	40	1	0,767	0,661	0,65	0,655	0,695	0,557	0,542	0,549
			2	0,644	0,456	0,467	0,461				
			3	0,611	0,423	0,417	0,42				
			4	0,633	0,46	0,45	0,455				
			5	0,701	0,568	0,554	0,561				
			6	0,724	0,586	0,585	0,586				
			7	0,685	0,523	0,517	0,52				
			8	0,731	0,608	0,602	0,605				
			9	0,718	0,621	0,579	0,599				
			10	0,733	0,664	0,6	0,631				

Tabel 6.1 Hasil Pengujian Kombinasi Parameter X, Y, L terbaik terhadap Hasil Evaluasi (lanjutan)

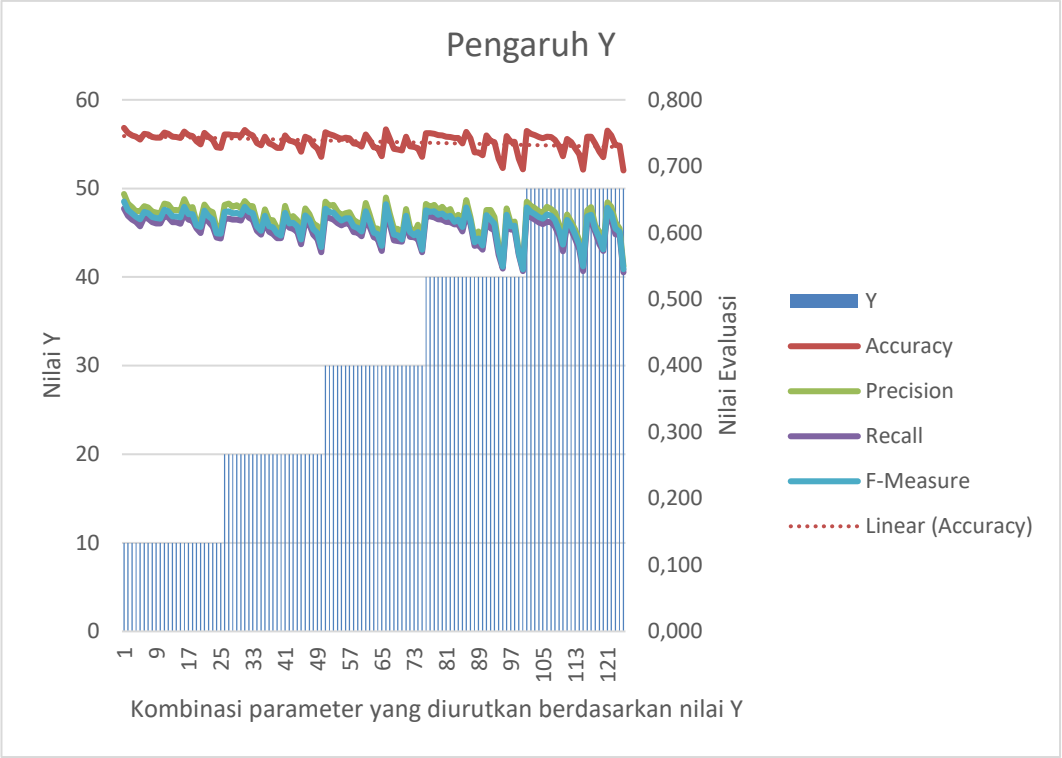
X	Y	L	K-Fold	Accuracy	Precision	Recall	F-Measure	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F-Measure
50	50	50	1	0,678	0,521	0,517	0,519	0,694	0,55	0,54	0,545
			2	0,678	0,516	0,517	0,516				
			3	0,656	0,495	0,483	0,489				
			4	0,638	0,467	0,458	0,462				
			5	0,701	0,55	0,554	0,552				
			6	0,731	0,596	0,596	0,596				
			7	0,66	0,485	0,489	0,487				
			8	0,673	0,53	0,506	0,518				
			9	0,767	0,704	0,65	0,676				
			10	0,756	0,633	0,633	0,633				

Berdasarkan hasil pengujian pada Tabel 6.1 didapatkan bahwa kombinasi dengan nilai X bernilai 10, Y bernilai 10, dan L bernilai 40 memiliki akurasi terbaik dengan nilai 0,758 atau 75.8% sedangkan kombinasi yang memiliki akurasi terburuk dengan nilai 0,694 atau 69.4% adalah X yang bernilai 50, Y bernilai 50, dan L bernilai 50. Berdasarkan hasil kombinasi tersebut dapat dianalisis bahwa terdapat pengaruh pemilihan besarnya nilai parameter. Hasil Evaluasi untuk pengaruh X akan ditampilkan pada Gambar 6.1.



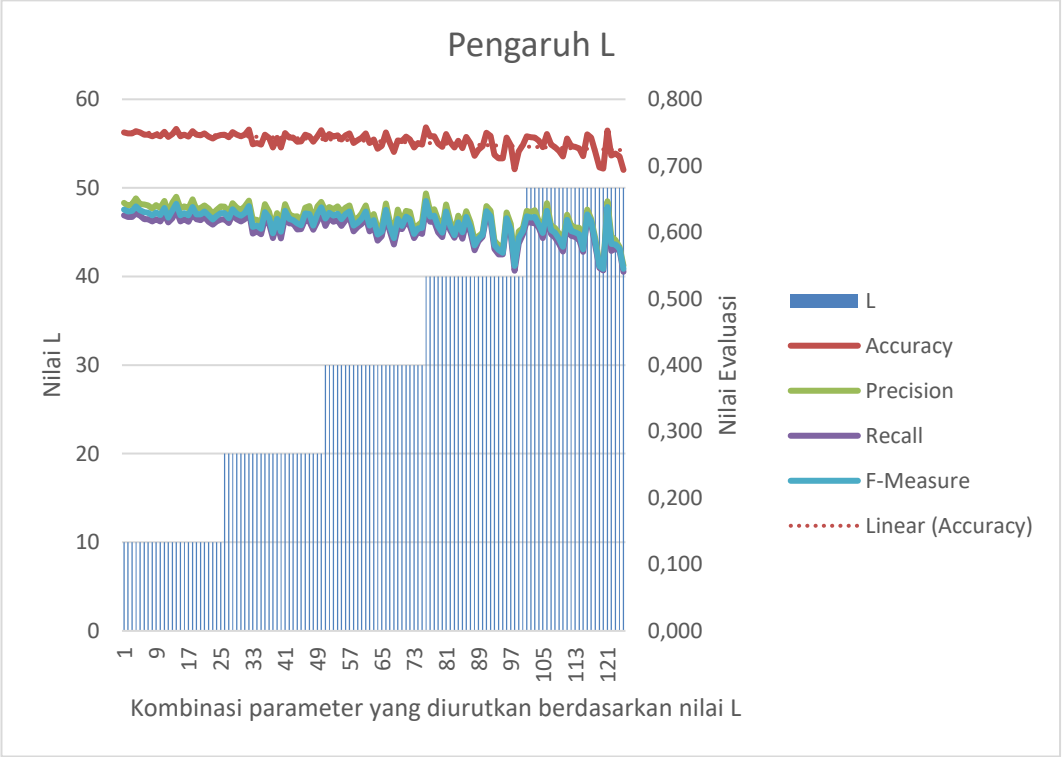
Gambar 6.1 Grafik Pengaruh X

Hasil Evaluasi untuk pengaruh Y akan ditampilkan pada Gambar 6.2.



Gambar 6.2 Grafik Pengaruh Y

Hasil Evaluasi untuk pengaruh L akan ditampilkan pada Gambar 6.3.



Gambar 6.3 Grafik Pengaruh L

Berdasarkan pada Gambar 6.1, 6.2, dan 6.3 dapat dilihat pada bagian bawah yang merupakan 125 kombinasi parameter yang sudah diurutkan

berdasarkan nilai X, Y, dan L, dan diketahui bahwa garis *accuracy, precision, recall,* dan *f-measure* semakin kanan semakin menurun menunjukkan bahwa semakin besar nilai X, Y, dan L maka semakin tinggi kemungkinannya untuk *accuracy, precision, recall,* dan *f-measure* turun. Hal ini dibuktikan ketika X bernilai 10, garis *accuracy* lebih tinggi dan lebih stabil jika dibandingkan dengan X bernilai 50, hal ini berlaku juga untuk kedua parameter lainnya yaitu Y dan L.

Dapat dilihat juga bahwa ketiga grafik tersebut bersifat fluktuatif, hal ini terjadi karena disaat parameter tersebut digunakan, terdapat pengaruh parameter-parameter lainnya yang mempengaruhi hasil evaluasi. Berikut adalah Tabel yang menggambarkan mengapa terjadinya fluktuatif nilai *accuracy, precision, recall* dan *f-measure*.

Tabel 6.2 Pengaruh Parameter

No	X	Y	L	Accuracy	Precision	Recall	F-Measure
1	10	10	40	0,758	0,658	0,636	0,647
2	30	50	40	0,695	0,557	0,542	0,549

Dapat dilihat dalam Tabel 6.2 terdapat 2 kombinasi parameter yang keduanya menggunakan nilai L sebesar 40, namun terjadi perbedaan yang signifikan antara perbandingan *accuracy, precision, recall*, dan *f-measure* yang dimana ketika X dan Y bernilai 10, 10 lebih baik ketika nilai X dan Y bernilai 30, 50.

Untuk meyakinkan analisis kombinasi terbaik, peneliti melakukan pengujian ulang dan mendapatkan hasil evaluasi terbaik didapatkan pada kombinasi dengan nilai X bernilai 10, Y bernilai 40, dan L bernilai 30 yang mendapatkan akurasi dengan nilai 0,755 atau 75.5%. Lalu pengujian selanjutnya mendapatkan nilai X bernilai 30, Y bernilai 20, dan L bernilai 10 dengan akurasi 75.3%. Dan pada pengujian terakhir mendapatkan nilai X bernilai 40, Y bernilai 10, dan L bernilai 30 dengan akurasi 75.8%.

Dalam hasil 4 pengujian yang sudah diuji sebelumnya, dilakukan pengurutan akurasi tertinggi hingga terendah dan diambil 25 kombinasi yang memiliki akurasi terbaik yang akan ditampilkan pada Tabel 6.3.

Tabel 6.3 Daftar 25 Kombinasi Terbaik

No	**Pengujian 1**			**Pengujian 2**			**Pengujian 3**			**Pengujian 4**		
	X	**Y**	**L**	**X**	**Y**	**L**	**X**	**Y**	**L**	**X**	**Y**	**L**
1	10	10	40	10	40	30	30	20	10	40	10	30
2	40	30	10	10	20	20	40	10	10	10	30	50
3	20	20	20	20	40	20	10	10	40	20	10	20
4	50	50	20	10	10	40	30	10	20	30	10	10
5	10	50	50	50	50	10	10	10	30	20	10	10

Tabel 6.3 Daftar 25 Kombinasi Terbaik (lanjutan)

No	Pengujian 1			Pengujian 2			Pengujian 3			Pengujian 4		
	X	Y	L	X	Y	L	X	Y	L	X	Y	L
6	40	10	10	20	20	30	10	40	50	50	20	10
7	30	40	10	10	40	10	40	50	10	10	50	10
8	10	30	10	10	10	20	20	10	40	30	10	40
9	30	10	20	10	10	50	10	30	40	10	10	50
10	10	10	10	30	10	10	50	30	10	40	20	20
11	50	10	10	10	50	40	30	50	10	30	10	20
12	10	40	30	10	50	20	40	30	10	20	20	10
13	10	40	40	30	40	10	50	50	10	10	10	20
14	10	40	20	10	20	30	10	10	50	50	30	10
15	20	10	10	50	10	20	40	20	10	10	30	20
16	10	50	10	10	10	30	30	50	20	10	20	30
17	30	10	10	10	20	10	20	50	40	40	30	10
18	20	20	30	20	20	20	10	30	50	10	40	30
19	10	30	30	10	40	40	10	40	10	10	40	40
20	10	20	40	10	10	10	50	10	10	10	40	20
21	10	20	50	10	30	20	10	20	20	20	20	30
22	20	10	30	20	10	10	10	50	50	10	10	40
23	30	30	10	10	30	10	20	20	10	30	20	10
24	10	40	50	20	40	10	30	20	50	40	40	10
25	10	50	20	10	50	10	10	50	20	10	20	50

Berdasarkan pada Tabel 6.3 dapat dilihat bahwa setiap pengujian yang dilakukan menghasilkan kombinasi terbaik yang berbeda, namun dapat terlihat bahwa kombinasi parameter X bernilai 10, Y bernilai 10, dan L bernilai 40 selalu berada di 25 peringkat teratas setiap pengujiannya. Perbedaan peringkat setiap pengujiannya ini terjadi karena dalam algoritme *Term Based Random Sampling*, terdapat unsur *random* dimana dalam perhitungannya terdapat kata yang diambil secara *random* untuk menjadi penentu langkah selanjutnya. Sehingga jika kata *random* yang diambil merupakan kata yang dimiliki oleh banyak dokumen, maka bobot tiap *term* yang dihasilkan memang mencerminkan *term* tersebut, sedangkan jika suatu kata *random* hanya dimiliki sedikit dokumen dan terdapat kata yang seharusnya berupa *stopword* namun tidak terbobot dengan rendah karena jumlahnya yang sangat sedikit di dokumen tersebut sehingga kata

tersebut tidak mendapatkan bobot yang seharusnya mencerminkan kata tersebut.

Dengan akurasi kombinasi tertinggi hanya 0,758 atau 75,8% hal ini disebabkan karena terdapat beberapa fold yang hampir selalu memiliki akurasi rendah. Hal ini dapat dilihat dalam setiap kombinasinya, fold ke 2 dan fold ke 3 sering dan hampir selalu mendapatkan nilai akurasi terendah. Dalam fold ke 3 ditemukan beberapa penyebab dari rendahnya akurasi pada fold ini. Berikut adalah salah satu contoh data uji yang memiliki kesalahan klasifikasi serta analisis dari kesalahan tersebut yang ditampilkan pada Tabel 6.4.

Tabel 6.4 Contoh Kalimat mengenai rendahnya akurasi

Aktual	Prediksi	Kalimat	*Term* yang digunakan setelah melalui *training* dan *testing*
Negatif	Positif	kalau kata anak sekolah "sekolah daring itu bikin hp ngehang soalnya banyak grup mata pelajaran" Buat Saya yang mahasiswa "itu bukan apa dibanding saya yang kuliah online anjir drive laptop penuh, hp kepenuhan grup gosip,ghibah, kelas sudah biasa"	['kata', 'anak', 'sekolah', 'sekolah', 'bikin', 'hp', 'soal', 'banyak', 'mata', 'ajar', 'mahasiswa', 'bukan', 'banding', 'penuh', 'hp', 'penuh', 'kelas', 'biasa']

Berdasarkan likelihood *term* tersebut didapatkan bahwa kata-kata seperti “kata”, “sekolah”, ”soal”, dan ”kelas” memiliki likelihood di kelas Positif lebih tinggi dibanding kelas Negatif, hal ini didapatkan dalam data latih yang digunakan kata-kata tersebut lebih dominan atau lebih banyak di kelas Positif, sehingga hal ini menyebabkan kesalahan klasifikasi. Selain itu, dalam kesalahan klasifikasi ini terdapat kata-kata yang terdapat dalam data uji yang dapat berkontribusi dalam sentimen Negatif, namun tidak ada didalam data latih, beberapa contoh kata tersebut adalah “anjir”, “gosip”, “ghibah”, dan “ngehang”.

6.2 Pengujian dan Analisis pengaruh *Stopword Term Based Random Sampling* dalam Hasil Evaluasi Sistem.

Pada pengujian ini ditujukan untuk membandingkan hasil evaluasi yang didapatkan jika menggunakan *Stopword Term Based Random Sampling* dan dibandingkan dengan tanpa menggunakan *Stopword* Removal. Penggunaan parameter yang digunakan dalam proses pembuatan *stopword* dengan *Term Based Random Sampling* sesuai dengan pengujian sebelumnya yang memiliki nilai akurasi terbaik yaitu dengan X bernilai 10, Y bernilai 10, dan L bernilai 40.

Hasil pengujian perbandingan *stopword Term Based Random Sampling* dengan tanpa proses *Stopword* Removal akan ditampilkan pada Tabel 6.5.

Tabel 6.5 Hasil Pengujian Pengaruh *Stopword Term Based Random Sampling* dalam Hasil Evaluasi Sistem

k-fold	*Stopword*	Accuracy	Precision	Recall	F-Measure
1	Tanpa *Stopword*	0,756	0,668	0,633	0,65
	TBRS	0,789	0,717	0,683	0,7
2	Tanpa *Stopword*	0,667	0,491	0,5	0,5
	TBRS	0,667	0,495	0,5	0,498
3	Tanpa *Stopword*	0,656	0,487	0,483	0,49
	TBRS	0,678	0,53	0,517	0,523
4	Tanpa *Stopword*	0,722	0,591	0,583	0,59
	TBRS	0,722	0,589	0,583	0,586
5	Tanpa *Stopword*	0,756	0,622	0,633	0,63
	TBRS	0,767	0,639	0,65	0,645
6	Tanpa *Stopword*	0,833	0,766	0,75	0,76
	TBRS	0,819	0,757	0,725	0,741
7	Tanpa *Stopword*	0,756	0,638	0,633	0,64
	TBRS	0,729	0,594	0,593	0,594
8	Tanpa *Stopword*	0,8	0,704	0,7	0,7
	TBRS	0,797	0,702	0,696	0,699
9	Tanpa *Stopword*	0,778	0,715	0,667	0,69
	TBRS	0,8	0,768	0,7	0,732
10	Tanpa *Stopword*	0,811	0,791	0,717	0,75

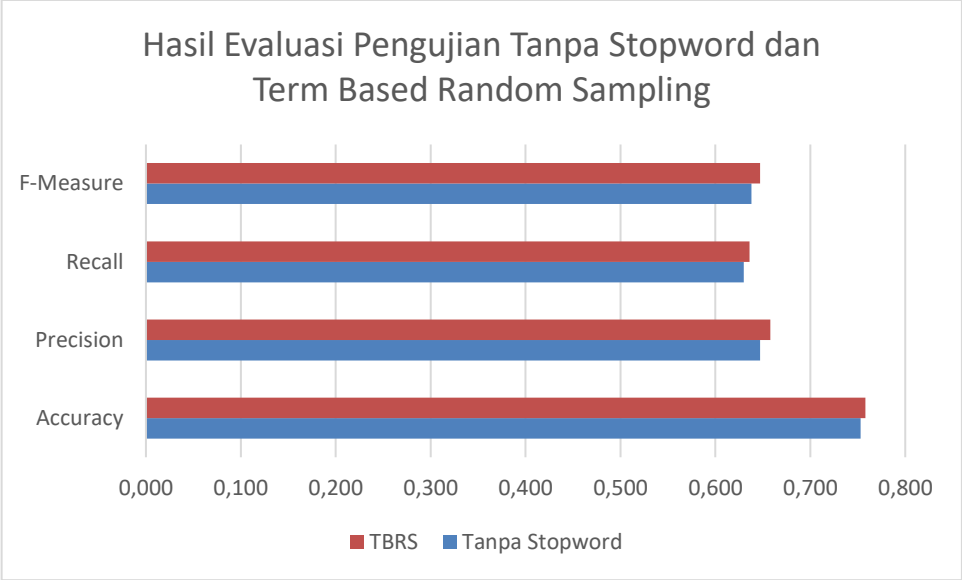
	TBRS	0,811	0,791	0,717	0,752

Berdasarkan hasil pengujian diatas dapat dihitung akurasi keseluruhan fold dari masing-masing metode. Berikut hasil rata-rata dari 10 fold *accuracy, precision, recall,* dan *f-measure* dari masing-masing metode yang akan ditampilkan pada Tabel 6.6.

Tabel 6.6 Hasil Evaluasi Pengujian Tanpa *Stopword* dan TBRS

Stopword	***Avg. Accuracy***	***Avg. Precision***	***Avg. Recall***	***Avg. F-Measure***
Tanpa *Stopword*	0,753	0,647	0,63	0,638
TBRS	0,758	0,658	0,636	0,647

Untuk mempermudah analisis, berikut hasil pengujian tanpa *stopword* dan *Term Based Random Sampling* yang disajikan dalam bentuk grafik pada Gambar 6.4.



Gambar 6.4 Grafik Pengujian Tanpa *Stopword* dan *Term Based Random Sampling*

Pada Tabel 6.6 dan Gambar 6.4 didapatkan bahwa akurasi keseluruhan dari penggunaan metode *Term Based Random Sampling* ini sedikit lebih baik 0,5% dibandingkan dengan tanpa menggunakan proses *stopword* removal. Metode tanpa *stopword* memiliki *macroaverage accuracy* sebesar 75,3%, *macroaverage precision* sebesar 64,7%, *macroaverage recall* sebesar 63,0%, *macroaverage f-measure* sebesar 63,8% sedangkan untuk metode dengan *Term Based Random Sampling* memiliki *macroaverage accuracy* sebesar 75,8%, *macroaverage precision* sebesar 65,8%, *macroaverage recall* sebesar 63,6%, *macroaverage f-measure* sebesar 64,7%. Sehingga dapat disimpulkan bahwa penggunaan Term Based Random Sampling sedikit lebih baik dalam kasus ini.

6.3 Perancangan Pengujian Perbandingan Hasil Evaluasi *Stopword* Tala dan *Stopword Term Based Random Sampling*.

Pada pengujian ini ditujukan untuk membandingkan hasil evaluasi yang didapatkan jika menggunakan *stopword* Tala dan dibandingkan dengan *Stopword* yang dihasilkan oleh algoritme *Term Based Random Sampling*. Penggunaan parameter yang digunakan dalam proses pembuatan *stopword* dengan *Term Based Random Sampling* sesuai dengan pengujian sebelumnya yang memiliki nilai akurasi terbaik yaitu dengan X bernilai 10, Y bernilai 10, dan L bernilai 40.

Tabel 6.7 Hasil Pengujian Perbandingan Evaluasi Pengunaan *Stopword* Tala dan *Stopword Term Based Random Sampling*

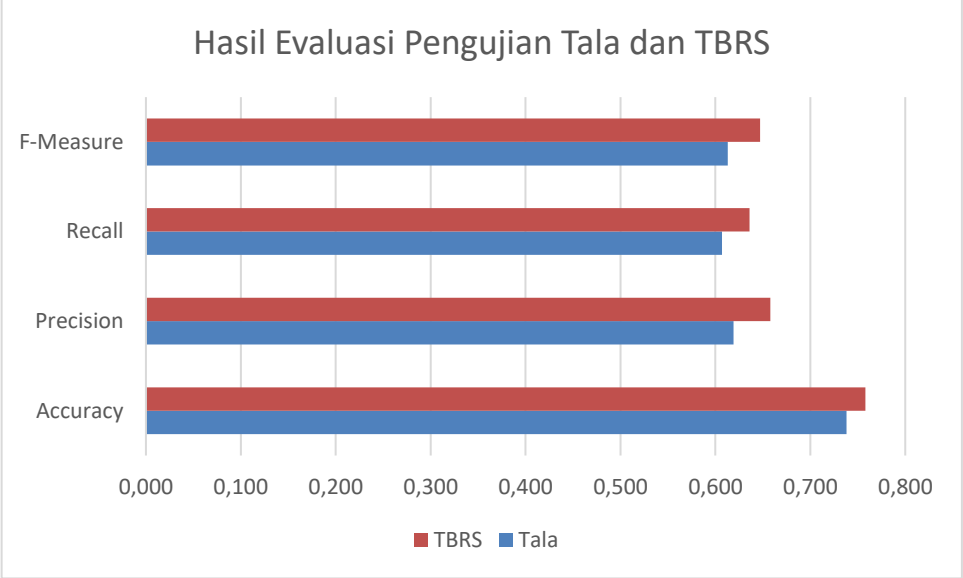
k-fold	*Stopword*	Accuracy	Precision	Recall	F-Measure
1	Tala	0,733	0,643	0,6	0,621
	TBRS	0,789	0,717	0,683	0,7
2	Tala	0,667	0,509	0,5	0,504
	TBRS	0,667	0,495	0,5	0,498
3	Tala	0,633	0,447	0,45	0,449
	TBRS	0,678	0,53	0,517	0,523
4	Tala	0,667	0,498	0,5	0,499
	TBRS	0,722	0,589	0,583	0,586
5	Tala	0,744	0,607	0,617	0,612
	TBRS	0,767	0,639	0,65	0,645
6	Tala	0,833	0,765	0,75	0,758
	TBRS	0,819	0,757	0,725	0,741
7	Tala	0,767	0,656	0,65	0,653
	TBRS	0,729	0,594	0,593	0,594
8	Tala	0,767	0,653	0,65	0,651
	TBRS	0,797	0,702	0,696	0,699
9	Tala	0,789	0,71	0,683	0,696
	TBRS	0,8	0,768	0,7	0,732
10	Tala	0,778	0,706	0,667	0,686
	TBRS	0,811	0,791	0,717	0,752

Berdasarkan hasil pengujian diatas dapat dihitung akurasi keseluruhan fold dari masing-masing metode. Berikut hasil rata-rata dari 10 fold *accuracy, precision, recall,* dan *f-measure* dari masing-masing metode yang akan ditampilkan pada Tabel 6.8.

Tabel 6.8 Hasil Evaluasi Pengujian Tala dan TBRS

Stopword	***Avg. Accuracy***	***Avg. Precision***	***Avg. Recall***	***Avg. F-Measure***
Tala	0,738	0,619	0,607	0,613
TBRS	0,758	0,658	0,636	0,647

Untuk mempermudah analisis, berikut hasil pengujian stopword Tala dan Term Based Random Sampling yang disajikan dalam bentuk grafik pada Gambar 6.5.



Gambar 6.5 Grafik Pengujian Tala dan Term Based Random Sampling

Pada Tabel 6.8 dan Gambar 6.5 didapatkan bahwa akurasi keseluruhan dari penggunaan metode *Term Based Random Sampling* 2% lebih baik dibandingkan dengan penggunaan *stopword* Tala. Metode *stopword* Tala memiliki *macroaverage accuracy* sebesar 73,8%, *macroaverage precision* sebesar 61,9%, *macroaverage recall* sebesar 60,7%, *macroaverage f-measure* sebesar 61,3% sedangkan untuk metode dengan *Term Based Random Sampling* memiliki *macroaverage accuracy* sebesar 75,8%, *macroaverage precision* sebesar 65,8%, *macroaverage recall* sebesar 63,6%, *macroaverage f-measure* sebesar 64,7%. Sehingga dapat disimpulkan bahwa penggunaan Term Based Random Sampling sedikit lebih baik dalam kasus ini.

BAB 7 PENUTUP

Pada bab ini akan dijelaskan beberapa kesimpulan serta saran yang didapatkan dari penelitian sehingga dapat membantu penelitian selanjutnya.

7.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, didapatkan beberapa poin kesimpulan antara lain:

1. Dalam pencarian kombinasi parameter X, Y, dan L terbaik dilakukan sejumlah 4 kali pengujian setiap parameter dengan angka 10, 20, 30, 40, dan 50 dan didapatkan sebanyak 125 kombinasi yang setiap kombinasinya dilakukan 10 fold cross validation dan setiap pengujiannya dianalisis 25 kombinasi terbaik dan dapat disimpulkan bahwa terdapat kombinasi yang selalu terdapat didalam 25 kombinasi terbaik yakni X sebesar 10, Y sebesar 10, dan L sebesar 40 untuk analisis sentimen dengan *Naïve Bayes* yang mendapatkan *macroaverage accuracy* sebesar 75,8%, *macroaverage precision* sebesar 65,8%, *macroaverage recall* sebesar 63,6%, dan *macroaverage f-measure* sebesar 64,7%. Perbedaan peringkat-peringkat kombinasi ini disebabkan karena dalam algoritme *Term Based Random Sampling*, terdapat unsur *random* dimana dalam perhitungannya terdapat kata yang diambil secara *random* untuk menjadi penentu langkah selanjutnya. Sehingga setiap pengujian yang dilakukan akan menghasilkan kombinasi-kombinasi yang berbeda. Selain itu, dapat disimpulkan bahwa semakin besar nilai X, Y, dan L maka semakin tinggi kemungkinannya untuk *accuracy, precision, recall,* dan *f-measure* turun. Hal ini dibuktikan ketika X bernilai 10, garis accuracy lebih tinggi dan lebih stabil jika dibandingkan dengan X bernilai 50, hal ini berlaku juga untuk kedua parameter lainnya yaitu Y dan L.
2. Pengunaan metode pembentukan *stopword Term Based Random Sampling* untuk analisis sentimen dengan *Naïve Bayes* dapat diterapkan dengan baik, hal ini dapat dilihat dengan meningkatnya akurasi sistem yang dilakukan sebanyak 10-*fold* ketika menggunakan *stopword Term Based Random Sampling* sebesar 0,5% jika dibandingkan dengan tidak menggunakan proses *stopword removal.*
3. Berdasarkan pengujian perbandingan antara *Naïve Bayes* dan *stopword Term Based Random Sampling* mendapatkan rata-rata akurasi dari 10 *fold*, *stopword Term Based Random Sampling* memiliki akurasi sebesar 75,8% sedangkan jika menggunakan *stopword* Tala adalah sebesar 73,8%. Penggunaan *stopword Term Based Random Sampling* terbukti dapat meningkatkan akurasi pada analisis sentimen dengan *Naïve Bayes* sebesar 2%.

7.2 Saran

Penelitian yang dilakukan masih memiliki banyak kekurangan yang perlu diperbaiki. Adapun saran yang dapat diberikan untuk penelitian berikutnya

adalah pada tahap *preprocessing* sebaiknya dilakukan proses normalisasi kata untuk dapat meningkatkan akurasi sistem serta pemilihan data yang lebih baik.

DAFTAR REFERENSI

Arnani, M., 2020. *KOMPAS.* [Online] Available at: https://www.kompas.com/tren/read/2020/03/13/111245765/kasus-pertama-virus-corona-di-china-dilacak-hingga-17-november-2019

Devita, R. N., Herwanto, H. W. & Wibawa, A. P., 2018. PERBANDINGAN KINERJA METODE NAIVE *BAYES* DAN K-NEAREST NEIGHBOR UNTUK KLASIFIKASI ARTIKEL BERBAHASA INDONESIA. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* 5(4), pp. 427-434.

Dila Purnama Sari, D. E., Sari, Y. A. & Furqon, M. T., 2020. Pembentukan Daftar Stopword menggunakan Zipf Law dan Pembobotan Augmented TF - Probability IDF pada Klasifikasi Dokumen Ulasan Produk. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* pp. 406-412.

Gaddam, S. H. R., 2019. *Text Preprocessing in Natural Language Processing.* [Online] Available at: https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8

Imtiyazi, M. A., S. & Bijaksana, M. A., 2015. Sentiment Analysis Berbahasa Indonesia Menggunakan Improved Multinomial Naive *Bayes*. *e-Proceeding of Engineering,* 2(2), p. 6331.

Jones, S., 2004. A Statistical Interpretation of Term Specificity and Its Retrieval. *Journal Of Documentation,* 60(5), pp. 11-21.

Liu, B., 2012. *Sentiment Analysis and Opinion Mining.* Chicago: Morgan & Claypool.

Lo, R. T.-W., He, B. & Ounis, I., 2005. *Automatically Building a Stopword List for an Information Retrieval System,* Glasgow, UK: Department of Computing Science.

Narkhede, S., 2018. *Understanding Confusion Matrix.* [Online] Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Neale, C., Workman, D. & Dommalapati, A., 2019. *Cross Validation: A Beginner's Guide.* [Online] Available at: https://towardsdatascience.com/cross-validation-a-beginners-guide-5b8ca04962cd [Diakses 23 September 2020].

Prabowo, D. A., Fhadli, M., Najib, M. A. & Fauzi, H. A., 2016. *TF-IDF*-Enhanced Genetic Algorithm Untuk Extractive Automatic Text Summarization. *Jurnal Teknologi Informasi dan Ilmu Komputer,* 3(3), pp. 208-215.

Putsanra, D. V., 2020. *tirto.* [Online] Available at: https://tirto.id/apa-itu-new-normal-dan-bagaimana-penerapannya-saat-pandemi-corona-fCSg

Rahman, A., Wiranto & Doewes, A., 2017. Online News Classification Using Multinomial Naive *Bayes*. *ITSMART: Jurnal Ilmiah Teknologi dan Informasi.*

Rahutomo, F. & Ririd, A. R. T. H., 2018. EVALUASI DAFTAR STOPWORD BAHASA INDONESIA. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* pp. 41-48.

Ramadhan, A., Nugraheny, D. E. & Maharani, T., 2020. *KOMPAS.* [Online] Available at: https://nasional.kompas.com/read/2020/09/05/15204581/update-kembali-bertambah-di-atas-3000-kasus-covid-19-lewati-190000?page=all

Sa'rony, A., Adikara, P. P. & Wihandika, R. C., 2019. Analisis Sentimen Kebijakan Pemindahan Ibukota Republik Indonesia dengan Menggunakan Algoritme Term-Based Random Samplingdan Metode Klasifikasi *Naïve Bayes*. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* pp. 10086-10094.

Sawla, S., 2018. *Introduction to Naive Bayes for Classification.* [Online] Available at: https://medium.com/@srishtisawla/introduction-to-naive-*Bayes*-for-classification-baefefb43a2d

Septian, J. A., Fahrudin, T. M. & Nugroho, A., 2019. Analisis Sentimen Pengguna Twitter Terhadap Polemik Persepakbolaan Indonesia Menggunakan Pembobotan *TF-IDF* dan K-Nearest Neighbor. *JOURNAL OF INTELLIGENT SYSTEMS AND COMPUTATION.*

Singh, S. & Shukla, S., 2016. *Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification.* Bhimavaram, IEEE.

Tania, A., 2020. *Muda Kompas.* [Online] Available at: https://muda.kompas.id/baca/2020/05/13/perlu-kerjasama-dosen-dan-mahasiswa-dalam-kuliah-daring/
[Diakses 22 September 2020].

LAMPIRAN A PENGUJIAN PENGARUH PARAMETER X,Y,L

Untuk selengkapnya dapat dilihat pada
http://bit.ly/PengujianPengaruhParameterXYL

X	Y	L	Avg. Accuracy	Avg. Precision	Avg. Recall	Avg. F-Measure
10	10	40	0,758	0,658	0,636	0,647
40	30	10	0,756	0,653	0,633	0,643
20	20	20	0,754	0,647	0,631	0,639
40	10	10	0,752	0,651	0,628	0,639
50	50	20	0,754	0,646	0,629	0,637
10	50	50	0,753	0,647	0,63	0,638
10	10	10	0,75	0,644	0,625	0,634
10	30	10	0,751	0,647	0,626	0,636
30	40	10	0,752	0,649	0,628	0,638
50	10	10	0,75	0,642	0,625	0,633
...	...	...	...	...	...	...
30	50	50	0,718	0,588	0,575	0,581
30	40	40	0,717	0,587	0,575	0,581
50	20	50	0,714	0,586	0,571	0,578
50	30	50	0,714	0,578	0,571	0,574
40	40	40	0,711	0,58	0,567	0,573
50	40	40	0,711	0,572	0,567	0,569
40	50	50	0,714	0,578	0,572	0,575
40	40	50	0,698	0,552	0,546	0,549
50	40	50	0,696	0,547	0,542	0,545
30	50	40	0,695	0,557	0,542	0,549
50	50	50	0,694	0,55	0,54	0,545