

**ANALISIS SENTIMEN MASYARAKAT TERHADAP UJI COBA
LRT JAKARTA MENGGUNAKAN *IMPROVED K-NEAREST
NEIGHBOR DAN INFORMATION GAIN***

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Mahendra Okza Pradhana
NIM: 165150200111079



PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

ANALISIS SENTIMEN MASYARAKAT TERHADAP UJI COBA LRT JAKARTA
MENGGUNAKAN IMPROVED K-NEAREST NEIGHBOR DAN INFORMATION GAIN

SKRIPSI

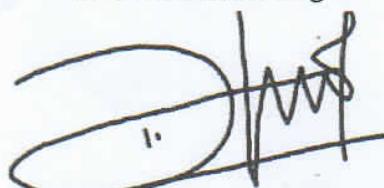
Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Mahendra Okza Pradhana
NIM: 165150200111079

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Juni 2020

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I



Indriati, S.T., M.Kom.
NIP: 19831013 201504 2 002

Dosen Pembimbing 2



Sigit Adinugroho, S.Kom., M.Sc.
NIK: 201607 880701 1 000

Mengetahui

Ketua Jurusan Teknik Informatika



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

NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar 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, 3 Juni 2020



Mahendra Okza Pradhana

NIM: 165150200111079

PRAKATA

Puji syukur penulis panjatkan pada Kehadirat ALLAH SWT karena atas segala limpahan rezeki, rahmat, dan hidayahnya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Analisis Sentimen Masyarakat terhadap Uji Coba LRT Jakarta menggunakan *Improved K-Nearest Neighbor* dan *Information Gain* . Penulis juga mengucapkan banyak terima kasih pada pihak - pihak yang telah memberikan pencerahan sehingga skripsi ini dapat selesai, diantaranya adalah :

1. Kedua orang tua penulis, Edy Santoso, S.Pd dan Suliani, S.Pd serta saudara penulis, Ridjky Tegar Perkasa dan segenap keluarga besar penulis yang telah memberikan dukungan dan doa kepada penulis.
2. Ibu Indriati, S.T., M.Kom. dan Bapak Sigit Adinugroho, S.Kom., M.Sc. sebagai dosen pembimbing I dan dosen pembimbing II penulis yang telah memberikan arahan serta bimbingan selama pengerjaan skripsi.
3. Bapak Wayan Firdaus Mahmudi, S.Si., M.T., Ph.D. selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Agus Wahyu Widodo, S.T., M.Cs., selaku Ketua Program Studi Teknik Informatika yang telah memberikan kesempatan kepada penulis untuk menyelesaikan skripsi.
6. Seluruh bapak dan ibu dosen pengajar yang telah mendidik penulis selama di bangku perkuliahan.
7. Teman - teman dari Program Studi Teknik Informatika 2016 khususnya Aditya Pratama Putra, Farhan Setya, Titus Christian, Ludgerus Darell, Syarief Noor, Muhammad Hasan Johan Alfarizi, Feri Setyo Efendi, Candra Ardiansyah, Muhammad Fadel Yudhistira, Edo Ergi, Muhamajir Kurniawan, Muhammad Ikhsan Romadhoni, Iqro Ilhamsyah, dan Iqbal Setya Nurfirmansyah yang telah mendampingi perjalanan selama masa perkuliahan.
8. Seluruh pihak yang tidak bisa disebutkan satu-persatu atas bantuannya selama penulisan skripsi.

Penulis sangat menyadari bahwa selama proses pengerjaan skripsi terdapat banyak kekurangan. Penulis berharap skripsi ini dapat bermanfaat bagi pembaca dan penelitian selanjutnya.

Malang, 3 Juni 2020

Penulis

okzapradhana@student.ub.ac.id

ABSTRAK

Mahendra Okza Pradhana, Analisis Sentimen Masyarakat terhadap Uji Coba LRT Jakarta menggunakan *Improved K-Nearest Neighbor* dan *Information Gain*

Pembimbing: Indriati, S.T., M.Kom. dan Sigit Adinugroho, S.Kom., M.Sc.

Perkembangan sistem transportasi saat ini sangat memberikan kemudahan bagi masyarakat untuk berpindah dari satu tempat ke tempat lainnya. Salah satu transportasi umum yang cukup baru yaitu *Light Rail Transit* (LRT). LRT atau kereta api ringan sempat mengadakan uji coba public secara gratis hanya dengan mendaftarkan diri melalui situs LRT Jakarta. Untuk meningkatkan dan memaksimalkan pelayanan dari LRT Jakarta, mereka memiliki akun media sosial di mana masyarakat dapat memberikan masukan atau *feedback* maupun penilaian. Salah satu cara yang dapat dilakukan yaitu dengan analisis sentimen untuk mengetahui apakah masyarakat menyukai pelayanan yang diberikan oleh LRT Jakarta. Pada penelitian ini menggunakan *Improved KNN* sebagai metode klasifikasi untuk mengetahui sentimen masyarakat ditambah dengan *Information Gain* untuk menyeleksi fitur yang digunakan saat proses klasifikasi. Proses dari analisis sentimen ini meliputi pengumpulan data, *text preprocessing* yang menghasilkan data bersih, kemudian pembobotan pada term dengan tf idf dilanjutkan dengan seleksi fitur menggunakan *Information Gain*. Selanjutnya, melakukan klasifikasi dengan *Improved KNN* menggunakan fitur hasil seleksi sebelumnya. Data yang digunakan merupakan data primer yang bersumber dari tiga media sosial yakni Youtube, Twitter dan Facebook. Hasil dari penelitian ini yaitu *f-measure* terbaik diperoleh saat $k=11$ menggunakan *threshold* 100% atau seluruh term digunakan yakni sebesar 85.51% dengan rata-rata waktu komputasi yang dihitung dari 5-fold sebesar 0.4647 menit.

Kata kunci: LRT, klasifikasi, analisis sentimen, *improved KNN*, *information gain*, *tf idf*

ABSTRACT

Mahendra Okza Pradhana, Analysis of Society Sentiments on LRT Jakarta Trial using Improved K-Nearest Neighbor and Information Gain

Supervisors: Indriati, S.T., M.Kom. and Sigit Adinugroho, S.Kom., M.Sc.

Current transportation development system has giving easiness for society to moving from place to another places. One of quite new public transportation is Light Rail Transit (LRT). LRT or light railroad have an opportunity to held public trial access for free just by registering yourself in LRT Jakarta website. For improving and maximize LRT Jakarta services, they have social media account where people may give feedback and assessment. One of way that could be done is by sentiment analysis to find out whether the society likes the services provided by LRT Jakarta. This study is using the Improved KNN as a classification method to determine people sentiment coupled with Information Gain to select features used during the classification process. The process of sentiment analysis includes data collection, text preprocessing that produces clean data, then weighting the terms with tf idf followed by feature selection using Information Gain. The next step is classification with Improved KNN using the features of the previous selection. The data used are primary data sourced from three social media namely Youtube, Twitter and Facebook. The results of this study are the best f-measure obtained when k = 11 using a 100% threshold or the whole term used that is equal to 85.51% with an average computational time calculated from 5-fold of 0.4647 minutes.

Keywords: *LRT, classification, sentiment analysis, improved KNN, information gain, tf idf.*

DAFTAR ISI

PENGESAHAN	ii
PERNYATAAN ORISINALITAS	iii
PRAKATA.....	iv
ABSTRAK.....	v
ABSTRACT.....	vi
DAFTAR ISI.....	vii
DAFTAR TABEL.....	x
DAFTAR GAMBAR.....	xii
BAB 1 PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Tujuan	2
1.4 Manfaat.....	3
1.5 Batasan Masalah.....	3
1.6 Sistematika Pembahasan.....	3
BAB 2 LANDASAN KEPUSTAKAAN	5
2.1 Kajian Pustaka	5
2.1.1 Analisis Sentimen	7
2.1.2 Media Sosial	8
2.1.3 <i>Light Rail Transit (LRT)</i>	9
2.2 <i>Preprocessing</i> pada Teks.....	9
2.2.1 <i>Cleaning</i>	9
2.2.2 <i>Case Folding</i>	9
2.2.3 <i>Tokenization</i>	9
2.2.4 <i>Stemming</i>	9
2.3 <i>Term Weighting</i>	10
2.4 <i>Improved K-Nearest Neighbor</i>	10
2.5 <i>Cosine Similarity</i>	12
2.6 <i>Information Gain</i>	12
2.7 <i>Confusion Matrix</i>	13

2.8 <i>K-fold Cross Validation</i>	14
BAB 3 METODOLOGI	16
3.1 Tipe Penelitian	16
3.2 Strategi Penelitian.....	16
3.3 Subjek Penelitian	16
3.4 Lokasi Penelitian	16
3.5 Teknik Pengumpulan Data.....	16
3.6 Teknik Analisis Data	17
3.7 Peralatan Pendukung	17
3.8 Implementasi Algoritme	17
BAB 4 PERANCANGAN.....	18
4.1 Diagram Alir Sistem.....	18
4.1.1 Diagram Alir <i>Text Preprocessing</i>	19
4.1.2 Diagram Alir <i>Feature Selection</i>	24
4.1.3 Diagram Alir <i>Term Weighting</i>	25
4.1.4 Diagram Alir Menghitung Nilai <i>Cosine Similarity</i>	33
4.1.5 Diagram Alir Klasifikasi dengan <i>Improved K-Nearest Neighbor</i>	34
4.2 Manualisasi Perhitungan Data.....	36
4.2.1 <i>Text Preprocessing</i>	37
4.2.2 <i>Feature Selection</i> dengan <i>Information Gain</i>	41
4.2.3 Pemilihan Term	52
4.2.4 <i>Term Weighting</i>	55
4.2.5 Menghitung <i>Cosine Similarity</i>	64
4.2.6 Klasifikasi dengan <i>Improved K-Nearest Neighbor</i>	65
4.3 Perancangan Skenario Pengujian	67
4.3.1 Pengujian Nilai k Awal	67
4.3.2 Pengujian Seleksi Fitur <i>Information Gain</i>	67
BAB 5 IMPLEMENTASI	69
5.1 Implementasi <i>Preprocessing</i>	69
5.2 Implementasi <i>Term Weighting</i>	73
5.3 Implementasi <i>Information Gain</i>	79

5.4 Implementasi <i>Cosine Similarity</i>	87
5.5 Implementasi Klasifikasi dengan <i>Improved KNN</i>	91
BAB 6 PENGUJIAN DAN ANALISIS.....	96
6.1 Pengujian Nilai <i>k</i> (Ketetanggaan) dengan <i>5-Fold Cross Validation</i>	96
6.1.1 <i>Fold</i> ke-1.....	96
6.1.2 <i>Fold</i> ke-2.....	97
6.1.3 <i>Fold</i> ke-3.....	97
6.1.4 <i>Fold</i> ke-4.....	98
6.1.5 <i>Fold</i> ke-5.....	98
6.1.6 Pemilihan Nilai <i>k</i> Terbaik dari Rata-rata Pengujian pada <i>5-fold</i>	99
6.2 Pengujian Seleksi Fitur <i>Information Gain</i> dengan <i>5-Fold Cross Validation</i>	100
6.2.1 <i>Fold</i> ke-1.....	100
6.2.2 <i>Fold</i> ke-2.....	101
6.2.3 <i>Fold</i> ke-3.....	101
6.2.4 <i>Fold</i> ke-4.....	102
6.2.5 <i>Fold</i> ke-5.....	103
6.2.6 Pemilihan <i>Threshold</i> Terbaik dari Rata-rata Pengujian pada <i>5-fold</i>	104
6.3 Analisis	105
6.3.1 Analisis Pengujian Nilai <i>k</i> dengan <i>5-fold</i>	105
6.3.2 Analisis Pengujian <i>Information Gain</i>	106
BAB 7 PENUTUP	109
7.1 Kesimpulan.....	109
7.2 Saran	109
DAFTAR REFERENSI	111

DAFTAR TABEL

Tabel 2.2.1 Tabel Kajian Pustaka.....	6
Tabel 2.2.2 Tabel Kajian Pustaka (Lanjutan)	7
Tabel 2.2.3 <i>Confusion Matrix</i>	13
Tabel 2.3.1 Spesifikasi <i>Hardware</i>	17
Tabel 2.3.2 Spesifikasi <i>Software</i>	17
Tabel 4.1 Data Latih	36
Tabel 4.2 Data Uji	37
Tabel 4.3 Hasil <i>Cleaning</i> pada Data Latih.....	37
Tabel 4.4 Hasil <i>Cleaning</i> pada Data Uji	38
Tabel 4.5 Hasil <i>Case Folding</i> pada Data Latih.....	38
Tabel 4.6 Hasil <i>Case Folding</i> pada Data Uji	39
Tabel 4.7 Hasil <i>Tokenization</i> pada Data Latih	39
Tabel 4.8 Hasil <i>Tokenization</i> pada Data Uji.....	40
Tabel 4.9 Hasil <i>Stemming</i> pada Data Latih.....	40
Tabel 4.10 Hasil <i>Stemming</i> pada Data Uji	40
Tabel 4.11 Kemunculan Term pada Dokumen untuk Perhitungan <i>Information Gain</i>	41
Tabel 4.12 Perhitungan <i>Information Gain</i> pada Setiap Term.....	44
Tabel 4.13 Pengurutan Nilai <i>Information Gain</i>	47
Tabel 4.14 Term Terpilih berdasarkan <i>Mean Threshold</i> dari <i>Information Gain</i> ...	50
Tabel 4.15 Pemilihan Term Data Uji	52
Tabel 4.16 Term yang Digunakan Saat Klasifikasi pada Data Uji	54
Tabel 4.17 Hasil <i>Raw Term Frequency Weighting</i>	55
Tabel 4.18 Hasil <i>Log Term Frequency Weighting</i>	57
Tabel 4.19 Hasil <i>Inverse Document Frequency</i>	59
Tabel 4.20 Hasil <i>Term Frequency Inverse Document Frequency</i>	62
Tabel 4.21 TF-IDF pada Data Latih dengan Contoh Dokumen Dua serta Data Uji	64
Tabel 4.22 Perkalian Vektor antara Dokumen Dua dengan Dokumen Uji	64
Tabel 4.23 Hasil Penjumlahan dari Perkalian Vektor Data Latih dengan Data Uji	64
Tabel 4.24 Hasil Akar Kuadrat Jumlah Vektor Data Latih serta Data Uji	65

Tabel 4.25 Nilai <i>Cosine Similarity</i> Data Latih dengan Data Uji.....	65
Tabel 4.26 Penentuan Nilai K Awal	65
Tabel 4.27 Nilai k baru pada Kelas Positif dan Negatif	66
Tabel 4.28 Hasil Klasifikasi dari Data Uji	66
Tabel 4.29 Skenario Pengujian Nilai k Awal	67
Tabel 4.30 Skenario Pengujian Seleksi Fitur <i>Information Gain</i>	68
Tabel 6.1 Hasil Pengujian Nilai k pada Fold ke-1.....	96
Tabel 6.2 Hasil Pengujian Nilai k pada Fold ke-2.....	97
Tabel 6.3 Hasil Pengujian Nilai k pada Fold ke-3.....	97
Tabel 6.4 Hasil Pengujian Nilai k pada Fold ke-4.....	98
Tabel 6.5 Hasil Pengujian pada Fold ke-5	99
Tabel 6.6 Rata-rata Hasil Pengujian pada Setiap k	99
Tabel 6.7 Hasil Pengujian <i>Information Gain</i> pada Fold ke-1.....	100
Tabel 6.8 Hasil Pengujian <i>Information Gain</i> pada Fold ke-2.....	101
Tabel 6.9 Hasil Pengujian <i>Information Gain</i> pada Fold ke-3.....	102
Tabel 6.10 Hasil Pengujian <i>Information Gain</i> pada Fold ke-4.....	102
Tabel 6.11 Hasil Pengujian <i>Information Gain</i> pada Fold ke-5.....	103
Tabel 6.12 Rata-rata Hasil Pengujian Seleksi Fitur <i>Information Gain</i> pada Setiap <i>Threshold</i>	104
Tabel 6.13 Term beserta Nilai <i>Information Gain</i> , Kemunculan pada Dokumen dengan Kelas Positif dan Negatif	107

DAFTAR GAMBAR

Gambar 2.1 Pengguna Media Sosial di Indonesia	8
Gambar 2.2 Ilustrasi <i>K-Fold Cross Validation</i>	15
Gambar 4.1 Diagram Alir Sistem	18
Gambar 4.2 Diagram Alir <i>Text Preprocessing</i>	20
Gambar 4.3 Diagram Alir <i>Cleaning</i>	21
Gambar 4.4 Diagram Alir <i>Case Folding</i>	22
Gambar 4.5 Diagram Alir <i>Tokenization</i>	22
Gambar 4.6 Diagram Alir <i>Stemming</i>	24
Gambar 4.7 Diagram Alir <i>Feature Selection</i> dengan <i>Information Gain</i>	25
Gambar 4.8 Diagram Alir <i>Term Weighting</i>	26
Gambar 4.9 Diagram Alir <i>Raw Term Frequency Weighting</i>	28
Gambar 4.10 Diagram Alir <i>Log Term Frequency Weighting</i>	29
Gambar 4.11 Diagram Alir <i>Inverse Document Frequency Weighting</i>	31
Gambar 4.12 Diagram Alir <i>Term Frequency Inverse Document Frequency Weighting</i>	32
Gambar 4.13 Menghitung Nilai <i>Cosine Similarity</i>	34
Gambar 4.14 Diagram Alir Klasifikasi dengan <i>Improved KNN</i>	36
Gambar 6.1 Grafik Variasi Nilai k Terhadap <i>Precision</i> , <i>Recall</i> dan <i>F-measure</i>	105
Gambar 6.2 Hasil Pengujian <i>Information Gain</i> terhadap <i>Precision</i> , <i>Recall</i> dan <i>F-measure</i>	106
Gambar 6.3 Rata-rata Hasil Pengujian <i>Information Gain</i> dengan 5-fold terhadap Rata-rata <i>Running Time</i>	106

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

Sistem transportasi saat ini telah berkembang pesat dan semakin memberikan mobilitas tinggi kepada masyarakat. Sebagai contoh salah satu transportasi umum di ibukota yaitu *Light Rail Transit* (LRT). LRT merupakan kereta api ringan yang memiliki kecepatan sekitar 70 km/jam hingga 80 km/jam. Bangunan rangka stasiun LRT dibangun menggunakan *sandwich panel* yang ramah lingkungan (Fauziah, 2019). Pada bulan Juni 2019 lalu, LRT Jakarta mengadakan uji coba publik melalui registrasi secara *online* tanpa dipungut biaya. Tercatat bahwa total penumpang LRT Jakarta yang telah mengikuti uji coba publik sebanyak 242.791 orang terhitung sejak uji coba dimulai hingga bulan Juli 2019 berdasarkan data LRT Jakarta yang disampaikan oleh *Corporate Communications Manager* (Halwi & Dewi, 2019). Agar pelayanan LRT Jakarta ke depannya dapat lebih baik dan menjadi moda transportasi umum yang semakin diminati, mereka menyediakan wadah bagi masyarakat untuk memberikan *feedback* berupa masukan maupun kritik ke dalam media sosial LRT Jakarta. Salah satu cara untuk membantu mengetahui *feedback* masyarakat apakah mereka menyukai atau tidak dari layanan LRT Jakarta adalah dengan melakukan analisis sentimen.

Analisis sentimen adalah bidang studi yang melakukan analisis terhadap pendapat, emosi, sikap yang diutarakan seseorang terhadap produk, pelayanan, organisasi dan hal lainnya. Analisis sentimen berfokus pada pendapat yang mengindikasikan sentimen positif maupun negatif. Kapanpun keputusan akan dibuat baik saat hendak membeli barang maupun menggunakan suatu layanan, opini orang lain pasti akan diperhatikan terlebih dahulu (Liu, 2012). Maka dari itu, *feedback* berupa komentar maupun tulisan dari masyarakat kemudian dianalisis secara otomatis untuk mengetahui penilaian mereka terhadap uji coba publik LRT Jakarta agar pelayanan LRT dapat ditingkatkan secara terus menerus serta menjadi pertimbangan berikutnya bagi pemerintah ketika ingin membangun fasilitas baru.

Terdapat beberapa pendekatan *Machine Learning* dalam melakukan analisis sentimen untuk mengetahui emosi dari seseorang terhadap suatu kejadian maupun kebijakan dengan *Supervised Learning* yaitu menggunakan *Decision Tree*, *Support Vector Machine* (SVM), *K-Nearest Neighbor*, *Neural Network*, *Naïve Bayes*, *Maximum Entropy* (Walaa, et. al. 2014) . Penelitian sebelumnya terkait analisis sentimen pada Twitter sebanyak 1000 tweets menggunakan *K-Nearest Neighbor* dan *Support Vector Machine* dengan normalisasi dan 4 fitur didapatkan nilai *precision* sebesar 85%, *recall* sebesar 75%, *f-score* sebesar 79%, dan akurasi

sebesar 80,80% untuk algoritme KNN dan *precision* sebesar 56%, *recall* sebesar 69%, *f-score* sebesar 61%, dan akurasi sebesar 58,79% untuk algoritme SVM. Terlihat bahwa algoritme KNN memiliki hasil yang lebih baik dibandingkan SVM dengan jumlah data dan perlakuan yang sama (Huq, et.al., 2017). Hasil ini dapat bervariasi tergantung banyaknya fitur serta data yang digunakan, seperti SVM yang memiliki kelemahan jika jumlah fitur yang digunakan sedikit atau jumlah dimensi yang kecil.

Di sisi lain, algoritme KNN memiliki kekurangan yaitu sensitif dengan nilai k untuk jumlah anggota yang berbeda pada tiap kategori baik positif maupun negatif. Kekurangan tersebut diperbaiki oleh *Improved K-Nearest Neighbor* (*Improved KNN*) dimana memiliki nilai k baru atau n yang berbeda pada tiap kategori. Rata-rata hasil *f1 score* yang diperoleh dari *Improved KNN* 1,4% lebih tinggi dibandingkan KNN. Performa *Improved KNN* lebih stabil dilihat dari penurunan nilai *f1-score* ketika nilai k yang diuji semakin tinggi dengan rentang nilai k dimulai dari 5 hingga 60 (Baoli, et.al., 2003). Fitur yang dihasilkan dari data berupa teks sangat banyak, dibutuhkan pemilihan fitur yang tepat agar *training* dapat berjalan lebih cepat dan hasil klasifikasi yang lebih akurat. Beberapa teknik pemilihan fitur untuk digunakan oleh algoritme saat proses *training* yaitu *Chi-square*, *Information Gain* (IG) dan *Mutual Information* (MI) yang diterapkan pada KNN. Penelitian tersebut menggunakan 100 fitur dengan teks yang telah dilakukan *stemming* dan diperoleh bahwa seleksi fitur menggunakan IG memperoleh nilai akurasi sebesar 74,47% sedangkan menggunakan *Chi-square* sebesar 70,11% dan MI sebesar 38,22% (Jodha, et.al., 2018).

Berdasarkan permasalahan yang dikemukakan dan penelitian sebelumnya dan maka penulis melakukan penelitian dengan judul ANALISIS SENTIMEN MASYARAKAT TERHADAP UJI COBA LRT JAKARTA MENGGUNAKAN IMPROVED K-NEAREST NEIGHBOR DAN INFORMATION GAIN dengan harapan dapat membantu pemerintah untuk meningkatkan mutu fasilitas berdasarkan penilaian masyarakat.

1.2 Rumusan Masalah

Berdasarkan latar belakang dan beberapa penelitian sebelumnya terkait sentimen masyarakat, diperoleh rumusan masalah sebagai berikut.

1. Bagaimana pengaruh nilai k terhadap evaluasi *f-measure* dari *Improved K-Nearest Neighbor*?
2. Bagaimana pengaruh *threshold* pada seleksi fitur *Information Gain* terhadap hasil *f-measure* dan *running time* dalam klasifikasi sentimen pada LRT Jakarta?

1.3 Tujuan

Tujuan diadakannya penelitian ini yaitu.

1. Menjelaskan pengaruh nilai ketetanggaan atau k terhadap evaluasi berdasarkan *f-measure* dari metode *Improved K-Nearest Neighbor*.

2. Menjelaskan pengaruh *threshold* atau jumlah fitur yang digunakan *Information Gain* sebagai salah satu metode seleksi fitur terhadap *f-measure* dalam klasifikasi dengan *Improved K-Nearest Neighbor*.

1.4 Manfaat

Adapun manfaat dari diadakannya penelitian ini yaitu.

1. Dapat mengetahui seberapa besar peran nilai ketetanggaan dalam mempengaruhi hasil evaluasi *f-measure* dalam klasifikasi sentimen.
2. Dapat mengetahui seberapa besar peran *threshold* dari seleksi fitur *Information Gain* terhadap hasil evaluasi *f-measure* dalam klasifikasi sentimen.

1.5 Batasan Masalah

Dalam penelitian ini, penulis menjabarkan beberapa batasan masalah yaitu.

1. Penelitian ini menggunakan data komentar dari Youtube serta tulisan dari Facebook dan Twitter.
2. Data yang digunakan dalam penelitian dikhususkan pada data berbahasa Indonesia.
3. Dalam penelitian ini, emoji tidak diikutsertakan sebagai penilaian untuk menentukan sentimen.
4. Pada penelitian ini, hasil klasifikasi sentimen berupa kelas positif atau negatif.
5. Jumlah data yang digunakan sebanyak 500 data.

1.6 Sistematika Pembahasan

Penelitian ini memiliki sistematika pembahasan yang tersusun atas beberapa bab yaitu.

BAB I PENDAHULUAN

Bab ini berisi mengenai latar belakang, rumusan masalah, tujuan, manfaat dan batasan masalah sesuai dengan kemampuan penulis dalam penelitian ini.

BAB II LANDASAN KEPUSTAKAAN

Bab ini mengandung kajian pustaka berisi penelitian sebelumnya yang serupa, serta teori pendukung penelitian ini seperti *preprocessing*, *term weighting*, *cosine similarity (cossim)*, *Improved KNN* maupun tabel *confusion matrix*.

BAB III METODOLOGI

Bab ini menjelaskan terkait jenis penelitian yang diterapkan, strategi penelitian berupa alur sistem yang digambarkan dalam *flowchart*, lokasi penelitian dilaksanakan, komponen pendukung yang meliputi *software* dan *hardware*,

metode pengumpulan data hingga analisis hasil dari perhitungan manual maupun *source code*.

BAB IV PERANCANGAN

Bab ini menjelaskan terkait tahap perancangan yang digunakan dalam penelitian ini.

BAB V IMPLEMENTASI

Bab ini berisi penerapan algoritme *Improved KNN* dengan *Information Gain* untuk menyeleksi fitur agar mengetahui sentimen masyarakat.

BAB VI PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan pengujian algoritme *Improved KNN* serta *Information Gain* terhadap akurasi dari segi akurasi, *precision*, *recall* dan *f-score* serta analisis berdasarkan hasil implementasi.

BAB VII: PENUTUP

Bab ini menjelaskan kesimpulan serta saran yang dapat dijadikan bahan referensi untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini terdiri dari kajian pustaka dan dasar teori untuk mendukung penelitian ini. Kajian pustaka berisi penelitian yang telah ada, dengan topik yang sama namun menggunakan metode berbeda serta metode yang serupa pada kasus yang berbeda. Dasar teori berisi teori yang mendukung dan digunakan dalam penelitian ini.

2.1 Kajian Pustaka

Penelitian sebelumnya terkait hasil dari algoritme *Improved K-Nearest Neighbor* dilakukan oleh Nathania et.al. (2018). Penelitian tersebut bertujuan untuk melakukan klasifikasi *spam* pada *tweets* dengan membuat lima skenario di mana masing-masing skenario memiliki nilai k awal dan jumlah data latih yang berbeda. Berdasarkan hasil dari lima skenario tersebut, diperoleh hasil rata-rata *precision* sebesar 89,46%, *recall* sebesar 94,05%, *f-measure* sebesar 91,55%, akurasi sebesar 89,57%.

Penelitian terkait analisis sentimen juga dilakukan oleh Huq, et.al. (2017) menggunakan algoritme *K-Nearest Neighbor* dan *Support Vector Machine* (SVM) pada data Twitter sebanyak 1000 *tweets*. Penelitian tersebut menggunakan beberapa jenis fitur yaitu *word feature*, *n-gram feature*, *pattern feature*, *punctuation feature* dan *key based feature*. Berdasarkan seluruh fitur yang digunakan, didapat hasil akurasi pada algoritme KNN dengan normalisasi serta menggunakan lima fitur sebesar 84,32%, *recall* sebesar 81%, *precision* sebesar 85% dan *f-score* sebesar 83%. Sedangkan pada algoritme SVM diperoleh hasil akurasi sebesar 77,97%, *precision* sebesar 72%, *recall* sebesar 89%, dan *f-score* sebesar 80% dengan perlakuan normalisasi data, menggunakan lima fitur ditambah *Grid Search* untuk mencari pasangan nilai c dan gamma terbaik.

Jodha et.al. (2018) melakukan penelitian mengenai klasifikasi pada teks menggunakan KNN dengan membandingkan berbagai seleksi fitur yaitu *Chi-square*, *Information Gain*, dan *Mutual Information*. Dataset yang digunakan yaitu 20 Newsgroup yang dibagi menjadi 60% data latih dan 40% data uji. Teks tersebut dilakukan *stemming* dengan metode *Lancaster Stemmer*. Hasil terbaik diperoleh menggunakan *Information Gain* dengan *Lancaster Stemmer* dengan akurasi sebesar 74,47%, *precision* 76,63%, *recall* sebesar 74,47% dan *f1-score* sebesar 75,14%.

Penelitian terkait analisis sentimen juga dilakukan oleh Sudheer & Valarmathi (2018) yang membandingkan algoritme *Naïve Bayes*, *Decision Tree* dan *Maximum Entropy*. Dataset yang digunakan pada penelitian tersebut berasal dari situs Amazon, E-bay dan Alibaba dengan jumlah fitur berada pada rentang 200 hingga 1000. Hasil terbaik yang diperoleh pada seleksi fitur menggunakan *document frequency* dan 1000 fitur pada masing-masing situs *e-commerce* dengan rata-rata akurasi yang diperoleh sebesar 91,65%.

Penelitian terkait algoritme *Improved K-Nearest Neighbor* (IKNN) dilakukan oleh Baoli dengan dataset dari Universitas Peking. Penelitian tersebut membandingkan antara KNN yang disebut sebagai kNN-A dengan IKNN yang disebut dengan kNN-B. Secara rata-rata keseluruhan, hasil evaluasi dari KNN-B 1,4% lebih tinggi dari KNN-A dilihat dari *precision*, *recall* dan *f1-score*. Tabel 2.1 menjelaskan mengenai penelitian sejenis yang telah dilakukan sebelumnya.

Tabel 2.2.1 Tabel Kajian Pustaka

Judul Pustaka	Objek Penelitian	Metode	Hasil
Klasifikasi Spam Pada Twitter Menggunakan Metode <i>Improved K-Nearest Neighbor</i> (Nathania et.al., 2018)	Data Twitter	<i>Improved K-Nearest Neighbor</i>	Memperoleh rata-rata <i>precision</i> yaitu 89,46%, <i>recall</i> 94,05%, <i>F-Measure</i> 91,55% dan <i>akurasi</i> sebesar 89,57%.
<i>Sentiment Analysis on Twitter Data using KNN and SVM</i> (Huq, et.al., 2017)	Data Twitter	<i>K-Nearest Neighbor</i> dan <i>Support Vector Machine</i>	Pada metode KNN dengan normalisasi ditambah 5 teknik fitur, diperoleh akurasi sebesar 84,32%, <i>recall</i> 0.81, <i>precision</i> 85% dan <i>f-score</i> 83%. Sedangkan pada metode SVM dengan normalisasi dan 5 teknik fitur ditambah <i>Grid Search</i> , diperoleh akurasi sebesar 77,97%, <i>precision</i> 72%, <i>recall</i> 89% dan <i>f-score</i> 80% dari 1000 tweets.

Tabel 2.2.2 Tabel Kajian Pustaka (Lanjutan)

<i>Text Classification using KNN with different Features Selection Methods</i> (Jodha, et.al., 2018)	20 Newsgroup Dataset berisi mengenai berita	<i>K-Nearest Neighbor</i> dengan membandingkan beberapa seleksi fitur yaitu <i>chi square</i> , <i>information gain</i> , dan <i>mutual information</i> .	Hasil evaluasi terbesar yaitu saat menggunakan 100 fitur, metode <i>Lancaster Stemmer</i> untuk <i>stemming</i> dan <i>Information Gain</i> sebagai seleksi fitur. Akurasi terbesar diperoleh yaitu 74.47%, <i>precision</i> 76.63%, <i>recall</i> 74.47% dan <i>f1-score</i> sebesar 75.14%.
<i>Real Time Sentiment Analysis of E-commerce Websites using Machine Learning Algorithms</i> (Sudheer & Valarmathi, 2018)	<i>Realtime Data</i> dari Twitter pada situs Amazon, E-bay dan Alibaba	<i>Naïve Bayes</i> , <i>Decision Tree</i> dan <i>Maximum Entropy</i>	Hasil terbaik diperoleh pada seleksi fitur dengan <i>document frequency</i> dan menggunakan 1000 fitur pada masing-masing situs <i>e-commerce</i> . Rata-rata akurasi yang diperoleh sebesar 91.65%.
<i>An Improved k-Nearest Neighbor Algorithm for Text Categorization</i> (Baoli, et.al., 2003)	Dataset dari Universitas Peking	<i>K-Nearest Neighbor</i> yang disebut sebagai kNN-A dan <i>Improved K-Nearest Neighbor</i> yang disebut sebagai kNN-B	Secara rata-rata, hasil evaluasi dari KNN-B 1.4% lebih tinggi dari KNN-A dilihat dari <i>precision</i> , <i>recall</i> dan <i>f1-score</i> .

2.1.1 Analisis Sentimen

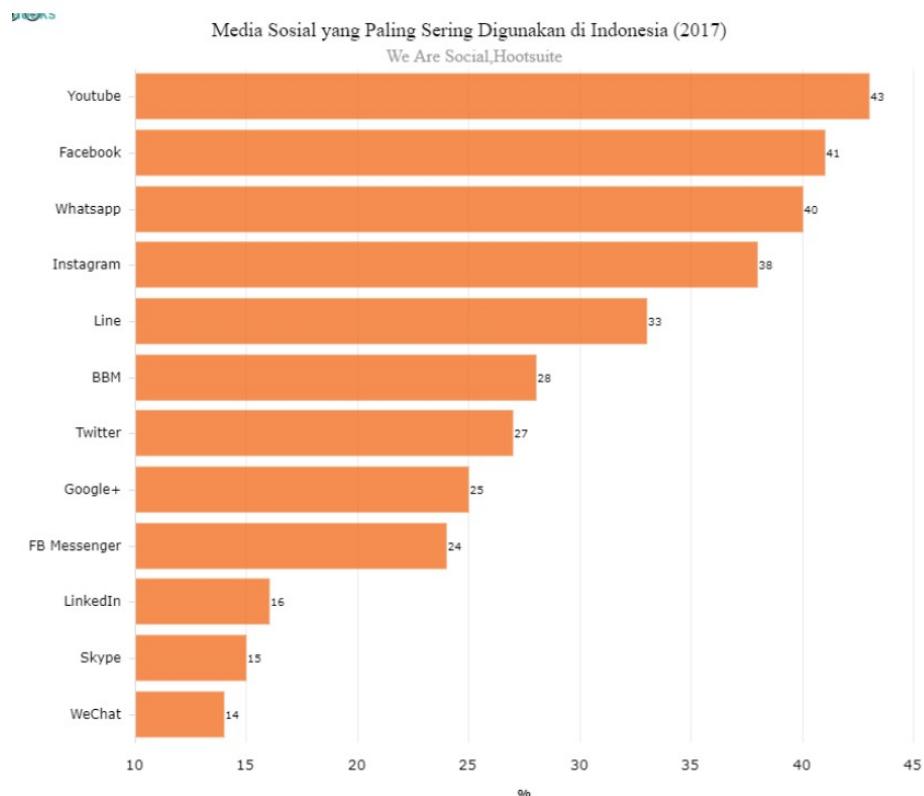
Analisis sentimen adalah bidang studi yang melakukan analisis terhadap pendapat, emosi, sikap yang diutarakan seseorang terhadap produk, pelayanan, organisasi dan hal lainnya. Analisis sentimen berfokus pada pendapat yang mengindikasikan sentimen positif maupun negatif. Berbagai jenis keputusan dapat dibuat baik saat hendak membeli barang maupun menggunakan suatu layanan, opini orang lain pasti akan diperhatikan terlebih dahulu (Liu, 2012). Penggunaan analisis sentimen tidak hanya pada *domain* bisnis dan penilaian terhadap produk

namun juga diterapkan pada bidang lain diantaranya sebagai berikut (Bhandari and Ghosh, 2016):

- a. *Cyber Security*
- b. Prediksi pasar saham
- c. Mengukur kesejahteraan seseorang

2.1.2 Media Sosial

Media sosial merupakan wadah yang mengizinkan untuk membuat profil, dan hubungan yang terbuka antar pengguna (Boyd & Ellison, 2008). Kietzmann et. al. (2011) mendefinisikan media sosial sebagai aplikasi berbasis *website* yang menyediakan layanan untuk berbagi, hubungan pertemanan, grup hingga profil. Adapun media sosial merupakan sekumpulan teknologi informasi yang memberikan fasilitas dalam berinteraksi dan bersosialisasi (Kapoor et.al., 2017).



Gambar 2.1 Pengguna Media Sosial di Indonesia

Sumber: <https://databoks.katadata.co.id/datapublish/2018/02/01/media-sosial-apa-yang-paling-sering-digunakan-masyarakat-indonesia>

Berdasarkan data pengguna media sosial di Indonesia pada Gambar 2.1, pengguna media sosial Facebook mencapai 41%, Instagram mencapai 38% dan Twitter mencapai 27%. Ini mencerminkan bahwa ketiga sosial media tersebut ramai digunakan oleh masyarakat sebagai alat untuk membagikan opini, ide hingga kejadian terkini.

2.1.3 Light Rail Transit (LRT)

LRT merupakan salah satu transportasi yang menggunakan rel (Rudi, 2015). Kecepatan LRT berkisar antara 70 hingga 80 km per jam. LRT dapat mengangkut 278 penumpang pada setiap rangkaian kereta. Proyek LRT yang dikelola oleh Pemerintah Provinsi Jakarta bernama LRT Jakarta sedangkan LRT Jabodebek dikelola oleh PT Adhi Karya Tbk. yang menghubungkan antara Jakarta dengan daerah disekitarnya. LRT Jakarta menggunakan 750 volt DC sebagai daya pada rel listrik (Widowati, 2019).

2.2 Preprocessing pada Teks

Sebelum dokumen diolah untuk dilakukan analisis sentimen, dokumen tersebut perlu untuk diproses terlebih dahulu untuk mempermudah ketika diolah dan menghasilkan akurasi yang lebih tinggi yang dikenal sebagai tahap *preprocessing* (Angiani, et.al., 2016). Tahapan dalam *preprocessing* diantaranya.

2.2.1 Cleaning

Tahap ini merupakan proses yang pada umumnya untuk membersihkan data dari tanda baca, *url*, *tag HTML* maupun *emoticon*. Data berupa tanda baca (*punctuation*), angka, dan alamat *website*, simbol *hashtag* (#) dan *username* seperti “@Transit” dihapus (Angiani, et.al., 2016). Selain itu, karakter tunggal juga akan dihilangkan (Al-Khafaji & Habeeb, 2017). Contohnya yaitu “Yuk jaga kebersihan #cintaIndonesia.” menjadi “Yuk jaga kebersihan”.

2.2.2 Case Folding

Pada tahap *case folding*, teks akan dirubah menjadi huruf kecil. Hal ini ditujukan agar kedua kata dianggap sama saat perhitungan kemiripan dokumen. Contohnya yaitu “Ani sedang memasak Nasi Goreng” menjadi “ani sedang memasak nasi goreng”. Jika tidak dilakukan *case folding* maka kata “Ani” dan “ani” akan dianggap berbeda, sehingga tingkat kemiripan dokumen akan turun.

2.2.3 Tokenization

Tokenization atau dalam Bahasa Indonesia disebut tokenisasi merupakan proses untuk merubah tiap kalimat dalam dokumen menjadi kata yang memenuhi kondisi pada *regular expression*. Proses tokenisasi meliputi pemisahan kalimat menjadi kata, frasa, symbol yang disebut *token* (Jodha, et.al, 2018).

2.2.4 Stemming

Stemming merupakan tahap *preprocessing* pada teks yang bertujuan untuk mencari bentuk dasar (*root*) dari sebuah kata (Jodha, et.al, 2018). *Stemming* juga dapat digunakan untuk mengurangi fitur dari kata yang serupa namun dianggap berbeda karena memiliki imbuhan. Dimisalkan terdapat dokumen dengan frekuensi kata sebagai berikut; meraba: 4, teraba: 2, dan raba: 4 jika dilakukan *stemming* maka kata “meraba” dan “teraba” akan menjadi “raba” dengan

kemunculan kata sebanyak 12 karena ketiga kata tersebut memiliki makna yang sama hanya memiliki imbuhan.

Teknik *stemming* berbasis Python yang digunakan pada penelitian ini yaitu Sastrawi. Sastrawi merupakan *library* yang digunakan untuk mereduksi kata berimbuhan menjadi bentuk kata dasar dalam Bahasa Indonesia (Robbani, 2016).

2.3 Term Weighting

Pembobotan kata (*term weighting*) merupakan teknik untuk merubah teks menjadi numerik yang merepresentasikan kemunculan tiap kata dalam dokumen. Salah satu metode *term weighting* yaitu *Term Frequency – Inverse Document Frequency* (*tf.idf*) (Jones, 1972). *tf.idf* merupakan pembobotan *unsupervised* yang merupakan hasil perkalian dari *term frequency* dengan *inverse document frequency*. Rumus untuk menghitung log *tf* dijelaskan pada Persamaan 2.1.

$$tf_{t,d} = 1 + \log(f_{t,d}) \quad (2.1)$$

Lalu *idf* dihitung dengan Persamaan 2.2.

$$idf(t) = \frac{\log(N)}{df_t} \quad (2.2)$$

Setelah mendapatkan nilai *idf* maka untuk menghitung *tf.idf* dijelaskan pada Persamaan 2.3.

$$tf.idf(t) = tf_{t,d} * idf(t) \quad (2.3)$$

Dimana:

- $f_{t,d}$ merupakan tingkat kemunculan term t pada dokumen d.
- $\log(N)$ merupakan nilai logaritmik basis 10 dari jumlah semua dokumen.
- df_t merupakan banyaknya dokumen yang mengandung term t.

2.4 Improved K-Nearest Neighbor

Algoritme *K-Nearest Neighbor* nilai *k* untuk tiap kelas akan sama tanpa melihat jumlah dokumen latih dari tiap kelas. Hal tersebut akan berpengaruh terhadap hasil prediksi sebuah dokumen ketika anggota tiap kelas tidak seimbang dimana sebuah dokumen akan cenderung masuk ke dalam *majority class* atau dalam Bahasa Indonesia disebut kelas dengan anggota lebih banyak dibandingkan ke dalam *minority class* atau dalam Bahasa Indonesia disebut kelas dengan anggota lebih sedikit. Sehingga diperlukan adaptasi nilai *k* pada tiap kelas agar hasil klasifikasi tidak terpengaruh pada kelas dengan anggota yang lebih banyak.

Improved K-Nearest Neighbor (*Improved KNN*) merupakan salah satu varian dari *K-Nearest Neighbor* yang memperhatikan nilai *k* pada tiap kelas/kategori dengan tujuan memperbaiki kekurangan dari algoritme KNN. Sebagian besar data pada kenyataannya memiliki jumlah anggota yang berbeda pada tiap kelas atau kategori. Akibatnya, *majority class* akan menguasai hasil klasifikasi dibandingkan *minority class*. Maka dari itu, nilai *k* pada *Improved K-Nearest Neighbor* berbeda

pada tiap kelas dibandingkan menggunakan nilai k yang tetap untuk semua kelas (Baoli, et. al., 2003). Penentuan nilai k baru atau n pada algoritme *Improved KNN* dijelaskan dengan Persamaan 2.3.

$$n = \left[\frac{k \times N(c_m)}{\max \{N(c_i) | i=1 \dots N_c\}} \right] \quad (2.3)$$

Dimana:

- n merupakan nilai k baru yang berbeda pada tiap kategori
- k merupakan nilai tetangga terdekat sebelumnya
- $N(c_m)$ merupakan jumlah dokumen latih pada kelas/kategori m
- $\max \{N(c_i) | i = 1 \dots N_c\}$ merupakan jumlah dokumen latih terbanyak pada seluruh kategori.

Setelah mendapatkan nilai k baru, penentuan kelas untuk tiap dokumen uji menggunakan perhitungan kemiripan antara dokumen latih pada tiap kelas dengan dokumen uji. Selanjutnya, nilai kemiripan tersebut digunakan untuk menentukan peluang tiap dokumen uji terhadap kelas m berdasarkan nilai k baru atau n tetangga terdekat yang berbeda pada tiap kelas. Perhitungan tersebut dijelaskan pada Persamaan 2.4.

$$p(d_i, c_m) = \operatorname{argmax}_m \frac{\sum_{x_j \in \text{top_}n\text{-}kNN(c_m)} \text{similarity}(d_i, x_j) y(x_j, c_m)}{\sum_{x_j \in \text{top_}n\text{-}kNN(c_m)} \text{similarity}(d_i, x_j)} \quad (2.4)$$

Dimana:

- $p(d_i, c_m)$ merupakan peluang dokumen uji ke-i masuk ke dalam kelas c_m
- $\text{similarity}(d_i, x_j)$ merupakan nilai kemiripan antara dokumen uji ke-i dengan dokumen latih ke-j
- $\text{top_}n\text{-}kNN(c_m)$ merupakan n tetangga terdekat pada kelas c_m
- $y(x_j, c_m)$ merupakan fungsi dengan aturan sebagai berikut:
 $y(x_j, c_m)$ bernilai 1 jika dokumen latih ke-j merupakan anggota kelas c_m
 $y(x_j, c_m)$ bernilai 0 jika dokumen latih ke-j bukan merupakan anggota kelas c_m

Adapun dalam melakukan klasifikasi dokumen menggunakan algoritme *Improved KNN* dapat dijabarkan melalui langkah-langkah sebagai berikut.

- Melakukan *preprocessing* terhadap dokumen uji dan dokumen latih hingga didapatkan *term* atau kata dasar.
- Menghitung *term weighting* menggunakan *tf.idf* pada tiap *term* untuk dokumen uji dan dokumen latih yang akan membentuk vektor hasil pembobotan.
- Menggunakan *cosine similarity* untuk menghitung kemiripan antara dokumen uji dan dokumen latih lalu diurutkan secara *descending* atau

menurun. Semakin tinggi nilai *cosine similarity* maka tingkat kemiripan antara kedua dokumen tersebut semakin tinggi.

- Mengelompokkan hasil perhitungan *cosine similarity* berdasarkan kelas.
- Menentukan nilai awal k seperti pada algoritme KNN.
- Menghitung nilai k baru untuk masing-masing kelas menggunakan Persamaan 2.3.
- Menghitung peluang dokumen uji terhadap masing-masing kelas menggunakan Persamaan 2.4 setelah mendapatkan nilai k baru yang merupakan tetangga terdekat.
- Melakukan klasifikasi dokumen ke dalam sebuah kelas berdasarkan nilai $p(d_i, c_m)$ tertinggi.

2.5 Cosine Similarity

Cosine Similarity merupakan metrik yang digunakan untuk mengukur nilai sudut cosinus antara vektor term dari dua dokumen (Salton & Buckley, 1988). Semakin tinggi nilai cosinus atau mendekati 1, maka relevansi (kemiripan) antara kedua dokumen tersebut semakin besar. Rumus untuk menghitung nilai *cosine similarity* dijelaskan pada Persamaan 2.5

$$cosSim(\vec{d}, \vec{q}) = \frac{\sum_{i=1}^W v_{di} * v_{qi}}{\sqrt{\sum_{i=1}^W (v_{di})^2} \sqrt{\sum_{i=1}^W (v_{qi})^2}} \quad (2.5)$$

Dimana:

- v_{di} merupakan vektor yang berisi term pada dokumen latih.
- v_{qi} merupakan vektor yang berisi term pada dokumen uji (query).

2.6 Information Gain

Information Gain (IG) merupakan metode evaluasi fitur berbasis *entropy* yang umum digunakan dalam permasalahan *Machine Learning*. IG digunakan sebagai seleksi fitur yang merupakan banyaknya informasi pada tiap fitur atau term terhadap kategori (Lei, 2012). Nilai tiap fitur yang merepresentasikan seberapa baik dalam membedakan aspek penting dari kategori sentimen dapat diketahui melalui *Information Gain* (Schouten, et.al., 2016). IG dihitung berdasarkan seberapa besar pengaruh sebuah term digunakan untuk klasifikasi (Lei, 2016). Perhitungan IG dilakukan pada setiap term dari data latih, sebuah term akan dihapus jika nilai IG term tersebut lebih rendah dibandingkan *threshold* yang telah ditentukan (Wang, et.al., 2006). Persamaan 2.6 menunjukkan perhitungan *Information Gain*.

$$IG(t) = -\sum_{i=1}^m P(C_i) \log P(C_i) + P(t) \sum_{i=1}^m P(C_i|t) \log (P(C_i|t)) + P(\bar{t}) \sum_{i=1}^m P(C_i|\bar{t}) \log P(C_i|\bar{t}) \quad (2.6)$$

Dimana:

- t merupakan term unik.
- m merupakan banyaknya kategori atau kelas.
- $IG(t)$ merupakan nilai *Information Gain* dari term t .
- $P(C_i)$ merupakan peluang dari kelas C_i yang dihitung berdasarkan bagian dokumen yang termasuk kelas C_i .
- $P(t)$ merupakan peluang kemunculan dari term t .
- $P(C_i|t)$ merupakan peluang kelas C_i dengan syarat kemunculan term t yang dihitung berdasarkan bagian dokumen dengan kelas C_i yang memiliki kemunculan term t minimal satu kali.
- $P(\bar{t})$ merupakan peluang term t tidak muncul.
- $P(C_i|\bar{t})$ merupakan peluang kelas C_i dengan syarat tidak mengandung term t yang dihitung berdasarkan bagian dokumen dengan kelas C_i namun tidak memiliki term t .

2.7 Confusion Matrix

Confusion matrix berisi informasi hasil evaluasi dari kelas sebenarnya (*actual class*) dan kelas hasil prediksi (*predicted class*) yang dilakukan oleh algoritme klasifikasi (Kohavi and Provost, 1998). Berikut merupakan tabel dari *confusion matrix*.

Tabel 2.2.3 Confusion Matrix

		Kelas hasil prediksi	
		Negatif	Positif
Kelas sebenarnya	Negatif	TN	FP
	Positif	FN	TP

Dari tabel diatas, dapat dijabarkan sebagai berikut.

- *True Negative* (TN) merupakan jumlah dokumen yang diklasifikasikan sebagai kelas negatif dan kelas sebenarnya juga negatif.
- *False Positive* (FP) merupakan jumlah dokumen yang diklasifikasikan sebagai kelas positif namun sebenarnya adalah kelas negatif.
- *False Negative* (FN) merupakan jumlah dokumen yang diklasifikasikan sebagai kelas negative namun sebenarnya adalah kelas positif.
- *True Positive* (TP) merupakan jumlah dokumen yang diklasifikasikan sebagai kelas positif dan kelas sebenarnya juga positif.

Pada *confusion matrix* juga dapat dihitung metrik seperti *precision*, *recall*, maupun *f-measure* untuk mengevaluasi hasil klasifikasi dari metode yang digunakan.

Recall merupakan bagian dari dokumen yang relevan dan berhasil diperoleh (*retrieve*) dari seluruh dokumen yang relevan. *Recall* memiliki perhitungan yang ditunjukkan oleh Persamaan 2.9.

$$recall = \frac{TP}{TP+FN} \quad (2.9)$$

Precision merupakan bagian (fraksi) dari dokumen yang relevan dari seluruh dokumen yang diperoleh. Rumus untuk menghitung *precision* dituliskan pada Persamaan 2.10.

$$precision = \frac{TP}{TP+FP} \quad (2.10)$$

F-measure merupakan metrik pengukuran yang menggabungkan *precision* dan *recall* dengan rumus yang dijelaskan pada Persamaan 2.11.

$$fmeasure = \frac{2*precision*recall}{precision+recall} \quad (2.11)$$

2.8 K-fold Cross Validation

Secara umum, *cross validation* merupakan metode yang digunakan untuk membandingkan dan mengevaluasi algoritme dengan membagi dataset menjadi dua bagian yaitu data untuk dipelajari oleh algoritme dan data yang digunakan untuk memvalidasi algoritme. Pada *k-fold cross validation*, dataset dibagi sama rata atau mendekati sama rata menjadi lipatan atau segmen sebanyak *k*. Selanjutnya pada tiap iterasi sejumlah *k-1* lipatan digunakan untuk tahap pembelajaran sedangkan sisanya digunakan untuk *validation* (Refaeilzadeh, et al., 2009). Menurut Kohavi (1995) *k-fold cross validation* juga disebut sebagai *rotation estimation* yang membagi dataset secara *random* menjadi *subset* atau himpunan bagian yang saling ekslusif dengan ukuran yang hampir sama pada tiap bagian. Ilustrasi dari *k-fold cross validation* ditunjukkan pada Gambar 2.2



Gambar 2.2 Ilustrasi *K-Fold Cross Validation*

Sumber: <http://www.ebc.cat/2017/01/31/cross-validation-strategies/#k-fold>
(2017)

Berdasarkan Gambar 2.2 dapat dilihat setiap grup yang berisi $k-1$ lipatan untuk pembelajaran ditandai dengan warna biru dan sisanya untuk validasi yang ditandai dengan warna oranye memiliki nilai evaluasi masing-masing. Untuk menghitung nilai evaluasi akhir maka dihitung rata-rata dari seluruh evaluasi pada tiap grup (Neale, et al., 2019).

BAB 3 METODOLOGI

Bab ini membahas mengenai metodologi yang diterapkan pada penelitian ini dimulai dari tipe penelitian, strategi, subjek penelitian sebagai objek yang difokuskan dalam mengambil data, lokasi penelitian dilaksanakan, teknik yang digunakan dalam mengumpulkan data, sumber data yang digunakan , teknik untuk menganalisis data hingga penyusunan jadwal penelitian.

3.1 Tipe Penelitian

Penelitian ini menggunakan tipe non implementatif analitik yang memiliki definisi yaitu penelitian yang menguji keterkaitan antara variabel *dependent* dengan variabel *independent* dimana dalam kasus ini adalah menganalisis hubungan antara fitur pada dokumen dari media sosial terhadap LRT Jakarta dengan kelas sentimen.

3.2 Strategi Penelitian

Strategi pada penelitian ini menggunakan studi eksperimen pada sentimen masyarakat terhadap uji coba publik LRT Jakarta melalui media sosial. Studi eksperimen berfokus pada memberikan perlakuan tertentu pada objek penelitian seperti menguji nilai k , maupun *threshold* pada *Information Gain*.

3.3 Subjek Penelitian

Subjek yang difokuskan pada penelitian ini yaitu adalah pengguna media sosial Facebook , Youtube atau Twitter dari segala umur dengan syarat opini mereka membahas mengenai LRT Jakarta baik dari segi pelayanan, tarif, dan lainnya.

3.4 Lokasi Penelitian

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

3.5 Teknik Pengumpulan Data

Teknik pengumpulan data yang dilakukan pada penelitian ini yaitu studi dokumen dengan menggunakan data primer. Data yang digunakan bersumber dari tiga media sosial yakni Youtube, Twitter dan Facebook dengan pembagian 414 data untuk Youtube, 70 data untuk Twitter, dan 16 data Facebook. Data tersebut dilabeli oleh seorang pakar bernama Kusno Hermawan S.Pd yang merupakan Guru Bahasa Indonesia.

Data yang digunakan dalam Youtube melalui komentar pengguna pada konten mengenai uji coba LRT dengan memanfaatkan *scraper*. Kemudian data yang digunakan pada Twitter diambil dengan memanfaatkan API (*Application Programming Interface*) melalui fitur premium pada *Developer Account* dengan

mengambil *tweets* berisi tagar #LRTJakarta. Dataset tersebut dapat diakses pada <https://gist.github.com/okzapradhana/68557bb94cea930f1ad22467a575b720>

3.6 Teknik Analisis Data

Teknik analisis data pada penelitian ini berdasarkan hasil dari pengujian dari hasil perhitungan menggunakan *Improved K-Nearest Neighbor* dengan *Information Gain*. Pengujian dilakukan dengan menghitung *precision*, *recall*, *f-measure* dari setiap *fold* yang digunakan.

3.7 Peralatan Pendukung

Dalam pelaksanaan penelitian ini, peralatan pendukung yang akan digunakan yaitu *Hardware* dan *Software* dengan spesifikasi sebagai berikut.

Tabel 2.3.1 Spesifikasi Hardware

Spesifikasi	Keterangan
Laptop	Acer Aspire E14 E5-475G
CPU	Core i5
GPU	NVIDIA GeForce 940MX
RAM	4 GB
Tipe memori	DDR 4
SSD	Samsung SSD 860 EVO (250 GB)

Tabel 2.3.2 Spesifikasi Software

Jenis	Keterangan
<i>Operating System</i>	Windows 10
Bahasa pemrograman	Python 3.6.0
IDE	Jupyter dan Pycharm
Library	Sastrawi, Pandas, Numpy

3.8 Implementasi Algoritme

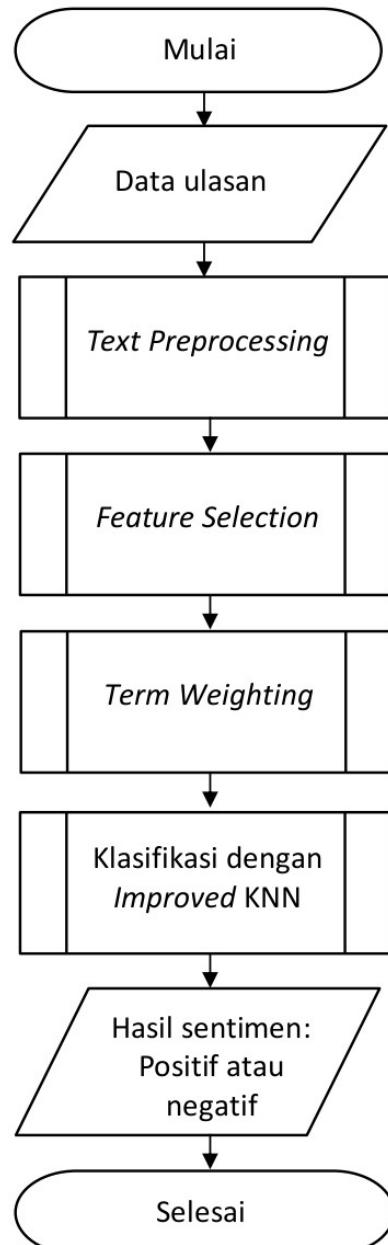
Tahap implementasi algoritme menerapkan *preprocessing* pada data untuk merubah dari kumpulan dokumen menjadi kumpulan *term* yang telah *distemming*. Lalu menggunakan pembobotan kata dengan *tf.idf* untuk merubah teks menjadi numerik dan memilih fitur yang penting menggunakan *Information Gain* yang selanjutnya dilatih oleh *Improved KNN* dan klasifikasi *query* / dokumen uji untuk mengetahui sentimen masyarakat dengan menerapkan *Cosine Similarity* untuk mencari kemiripan antara dokumen uji dengan dokumen latih.

BAB 4 PERANCANGAN

Pada bab ini menjelaskan perancangan diagram alir dari metode yang digunakan pada penelitian ini serta perhitungan manualisasi dari sistem klasifikasi dengan *Improved KNN* serta *Information Gain* sebagai seleksi fitur.

4.1 Diagram Alir Sistem

Diagram alir sistem memberikan penjelasan mengenai tahapan yang terjadi pada sistem. Proses pada sistem dijelaskan pada Gambar 4.1.

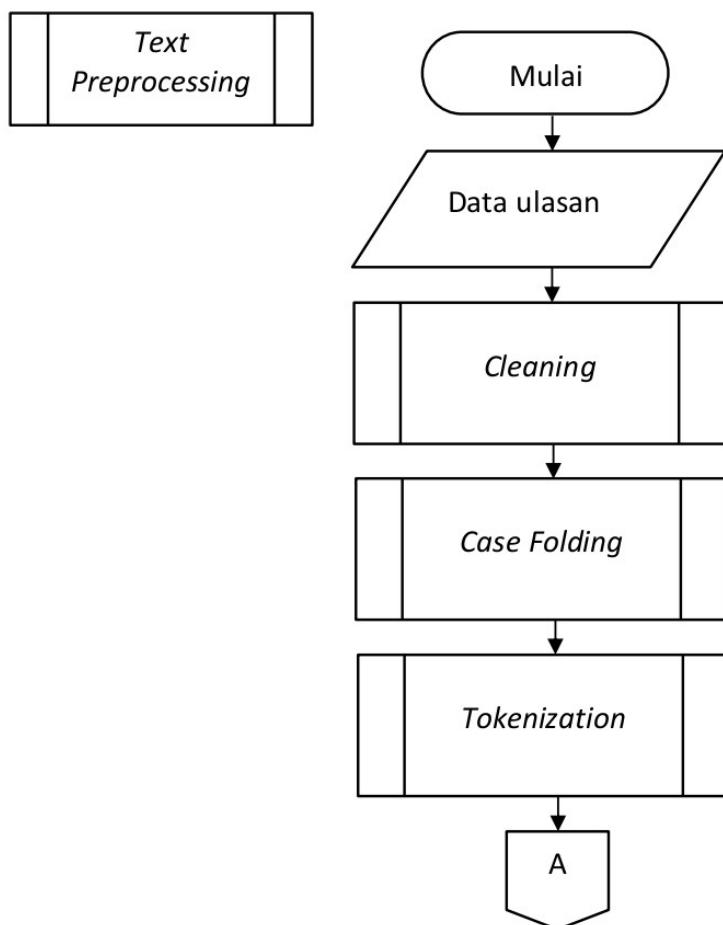


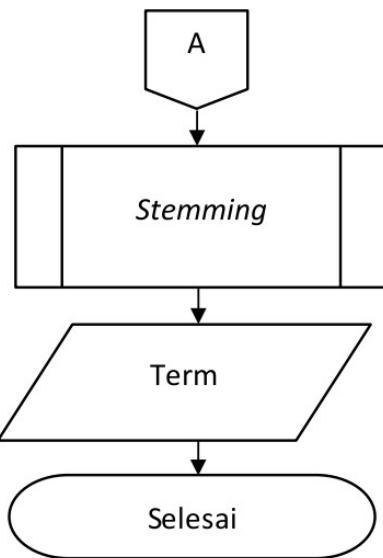
Gambar 4.1 Diagram Alir Sistem

Berdasarkan Gambar 4.1, proses yang terjadi pada sistem diawali dengan melakukan *preprocessing* pada teks ulasan masyarakat dari Youtube, Twitter dan Facebook. Selanjutnya *term* hasil *preprocessing* dilakukan proses seleksi fitur dengan *Information Gain*. Hasil *term* yang telah diseleksi dilakukan pembobotan dengan *term frequency inverse document frequency*. *Term* hasil pembobotan tersebut akan menjadi *term training* untuk kemudian diklasifikasikan menggunakan algoritme *Improved KNN*. Nilai peluang dari *Improved KNN* untuk masing-masing sentimen akan dibandingkan dan diambil nilai terbesar untuk kemudian menjadi kelas sentimen baru dari hasil prediksi.

4.1.1 Diagram Alir *Text Preprocessing*

Diagram alir ini menjelaskan proses *text preprocessing* untuk mendapatkan data yang bersih dan representasi vektor kemunculan term pada dokumen. Tahapan yang dilakukan pada *text preprocessing* dijelaskan melalui diagram alir pada Gambar 4.2.



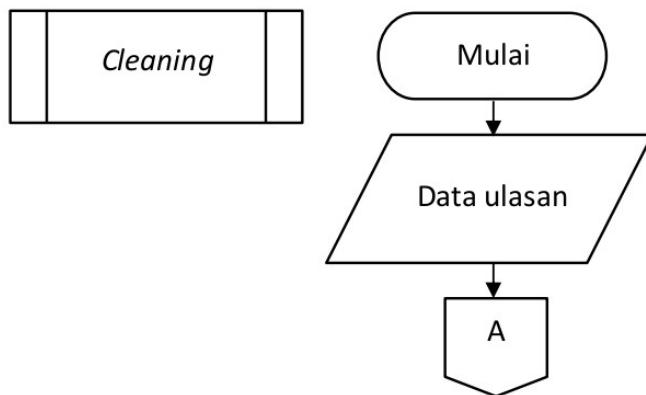


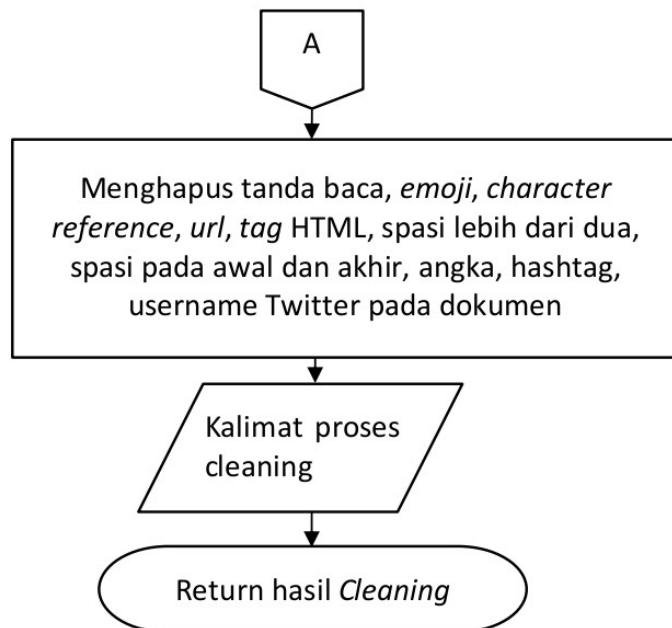
Gambar 4.2 Diagram Alir *Text Preprocessing*

Text preprocessing merupakan tahapan awal sebelum masuk pada pembobotan *term* hingga klasifikasi. Berdasarkan Gambar 4.2 dijelaskan bahwa data ulasan masyarakat akan diproses melalui tahap *cleaning*, *case folding*, *tokenization*, *stemming*. Hasil dari *text preprocessing* merupakan term yang memiliki *base word* atau dalam Bahasa Indonesia disebut kata dasar.

4.1.1.1 Diagram Alir Cleaning

Diagram alir ini berisi tahapan *Cleaning* untuk membersihkan dokumen sebelum dilakukan pembobotan. Proses yang dilalui saat *Cleaning* dijelaskan dalam Gambar 4.3.



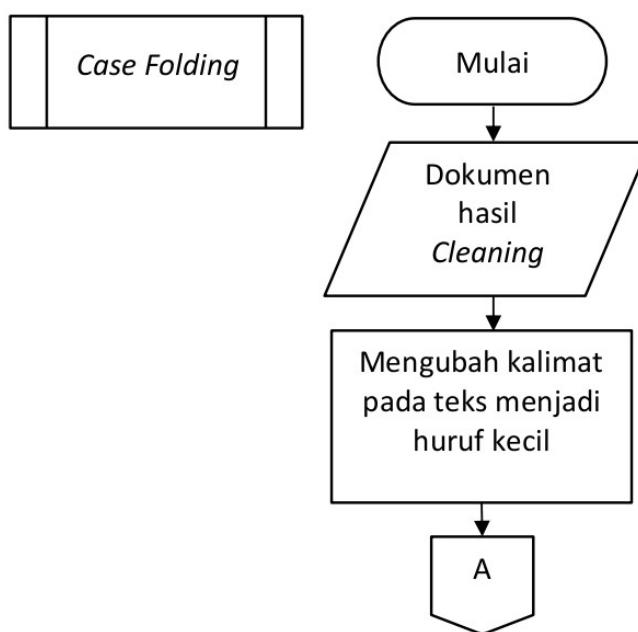


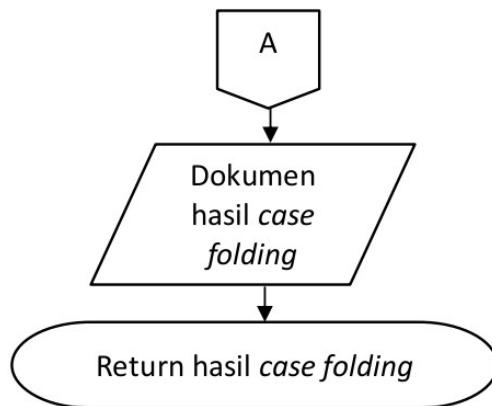
Gambar 4.3 Diagram Alir Cleaning

Berdasarkan Gambar 4.3 proses yang dilalui pada *Cleaning* yaitu data ulasan masyarakat akan dibersihkan dengan menghilangkan tanda baca, *emoji*, *character reference* seperti '&', '>', kemudian menghapus *url*, *tag HTML*, spasi lebih dari dua, spasi pada awal dan akhir dokumen, angka, *hashtag*, hingga username Twitter menggunakan fungsi *apply* dan *replace* untuk menghapus *emoji*. Selanjutnya dokumen telah bersih dan masuk pada tahapan berikutnya yaitu *case folding*.

4.1.1.2 Diagram Alir Case Folding

Diagram alir ini menjelaskan proses *case folding* yang dijelaskan pada Gambar 4.4.



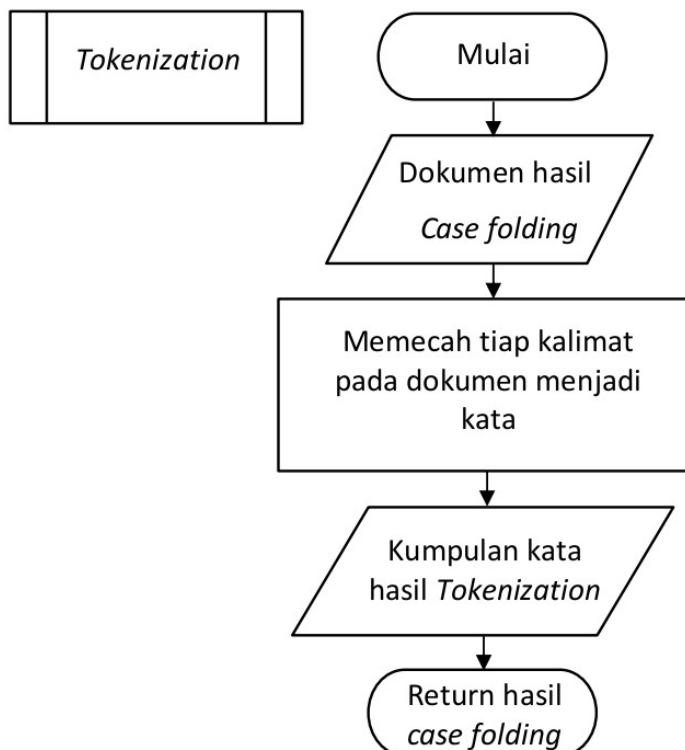


Gambar 4.4 Diagram Alir Case Folding

Berdasarkan Gambar 4.4, proses *case folding* meliputi perubahan tiap kalimat pada dokumen menjadi huruf kecil dengan fungsi *lower*. *Case folding* dilakukan agar kata yang sama namun memiliki perbedaan besar kecil huruf tetap dianggap kata yang sama.

4.1.1.3 Diagram Alir Tokenization

Diagram alir proses *tokenization* untuk memecah kalimat menjadi kata dijelaskan pada Gambar 4.5.



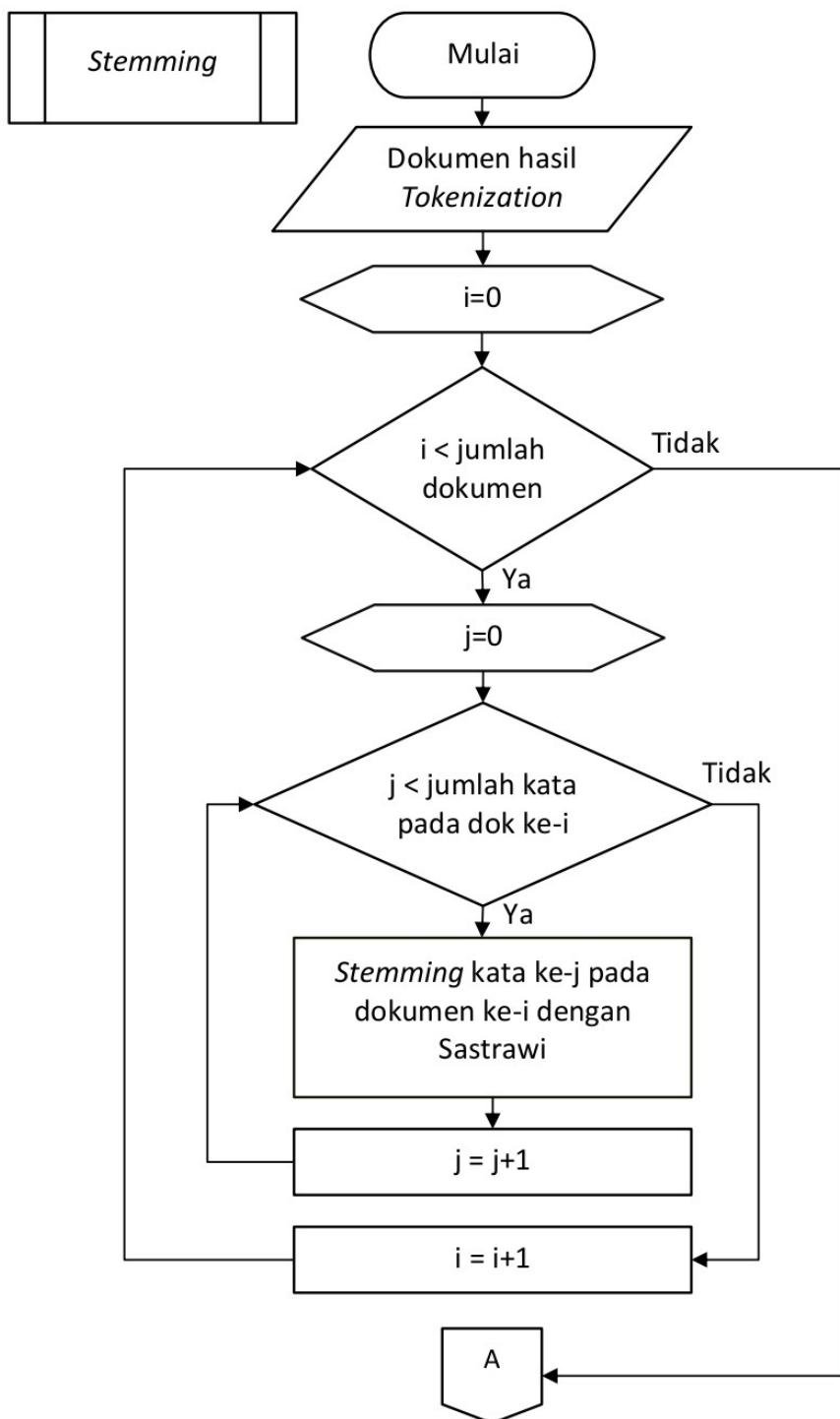
Gambar 4.5 Diagram Alir Tokenization

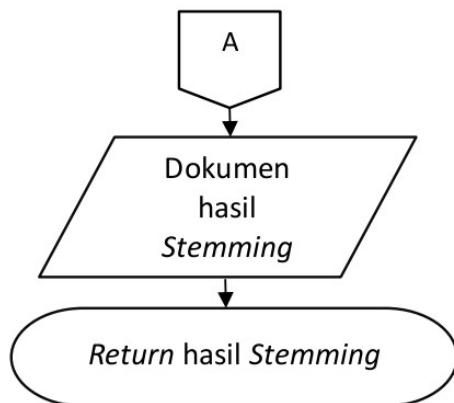
Berdasarkan Gambar 4.5, proses *Tokenization* diawali dengan memecah tiap kalimat pada dokumen hasil *case folding* menjadi kata menggunakan fungsi

apply yang telah disediakan oleh Pandas. Hasil *tokenization* selanjutnya akan masuk pada tahapan *Stemming*.

4.1.1.4 Diagram Alir Stemming

Diagram alir ini menggambarkan proses *stemming* untuk merubah kata ke dalam bentuk kata dasar menggunakan Sastrawi. Proses tersebut dijelaskan pada Gambar 4.6.



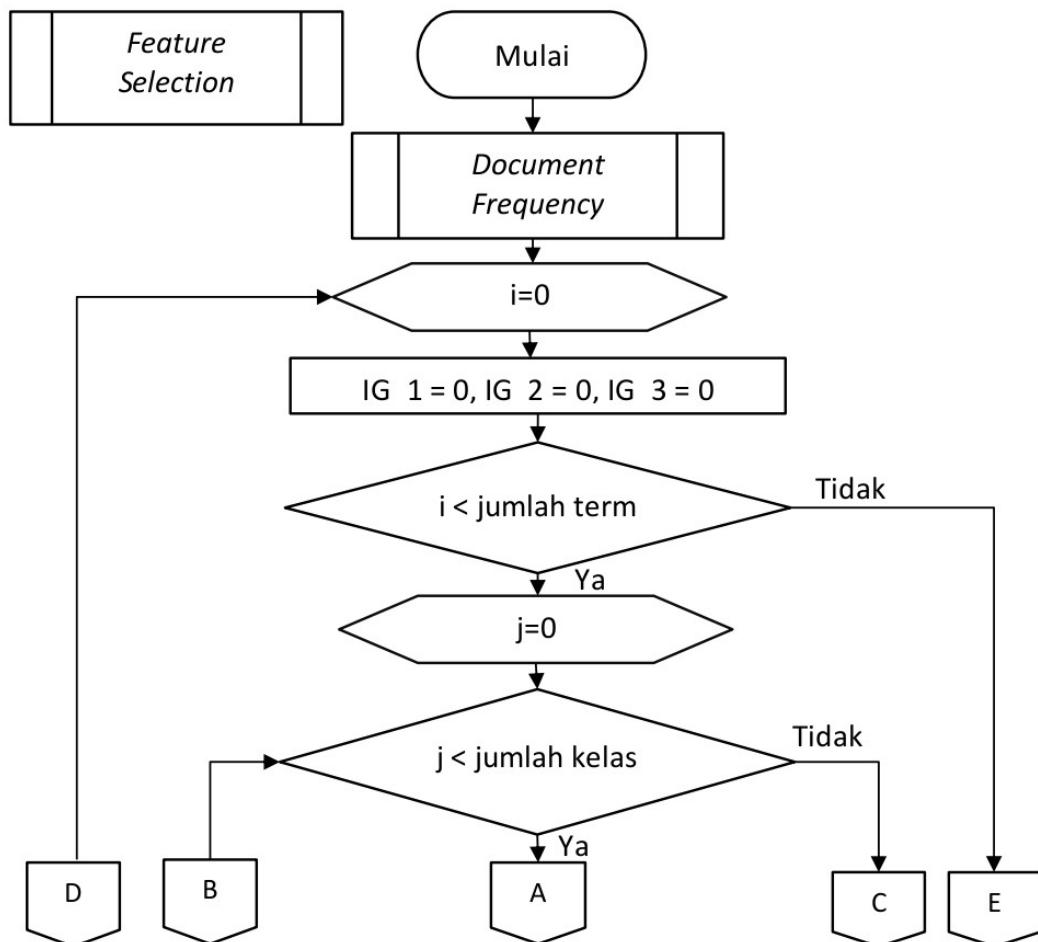


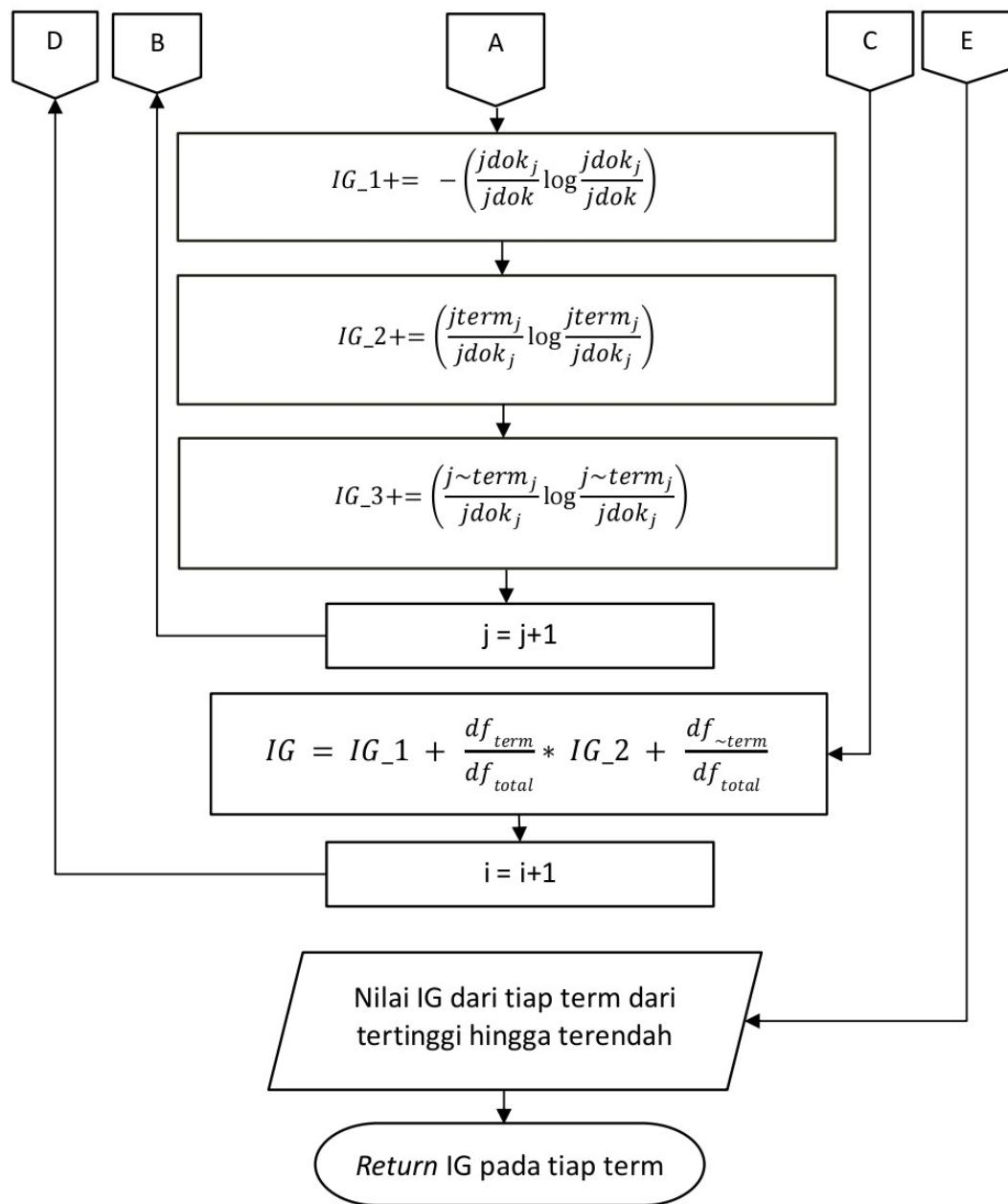
Gambar 4.6 Diagram Alir Stemming

Proses *Stemming* diawali oleh perulangan pada dokumen dan kata dari tiap dokumen. Selanjutnya tiap kata dilakukan *stemming* dengan Sastrawi. Hasil dari tahap ini berupa kumpulan kata dari tiap dokumen yang telah menjadi bentuk dasar dan akan masuk pada proses pembobotan *term*.

4.1.2 Diagram Alir Feature Selection

Diagram alir ini menjelaskan proses menyeleksi fitur dengan *Information Gain* agar mempercepat proses sistem karena dimensi dari fitur berkurang. Adapun proses tersebut dijelaskan pada Gambar 4.7.





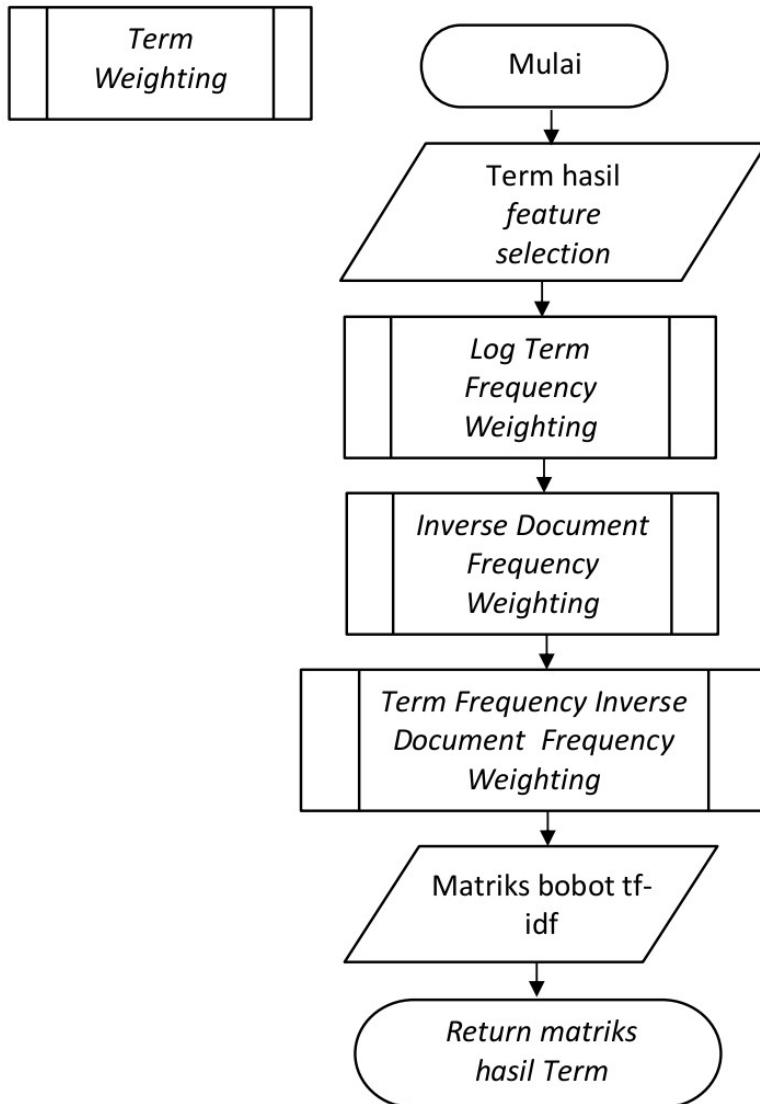
Gambar 4.7 Diagram Alir *Feature Selection* dengan *Information Gain*

Berdasarkan Gambar 4.7, proses diawali dengan menggunakan bobot *Document Frequency* yang didapatkan dari *Raw Term Frequency* pada tiap term. Kemudian, setiap term dilakukan perhitungan *Information Gain* yang dibagi menjadi tiga bagian yaitu IG_1 , IG_2 dan IG_3 pada setiap kelas yang akan dijumlahkan untuk mendapatkan nilai *information gain* term tersebut. Keluaran dari proses ini yaitu nilai IG dari setiap term. Semakin besar nilai IG maka term tersebut semakin penting.

4.1.3 Diagram Alir *Term Weighting*

Diagram ini menjelaskan proses pembobotan term setelah proses *feature selection*. Pembobotan pada term dilakukan setelah proses pemilihan term agar

proses yang dikerjakan lebih efisien. Gambar 4.8 menggambarkan proses pembentukan matriks *term weighting*.



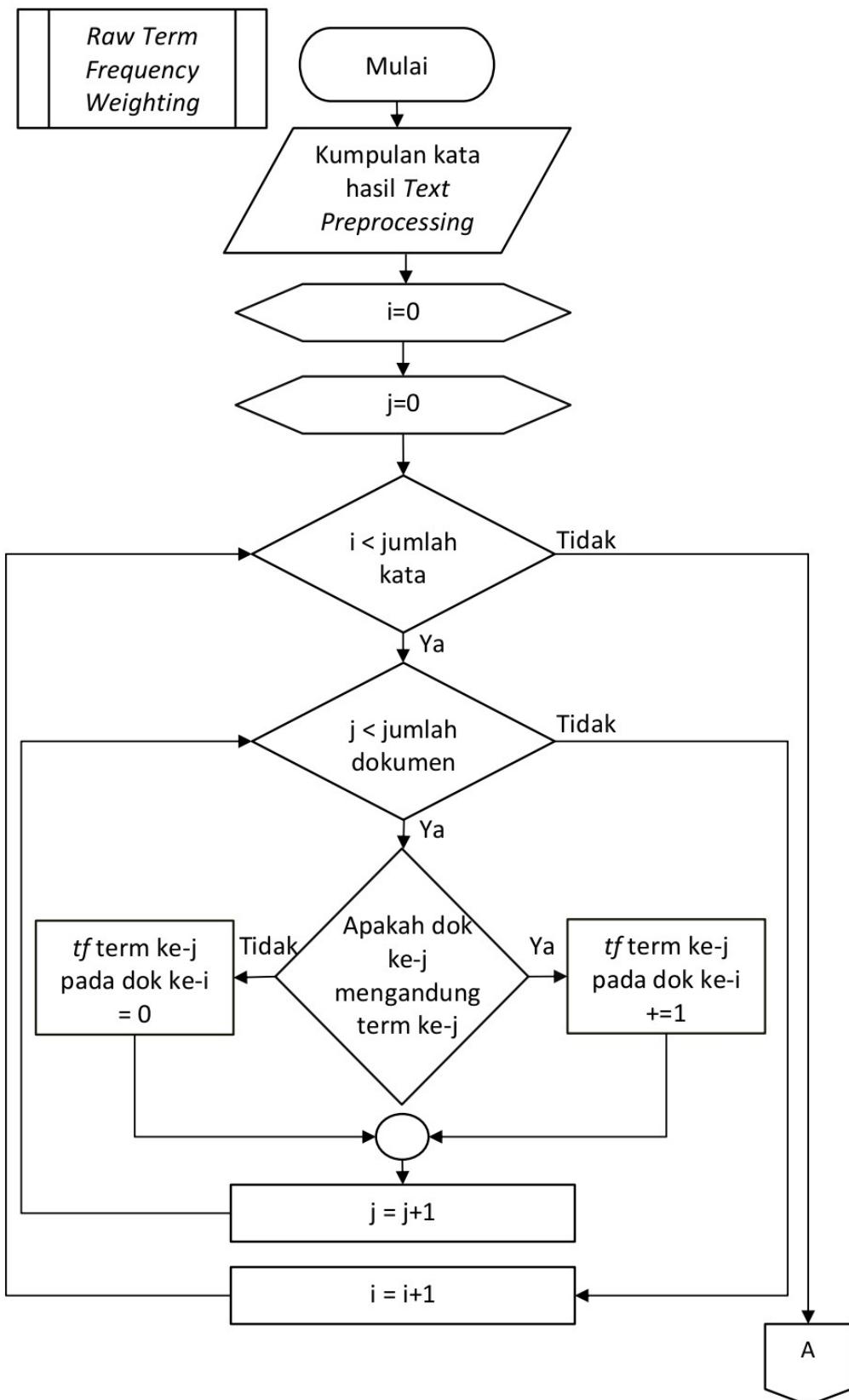
Gambar 4.8 Diagram Alir Term Weighting

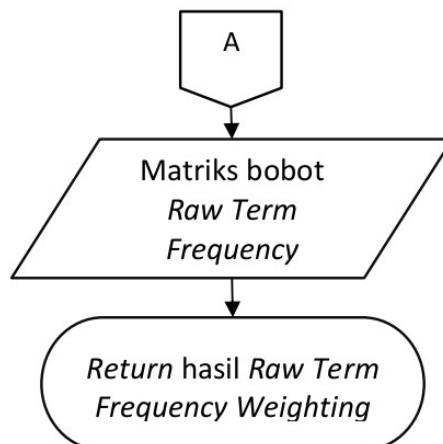
Berdasarkan Gambar 4.8, tahapan yang dilalui saat *term weighting* diantaranya *log term frequency weighting* dengan term hasil dari seleksi fitur, *inverse document frequency weighting*, dan yang terakhir yaitu *term frequency inverse document frequency weighting*. Keluaran dari proses ini yaitu matriks bobot *tf idf*.

4.1.3.1 Diagram Alir Raw Term Frequency Weighting

Setelah melakukan proses *stemming*, dilakukan proses pembobotan dokumen dengan *Raw Term frequency weighting* atau dikenal dengan *tf* yang

merupakan nilai kemunculan sebuah *term* pada suatu dokumen. Gambar 4.9 menjelaskan proses *Raw Term frequency weighting*.



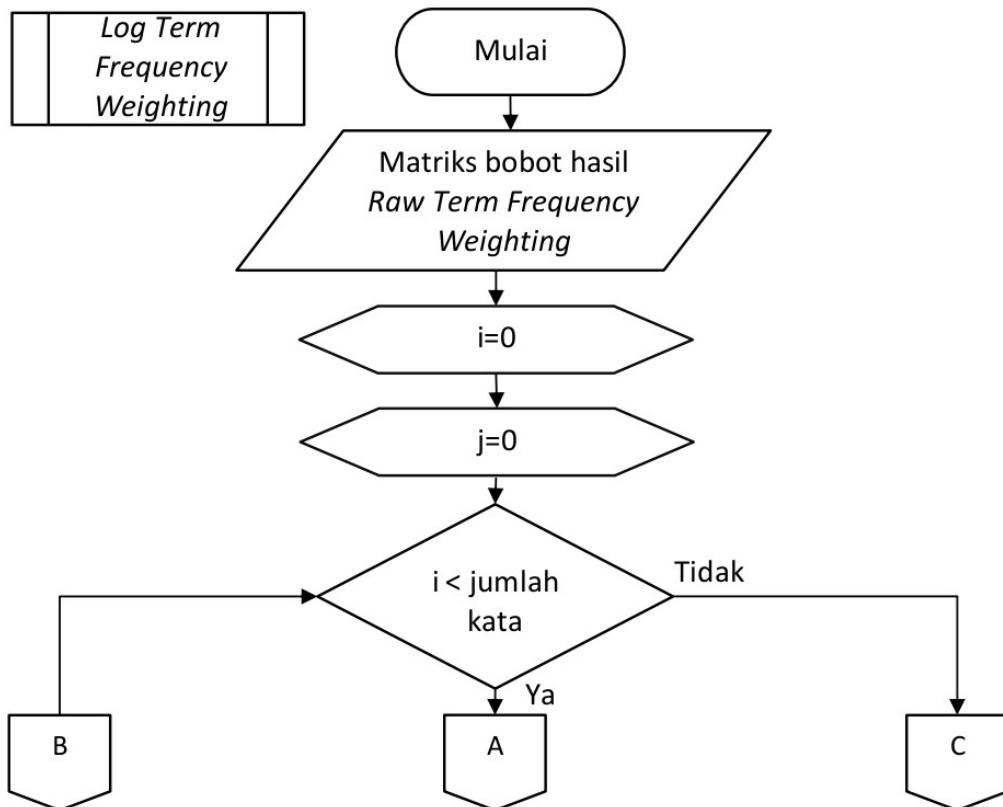


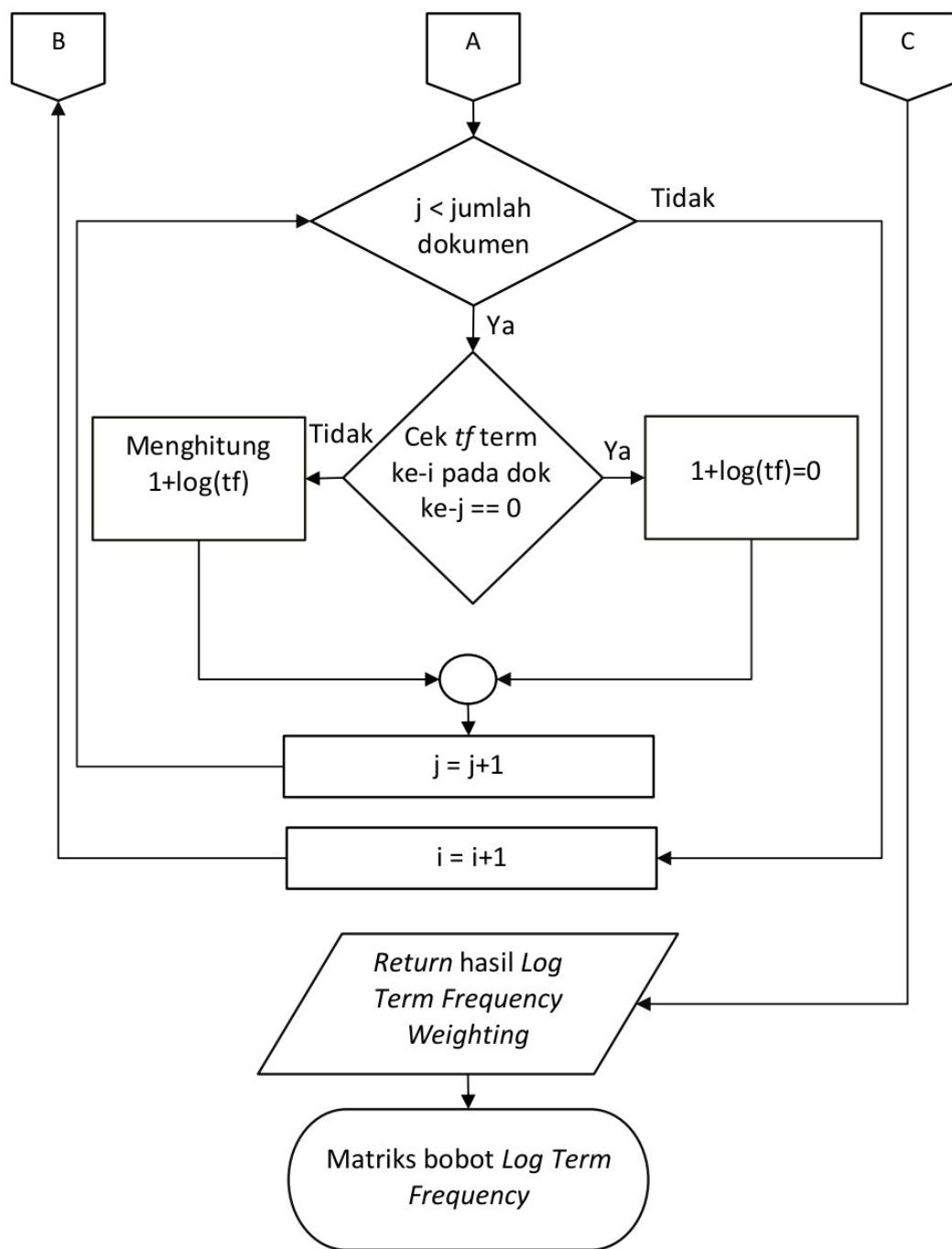
Gambar 4.9 Diagram Alir Raw Term Frequency Weighting

Berdasarkan Gambar 4.9, kumpulan kata hasil *text preprocessing* atau disebut dengan term. Selanjutnya dilakukan perulangan pada dokumen dan kata dalam dokumen tersebut dan mengecek jika dokumen mengandung term tertentu maka nilai frekuensi kemunculan kata pada term tersebut ditambah dengan 1. Keluaran dari tahapan ini merupakan matriks yang berisi nilai bobot kemunculan term pada dokumen.

4.1.3.2 Diagram Alir Log Term Frequency Weighting

Diagram alir ini menjelaskan proses *log term frequency weighting* sebagai bagian dalam menghitung *term frequency inverse document frequency*. Gambar 4.10 menggambarkan proses *log term frequency weighting*.



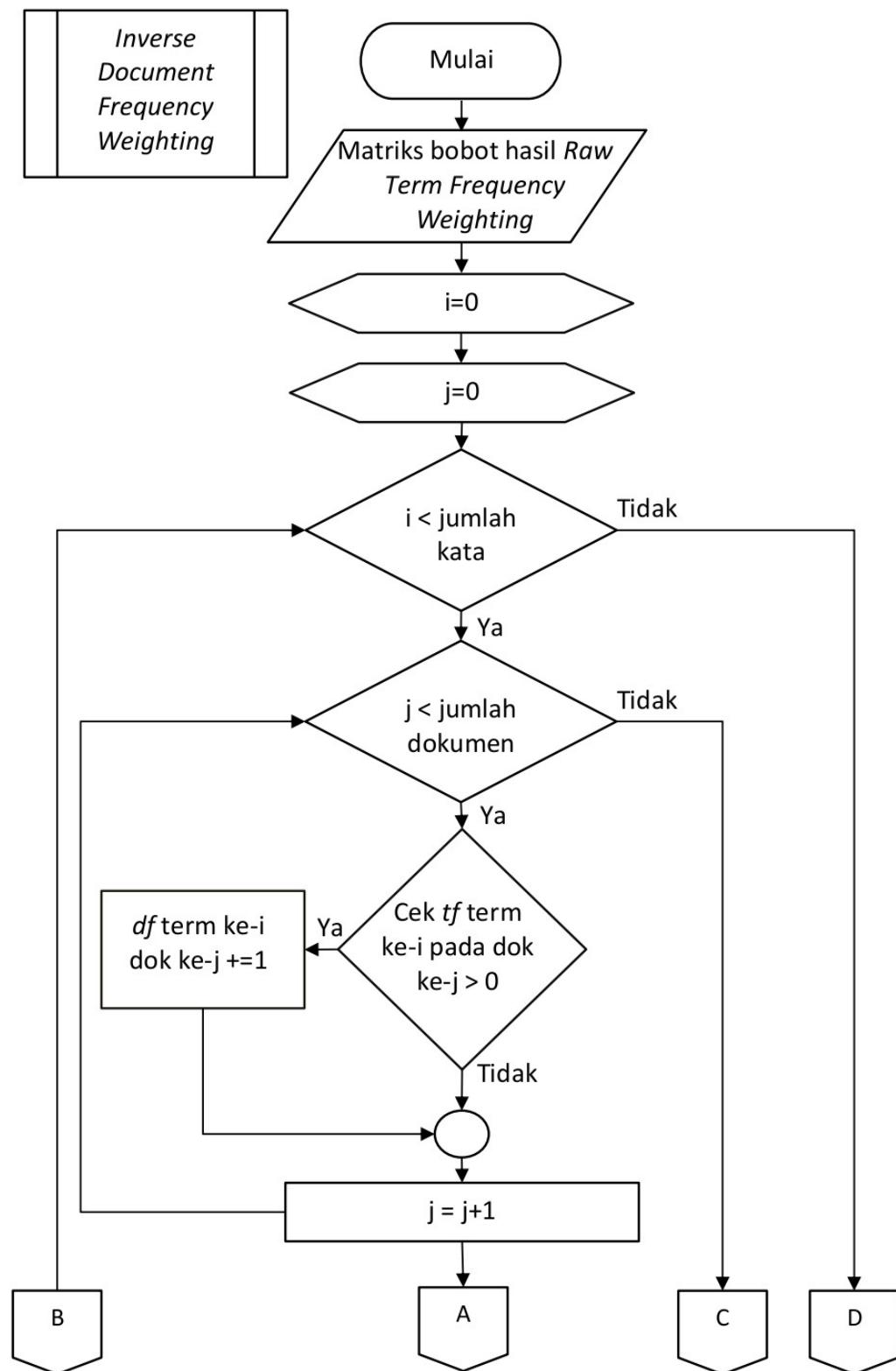


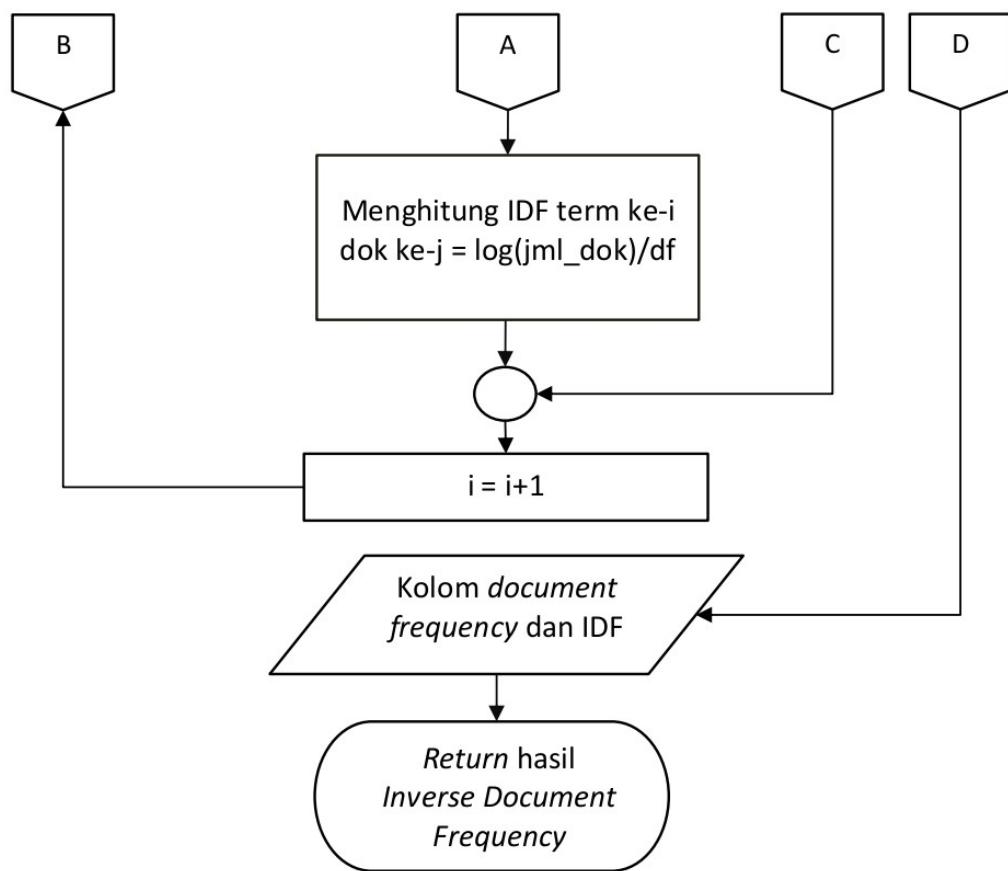
Gambar 4.10 Diagram Alir *Log Term Frequency Weighting*

Berdasarkan Gambar 4.10, proses ini diawali oleh perulangan pada matriks bobot hasil dari *raw term frequency weighting* dimana baris merupakan term dan kolom merupakan dokumen. Pada tiap term suatu dokumen dilakukan pengecekan nilai *tf*, jika bernilai 0 maka nilai *log term* akan bernilai 0 namun jika tidak bernilai 0 maka nilai *log term* tersebut akan dihitung sesuai dengan nilai *raw term frequency*. Keluaran pada tahapan ini berupa matriks bobot dengan nilai *log* dari *term frequency*.

4.1.3.3 Diagram Alir Inverse Document Frequency Weighting

Diagram alir ini menjelaskan proses untuk menghitung *Inverse Document Frequency (IDF) Weighting*. Gambar 4.11 menggambarkan proses mendapatkan nilai IDF dari tiap dokumen.



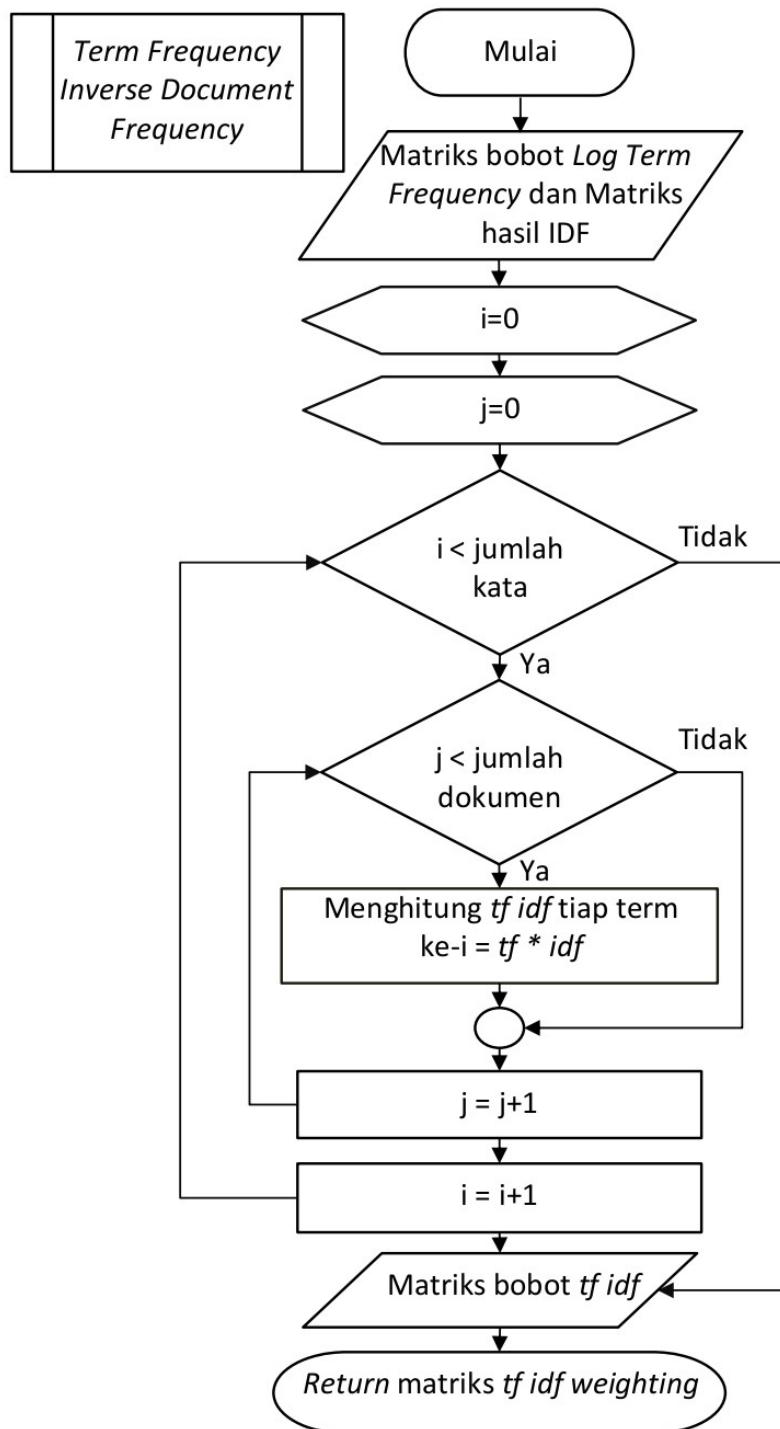


Gambar 4.11 Diagram Alir Inverse Document Frequency Weighting

Pada tahap ini yang dijelaskan pada Gambar 4.11 dimulai oleh perulangan tiap term dan dokumen. Selanjutnya dilakukan pengecekan nilai *term frequency* pada masing-masing dokumen, jika *tf* bernilai 0 maka *document frequency* dari dokumen tersebut tidak akan ditambahkan. Sedangkan jika *tf* bernilai 1 maka *document frequency* akan ditambah oleh 1. Pada akhir perulangan masing-masing dokumen dilakukan proses perhitungan nilai IDF. Keluaran pada proses ini berupa kolom baru yaitu kolom *document frequency* dan kolom IDF.

4.1.3.4 Diagram Alir Tf Idf Weighting

Diagram ini menjelaskan proses terakhir pada pembobotan term yaitu *term frequency inverse document frequency weighting (tf idf)*. Gambar 4.12 menggambarkan proses menghitung *tf idf* pada masing-masing term dalam dokumen.

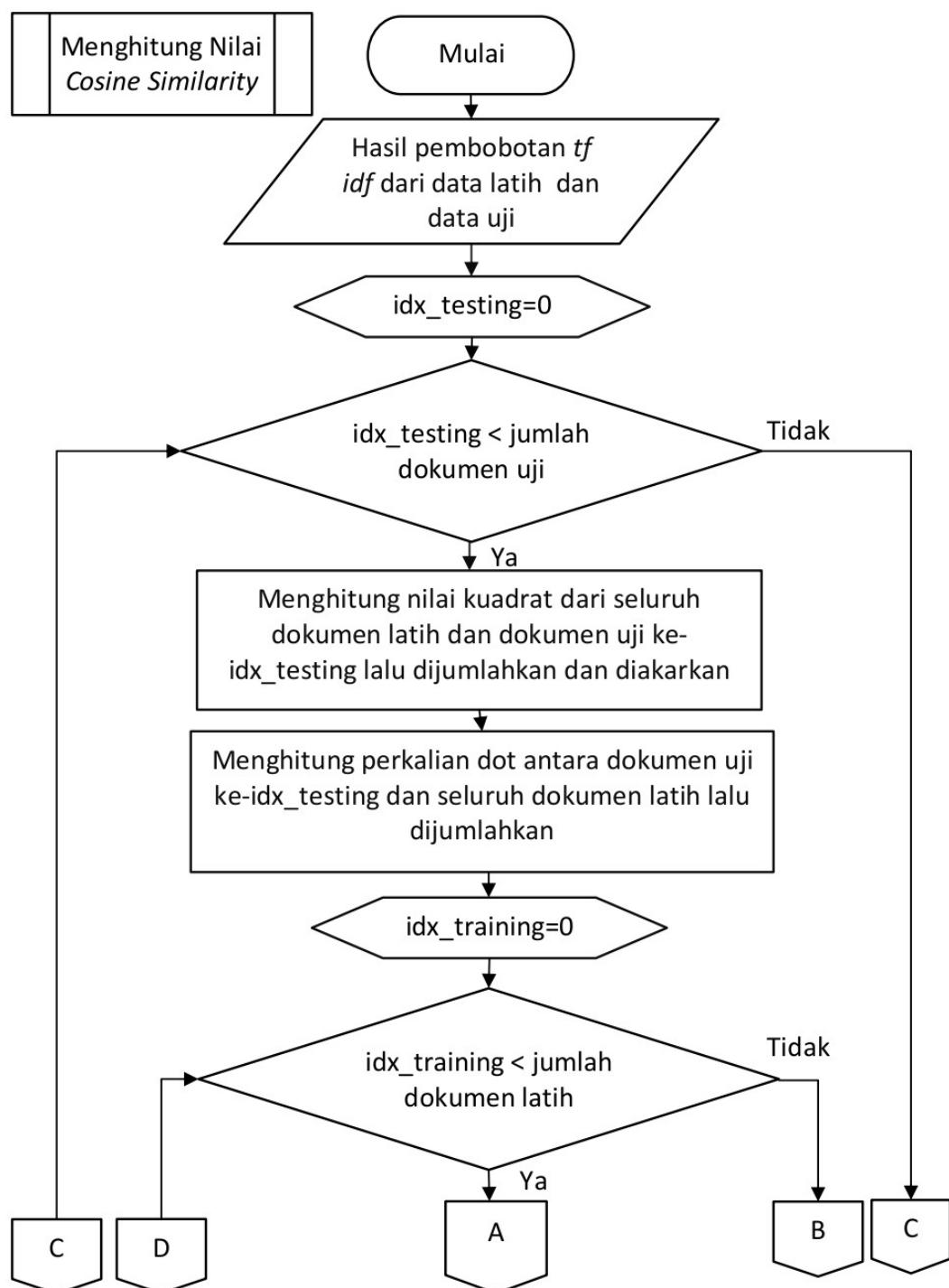


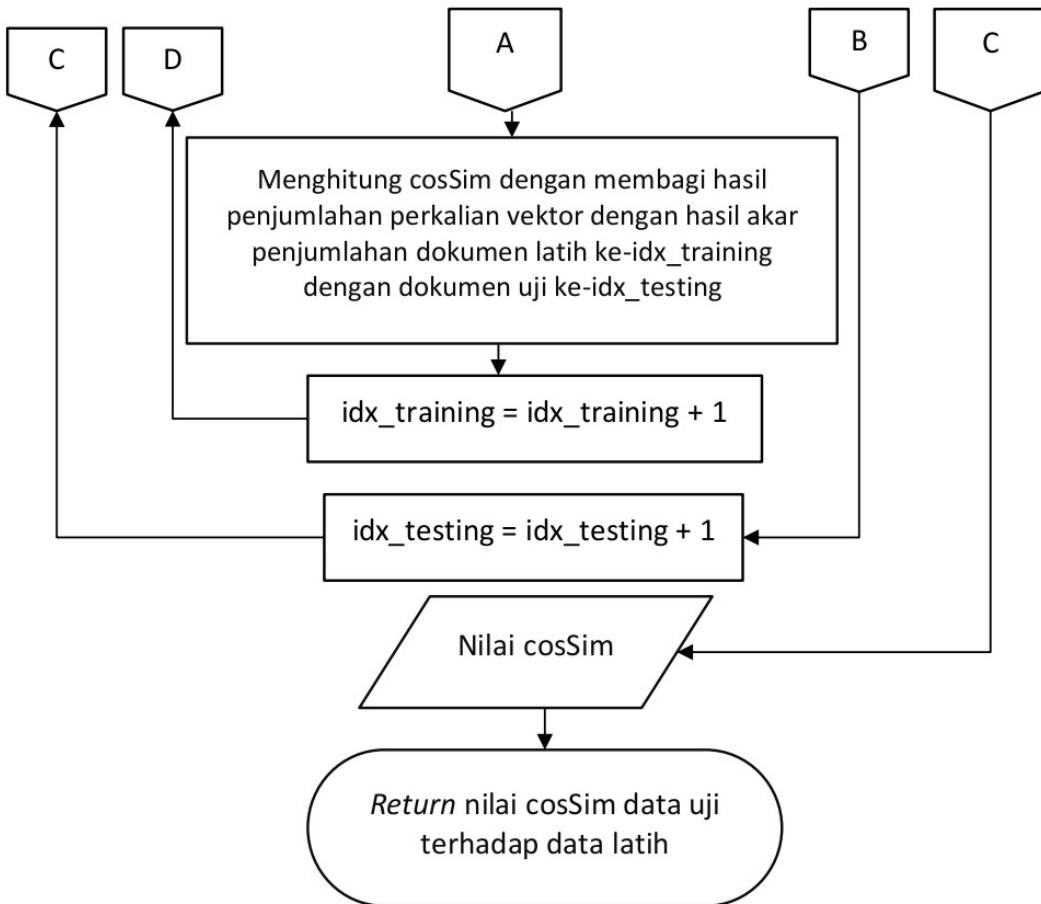
Gambar 4.12 Diagram Alir Term Frequency Inverse Document Frequency Weighting

Berdasarkan Gambar 4.12, proses diawali oleh perulangan term dan dokumen pada matriks *log term frequency*. Selanjutnya nilai *log* dari masing-masing term dikalikan dengan *idf* term tersebut yang menghasilkan nilai *tf idf*. Keluaran dari proses ini berupa matriks bobot *term frequency inverse document frequency* yang selanjutnya akan melalui tahap seleksi fitur.

4.1.4 Diagram Alir Menghitung Nilai *Cosine Similarity*

Salah satu bagian dari proses klasifikasi dengan *Improved KNN* yaitu menghitung kemiripan antara data uji dengan data latih. Metode yang dapat digunakan untuk menghitung kemiripan tersebut yaitu *Cosine Similarity* sebagaimana digambarkan pada Gambar 4.13.



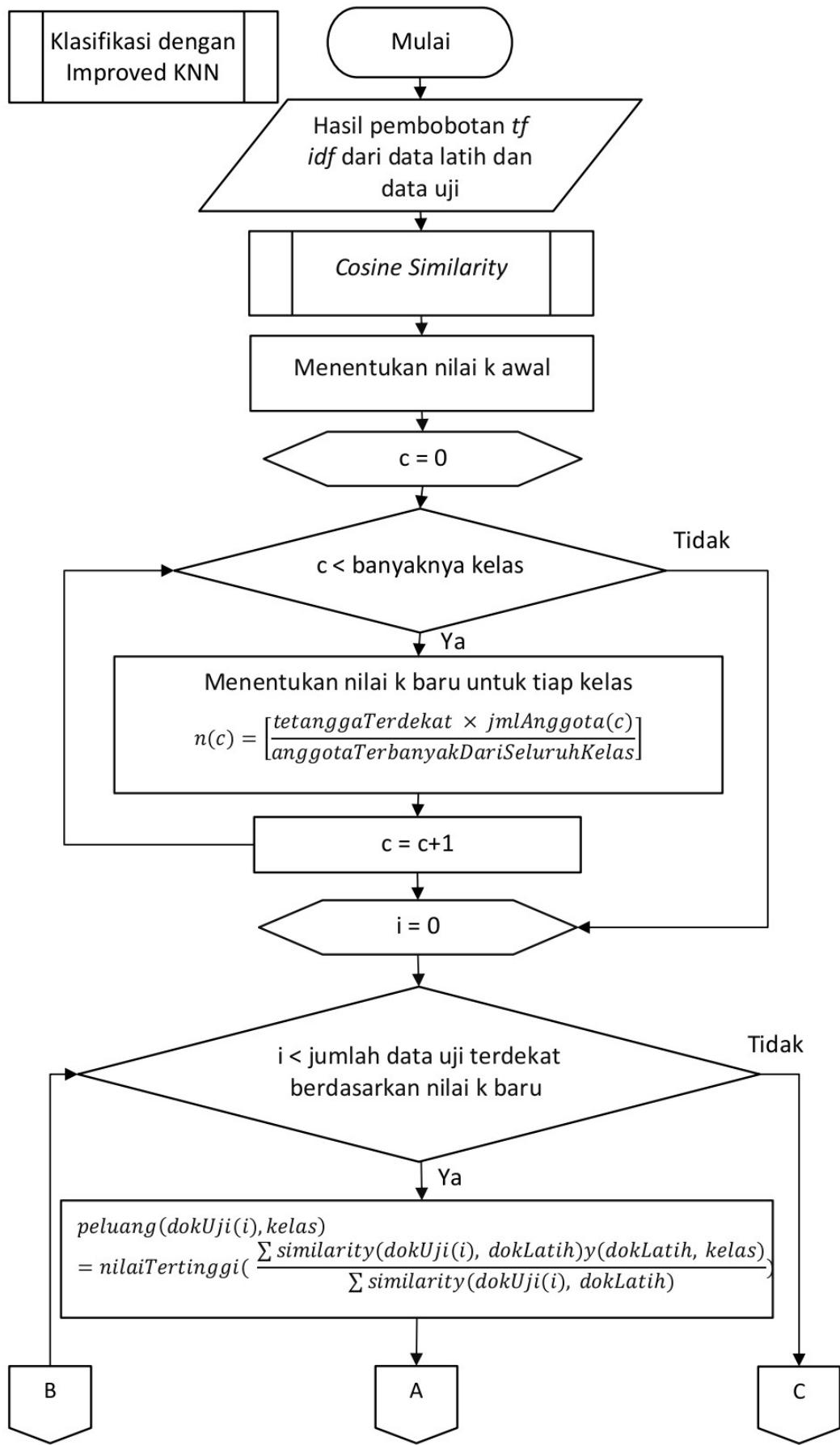


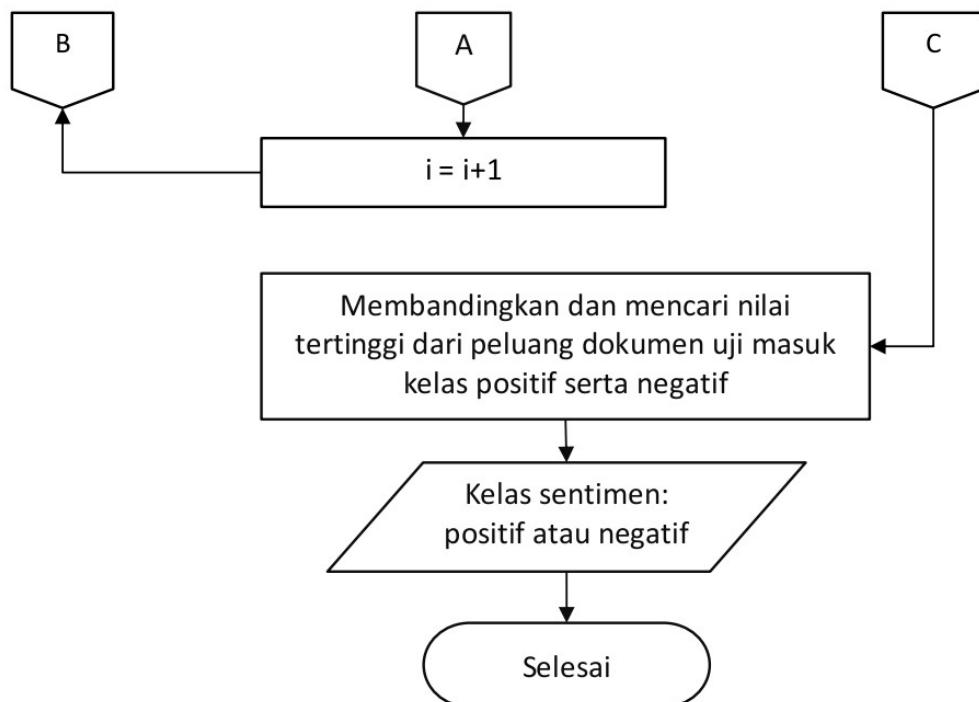
Gambar 4.13 Menghitung Nilai *Cosine Similarity*

Berdasarkan Gambar 4.13, untuk menghitung nilai *Cosine Similarity* diawali oleh menghitung nilai kuadrat dari tiap vektor data latih serta data uji dari perulangan pada tiap dokumen uji kemudian perulangan pada dokumen latih. Pada perulangan ini dilakukan perhitungan nilai kuadrat dari tiap dokumen latih dan uji lalu dijumlahkan kemudian diakarkan. Berikutnya juga dilakukan perkalian antara dokumen uji dan dokumen latih lalu dijumlahkan. Selanjutnya dilakukan perulangan pada dokumen latih untuk menghitung nilai *cosine similarity* dengan membagi hasil penjumlahan perkalian vektor dengan hasil akar penjumlahan tiap data latih dengan data uji. Keluaran dari proses ini yaitu nilai *cosine similarity* atau kemiripan antara data uji dengan seluruh data latih yang akan digunakan pada klasifikasi dengan *Improved K-Nearest Neighbor*.

4.1.5 Diagram Alir Klasifikasi dengan *Improved K-Nearest Neighbor*

Proses terakhir untuk menentukan sentimen dari ulasan masyarakat terhadap LRT Jakarta yaitu dengan melakukan klasifikasi menggunakan *Improved KNN*. Adapun proses yang dilalui digambarkan pada Gambar 4.14.





Gambar 4.14 Diagram Alir Klasifikasi dengan *Improved KNN*

Berdasarkan Gambar 4.14, proses diawali dengan menggunakan nilai *cosine similarity* kemudian diurutkan secara menurun. Selanjutnya menentukan nilai k awal seperti pada algoritme KNN lalu menghitung nilai k baru pada kelas positif maupun negatif. Nilai k tersebut digunakan untuk menentukan tetangga terdekat baru yang berada dalam perulangan untuk menghitung peluang dokumen uji merupakan anggota kelas positif atau negatif. Peluang tertinggi dari dokumen uji masuk suatu kelas akan dipilih sebagai anggota kelas tersebut.

4.2 Manualisasi Perhitungan Data

Manualisasi perhitungan pada data merupakan tahapan untuk menghitung secara manual sebelum memasuki tahap implementasi klasifikasi ulasan masyarakat terhadap LRT Jakarta menggunakan *Improved KNN* serta *Information Gain*. Data latih yang digunakan pada perhitungan manual ini terdapat pada Tabel 4.1.

Tabel 4.1 Data Latih

No	Teks	Kelas
1	yuk yg mau nyoba jg, lgsung dtg aja k stasiun LRT trdekat trus jgn lupa bawa ktp gengs.. karna syarat uji coba gretongnya cm itu aja, ditunjukin d loket dpt deh tiket masuknya. Tenang, petugasnya dr pintu masuk smp dalem LRT ramah2 bgts.. sukses buat #LRTJakarta ☺	positif
2	bersih, nyaman, murah & praktis...keren banget	positif

Tabel 4.1 Data Latih (Lanjutan)

3	Mencoba . Hanya lima stasiun. Jumlah gerbong hanya tiga. Keretanya mungil. Sepi. Nyaman. . #lrtjakarta #lrt #wajahbarujakarta @ Lrt Jakarta Corridor 1 phase 1 Kelapa Gading – Velodrome https://t.co/t5oeqS7oxL	Jakarta.	positif
4	terlalu ribet jalan masuk dan keluar yang diputar-putar. Pembayaran jam tertentu yang gak bisa pakai e-money dll. Antrian yang dipindah-pindah. Di luar negeri naik LRT gak seribet ini.		Negatif
5	Pagi-pagi udah ngos-ngosan sebelum naik LRT? Tangganya tinggi banget? Liftnya antri?		Negatif

Adapun data uji yang digunakan pada perhitungan manual ini terdapat pada Tabel 4.2.

Tabel 4.2 Data Uji

No	Teks	Kelas
1	Mumpung di Kelapa Gading, nyobain LRT....masih bersih & gratis ØØ serasa di Jepun ØØ™S #lrt #lrtjakarta #samsung #samsungS9photography #withGalaxy #jakarta #indONESia @ Stasiun Velodrom https://t.co/RQrptGqXtR	?

4.2.1 *Text Preprocessing*

Proses pertama yang dilakukan yaitu *text preprocessing*. Pada proses ini data akan melalui tahapan yaitu *cleaning*, *case folding*, *tokenization* dan *stemming*. Penjelasan pada masing-masing tahapan akan dijelaskan pada sub bab berikutnya.

4.2.1.1 *Cleaning*

Tahapan ini merupakan proses untuk membersihkan data dari tanda baca, *hashtag*, *username*, karakter tunggal maupun *emoticon* serta *emoji*. Hasil dari *Cleaning* pada data latih terdapat pada Tabel 4.3.

Tabel 4.3 Hasil Cleaning pada Data Latih

No	Teks	Kelas
1	yuk yg mau nyoba jd lgsung dtg aja k stasiun LRT trdekat trus jgn lupa bawa ktp gengs karna syarat uji coba gretongnya cm itu aja ditunjukin d loket dpt deh tiket masuknya Tenang petugasnya dr pintu masuk smp dalem LRT ramah bgts sukses buat	positif

Tabel 4.3 Hasil *Cleaning* pada Data Latih (Lanjutan)

2	bersih nyaman murah praktis keren banget	positif
3	Mencoba LRT Jakarta Hanya lima stasiun Jumlah gerbong hanya tiga Keretanya mungil Sepi Nyaman Lrt Jakarta Corridor phase Kelapa Gading Velodrome	positif
4	terlalu ribet jalan masuk dan keluar yang diputar putar pembayaran jam tertentu yang gak bisa pakai e money dll antrian yang dipindah pindah di luar negeri naik LRT gak seribet ini	negatif
5	Pagi pagi udah ngos ngosan sebelum naik LRT Tangganya tinggi banget Lifnya antri	negatif

Dapat dilihat pada Tabel 4.3 bahwa tanda baca seperti “?”, “.”, “#” dihilangkan, serta *emoticon* maupun *emoji* juga dihapus. Selain itu karakter tunggal juga dihapus. Adapun hasil *Cleaning* pada data uji terdapat pada Tabel 4.4.

Tabel 4.4 Hasil *Cleaning* pada Data Uji

No	Teks	Kelas
1	Mumpung di Kelapa Gading nyobain LRT masih bersih gratis serasa di Jepun Stasiun Velodrom	?

4.2.1.2 *Case Folding*

Tahapan *case folding* bertujuan untuk mengubah huruf kapital menjadi huruf kecil pada data latih maupun data uji. Hasil *case folding* pada data latih terdapat pada Tabel 4.5.

Tabel 4.5 Hasil *Case Folding* pada Data Latih

No	Teks	Kelas
1	yuk yg mau nyoba jd lgsung dtg aja k stasiun lrt trdekat trus jgn lupa bawa ktp gengs karna syarat uji coba gretongnya cm itu aja ditunjukin d loket dpt deh tiket masuknya tenang petugasnya dr pintu masuk smp dalem lrt ramah bgts sukses buat	positif
2	bersih nyaman murah praktis keren banget	positif
3	mencoba lrt jakarta hanya lima stasiun jumlah gerbong hanya tiga keretanya mungil sepi nyaman lrt jakarta corridor phase kelapa gading velodrome	positif

Tabel 4.5 Hasil *Case Folding* pada Data Latih (Lanjutan)

4	terlalu ribet jalan masuk dan keluar yang diputar putar pembayaran jam tertentu yang gak bisa pakai e money dll antrian yang dipindah pindah di luar negeri naik lrt gak seribet ini	negatif
5	pagi pagi udah ngos ngosan sebelum naik lrt tangganya tinggi banget liftnya antri	negatif

Berdasarkan Tabel 4.5, terlihat bahwa seluruh huruf kapital dirubah menjadi huruf kecil. Adapun hasil *case folding* pada data uji terdapat pada Tabel 4.6.

Tabel 4.6 Hasil *Case Folding* pada Data Uji

No	Teks	Kelas
1	mumpung di kelapa gading nyobain lrt masih bersih gratis serasa di jepun stasiun velodrom	?

4.2.1.3 *Tokenization*

Tokenization merupakan tahapan untuk memecah kalimat menjadi kata berdasarkan spasi. Hasil dari *tokenization* pada data latih terdapat pada Tabel 4.7.

Tabel 4.7 Hasil *Tokenization* pada Data Latih

No	Teks	Kelas
1	yuk yg mau nyoba jg lgsung dtg aja k stasiun lrt trdekat trus jgn lupa bawa ktp gengs karna syarat uji coba gretongnya cm itu aja ditunjukin d loket dpt deh tiket masuknya tenang petugasnya dr pintu masuk smp dalem lrt ramah bgts sukses buat	positif
2	bersih nyaman murah praktis keren banget	positif
3	mencoba lrt jakarta hanya lima stasiun jumlah gerbong hanya tiga keretanya mungil sepi nyaman lrt jakarta corridor phase kelapa gading velodrome	positif
4	terlalu ribet jalan masuk dan keluar yang diputar putar pembayaran jam tertentu yang gak bisa pakai e money dll antrian yang dipindah pindah di luar negeri naik lrt gak seribet ini	negatif
5	pagi pagi udah ngos ngosan sebelum naik lrt tangganya tinggi banget liftnya antri	negatif

Berdasarkan Tabel 4.7 terlihat bahwa tiap kata dipisahkan dengan garis tegak “|” yang menandakan bahwa kalimat tersebut telah dipecah menjadi

kumpulan kata. Adapun hasil dari *tokenization* pada data uji terdapat pada Tabel 4.8.

Tabel 4.8 Hasil *Tokenization* pada Data Uji

No	Teks	Kelas
1	mumpung di kelapa gading nyobain lrt masih bersih gratis serasa di jepun stasiun velodrom	?

4.2.1.4 *Stemming*

Stemming merupakan tahap terakhir dalam *text preprocessing* untuk merubah kata menjadi kata dasar. Hasil *stemming* pada data latih ditunjukkan oleh Tabel 4.9.

Tabel 4.9 Hasil *Stemming* pada Data Latih

No	Teks	Kelas
1	yuk yg mau nyoba jg lgsung dtg aja k stasiun lrt trdekat trus jgn lupa bawa ktp gengs karna syarat uji coba gretongnya cm itu aja ditunjukin d loket dpt deh tiket masuk tenang tugas dr pintu masuk smp dalem lrt ramah bgts sukses buat	positif
2	bersih nyaman murah praktis keren banget	positif
3	coba lrt jakarta hanya lima stasiun jumlah gerbong hanya tiga kereta mungil sepi nyaman lrt jakarta corridor phase kelapa gading velodrome	positif
4	terlalu ribet jalan masuk dan keluar yang putar putar bayar jam tentu yang gak bisa pakai e money dll antri yang pindah pindah di luar negeri naik lrt gak seribet ini	negatif
5	pagi pagi udah ngos ngosan belum naik lrt tangga tinggi banget lift antri	negatif

Berdasarkan Tabel 4.9, dapat dilihat bahwa beberapa kata dirubah menjadi kata dasar seperti “antrian” menjadi “antri” pada dokumen 4. Adapun hasil *stemming* pada data uji terdapat pada Tabel 4.10.

Tabel 4.10 Hasil *Stemming* pada Data Uji

No	Teks	Kelas
1	mumpung di kelapa gading nyobain lrt masih bersih gratis serasa di jepun stasiun velodrom	?

4.2.2 Feature Selection dengan *Information Gain*

Fitur pada teks yang berupa term memiliki jumlah yang sangat banyak, maka dari itu dilakukan proses seleksi fitur untuk mengurangi fitur yang kurang representatif terhadap kelas. Metode seleksi fitur yang digunakan yaitu *Information Gain* yang membutuhkan nilai *document frequency* yang didapat dari *raw term frequency* dari term. Adapun kemunculan term pada setiap dokumen yang diperlukan untuk perhitungan manual yang ditunjukkan pada Tabel 4.11.

Tabel 4.11 Kemunculan Term pada Dokumen untuk Perhitungan *Information Gain*

Term	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5
aja	1	0	0	0	0
antri	0	0	0	1	1
banget	0	1	0	0	1
bawa	1	0	0	0	0
bayar	0	0	0	1	0
belum	0	0	0	0	1
bersih	0	1	0	0	0
bgts	1	0	0	0	0
bisa	0	0	0	1	0
buat	1	0	0	0	0
cm	1	0	0	0	0
coba	1	0	1	0	0
corridor	0	0	1	0	0
dalem	1	0	0	0	0
dan	0	0	0	1	0
deh	1	0	0	0	0
di	0	0	0	1	0
ditunjukin	1	0	0	0	0
dll	0	0	0	1	0
dpt	1	0	0	0	0
dr	1	0	0	0	0
dtg	1	0	0	0	0
gading	0	0	1	0	0

Tabel 4.11 Kemunculan Term pada Dokumen untuk Perhitungan *Information Gain* (Lanjutan)

gak	0	0	0	1	0
gengs	1	0	0	0	0
gerbong	0	0	1	0	0
gretongnya	1	0	0	0	0
hanya	0	0	1	0	0
ini	0	0	0	1	0
itu	1	0	0	0	0
jakarta	0	0	1	0	0
jalan	0	0	0	1	0
jam	0	0	0	1	0
jg	1	0	0	0	0
jgn	1	0	0	0	0
jumlah	0	0	1	0	0
karna	1	0	0	0	0
kelapa	0	0	1	0	0
keluar	0	0	0	1	0
keren	0	1	0	0	0
kereta	0	0	1	0	0
ktp	1	0	0	0	0
lgsung	1	0	0	0	0
lift	0	0	0	0	1
lima	0	0	1	0	0
loket	1	0	0	0	0
lrt	1	0	1	1	1
luar	0	0	0	1	0
lupa	1	0	0	0	0
masuk	1	0	0	1	0
mau	1	0	0	0	0
money	0	0	0	1	0
mungil	0	0	1	0	0

Tabel 4.11 Kemunculan Term pada Dokumen untuk Perhitungan *Information Gain* (Lanjutan)

murah	0	1	0	0	0
naik	0	0	0	1	1
negeri	0	0	0	1	0
ngos	0	0	0	0	1
ngosan	0	0	0	0	1
nyaman	0	1	1	0	0
nyoba	1	0	0	0	0
pagi	0	0	0	0	1
pakai	0	0	0	1	0
phase	0	0	1	0	0
pindah	0	0	0	1	0
pintu	1	0	0	0	0
praktis	0	1	0	0	0
putar	0	0	0	1	0
ramah	1	0	0	0	0
ribet	0	0	0	1	0
sepi	0	0	1	0	0
seribet	0	0	0	1	0
smp	1	0	0	0	0
stasiun	1	0	1	0	0
sukses	1	0	0	0	0
syarat	1	0	0	0	0
tangga	0	0	0	0	1
tenang	1	0	0	0	0
tentu	0	0	0	1	0
terlalu	0	0	0	1	0
tiga	0	0	1	0	0
tiket	1	0	0	0	0
tinggi	0	0	0	0	1
trdekat	1	0	0	0	0

Tabel 4.11 Kemunculan Term pada Dokumen untuk Perhitungan *Information Gain* (Lanjutan)

trus	1	0	0	0	0
tugas	1	0	0	0	0
udah	0	0	0	0	1
uji	1	0	0	0	0
velodrome	0	0	1	0	0
yang	0	0	0	1	0
yg	1	0	0	0	0
yuk	1	0	0	0	0

Di bawah ini menunjukkan contoh perhitungan nilai *Information Gain* dari term “bersih” berdasarkan kemunculan pada dokumen yang ditunjukkan pada Tabel 4.11 dengan nilai $df_{total} = 101$, $P(positif) = \frac{3}{5}$, $P(negatif) = \frac{2}{5}$, $P(bersih) = \frac{1}{101}$, $P(\sim bersih) = \frac{4}{101}$, $P(positif|bersih) = \frac{1}{3}$, $P(negatif|bersih) = \frac{0}{2}$, $P(positif|\sim bersih) = \frac{2}{3}$, $P(negatif|\sim bersih) = \frac{2}{2}$ dengan menggunakan Persamaan 2.6.

$$IG(bersih) = -\left(\frac{3}{5} \log\left(\frac{3}{5}\right) + \frac{2}{5} \log\left(\frac{2}{5}\right)\right) + \left(\left(\frac{1}{101}\right)\left(\frac{1}{3}\right) \log\left(\frac{1}{3}\right) + \left(\frac{0}{2}\right) \log\left(\frac{0}{2}\right)\right) * \left(\left(\frac{4}{101}\right)\left(\frac{2}{3}\right) \log\left(\frac{2}{3}\right) + \left(\frac{2}{2}\right) \log\left(\frac{2}{2}\right)\right)$$

$$IG(bersih) = 0.292285253 + -0.001574658 + -0.004649274$$

$$IG(bersih) = 0.286061321$$

Adapun hasil perhitungan *Information Gain* untuk term pada data latih ditunjukkan pada Tabel 4.12.

Tabel 4.12 Perhitungan *Information Gain* pada Setiap Term

No	Term	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5	IG
1	aja	1	0	0	0	0	0.286061321
2	antri	0	0	0	1	1	0.292285253
3	banget	0	1	0	0	1	0.278197745
4	bawa	1	0	0	0	0	0.286061321
5	bayar	0	0	0	1	0	0.284834016
6	belum	0	0	0	0	1	0.284834016
7	bersih	0	1	0	0	0	0.286061321
8	bgts	1	0	0	0	0	0.286061321
9	bisa	0	0	0	1	0	0.284834016
10	buat	1	0	0	0	0	0.286061321

Tabel 4.12 Perhitungan *Information Gain* pada Setiap Term (Lanjutan)

11	cm	1	0	0	0	0	0.286061321
12	coba	1	0	1	0	0	0.285236643
13	corridor	0	0	1	0	0	0.286061321
14	dalem	1	0	0	0	0	0.286061321
15	dan	0	0	0	1	0	0.284834016
16	deh	1	0	0	0	0	0.286061321
17	di	0	0	0	1	0	0.284834016
18	ditunjukin	1	0	0	0	0	0.286061321
19	dll	0	0	0	1	0	0.284834016
20	dpt	1	0	0	0	0	0.286061321
21	dr	1	0	0	0	0	0.286061321
22	dtg	1	0	0	0	0	0.286061321
23	gading	0	0	1	0	0	0.286061321
24	gak	0	0	0	1	0	0.284834016
25	gengs	1	0	0	0	0	0.286061321
26	gerbong	0	0	1	0	0	0.286061321
27	gretongnya	1	0	0	0	0	0.286061321
28	hanya	0	0	1	0	0	0.286061321
29	ini	0	0	0	1	0	0.284834016
30	itu	1	0	0	0	0	0.286061321
31	jakarta	0	0	1	0	0	0.286061321
32	jalan	0	0	0	1	0	0.284834016
33	jam	0	0	0	1	0	0.284834016
34	jk	1	0	0	0	0	0.286061321
35	jgn	1	0	0	0	0	0.286061321
36	jumlah	0	0	1	0	0	0.286061321
37	karna	1	0	0	0	0	0.286061321
38	kelapa	0	0	1	0	0	0.286061321
39	keluar	0	0	0	1	0	0.284834016
40	keren	0	1	0	0	0	0.286061321
41	kereta	0	0	1	0	0	0.286061321
42	ktp	1	0	0	0	0	0.286061321
43	lgsung	1	0	0	0	0	0.286061321
44	lift	0	0	0	0	1	0.284834016
45	lima	0	0	1	0	0	0.286061321
46	loket	1	0	0	0	0	0.286061321
47	lrt	1	0	1	1	1	0.286061321
48	luar	0	0	0	1	0	0.284834016
49	lupa	1	0	0	0	0	0.286061321
50	masuk	1	0	0	1	0	0.278197745
51	mau	1	0	0	0	0	0.286061321

Tabel 4.12 Perhitungan *Information Gain* pada Setiap Term (Lanjutan)

52	money	0	0	0	1	0	0.284834016
53	mungil	0	0	1	0	0	0.286061321
54	murah	0	1	0	0	0	0.286061321
55	naik	0	0	0	1	1	0.292285253
56	negeri	0	0	0	1	0	0.284834016
57	ngos	0	0	0	0	1	0.284834016
58	ngosan	0	0	0	0	1	0.284834016
59	nyaman	0	1	1	0	0	0.285236643
60	nyoba	1	0	0	0	0	0.286061321
61	pagi	0	0	0	0	1	0.284834016
62	pakai	0	0	0	1	0	0.284834016
63	phase	0	0	1	0	0	0.286061321
64	pindah	0	0	0	1	0	0.284834016
65	pintu	1	0	0	0	0	0.286061321
66	praktis	0	1	0	0	0	0.286061321
67	putar	0	0	0	1	0	0.284834016
68	ramah	1	0	0	0	0	0.286061321
69	ribet	0	0	0	1	0	0.284834016
70	sepi	0	0	1	0	0	0.286061321
71	seribet	0	0	0	1	0	0.284834016
72	smp	1	0	0	0	0	0.286061321
73	stasiun	1	0	1	0	0	0.285236643
74	sukses	1	0	0	0	0	0.286061321
75	syarat	1	0	0	0	0	0.286061321
76	tangga	0	0	0	0	1	0.284834016
77	tenang	1	0	0	0	0	0.286061321
78	tentu	0	0	0	1	0	0.284834016
79	terlalu	0	0	0	1	0	0.284834016
80	tiga	0	0	1	0	0	0.286061321
81	tiket	1	0	0	0	0	0.286061321
82	tinggi	0	0	0	0	1	0.284834016
83	trdekat	1	0	0	0	0	0.286061321
84	trus	1	0	0	0	0	0.286061321
85	tugas	1	0	0	0	0	0.286061321
86	udah	0	0	0	0	1	0.284834016
87	uji	1	0	0	0	0	0.286061321
88	velodrome	0	0	1	0	0	0.286061321
89	yang	0	0	0	1	0	0.284834016
90	yg	1	0	0	0	0	0.286061321
91	yuk	1	0	0	0	0	0.286061321

4.2.2.1 Pengurutan Term dari Hasil Information Gain

Setelah mendapatkan hasil perhitungan *Information Gain* untuk setiap term pada data latih dilakukan proses pengurutan nilai *Information Gain* dari tertinggi hingga terendah. Pada perhitungan manual ini, adapun *threshold* yang ditentukan untuk memilih fitur yaitu menggunakan *mean* atau rata-rata dari seluruh *Information Gain*. Pengurutan nilai IG ditunjukkan pada Tabel 4.13.

Tabel 4.13 Pengurutan Nilai *Information Gain*

Term	IG
antri	0.292285253
naik	0.292285253
aja	0.286061321
bawa	0.286061321
bersih	0.286061321
bgts	0.286061321
buat	0.286061321
cm	0.286061321
corridor	0.286061321
dalem	0.286061321
deh	0.286061321
ditunjukin	0.286061321
dpt	0.286061321
dr	0.286061321
dtg	0.286061321
gading	0.286061321
gengs	0.286061321
gerbong	0.286061321
gretongnya	0.286061321
hanya	0.286061321
itu	0.286061321
jakarta	0.286061321
jg	0.286061321
jgn	0.286061321
jumlah	0.286061321

Tabel 4.13 Pengurutan Nilai *Information Gain* (Lanjutan)

karna	0.286061321
kelapa	0.286061321
keren	0.286061321
kereta	0.286061321
ktp	0.286061321
lgsung	0.286061321
lima	0.286061321
loket	0.286061321
lrt	0.286061321
lupa	0.286061321
mau	0.286061321
mungil	0.286061321
murah	0.286061321
nyoba	0.286061321
phase	0.286061321
pintu	0.286061321
praktis	0.286061321
ramah	0.286061321
sepi	0.286061321
smp	0.286061321
sukses	0.286061321
syarat	0.286061321
tenang	0.286061321
tiga	0.286061321
tiket	0.286061321
trdekat	0.286061321
trus	0.286061321
tugas	0.286061321
uji	0.286061321
velodrome	0.286061321

Tabel 4.13 Pengurutan Nilai *Information Gain* (Lanjutan)

yg	0.286061321
yuk	0.286061321
coba	0.285236643
nyaman	0.285236643
stasiun	0.285236643
bayar	0.284834016
belum	0.284834016
bisa	0.284834016
dan	0.284834016
di	0.284834016
dll	0.284834016
gak	0.284834016
ini	0.284834016
jalan	0.284834016
jam	0.284834016
keluar	0.284834016
lift	0.284834016
luar	0.284834016
money	0.284834016
negeri	0.284834016
ngos	0.284834016
ngosan	0.284834016
pagi	0.284834016
pakai	0.284834016
pindah	0.284834016
putar	0.284834016
ribet	0.284834016
seribet	0.284834016
tangga	0.284834016
tentu	0.284834016

Tabel 4.13 Pengurutan Nilai *Information Gain* (Lanjutan)

terlalu	0.284834016
tinggi	0.284834016
udah	0.284834016
yang	0.284834016
banget	0.278197745
masuk	0.278197745

Setelah mengurutkan nilai *Information Gain* dari tertinggi hingga terendah, akan dipilih term dengan nilai *Information Gain* yang lebih tinggi dibandingkan nilai *mean* dari seluruh *Information Gain*. Adapun nilai mean dari keseluruhan yaitu 0.285606979. Berikut merupakan term yang terpilih sebagaimana ditunjukkan pada Tabel 4.14.

Tabel 4.14 Term Terpilih berdasarkan *Mean Threshold* dari *Information Gain*

No	Term
1	antri
2	naik
3	aja
4	bawa
5	bersih
6	bgts
7	buat
8	cm
9	corridor
10	dalem
11	deh
12	ditunjukin
13	dpt
14	dr
15	dtg
16	gading
17	gengs
18	gerbong
19	gretongnya

Tabel 4.14 Term Terpilih berdasarkan *Mean Threshold* dari *Information Gain* (Lanjutan)

20	hanya
21	itu
22	jakarta
23	jg
24	jgn
25	jumlah
26	karna
27	kelapa
28	keren
29	kereta
30	ktp
31	lgsung
32	lima
33	loket
34	lrt
35	lupa
36	mau
37	mungil
38	murah
39	nyoba
40	phase
41	pintu
42	praktis
43	ramah
44	sepi
45	smp
46	sukses
47	syarat
48	tenang
49	tiga

Tabel 4.14 Term Terpilih berdasarkan *Mean Threshold* dari *Information Gain* (Lanjutan)

50	tiket
51	trdekat
52	trus
53	tugas
54	ujji
55	velodrome
56	yg
57	yuk

4.2.3 Pemilihan Term

Pada tahapan ini, akan dilakukan pemilihan term data uji setelah dilakukan *preprocessing* yang dibandingkan dengan term dari data latih yang dihasilkan dari *Information Gain*. Adapun langkah-langkah yang dilakukan saat pemilihan term.

Langkah 1: Melakukan pencocokan term hasil *preprocessing* dari data uji dengan term hasil *Information Gain* dari data latih.

Langkah 2: Jika term data uji terdapat pada term hasil *Information Gain*, maka term tersebut disimpan namun jika tidak terdapat dalam term hasil *Information Gain* maka term tersebut akan dihapus. Hasil dari pemilihan term terdapat pada Tabel 4.15.

Tabel 4.15 Pemilihan Term Data Uji

No.	Term dari <i>Information Gain</i>	Term dari <i>Preprocessing</i> Data Uji	Hasil
1	antri	-	Simpan Term
2	naik	-	Simpan Term
3	aja	-	Simpan Term
4	bawa	-	Simpan Term
5	bersih	bersih	Simpan Term
6	bgts	-	Simpan Term
7	buat	-	Simpan Term
8	cm	-	Simpan Term
9	corridor	-	Simpan Term
10	dalem	-	Simpan Term

Tabel 4.15 Pemilihan Term Data Uji (Lanjutan)

11	deh	-	Simpan Term
12	ditunjukin	-	Simpan Term
13	dpt	-	Simpan Term
14	dr	-	Simpan Term
15	dtg	-	Simpan Term
16	gading	gading	Simpan Term
17	gengs	-	Simpan Term
18	gerbong	-	Simpan Term
19	gretongnya	-	Simpan Term
20	hanya	-	Simpan Term
21	itu	-	Simpan Term
22	jakarta	-	Simpan Term
23	jg	-	Simpan Term
24	jgn	-	Simpan Term
25	jumlah	-	Simpan Term
26	karna	-	Simpan Term
27	kelapa	kelapa	Simpan Term
28	keren	-	Simpan Term
29	kereta	-	Simpan Term
30	ktp	-	Simpan Term
31	lgsung	-	Simpan Term
32	lima	-	Simpan Term
33	loket	-	Simpan Term
34	lrt	lrt	Simpan Term
35	lupa	-	Simpan Term
36	mau	-	Simpan Term
37	mungil	-	Simpan Term
38	murah	-	Simpan Term
39	nyoba	-	Simpan Term
40	phase	-	Simpan Term

Tabel 4.15 Pemilihan Term Data Uji (Lanjutan)

41	pintu	-	Simpan Term
42	praktis	-	Simpan Term
43	ramah	-	Simpan Term
44	sepi	-	Simpan Term
45	smp	-	Simpan Term
46	sukses	-	Simpan Term
47	syarat	-	Simpan Term
48	tenang	-	Simpan Term
49	tiga	-	Simpan Term
50	tiket	-	Simpan Term
51	trdekat	-	Simpan Term
52	trus	-	Simpan Term
53	tugas	-	Simpan Term
54	uji	-	Simpan Term
55	velodrome	-	Simpan Term
56	yg	-	Simpan Term
57	yuk	-	Simpan Term
58	-	velodrom	Hapus Term
59	-	mumpung	Hapus Term
60	-	nyobain	Hapus Term
61	-	gratis	Hapus Term
62	-	serasa	Hapus Term
63	-	jepun	Hapus Term
64	-	stasiun	Hapus Term

Berikut merupakan term data uji yang terpilih dan digunakan saat klasifikasi dengan *Improved KNN* sebagaimana ditunjukkan pada Tabel 4.16.

Tabel 4.16 Term yang Digunakan Saat Klasifikasi pada Data Uji

No	Term pada Data Uji
1	bersih
2	gading

Tabel 4.16 Term yang Digunakan Saat Klasifikasi pada Data Uji (Lanjutan)

3	kelapa
4	lrt

4.2.4 Term Weighting

Pada tahapan ini, fitur data latih dibentuk menjadi vektor kemunculan term terhadap dokumen. *Term Weighting* yang digunakan yaitu *tf idf* dengan urutan langkah yaitu menghitung *raw term frequency*, menghitung *log frequency*, dilanjutkan dengan menghitung *tf idf*.

4.2.4.1 Raw Term Frequency Weighting

Nilai *raw term frequency* dihitung berdasarkan tingkat kemunculan term tersebut pada dokumen. Adapun hasil perhitungan *raw term frequency weighting* yang ditunjukkan pada Tabel 4.17.

Tabel 4.17 Hasil Raw Term Frequency Weighting

Term	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5	Dok Uji
antri	0	0	0	1	1	0
naik	0	0	0	1	1	0
aja	2	0	0	0	0	0
bawa	1	0	0	0	0	0
bersih	0	1	0	0	0	1
bgts	1	0	0	0	0	0
buat	1	0	0	0	0	0
cm	1	0	0	0	0	0
corridor	0	0	1	0	0	0
dalem	1	0	0	0	0	0
deh	1	0	0	0	0	0
ditunjukin	1	0	0	0	0	0
dpt	1	0	0	0	0	0
dr	1	0	0	0	0	0
dtg	1	0	0	0	0	0
gading	0	0	1	0	0	1
gengs	1	0	0	0	0	0
gerbong	0	0	1	0	0	0

Tabel 4.17 Hasil Raw Term Frequency Weighting (Lanjutan)

gretongnya	1	0	0	0	0	0
hanya	0	0	2	0	0	0
itu	1	0	0	0	0	0
jakarta	0	0	2	0	0	0
jg	1	0	0	0	0	0
jgn	1	0	0	0	0	0
jumlah	0	0	1	0	0	0
karna	1	0	0	0	0	0
kelapa	0	0	1	0	0	1
keren	0	1	0	0	0	0
kereta	0	0	1	0	0	0
ktp	1	0	0	0	0	0
lgsung	1	0	0	0	0	0
lima	0	0	1	0	0	0
loket	1	0	0	0	0	0
lrt	2	0	2	1	1	1
lupa	1	0	0	0	0	0
mau	1	0	0	0	0	0
mungil	0	0	1	0	0	0
murah	0	1	0	0	0	0
nyoba	1	0	0	0	0	0
phase	0	0	1	0	0	0
pintu	1	0	0	0	0	0
praktis	0	1	0	0	0	0
ramah	1	0	0	0	0	0
sepi	0	0	1	0	0	0
smp	1	0	0	0	0	0
sukses	1	0	0	0	0	0
syarat	1	0	0	0	0	0
tenang	1	0	0	0	0	0

Tabel 4.17 Hasil Raw Term Frequency Weighting (Lanjutan)

tiga	0	0	1	0	0	0
tiket	1	0	0	0	0	0
trdekat	1	0	0	0	0	0
trus	1	0	0	0	0	0
tugas	1	0	0	0	0	0
uji	1	0	0	0	0	0
velodrome	0	0	1	0	0	0
yg	1	0	0	0	0	0
yuk	1	0	0	0	0	0

4.2.4.2 Log Term Frequency Weighting

Setelah mendapatkan matriks bobot *raw term frequency*, langkah berikutnya yaitu menghitung *log term frequency* seperti pada Persamaan 2.1 dengan contoh perhitungan pada term “bersih” di dokumen kedua yang memiliki kemunculan satu kali.

$$\begin{aligned} tf_{t,d} &= 1 + \log(1) \\ &= 1 + 0 = 1 \end{aligned}$$

Berikut merupakan hasil perhitungan *log term frequency* yang ditunjukkan pada Tabel 4.18.

Tabel 4.18 Hasil Log Term Frequency Weighting

Term	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5	Dok Uji
antri	0	0	0	1	1	0
naik	0	0	0	1	1	0
aja	1.301	0	0	0	0	0
bawa	1	0	0	0	0	0
bersih	0	1	0	0	0	1
bgts	1	0	0	0	0	0
buat	1	0	0	0	0	0
cm	1	0	0	0	0	0
corridor	0	0	1	0	0	0
dalem	1	0	0	0	0	0
deh	1	0	0	0	0	0

Tabel 4.18 Hasil Log Term Frequency Weighting (Lanjutan)

ditunjukin	1	0	0	0	0	0
dpt	1	0	0	0	0	0
dr	1	0	0	0	0	0
dtg	1	0	0	0	0	0
gading	0	0	1	0	0	1
gengs	1	0	0	0	0	0
gerbong	0	0	1	0	0	0
gretongnya	1	0	0	0	0	0
hanya	0	0	1.301	0	0	0
itu	1	0	0	0	0	0
jakarta	0	0	1.301	0	0	0
jg	1	0	0	0	0	0
jgn	1	0	0	0	0	0
jumlah	0	0	1	0	0	0
karna	1	0	0	0	0	0
kelapa	0	0	1	0	0	1
keren	0	1	0	0	0	0
kereta	0	0	1	0	0	0
ktp	1	0	0	0	0	0
lgsung	1	0	0	0	0	0
lima	0	0	1	0	0	0
loket	1	0	0	0	0	0
lrt	1.301	0	1.301	1	1	1
lupa	1	0	0	0	0	0
mau	1	0	0	0	0	0
mungil	0	0	1	0	0	0
murah	0	1	0	0	0	0
nyoba	1	0	0	0	0	0
phase	0	0	1	0	0	0
pintu	1	0	0	0	0	0

Tabel 4.18 Hasil Log Term Frequency Weighting (Lanjutan)

praktis	0	1	0	0	0	0
ramah	1	0	0	0	0	0
sepi	0	0	1	0	0	0
smp	1	0	0	0	0	0
sukses	1	0	0	0	0	0
syarat	1	0	0	0	0	0
tenang	1	0	0	0	0	0
tiga	0	0	1	0	0	0
tiket	1	0	0	0	0	0
trdekat	1	0	0	0	0	0
trus	1	0	0	0	0	0
tugas	1	0	0	0	0	0
uji	1	0	0	0	0	0
velodrome	0	0	1	0	0	0
yg	1	0	0	0	0	0
yuk	1	0	0	0	0	0

4.2.4.3 Inverse Document Frequency

Sebelum menghitung $tf \ idf$, dilakukan proses perhitungan *Inverse Document Frequency* (IDF) yang sesuai pada Persamaan 2.2 dengan contoh perhitungan pada term “bersih” sebagai berikut.

$$idf(bersih) = \frac{\log(5)}{1} \\ = 0.69897$$

Jumlah dokumen yang digunakan pada data latih sebanyak 5 dokumen serta term “bersih” hanya muncul pada dokumen 2 sehingga nilai df yaitu 1. Adapun hasil perhitungan IDF ditunjukkan pada Tabel 4.19.

Tabel 4.19 Hasil Inverse Document Frequency

Term	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5	Dok Uji	IDF
antri	0	0	0	1	1	0	0.3979
naik	0	0	0	1	1	0	0.3979
aja	1	0	0	0	0	0	0.6990
bawa	1	0	0	0	0	0	0.6990

Tabel 4.19 Hasil *Inverse Document Frequency* (Lanjutan)

bersih	0	1	0	0	0	1	0.6990
bgts	1	0	0	0	0	0	0.6990
buat	1	0	0	0	0	0	0.6990
cm	1	0	0	0	0	0	0.6990
corridor	0	0	1	0	0	0	0.6990
dalem	1	0	0	0	0	0	0.6990
deh	1	0	0	0	0	0	0.6990
ditunjukin	1	0	0	0	0	0	0.6990
dpt	1	0	0	0	0	0	0.6990
dr	1	0	0	0	0	0	0.6990
dtg	1	0	0	0	0	0	0.6990
gading	0	0	1	0	0	1	0.6990
gengs	1	0	0	0	0	0	0.6990
gerbong	0	0	1	0	0	0	0.6990
gretongnya	1	0	0	0	0	0	0.6990
hanya	0	0	1	0	0	0	0.6990
itu	1	0	0	0	0	0	0.6990
jakarta	0	0	1	0	0	0	0.6990
jg	1	0	0	0	0	0	0.6990
jgn	1	0	0	0	0	0	0.6990
jumlah	0	0	1	0	0	0	0.6990
karna	1	0	0	0	0	0	0.6990
kelapa	0	0	1	0	0	1	0.6990
keren	0	1	0	0	0	0	0.6990
kereta	0	0	1	0	0	0	0.6990
ktp	1	0	0	0	0	0	0.6990
lgsung	1	0	0	0	0	0	0.6990
lima	0	0	1	0	0	0	0.6990
loket	1	0	0	0	0	0	0.6990
Irt	1	0	1	1	1	1	0.0969

Tabel 4.19 Hasil *Inverse Document Frequency* (Lanjutan)

lupa	1	0	0	0	0	0	0.6990
mau	1	0	0	0	0	0	0.6990
mungil	0	0	1	0	0	0	0.6990
murah	0	1	0	0	0	0	0.6990
nyoba	1	0	0	0	0	0	0.6990
phase	0	0	1	0	0	0	0.6990
pintu	1	0	0	0	0	0	0.6990
praktis	0	1	0	0	0	0	0.6990
ramah	1	0	0	0	0	0	0.6990
sepi	0	0	1	0	0	0	0.6990
smp	1	0	0	0	0	0	0.6990
sukses	1	0	0	0	0	0	0.6990
syarat	1	0	0	0	0	0	0.6990
tenang	1	0	0	0	0	0	0.6990
tiga	0	0	1	0	0	0	0.6990
tiket	1	0	0	0	0	0	0.6990
trdekat	1	0	0	0	0	0	0.6990
trus	1	0	0	0	0	0	0.6990
tugas	1	0	0	0	0	0	0.6990
uji	1	0	0	0	0	0	0.6990
velodrome	0	0	1	0	0	0	0.6990
yg	1	0	0	0	0	0	0.6990
yuk	1	0	0	0	0	0	0.6990

4.2.4.4 Term Frequency Inverse Document Frequency

Setelah menghitung IDF, langkah terakhir pada *term weighting* yaitu menghitung *Term Frequency Inverse Document Frequency* (*tf idf*) seperti pada Persamaan 2.3. Contoh perhitungan *tf idf* pada term “bersih” pada dokumen dua sebagai berikut.

$$\begin{aligned} tf.idf(bersih) &= 1 * 0.69897 \\ &= 0.69897 \end{aligned}$$

Adapun hasil perhitungan *tf idf* sebagaimana ditunjukkan pada Tabel 4.20.

Tabel 4.20 Hasil Term Frequency Inverse Document Frequency

Term	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5	Dok Uji
antri	0.0000	0.0000	0.0000	0.3979	0.3979	0.0000
naik	0.0000	0.0000	0.0000	0.3979	0.3979	0.0000
aja	0.9094	0.0000	0.0000	0.0000	0.0000	0.0000
bawa	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
bersih	0.0000	0.6990	0.0000	0.0000	0.0000	0.6990
bgts	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
buat	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
cm	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
corridor	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
dalem	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
deh	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
ditunjukin	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
dpt	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
dr	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
dtg	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
gading	0.0000	0.0000	0.6990	0.0000	0.0000	0.6990
gengs	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
gerbong	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
gretongnya	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
hanya	0.0000	0.0000	0.9094	0.0000	0.0000	0.0000
itu	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
jakarta	0.0000	0.0000	0.9094	0.0000	0.0000	0.0000
jg	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
jgn	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
jumlah	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
karna	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
kelapa	0.0000	0.0000	0.6990	0.0000	0.0000	0.6990
keren	0.0000	0.6990	0.0000	0.0000	0.0000	0.0000
kereta	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000

Tabel 4.20 Hasil Term Frequency Inverse Document Frequency (Lanjutan)

ktp	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
lgsung	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
lima	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
loket	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
Irt	0.1261	0.0000	0.1261	0.0969	0.0969	0.0969
lupa	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
mau	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
mungil	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
murah	0.0000	0.6990	0.0000	0.0000	0.0000	0.0000
nyoba	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
phase	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
pintu	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
praktis	0.0000	0.6990	0.0000	0.0000	0.0000	0.0000
ramah	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
sepi	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
smp	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
sukses	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
syarat	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
tenang	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
tiga	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
tiket	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
trdekat	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
trus	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
tugas	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
uji	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
velodrome	0.0000	0.0000	0.6990	0.0000	0.0000	0.0000
yg	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000
yuk	0.6990	0.0000	0.0000	0.0000	0.0000	0.0000

4.2.5 Menghitung *Cosine Similarity*

Sebelum melakukan klasifikasi *Improved KNN*, dilakukan perhitungan kemiripan antara data uji dengan seluruh data latih. Metode yang digunakan yaitu *cosine similarity*. Berikut merupakan nilai TF-IDF yang digunakan untuk menghitung perkalian vektor pada data latih dengan data uji dicontohkan oleh dokumen kedua sebagai data latih ditunjukkan oleh Tabel 4.21.

Tabel 4.21 TF-IDF pada Data Latih dengan Contoh Dokumen Dua serta Data Uji

Term	Dok 2	Dok Uji
bersih	0.69897	0.69897
gading	0	0.69897
kelapa	0	0.69897
Irt	0	0.09691

Contoh perhitungan dari perkalian vektor data latih dengan data uji pada term bersih ditunjukkan di bawah ini.

$$v_{d2} * v_q = 0.69897 * 0.69897 = 0.488559067$$

Sehingga didapatkan hasil perkalian vektor antara dokumen dua dengan dokumen uji yang ditunjukkan pada Tabel 4.22

Tabel 4.22 Perkalian Vektor antara Dokumen Dua dengan Dokumen Uji

Dok 2 * Dok Uji
0.488559067
0
0
0

Untuk menghitung *Cosine Similarity* seperti pada Persamaan 2.5 membutuhkan jumlah dari hasil perkalian vektor data latih dan data uji yang ditunjukkan pada Tabel 4.23.

Tabel 4.23 Hasil Penjumlahan dari Perkalian Vektor Data Latih dengan Data Uji

Dok 1 * Dok Uji	Dok 2 * Dok Uji	Dok 3 * Dok Uji	Dok 4 * Dok Uji	Dok 5 * Dok Uji
0.012218689	0.488559067	0.989336823	0.009391551	0.009391551

Kemudian, hasil perhitungan nilai akar kuadrat jumlah dari masing-masing vektor data latih serta data uji yang ditunjukkan pada Tabel 4.24.

Tabel 4.24 Hasil Akar Kuadrat Jumlah Vektor Data Latih serta Data Uji

Dok 1	Dok 2	Dok 3	Dok 4	Dok 5	Dok Uji
4.23585	1.39794	2.74455	0.57106	0.57106	1.21452

Perhitungan *cosine similarity* dengan contoh manualisasi pada dokumen kedua dengan dokumen uji sebagai berikut.

$$\text{cosSim}(\text{dokDua}, \text{dokUji}) = \frac{0.488559067}{1.39794 * 1.21452}$$

$$\text{cosSim}(\text{dokDua}, \text{dokUji}) = 0.287754691$$

Adapun hasil perhitungan *Cosine Similarity* pada data latih dengan data uji ditunjukkan pada Tabel 4.25.

Tabel 4.25 Nilai Cosine Similarity Data Latih dengan Data Uji

Data Latih	Nilai Cosine Similarity	Kelas Data Latih
Dok 3	0.296802081	Positif
Dok 2	0.287754691	Positif
Dok 1	0.002375077	Positif
Dok 4	0.013541073	Negatif
Dok 5	0.013541073	Negatif

4.2.6 Klasifikasi dengan *Improved K-Nearest Neighbor*

Langkah pertama, seperti pada algoritme KNN yaitu menentukan nilai k atau tetangga terdeka yang ditunjukkan pada Tabel 4.26.

Tabel 4.26 Penentuan Nilai K Awal

Nilai k awal
3

Langkah kedua, menentukan nilai k berdasarkan kelas dan nilai k awal menggunakan Persamaan 2.3. Pada bagian inilah yang membedakan *Improved KNN* dengan *base KNN*. Perhitungan nilai k pada kelas positif adalah sebagai berikut.

$$n_{positif} = \left\lceil \frac{3 \times 3}{3} \right\rceil$$

$$n_{positif} = 3$$

Kemudian, penentuan nilai k pada kelas negatif ditunjukkan oleh perhitungan di bawah ini.

$$n_{negatif} = \left\lceil \frac{2 \times 3}{3} \right\rceil$$

$$n_{negatif} = 2$$

Adapun hasil perhitungan nilai k baru pada masing-masing kelas ditunjukkan pada Tabel 4.27.

Tabel 4.27 Nilai k baru pada Kelas Positif dan Negatif

K kelas positif	K kelas negatif
3	2

Langkah ketiga, yaitu menghitung peluang data uji masuk pada kelas positif dan negatif sesuai dengan tetangga terdekat masing-masing kelas menggunakan Persamaan 2.4. Sebelumnya, dilakukan proses perhitungan jumlah nilai *Cosine Similarity* dari data latih seperti pada *base KNN* menggunakan nilai k awal yaitu sebagai berikut.

$$\begin{aligned}\sum \text{similarity}(d_i, x_j) &= \text{Dok 3} + \text{Dok 2} + \text{Dok 1} \\ &= 0.296802081 + 0.287754691 + 0.00237507 \\ &= 0.586931849\end{aligned}$$

$$\sum \text{similarity}(d_i, x_j) = 0.586931849$$

Perhitungan data uji masuk kelas positif ditunjukkan manualisasi di bawah ini.

$$p(d_6, \text{positif}) = \frac{\text{Dok 3} * 1 + \text{Dok 2} * 1 + \text{Dok 1} * 1}{0.586931849}$$

$$p(d_6, \text{positif}) = \frac{0.35585777 * 1 + 0.287754691 * 1 + 0.00237507 * 1}{0.586931849}$$

$$p(d_6, \text{positif}) = 1$$

Kemudian, perhitungan data uji masuk kelas negatif ditunjukkan manualisasi di bawah ini.

$$p(d_6, \text{negatif}) = \frac{\text{Dok 3} * 0 + \text{Dok 2} * 0}{0.586931849}$$

$$p(d_6, \text{negatif}) = \frac{0 + 0}{0.586931849}$$

$$p(d_6, \text{negatif}) = 0$$

Berdasarkan hasil di atas dapat dilihat bahwa peluang data uji menjadi anggot kelas positif lebih besar dibandingkan kelas negatif sehingga kelas dari data uji yaitu positif seperti pada Tabel 4.28.

Tabel 4.28 Hasil Klasifikasi dari Data Uji

No	Teks	Kelas
1	mumpung di kelapa gading nyobain lrt masih bersih gratis serasa di jepun	Positif

4.3 Perancangan Skenario Pengujian

Perancangan skenario pengujian bertujuan untuk mengetahui evaluasi dan hasil kinerja algoritme dalam melakukan klasifikasi. Skenario pengujian yang dilakukan pada penelitian ini yaitu pengujian pada nilai k awal dan pengujian seleksi fitur *Information Gain*.

4.3.1 Pengujian Nilai k Awal

Pengujian dengan menentukan nilai k awal ini bertujuan untuk mengetahui seberapa besar pengaruh nilai k terhadap evaluasi dari segi *recall*, *precision*, *f-measure*. Pengujian pada penelitian ini menggunakan variasi nilai k awal yaitu 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 100 dan 500 dengan *5-fold cross validation*. Skenario pengujian nilai k awal ditunjukkan pada Tabel 4.29.

Tabel 4.29 Skenario Pengujian Nilai k Awal

Nilai k Awal	Nilai k pada Kelas		<i>Recall</i>	<i>Precision</i>	<i>F-measure</i>
	Positif	Negatif			
3					
5					
7					
9					
11					
13					
15					
17					
19					
21					
100					
400					

4.3.2 Pengujian Seleksi Fitur *Information Gain*

Pengujian seleksi fitur ini bertujuan untuk mengetahui seberapa besar pengaruh *threshold* dari *Information Gain* terhadap hasil klasifikasi yang diukur dari evaluasi *recall*, *precision* dan *f-measure*. *Threshold* yang digunakan pada penelitian ini yaitu berdasarkan persentase jumlah fitur yang digunakan dimulai dari 10% hingga 100%, *mean* atau nilai rata-rata, dan *median* nilai tengah dari *information gain*. Skenario pengujian ini ditunjukkan pada Tabel 4.30.

Tabel 4.30 Skenario Pengujian Seleksi Fitur *Information Gain*

Metrik	Threshold											
	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	<i>mean</i>	<i>median</i>
<i>Recall</i>												
<i>Precision</i>												
<i>F-measure</i>												

BAB 5 IMPLEMENTASI

Pada bagian implementasi dijelaskan kode program berdasarkan perancangan yang telah dilakukan serta penjelasan dari tiap baris pada kode program.

5.1 Implementasi *Preprocessing*

Implementasi *pre-processing* meliputi *cleaning*, *case folding*, *tokenization*, *stopword removal*, dan *stemming*. Adapun pada tahap *preprocessing* dibutuhkan beberapa *library* untuk mendukung implementasi tahap ini yang dapat dilihat pada Kode Program 5.1.

Import Library	
1	import string
2	import re
3	import pandas as pd
4	import demoji
5	demoji.download_codes()
6	from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
7	from sklearn.feature_extraction.text import CountVectorizer

Kode Program 5.1 Import Library

Penjelasan baris pada Kode Program 5.1 adalah sebagai berikut.

1. Baris 1 digunakan untuk *import built-in library string*.
2. Baris 2 digunakan untuk *import built-in library re* untuk regex.
3. Baris 3 digunakan untuk *import library pandas*.
4. Baris 4 digunakan untuk *import library demoji*.
5. Baris 5 digunakan untuk mengunduh emoji menggunakan *library demoji*.
6. Baris 6 digunakan untuk *import library Sastrawi* yang digunakan saat *stemming*.
7. Baris 7 digunakan untuk *import library CountVectorizer* dari *sklearn* untuk menghasilkan term.

Berikutnya merupakan tahap *cleaning* sebagai tahap pertama pada *preprocessing* dengan kode program pada Kode Program 5.2.

Algoritme Cleaning	
1	def cleaning(self, data, col):
2	clean_data = data.copy()
3	#regex
4	remove_username_hash = r'[@]\w+'
5	remove_punc = '[%s]+' % re.escape(string.punctuation)
6	remove_number = r'[0-9]+'
7	remove_space = r'\s{2,}'
8	remove_space_begin_end = r'^\s+ \s+\$' #when removing '1 JUTA PERSEN....'
9	remove_url = r'(https? www) : \/{1,} \w+ \w+ \w+ \/{1,} \w+' #https://t.co/RQrptGqXtR remove_character_reference = r'& \w+;' # > <

```

10     clean_data[col] =
11         clean_data[col].str.replace(remove_character_reference, ' ', 
12             regex=True)
12         clean_data[col] = clean_data[col].apply(lambda row_text:
13             demoji.replace(row_text, ''))
13         clean_data[col] =
14             clean_data[col].str.replace(remove_username_hash, ' ', regex=True)
14             clean_data[col] = clean_data[col].str.replace(remove_url, ' ', 
15                 regex=True)
15                 clean_data[col] = clean_data[col].str.replace(remove_punc, ' 
16                 ', regex=True)
16                 clean_data[col] = clean_data[col].str.replace(remove_number, ' 
17                 ', regex=True)
17                 clean_data[col] =
18                     clean_data[col].str.replace(remove_space_begin_end, ' ', 
19                         regex=True)

20             return clean_data

```

Kode Program 5.2 Implementasi Cleaning

Penjelasan baris pada Kode Program 5.2 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *Cleaning* dengan parameter data berupa dataframe dan col berupa kolom pada data.
2. Baris 2 digunakan untuk melakukan *copy dataframe* yang disimpan dalam variabel *clean_data*.
3. Baris 3 digunakan untuk mendefinisikan *regex* untuk menghapus *hashtag* dan *username* yang diawali oleh @ pada Twitter.
4. Baris 4 digunakan untuk mendefinisikan *regex* untuk menghapus tanda baca.
5. Baris 5 digunakan untuk mendefinisikan *regex* untuk menghapus angka.
6. Baris 6 digunakan untuk mendefinisikan *regex* untuk menghapus spasi dengan jumlah lebih dari dua.
7. Baris 7 digunakan untuk mendefinisikan *regex* untuk menghapus spasi pada awal dan akhir kalimat.
8. Baris 8 digunakan untuk mendefinisikan *regex* untuk menghapus *url*.
9. Baris 9 digunakan untuk mendefinisikan *regex* untuk menghapus *character reference* seperti &#;
10. Baris 10 digunakan untuk mengganti kolom dokumen yang mengandung *character reference* menjadi spasi.
11. Baris 11 digunakan untuk mengganti kolom dokumen yang mengandung *emoji* menjadi *string* kosong.
12. Baris 12 digunakan untuk mengganti kolom dokumen yang mengandung *username* dan *hashtag* menjadi spasi.
13. Baris 13 digunakan untuk mengganti kolom dokumen yang mengandung *url* menjadi spasi.

14. Baris 14 digunakan untuk mengganti kolom dokumen yang mengandung tanda baca menjadi spasi.
15. Baris 15 digunakan untuk mengganti kolom dokumen yang mengandung angka menjadi spasi.
16. Baris 16 digunakan untuk mengganti kolom dokumen yang mengandung spasi lebih dari dua menjadi spasi.
17. Baris 17 digunakan untuk mengganti kolom dokumen yang mengandung spasi pada awal dan akhir teks menjadi *string* kosong.
18. Baris 18 digunakan untuk mengembalikan hasil *Cleaning*.

Tahap berikutnya yaitu *case folding* yang dapat dilihat pada Kode Program 5.3.

Algoritme Case Folding	
1	def case_folding(self, data, col):
	#we use copy to prevent object reference
2	case_fold_data = data.copy()
3	case_fold_data[col] = case_fold_data[col].str.lower()
4	return case_fold_data

Kode Program 5.3 Implementasi Case Folding

Penjelasan baris pada Kode Program 5.3 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *case_folding* dengan parameter data berupa *dataframe* dan col berupa kolom pada data.
2. Baris 2 digunakan untuk melakukan duplikasi pada *dataframe* yang disimpan pada variabel *case_fold_data*.
3. Baris 3 digunakan untuk merubah seluruh teks pada kolom data menjadi huruf kecil dengan fungsi *lower*.
4. Baris 4 digunakan untuk mengembalikan nilai dari fungsi *case_folding*.

Setelah melakukan *case folding*, dilakukan proses *tokenization* seperti pada Kode Program 5.4.

Algoritme Tokenization	
1	def tokenization(self, data, col):
2	token_data = data.copy()
3	token_data[col] = token_data[col].apply(lambda sentence: sentence.split(' '))
4	return token_data

Kode Program 5.4 Implementasi Tokenization

Penjelasan baris pada Kode Program 5.4 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *tokenization* dengan parameter data berupa *dataframe* dan col berupa kolom dari data.
2. Baris 2 digunakan untuk melakukan duplikasi pada *dataframe* yang disimpan pada variabel *token_data*.

3. Baris 3 digunakan untuk membentuk kata dari kalimat pada kolom *dataframe* dengan fungsi *split*.
4. Baris 4 digunakan untuk mengembalikan nilai dari fungsi *tokenization*.

Langkah berikutnya yaitu melakukan *stemming* seperti yang dapat dilihat pada Kode Program 5.5.

Algoritme Stemming

```
1 def stemming(self, data, col):
2     stem_data = data.copy()
3     stemmer_factory = StemmerFactory()
4     stemmer = stemmer_factory.create_stemmer()
5
6     for i in range(len(stem_data[col])):
7         for j in range(len(stem_data[col][i])):
8             stem_data.loc[i, col][j] =
stemmer.stem(stem_data[col][i][j])
9
10    return stem_data
```

Kode Program 5.5 Implementasi Stemming

Penjelasan baris pada Kode Program 5.5 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *stemming* dengan parameter data berupa *dataframe* dan col berupa kolom pada data.
2. Baris 2 digunakan untuk melakukan duplikasi pada *dataframe* yang disimpan pada variabel *stem_data*.
3. Baris 3 digunakan untuk instansiasi *class StemmerFactory* yang disimpan dalam variabel *stemmer_factory*.
4. Baris 4 digunakan untuk membuat *stemmer* dengan fungsi *create_stemmer* dari objek *stemmer_factory*.
5. Baris 5 hingga 7 digunakan untuk melakukan proses *stemming* pada tiap kata ke-j dari dokumen ke-i.
6. Baris 8 digunakan untuk mengembalikan nilai *stem_data*.

Berikut merupakan fungsi untuk menyatukan proses *preprocessing* seperti pada Kode Program 5.6.

Algoritme Preprocess Data

```
1 def preprocess_data(self, data, col, notes):
2     pre_data = data.copy()
3     clean_data = self.cleaning(pre_data, 'dokumen')
4     case_fold_data = self.case_folding(clean_data, 'dokumen')
5     tokenization_data = self.tokenization(case_fold_data,
'dokumen')
6     stemming_data = self.stemming(stopword_data, 'dokumen')
7
8     return stemming_data
```

Kode Program 5.6 Implementasi Preprocess Data untuk Preprocessing

Penjelasan baris pada Kode Program 5.6 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *preprocess_data* dengan parameter *data* berupa *dataframe* dan *col* berupa kolom pada *data*.
2. Baris 2 digunakan untuk melakukan duplikasi pada *dataframe* yang disimpan dalam variabel *pre_data*.
3. Baris 3 hingga 6 digunakan untuk melakukan proses *preprocessing* dengan tahapan *Cleaning*, *tokenization*, *stopword removal* dan *stemming*.
4. Baris 8 digunakan untuk mengembalikan nilai hasil dari *stemming*.

Berikut merupakan fungsi untuk mendapatkan *term* atau fitur setelah tahap *preprocessing* seperti pada Kode Program 5.7.

Algoritme Get Term	
1	def get_term(self, data, col): 2 count_vect = CountVectorizer() 3 term_data = data.copy() 4 term_data[col] = term_data[col].apply(lambda sentence: 5 ''.join(sentence)) 6 count_vect.fit(term_data[col]) 6 return count_vect.get_feature_names() #return term (single character will be removed)

Kode Program 5.7 Implementasi Get Term

Penjelasan dari Kode Program 5.7 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *get_term* dengan parameter *data* berupa *dataframe* dan *col* berupa kolom dari *data*.
2. Baris 2 digunakan untuk instansiasi *class CountVectorizer* yang disimpan dalam variabel *count_vect*.
3. Baris 3 digunakan untuk melakukan duplikasi pada *dataframe* yang disimpan dalam variabel *term_data*.
4. Baris 4 digunakan untuk menggabungkan kata pada tiap dokumen menjadi kalimat dengan fungsi *join*.
5. Baris 5 digunakan untuk mempelajari isi dari dokumen.
6. Baris 6 digunakan untuk mendapatkan fitur atau term dari dokumen yang telah dilakukan *fit* sebelumnya.

5.2 Implementasi Term Weighting

Implementasi *term weighting* merupakan implementasi untuk menghitung bobot pada tiap fitur atau term. Proses ini meliputi perhitungan *term frequency*, *log term frequency*, *document frequency*, *inverse document frequency*, dan *term frequency inverse document frequency*. Adapun pada tahap *term weighting* dibutuhkan beberapa *library* untuk mendukung implementasi tahap ini yang dapat dilihat pada Kode Program 5.8.

Import Library	
1	import pandas as pd
2	import warnings

```

3 import numpy as np
4 warnings.filterwarnings('ignore')

```

Kode Program 5.8 Import Library

Penjelasan dari Kode Program 5.8 adalah sebagai berikut.

1. Baris 1 digunakan untuk *import library* bernama Pandas.
2. Baris 2 digunakan untuk *import built-in library* bernama *warnings*.
3. Baris 3 digunakan untuk *import library* bernama *numpy*.
4. Baris 4 digunakan untuk mengabaikan pesan peringatan atau *warning* yang dihasilkan saat program dijalankan.

Tahap pertama pada *term weighting* yaitu dengan menghitung *term frequency* yang dapat dilihat pada Kode Program 5.9.

Algoritme Term Frequency	
1	def term_frequency(self, term, data, col): #term: the term produced by vectorizer, data: list of documents
2	tf_matrix = []
3	for i in range(len(term)):
4	tf_each_term = []
5	for j in range(len(data)):
6	if term[i] in data[col][j]:
7	tf = data[col][j].count(term[i])
8	else:
9	tf = 0
10	tf_each_term.append(tf)
11	tf_matrix.append(tf_each_term)
12	#df stands for DataFrame
13	term_df = pd.DataFrame({'term': term})
14	doc_df = pd.DataFrame(tf_matrix)
15	doc_columns = ['dok ' + str(i+1) for i in range(len(data[col]))]
16	doc_df.columns = doc_columns #replace column with document name
17	tf_matrix_df = pd.concat([term_df, doc_df], axis=1)
	return tf_matrix df

Kode Program 5.9 Implementasi Term Frequency

Penjelasan dari Kode Program 5.9 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *term_frequency* dengan parameter *term*, *data* dan *col*.
2. Baris 2 digunakan untuk mendefinisikan *list* kosong yang disimpan dalam variabel *tf_matrix*.
3. Baris 3 hingga 15 digunakan untuk melakukan pengecekan term hasil *preprocessing* terdapat pada dokumen. Jika dokumen ke-j mengandung term ke-l maka hitung kemunculan term dengan fungsi *count*. Jika sebaliknya maka kemunculan term tersebut adalah 0. Pada setiap dokumen dilakukan penambahan nilai *term frequency* ke dalam list *tf_each_term*.

4. Baris 4 digunakan untuk menambah *list* *tf_each_term* ke dalam *list* *tf_matrix*.
5. Baris 5 digunakan untuk membuat *dataframe* yang berisi term.
6. Baris 6 digunakan untuk membuat *dataframe* yang berisi *tf_matrix*.
7. Baris 7 digunakan untuk membuat *list* berisi *string* bernilai ‘dok 1’ hingga sepanjang dokumen disimpan dalam variabel *doc_columns*.
8. Baris 8 digunakan untuk mengganti kolom *dataframe* saat ini dengan *doc_columns*.
9. Baris 9 digunakan untuk menggabungkan *dataframe* berisi term dengan *dataframe* berisi nilai dari *term frequency*.
10. Baris 10 digunakan untuk mengembalikan nilai *tf_matrix_df*.

Selanjutnya, menghitung *log term frequency* yang dapat dilihat pada Kode Program 5.10.

Algoritme Log Term Frequency	
1	def log_fn(self, row):
2	if row > 0:
3	return np.log10(row)+1
4	else:
5	return 0
6	def log_tf(self, term_freq_matrix):
7	log_tf = term_freq_matrix.copy()
8	column_docs = list(log_tf.filter(regex='dok').columns)
9	for each_doc in column_docs:
10	log_tf.loc[:, each_doc] =
11	log_tf[each_doc].apply(self.log_fn)

Kode Program 5.10 Implementasi Log Term Frequency

Penjelasan dari Kode Program 5.11 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *log_fn* dengan parameter *row*.
2. Baris 2 hingga 5 digunakan untuk mengembalikan nilai *log term frequency* jika nilai *row* lebih dari 0. Jika sebaliknya maka mengembalikan nilai 0.
3. Baris 6 digunakan untuk mendefinisikan fungsi *log_tf* dengan parameter *term_freq_matrix*.
4. Baris 7 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *log_tf*.
5. Baris 8 digunakan untuk menyaring *dataframe* yang memiliki kolom dengan nama yang mengandung ‘dok’ kemudian mengambil kolom dan diubah menjadi *list*.
6. Baris 9 hingga 10 digunakan untuk melakukan *log term frequency* pada tiap baris dengan fungsi *apply* dan fungsi *log_fn* sebagai parameter.
7. Baris 11 digunakan untuk mengembalikan nilai *log_tf*.

Tahap berikutnya yaitu menghitung *document frequency* yang dapat dilihat pada Kode Program 5.11.

Algoritme Document Frequency

```

1 def document_frequency(self, tf_matrix_df):
2     list_col = list(tf_matrix_df.columns)
3     docfreq_matrix = tf_matrix_df.copy()
4
5     for i in range(len(docfreq_matrix['term'])):
6         for j in range(1, len(list_col)): #start from 1 because we
want to check the value on dok 1 column till end.
7             if docfreq_matrix.iat[i, j] > 0:
8                 docfreq_matrix.iat[i, j] = 1
9
#create df_total column
8     docfreq_matrix['df'] = docfreq_matrix.iloc[:, 1: ].apply(np.sum, axis=1) #1: means document column till end.
9
return docfreq_matrix

```

Kode Program 5.11 Implementasi *Document Frequency*

Penjelasan dari Kode Program 5.11 adalah sebagai berikut.

- Baris 1 digunakan untuk mendefinisikan fungsi *document_frequency* dengan parameter *tf_matrix_df*.
- Baris 2 digunakan untuk merubah nilai kolom dari *tf_matrix_df* menjadi *list* dan disimpan pada variabel *list_col*.
- Baris 3 digunakan untuk duplikasi *dataframe* yang disimpan dalam variabel *docfreq_matrix*.
- Baris 4 hingga 7 digunakan untuk memberi nilai term ke-i pada dokumen ke-j dengan 1 jika nilai *term_frequency* pada baris dan kolom tersebut lebih dari 0.
- Baris 8 digunakan untuk membuat kolom df yang berisi nilai penjumlahan baris pada tiap term menggunakan fungsi *apply* dengan parameter *sum* dan *axis* yang bernilai 1.
- Baris 9 digunakan untuk mengembalikan nilai *docfreq_matrix*.

Adapun fungsi untuk memilih *term* yang digunakan pada algoritme dari hasil seleksi fitur dengan *information gain* ditunjukkan pada Kode Program 5.12.

Algoritme Term Frequency Information Gain
--

```

1 def term_frequency_ig(self, tf_matrix_df, filter_ig_matrix):
2     tf_ig_matrix =
3     tf_matrix_df[tf_matrix_df['term'].isin(filter_ig_matrix['term'])]
4     tf_ig_matrix = tf_ig_matrix.reset_index(drop=True)

```

Kode Program 5.12 Implementasi *Term Frequency Information Gain*

Penjelasan dari Kode Program 5.12 adalah sebagai berikut.

- Baris 1 digunaakn untuk mendefinisikan fungsi *term_frequency_ig* dengan parameter *tf_matrix_df* dan *filter_ig_matrix*.

2. Baris 2 digunakan untuk menyaring term pada matriks *term frequency* sesuai dengan term yang dihasilkan pada proses seleksi fitur dengan *information gain*.
3. Baris 3 digunakan untuk mengembalikan urutan *index* pada *dataframe*.
4. Baris 4 digunakan untuk mengembalikan nilai *tf_ig_matrix*.

Selanjutnya menghitung *inverse document frequency* (IDF) yang dapat dilihat pada Kode Program 5.13.

Algoritme Inverse Document Frequency	
1 def inverse_document_frequency(self, training_data, 2 docfreq_matrix): 3 idf_list = [] 4 idf_matrix = docfreq_matrix.copy() 5 idf_matrix = idf_matrix.reset_index(drop=True) 6 print('Matrix idf copy\n', idf_matrix) 7 column_docs = list(idf_matrix.filter(regex='dok').columns) 8 print(column_docs) 9 N = len(training_data) #N using docs in training data. 10 for i in range(len(idf_matrix['term'])): 11 df_term = docfreq_matrix['df'].iloc[i] 12 print('df term', df_term) 13 if df_term > 0: 14 idf_list.append(np.log10(N/df_term)) 15 else: 16 idf_list.append(0) 17 idf_df = pd.DataFrame({'idf': idf_list}) 18 idf_matrix = pd.concat([idf_matrix, idf_df], axis=1) 19 return idf_matrix	

Kode Program 5.13 Implementasi Inverse Document Frequency

Penjelasan dari Kode Program 5.13 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *inverse_document_frequency* dengan parameter *training_data* dan *docfreq_matrix*.
2. Baris 2 digunakan untuk mendefinisikan *list* kosong yang disimpan dalam variabel *idf_list*.
3. Baris 3 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *idf_matrix*.
4. Baris 4 digunakan untuk mengembalikan urutan *index* pada *dataframe*.
5. Baris 5 digunakan untuk mencetak nilai *idf_matrix*.
6. Baris 6 digunakan untuk menyaring *dataframe* yang memiliki kolom dengan nama yang mengandung 'dok' kemudian mengambil kolom dan diubah menjadi *list*.
7. Baris 7 digunakan untuk mencetak *column_docs*.
8. Baris 8 digunakan untuk mendefinisikan panjang dari data latih yang disimpan dalam variabel N.

9. Baris 9 hingga 50 digunakan untuk menambah nilai dari IDF ke dalam *idf_list* jika nilai *document frequency* lebih dari 0. Jika sebaliknya maka nilai 0 akan ditambah pada *idf_list*.
10. Baris 10 digunakan untuk membuat *dataframe* dengan nama kolom ‘*idf*’ berisi nilai dari *idf_list*.
11. Baris 11 digunakan untuk menggabungkan *dataframe* *idf_matrix* dengan *idf_df*.
12. Baris 12 digunakan untuk mengembalikan nilai *idf_matrix*.

Langkah terakhir pada tahap *term weighting* yaitu menghitung *tf idf* yang dapat dilihat pada Kode Program 5.14.

Algoritme Term Frequency Inverse Document Frequency Weighting	
1	def tf_idf(self, log_tf_matrix, docfreq_idf_matrix):
2	idf_matrix = docfreq_idf_matrix.copy()
3	tf_matrix = log_tf_matrix.copy()
4	tf_matrix['idf'] = idf_matrix['idf']
5	tf_idf_matrix = tf_matrix.copy()
6	column_docs = list(tf_matrix.filter(regex='dok').columns)
7	for each_doc in column_docs:
8	tf_idf_matrix.loc[:, each_doc] =
	tf_idf_matrix.apply(lambda row: row[each_doc] * row['idf'], axis=1)
9	return tf_idf_matrix

Kode Program 5.14 Implementasi Term Frequency Inverse Document Frequency Weighting

Penjelasan dari Kode Program 5.14 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *tf_idf* dengan parameter *log_tf_matrix* dan *docfreq_idf_matrix*.
2. Baris 2 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam *idf_matrix*.
3. Baris 3 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam *tf_matrix*.
4. Baris 4 digunakan untuk memberi nilai pada kolom ‘*idf*’ yaitu nilai dari kolom ‘*idf*’ pada *dataframe* *idf_matrix*.
5. Baris 5 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam *tf_idf_matrix*.
6. Baris 6 digunakan untuk menyaring *dataframe* yang memiliki kolom dengan nama yang mengandung ‘*dok*’ kemudian mengambil kolom dan diubah menjadi *list*.
7. Baris 7 hingga 8 digunakan untuk mengalikan nilai dari baris tiap dokumen dengan nilai baris pada kolom ‘*idf*’ dengan fungsi *apply* dan fungsi *lambda* sebagai parameter.

- Baris 9 digunakan untuk mengembalikan nilai *tf_idf_matrix*.

5.3 Implementasi *Information Gain*

Implementasi *information gain* merupakan implementasi untuk melakukan seleksi fitur dari term yang didapatkan setelah proses *preprocessing*. Adapun pada tahap *information gain* dibutuhkan beberapa *library* untuk mendukung implementasi tahap ini yang dapat dilihat pada Kode Program 5.15.

Import Library	
1	import pandas as pd
2	import numpy as np

Kode Program 5.15 Import Library

Penjelasan baris pada Kode Program 5.15 adalah sebagai berikut.

- Baris 1 digunakan untuk *import library* bernama Pandas,
- Baris 2 digunakan untuk *import library* bernama *numpy*.

Adapun fungsi pendukung untuk mendapatkan nilai *document frequency* berdasarkan *term* yang dapat dilihat pada Kode Program 5.16.

Algoritme Get DF	
1	def get_df(self, docfreq_matrix, t, is_class=False): #get DF value based on term
2	if is_class == False:
3	filter_term = docfreq_matrix[docfreq_matrix[:, 1] == t] #1 for term
4	else:
5	filter_term = docfreq_matrix[docfreq_matrix[:, -2] == t] #-2 for term
6	df_term = filter_term[0][-1] #[0] for unpack 2d arr, and -1 for get the df value (last column)
7	return df_term

Kode Program 5.16 Implementasi Get DF

Penjelasan baris pada Kode Program 5.16 adalah sebagai berikut.

- Baris 1 digunakan untuk mendefinisikan fungsi *get_df* dengan *docfreq_matrix*, *t* dan *is_class* sebagai parameter.
- Baris 2 digunakan untuk mengecek apakah *is_class* bernilai *False*.
- Baris 3 digunakan untuk menyaring *array* dengan nilai *term* sama dengan parameter *t*.
- Baris 4 digunakan untuk mengecek apakah *is_class* bernilai *True*.
- Baris 6 digunakan untuk mendapatkan nilai *df* yang terletak pada baris ke 0 dan kolom ke -1 disimpan dalam variabel *df_term*.
- Baris 7 digunakan untuk mengembalikan nilai *df_term*.

Salah satu langkah pada *information gain* yaitu menghitung peluang term yang dapat dilihat pada Kode Program 5.17.

Algoritme Term Probability

```
1 def P_term(self, docfreq_matrix, t=None, is_contain_term=False):
2     #t stands for term
3     df_total = np.sum(docfreq_matrix[:, -1])
4     df_term = self.get_df(docfreq_matrix, t)
5
6     num_of_documents = self.get_docs(docfreq_matrix).shape[1] #get
7     column_docs
8     df_not_contain_term = num_of_documents - df_term
9
10    if is_contain_term:
11        prob_term = df_term / df_total
12    else:
13        prob_term = df_not_contain_term / df_total
14
15    return prob_term
```

Kode Program 5.17 Implementasi Term Probability

Penjelasan baris pada Kode Program 5.17 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *P_term* dengan *docfreq_matrix*, *t*, dan *is_contain_term* sebagai parameter.
2. Baris 2 digunakan untuk menjumlahkan seluruh nilai pada indeks -1 yaitu pada kolom 'df' yang disimpan dalam variabel *df_total*.
3. Baris 3 digunakan untuk mendapatkan nilai *document frequency* dari term *t* menggunakan fungsi *get_df* disimpan pada variabel *df_term*.
4. Baris 4 digunakan untuk mendapatkan banyaknya dokumen dengan fungsi *get_docs* disimpan pada variabel *num_of_documents*.
5. Baris 5 digunakan untuk mendapatkan nilai *df* yang tidak mengandung term *t* dengan mengurangi *num_of_documents* dengan *df_term* disimpan pada variabel *df_not_contain_term*.
6. Baris 6 hingga 9 digunakan untuk menghitung peluang term, jika *is_contain_term* bernilai *true* maka peluang dihitung dengan membagi nilai *df* pada term dengan *df_total* namun jika sebaliknya maka dihitung dengan membagi *df_not_contain_term* dengan *df_total*.
7. Baris 10 digunakan untuk mengembalikan nilai *prob_term*.

Adapun fungsi untuk mendukung implementasi *information gain* yaitu fungsi untuk mendapatkan jumlah dokumen yang dapat dilihat pada Kode Program 5.18.

Algoritme Get Docs

```
1 def get_docs(self, docfreq_matrix):
2     docs_df = docfreq_matrix[:, 2:-1]
3
4     return docs_df
```

Kode Program 5.18 Implementasi Get Docs

Penjelasan baris pada Kode Program 5.18 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *get_docs* dengan parameter *docfreq_matrix*.

2. Baris 2 digunakan untuk menyaring *dataframe docfreq_matrix* untuk mengambil kolom dokumen dimulai dari indeks dua hingga indeks kedua terakhir dan disimpan dalam variabel *docs_df*.
3. Baris 3 digunakan untuk mengembalikan nilai *docs_df*.

Berikutnya menghitung peluang term pada kelas yang dapat dilihat pada Kode Program 5.19.

Algoritme <i>Term Probability on Class</i>	
1	def P_term_class(self, dataset, docfreq_matrix, C, t, is_log=False, is_contain_term=False):
2	filter_doc_class = dataset[dataset[:, -1] == C]
3	docs_index = filter_doc_class[:, 0] #0 to get the Unnamed col
4	num_of_docs_class = filter_doc_class.shape[0]
5	
6	docs_df = self.get_docs(docfreq_matrix)
7	docfreq_matrix_filter_class = docs_df[:, list(docs_index)]
8	
9	docfreq_arr = np.sum(docfreq_matrix_filter_class, axis=1).reshape(len(docfreq_matrix_filter_class), -1) #reshape to 2d with (row x col)
10	term_arr = docfreq_matrix[:, 1].reshape(len(docfreq_matrix_filter_class), -1) #1 = term column
11	
12	docfreq_matrix_filter_class_append = np.append(docfreq_matrix_filter_class, term_arr, axis=1)
13	docfreq_matrix_filter_class_append = np.append(docfreq_matrix_filter_class_append, docfreq_arr, axis=1)
14	
15	df_term = self.get_df(docfreq_matrix_filter_class_append, t, is_class=True)
16	
17	df_not_contain_term = num_of_docs_class - df_term
18	
19	if is_log:
20	if is_contain_term:
21	if df_term/num_of_docs_class == 0:
22	prob_term_class = 0
23	else:
24	prob_term_class = np.log10(df_term/num_of_docs_class)
25	else:
26	if df_not_contain_term/num_of_docs_class == 0:
27	prob_term_class = 0
28	else:
29	prob_term_class = np.log10(df_not_contain_term/num_of_docs_class)

Kode Program 5.19 Implementasi *Term Probability on Class*

Penjelasan baris pada Kode Program 5.19 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *P_term_class* dengan parameter *dataset, docfreq_matrix, C, t, is_log, dan is_contain_term*.

2. Baris 2 digunakan untuk menyaring *dataframe* dengan indeks -1 atau *class* sama dengan nilai *C* lalu disimpan dalam variabel *filter_doc_class*.
3. Baris 3 digunakan untuk mendapatkan indeks dari *dataframe* yang terdapat pada kolom *Unnamed col*.
4. Baris 4 digunakan untuk mendapatkan panjang dokumen dari *filter_doc_class* yang disimpan dalam variabel *num_of_docs_class*.
5. Baris 5 digunakan untuk mendapatkan *dataframe* dokumen menggunakan fungsi *get_docs* yang disimpan dalam variabel *docs_df*.
6. Baris 6 digunakan untuk menyaring dokumen sesuai *index* dari *docs_index*.
7. Baris 7 digunakan untuk merubah dimensi dari *docfreq_matrix_filter_class* menjadi dua dimensi kemudian menghitung penjumlahan dengan fungsi sum.
8. Baris 8 digunakan untuk merubah dimensi dari term pada *docfreq_matrix* menjadi dua dimensi.
9. Baris 9 digunakan untuk menambah array *term_arr* ke dalam *docfreq_matrix_filter_class* yang disimpan dalam variabel *docfreq_matrix_filter_class_append*.
10. Baris 10 digunakan untuk menambah array *docfreq_arr* ke dalam *docfreq_matrix_filter_class_append* yang disimpan dalam variabel *docfreq_matrix_filter_class_append*.
11. Baris 11 digunakan untuk mendapatkan panjang dokumen yang disimpan dalam variabel *num_of_documents*.
12. Baris 12 digunakan untuk mengurangi nilai dari banyak dokumen dengan *document frequency* dari term t yang disimpan dalam variabel *df_not_contain_term*.
13. Baris 13 hingga 28 digunakan untuk menghitung peluang term pada *class*. Jika *is_log* dan *is_contain_term* bernilai *true* dan peluang term t muncul pada dokumen dari *class C* tidak bernilai 0 maka peluang dihitung dengan hasil logaritma dari peluang kemunculan term t pada *class C*. Jika *is_contain_term* bernilai *false* dan peluang tidak muncul term t pada *class C* tidak bernilai 0 maka peluang dihitung dengan hasil logaritma dari peluang tidak muncul term t pada *class C*. Jika *is_log* bernilai *false* dan *is_contain_term* bernilai *true* maka peluang dihitung dengan hasil pembagian dari *document frequency* term t dengan banyaknya dokumen pada *class C* namun jika *is_contain_term* bernilai *false* peluang dihitung dengan hasil pembagian dari *document frequency* yang tidak mengandung term t dengan banyaknya dokumen pada *class C*.
14. Baris 29 digunakan untuk mengembalikan nilai peluang term t pada *class C*.

Pada tahap *information gain* juga menghitung nilai peluang kelas yang dapat dilihat pada Kode Program 5.20.

Algoritme Class Probability

```

1 def P_class(self, dataset, C, is_log):
2     filter_doc_class = dataset[dataset['sentimen'] == C]
3     num_of_docs_class = len(filter_doc_class)
4     num_of_documents = len(dataset)
5
6     if is_log:
7         prob_class = np.log10(num_of_docs_class/num_of_documents)
8     else:
9         prob_class = num_of_docs_class/num_of_documents

```

Kode Program 5.20 Implementasi *Class Probability*

Penjelasan baris pada Kode Program 5.20 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *P_class* dengan dataset, C, dan *is_log* sebagai parameter.
2. Baris 2 digunakan untuk menyaring dokumen yang memiliki *class* C yang disimpan dalam variabel *filter_doc_class*.
3. Baris 3 digunakan untuk mendapatkan panjang dokumen dari *filter_doc_class*.
4. Baris 4 digunakan untuk mendapatkan panjang dari dataset yang disimpan dalam variabel *num_of_documents*.
5. Baris 5 digunakan untuk mencetak *num_of_docs_class*.
6. Baris 6 digunakan untuk mencetak *num_of_documents*.
7. Baris 7 hingga 10 digunakan untuk menghitung peluang *class* C. Jika *is_log* bernilai *true* maka peluang dihitung dari hasil logaritma peluang dokumen *class* C dari seluruh dokumen yang disimpan dalam variabel *prob_class*.
8. Baris 11 digunakan untuk mencetak peluang *class* C.
9. Baris 12 digunakan untuk mengembalikan nilai *prob_class*.

Proses utama pada tahap ini yaitu menghitung nilai *information gain* pada tiap term yang dapat dilihat pada Kode Program 5.21.

Algoritme <i>Information Gain</i>	
1	def create_IG_dataframe(self, list_IG_each_term, docfreq_matrix):
2	ig_matrix = docfreq_matrix.copy()
3	ig_matrix['IG'] = list_IG_each_term
4	return ig_matrix
5	def IG(self, dataset, docfreq_matrix):
6	sentimen_class = dataset['sentimen'].unique()
7	list_IG_each_term = []
8	for i, row in enumerate(docfreq_matrix.values):
9	#reset IG value each looping term
10	IG_1 = 0
11	IG_2 = 0
	IG_3 = 0

```

12         t = row[1]
13         prob_term =
14             self.P_term(docfreq_matrix=docfreq_matrix.values, t=t,
15             is_contain_term=True)
16                 prob_not_contain_term =
17                     self.P_term(docfreq_matrix=docfreq_matrix.values, t=t,
18                     is_contain_term=False)

19             for j in range(len(sentimen_class)): #j represents each
20                 class
21                     C = sentimen_class[j]

22                     IG_1 += - self.P_class(dataset=dataset,
23                         C=C, is_log=False) \
24                             * self.P_class(dataset=dataset,
25                               C=C, is_log=True)

26                     IG_2 += self.P_term_class(dataset=dataset.values,
27                         docfreq_matrix=docfreq_matrix.values, C=C, t=t,
28                             is_log=False,
29                             is_contain_term=True) \
30                                 * self.P_term_class(dataset=dataset.values,
31                                   docfreq_matrix=docfreq_matrix.values, C=C, t=t,
32                                       is_log=True,
33                                       is_contain_term=True)

34                     IG_3 += self.P_term_class(dataset=dataset.values,
35                         docfreq_matrix=docfreq_matrix.values, C=C, t=t,
36                             is_log=False,
37                             is_contain_term=False) \
38                                 * self.P_term_class(dataset=dataset.values,
39                                   docfreq_matrix=docfreq_matrix.values, C=C, t=t,
40                                       is_log=True,
41                                       is_contain_term=False)

42                     IG = IG_1 + (prob_term * IG_2) + (prob_not_contain_term *
43                     IG_3)

44                     list_IG_each_term.append(IG)

45                     ig_matrix =
46                     self.create_IG_dataframe(list_IG_each_term=list_IG_each_term,
47                         docfreq_matrix=docfreq_matrix)

48                     return ig_matrix

```

Kode Program 5.21 Implementasi *Information Gain*

Penjelasan baris pada Kode Program 5.21 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *create_IG_dataframe* dengan *list_IG_each_term* dan *docfreq_matrix* sebagai parameter.
2. Baris 2 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *ig_matrix*.
3. Baris 3 digunakan untuk membuat kolom 'IG' yang berisi nilai dari *list_IG_each_term*.

4. Baris 4 digunakan untuk mengembalikan nilai *ig_matrix*.
5. Baris 5 digunakan untuk mendefinisikan fungsi *IG* dengan dataset dan *docfreq_matrix* sebagai parameter.
6. Baris 6 digunakan untuk mendapatkan nilai unik dari kolom ‘sentimen’ yang disimpan dalam variabel *sentimen_class*.
7. Baris 7 digunakan untuk mendefinisikan *list* kosong yang disimpan dalam variabel *list_IG_each_term*.
8. Baris 8 digunakan untuk melakukan perulangan enumerasi dimulai pada *array* dari *numpy* pada variabel *docfreq_matrix*.
9. Baris 9 hingga 11 digunakan untuk memberi nilai awal dari variabel *IG_1*, *IG_2* dan *IG_3*.
10. Baris 12 digunakan untuk mendapatkan term ke-i yang disimpan dalam variabel *t*.
11. Baris 13 digunakan untuk mendapatkan peluang term menggunakan fungsi *P_term* yang disimpan dalam variabel *prob_term*.
12. Baris 14 digunakan untuk mendapatkan peluang tidak muncul term menggunakan fungsi *P_term* yang disimpan dalam variabel *prob_not_contain_term*.
13. Baris 15 hingga 21 digunakan untuk menghitung *information gain* dari tiap *class* ke-j yang disimpan dalam variabel *C*. Nilai *information gain* didapat dari penjumlahan *IG_1* dengan hasil perkalian dari *prob_term* dengan *IG_2* dan hasil perkalian *prob_not_contain_term* dengan *IG_3*. Kemudian ditambahkan ke dalam *list_IG_each_term*.
14. Baris 22 digunakan untuk membuat *dataframe* berisi nilai *information gain* menggunakan fungsi *create_IG_dataframe* yang disimpan dalam variabel *ig_matrix*.
15. Baris 23 digunakan untuk mengembalikan fungsi *ig_matrix*.

Setelah mendapatkan nilai *information gain* dilakukan proses penyaringan berdasarkan *threshold* tertentu yang dapat dilihat pada Kode Program 5.22.

Algoritme Filter Information Gain

```
1 def filter_ig(self, ig_matrix, threshold, is_percent=False):
2     filter_ig_matrix = ig_matrix.copy()
3     if is_percent is False:
4         statistics_ig_matrix = filter_ig_matrix.describe()
5         filter_threshold_ig_matrix =
6             statistics_ig_matrix.loc[threshold, 'IG'] # return the threshold
7             value
8             filter_ig_matrix = filter_ig_matrix[filter_ig_matrix['IG']
9             >= filter_threshold_ig_matrix]
10            sorted_ig_matrix = filter_ig_matrix.sort_values(by=['IG'],
11 ascending=False)
12            else:
13                num_of_documents = len(filter_ig_matrix['term'])
14                sorted_ig_matrix = filter_ig_matrix.sort_values(by=['IG'],
15 ascending=False)
16                filter_ig_matrix =
17                sorted_ig_matrix.head(int(num_of_documents*threshold))
18
19 return filter_ig_matrix
```

Kode Program 5.22 Implementasi Filter Information Gain

Penjelasan dari Kode Program 5.22 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *filter_ig* dengan *ig_matrix*, *threshold*, dan *is_percent* sebagai parameter.
2. Baris 2 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam *filter_ig_matrix*.
3. Baris 3 digunakan untuk mengecek jika *is_percent* bernilai *False*.
4. Baris 4 digunakan untuk mendapatkan rangkuman statistik dari *dataframe* *filter_ig* menggunakan fungsi *describe* yang disimpan dalam variabel *statistics_ig_matrix*.
5. Baris 5 digunakan untuk mengambil nilai berdasarkan baris *threshold* dan kolom 'IG' dari *statistics_ig_matrix* yang disimpan dalam variabel *filter_threshold_ig_matrix*.
6. Baris 6 digunakan untuk menyaring nilai *information gain* dengan nilai lebih dari sama dengan *threshold* yang disimpan dalam *filter_ig_matrix*.
7. Baris 7 digunakan untuk mengurutkan hasil *filter_ig_matrix* berdasarkan kolom 'IG' secara menurun yang disimpan dalam *sorted_ig_matrix*.
8. Baris 8 digunakan untuk mengecek jika *is_percent* bernilai *True*.
9. Baris 9 digunakan untuk mendapatkan jumlah dokumen.
10. Baris 10 digunakan untuk mengurutkan *filter_ig_matrix* secara *descending* yang disimpan pada variabel *num_of_documents*.

11. Baris 11 digunakan untuk mendapatkan term teratas sebanyak hasil perkalian dari *num_of_documents* dengan *threshold*.
12. Baris 12 digunakan untuk mengembalikan nilai *filter_ig_matrix*.

5.4 Implementasi Cosine Similarity

Implementasi *cosine similarity* merupakan implementasi untuk menghitung kedekatan antara data uji dengan data latih. Adapun pada tahap *cosine similarity* dibutuhkan beberapa *library* untuk mendukung implementasi tahap ini yang dapat dilihat pada Kode Program 5.23.

Import Library	
1	import numpy as np
2	import pandas as pd
3	import math
4	import improved_knn as iknn

Kode Program 5.23 Import Library

Penjelasan dari Kode Program 5.23 adalah sebagai berikut.

1. Baris 1 digunakan untuk *import library* bernama *numpy*.
2. Baris 2 digunakan untuk *import library* bernama Pandas.
3. Baris 3 digunakan untuk *import built-in library* bernama *math*.
4. Baris 4 digunakan untuk *import file* program Python bernama *improved_knn*.

Adapun salah satu tahap pada perhitungan *cosine similarity* yaitu menghitung nilai kuadrat dari vektor tiap dokumen dengan implementasi pada Kode Program 5.24.

Algoritme Calculate Square Docs	
1	def calculate_square_docs(self, matrix):
2	square_docs = matrix.copy()
3	docs = list(square_docs.filter(regex='dok').columns)
4	square_filter_docs = square_docs[docs]
5	return square_filter_docs**2

Kode Program 5.24 Implementasi Calculate Square Docs

Penjelasan dari Kode Program 5.24 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *calculate_square_docs* dengan *matrix* sebagai parameter.
2. Baris 2 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *square_docs*.
3. Baris 3 digunakan untuk mendapatkan kolom dari *dataframe* dokumen lalu diubah menjadi *list* yang disimpan dalam variabel *docs*.
4. Baris 4 digunakan untuk menarik *dataframe* dengan mengambil kolom dokumen saja disimpan pada variabel *square_filter_docs*.
5. Baris 5 digunakan untuk mengembalikan hasil kuadrat dari *square_filter_docs*.

```

Algoritme Calculate Square Root Docs
1 def calculate_square_root_docs(self, matrix):
2     square_root_docs = matrix.copy()
3     docs = list(square_root_docs.filter(regex='dok').columns)
4     square_root_docs_matrix = pd.DataFrame()
5
6     sum_sqrt_doc = np.sqrt(np.sum(square_root_docs))
7     square_root_docs_matrix = pd.DataFrame(sum_sqrt_doc)
8     square_root_docs_matrix = square_root_docs_matrix.transpose()

8     return square root docs matrix

```

Kode Program 5.25 Implementasi Calculate Square Root Docs

Penjelasan dari Kode Program 5.25 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *calculate_square_root_docs* dengan parameter *matrix*.
2. Baris 2 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *square_root_docs*.
3. Baris 3 digunakan untuk mendapatkan kolom dari *dataframe* dokumen lalu diubah menjadi *list* yang disimpan dalam variabel *docs*.
4. Baris 4 digunakan untuk membuat *dataframe* kosong yang disimpan dalam *list square_root_docs_matrix*.
5. Baris 5 digunakan untuk menghitung akar dari penjumlahan tiap dokumen yang disimpan pada variabel *sum_sqrt_doc*.
6. Baris 6 digunakan untuk merubah *Series* dari *sum_sqrt_doc* menjadi *Dataframe*.
7. Baris 7 digunakan untuk melakukan *transpose* pada *square_root_docs_matrix*.
8. Baris 8 digunakan untuk mengembalikan nilai *square_root_docs_matrix*.

```

Algoritme Calculate Dot Product
1 def calculate_dot_product(self, training_matrix, testing_matrix):
2     training_matrix_docs =
3     list(training_matrix.filter(regex='dok').columns)
4     dot_training_testing_matrix = training_matrix.copy()
5
6     dot_training_testing_matrix =
7     dot_training_testing_matrix[training_matrix_docs].multiply(testing_m
8     atrix, axis='index')
9
10    return dot training testing matrix

```

Kode Program 5.26 Implementasi Calculate Dot Product

Penjelasan dari Kode Program 5.26 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *calculate_dot_product* dengan parameter *training_matrix* dan *testing_matrix*.
2. Baris 2 digunakan untuk mendapatkan kolom dari *dataframe* dokumen lalu diubah menjadi *list* yang disimpan dalam variabel *training_matrix_docs*.

3. Baris 3 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *dot_training_testing_matrix*.
4. Baris 4 digunakan untuk menghitung *dot product* dari dokumen latih dan dokumen uji menggunakan *multiply* yang disimpan dalam variabel *dot_training_testing_matrix*.
5. Baris 5 digunakan untuk mengembalikan nilai *dot_training_testing_matrix*.

Algoritme Sum Dot Product

```

1 def sum_dot_product(self, dot_training_testing_matrix):
2     sum_of_dot_product_matrix = dot_training_testing_matrix.copy()
3     sum_of_dot_product_matrix =
4         sum_of_dot_product_matrix.filter(regex='dok') #only get the Docs
5             column because it's used for the calculation.
6
7         sum_dot_product = np.sum(sum_of_dot_product_matrix)
8         sum_of_dot_product_new_matrix = pd.DataFrame(sum_dot_product)
9         sum_of_dot_product_new_matrix =
10            sum_of_dot_product_new_matrix.transpose()
11
12    return sum_of_dot_product_new_matrix

```

Kode Program 5.27 Implementasi Sum Dot Product

Penjelasan dari Kode Program 5.27 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *sum_dot_product* dengan parameter *dot_training_testing_matrix*.
2. Baris 2 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *sum_dot_product_matrix*.
3. Baris 3 digunakan untuk menyaring *dataframe* berdasarkan nama kolom yang mengandung ‘dok’ yang disimpan dalam variabel *sum_of_dot_product_matrix*.
4. Baris 4 digunakan untuk menghitung penjumlahan vektor dokumen yang telah dilakukan perkalian *dot* yang disimpan dalam variabel *sum_dot_product*.
5. Baris 5 digunakan untuk merubah *Series* dari *sum_dot_product* menjadi *Dataframe* yang disimpan pada variabel *sum_of_dot_product_new_matrix*.
6. Baris 6 digunakan untuk melakukan *transpose* pada *sum_of_dot_product_new_matrix*.
7. Baris 7 digunakan untuk mengembalikan nilai *sum_of_dot_product_new_matrix*.

Algoritme Cosine Similarity

```

1 def similarity_and_predict(self, raw_training_data,
2     raw_testing_data, training_matrix, testing_matrix, initial_k):
3     iknn_step = iknn.ImprovedKNN()
4     training_matrix = training_matrix.copy()
5     testing_matrix = testing_matrix.copy()
6     testing_docs =
7         list(testing_matrix.filter(regex='dok').columns)
8     training_docs =
9         list(training_matrix.filter(regex='dok').columns)
10    list_predict_class = []

```

```

8      for idx_testing, each_doc_testing in enumerate(testing_docs):
9          similarity_list = []
10         each_raw_testing_doc = raw_testing_data.iat[idx_testing,
11             0]
12         dot_product_matrix =
13             self.calculate_dot_product(training_matrix,
14                 testing_matrix[each_doc_testing])
15         sum_dot_product_matrix =
16             self.sum_dot_product(dot_product_matrix)
17         training_square_matrix =
18             self.calculate_square_docs(training_matrix)
19         training_square_root_matrix =
20             self.calculate_square_root_docs(training_square_matrix)
21         testing_square_matrix =
22             self.calculate_square_docs(testing_matrix)
23         testing_square_root_matrix =
24             self.calculate_square_root_docs(testing_square_matrix)

25         for idx_training, each_doc_training in
26             enumerate(training_docs):
27             similarity_training_testing =
28                 sum_dot_product_matrix[each_doc_training].iat[0]/(training_square_
29                     root_matrix[each_doc_training].iat[0] *
30                     testing_square_root_matrix[each_doc_testing].iat[0])
31             if math.isnan(similarity_training_testing):
32                 similarity_list.append('dok ' +
33                     str(idx_training+1), 0, raw_training_data.iat[idx_training, -1]))
34             else:
35                 similarity_list.append(('dok ' +
36                     str(idx_training+1), similarity_training_testing,
37                     raw_training_data.iat[idx_training, -1]))

38             cosine_similarity_df = pd.DataFrame(similarity_list,
39             columns=['q x dok', 'nilai cossim', 'sentimen'])
40             prob_each_class =
41                 iknn_step.probability_each_class(raw_training_data,
42                     cosine_similarity_df, initial_k)
43             predict_class = iknn_step.predict(prob_each_class)
44             list_predict_class.append((each_raw_testing_doc,
45                 predict_class))

46             predict_class_df = pd.DataFrame(list_predict_class,
47             columns=['dokumen', 'sentimen'])

48     return predict_class_df

```

Kode Program 5.28 Implementasi Cosine Similarity

Penjelasan dari Kode Program 5.28 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *similarity_and_predict* dengan *raw_training_data* , *raw_testing_data* , *training_matrix*, *testing_matrix* dan *initial_k* sebagai parameter.
2. Baris 2 digunakan untuk instansiasi *class ImprovedKNN* yang disimpan dalam variabel *iknn_step*.
3. Baris 3 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *training_matrix*.
4. Baris 4 digunakan untuk melakukan duplikasi *dataframe* yang disimpan dalam variabel *testing_matrix*.

5. Baris 5 digunakan untuk mendapatkan kolom dari *dataframe* dokumen lalu diubah menjadi *list* yang disimpan dalam variabel *testing_docs*.
6. Baris 6 digunakan untuk mendapatkan kolom dari *dataframe* dokumen lalu diubah menjadi *list* yang disimpan dalam variabel *training_docs*.
7. Baris 7 digunakan untuk membuat *list* kosong yang disimpan dalam variabel *list_predict_class*.
8. Baris 8 hingga 16 digunakan untuk menghitung *dot product*, kuadrat dari vektor dokumen latih dan uji, serta *akar kuadrat* dari penjumlahan vektor dokumen maupun latih.
9. Baris 17 hingga 26 digunakan untuk menghitung kemiripan antara dokumen uji dengan latih, jika bernilai NaN maka kemiripan tersebut diisi dengan 0. Kemudian hasil dari *cosine similarity* dimasukkan ke dalam *dataframe* untuk dihitung peluang antar *class* menggunakan fungsi *probability_each_class* berdasarkan nilai k awal pada parameter *initial_k*. Kelas yang didapatkan dari *improved knn* disimpan dalam variabel *predict_class* menggunakan fungsi *predict* kemudian ditambahkan ke dalam *tuple* berisi dokumen uji serta kelas hasil prediksi.
10. Baris 27 digunakan untuk membuat *dataframe* yang berisi kolom ‘dokumen’ dan ‘sentimen’ yang disimpan dalam variabel *predict_class_df*.
11. Baris 28 digunakan untuk mengembalikan nilai *predict_class_df*.

5.5 Implementasi Klasifikasi dengan *Improved KNN*

Implementasi klasifikasi dengan *improved KNN* merupakan tahap untuk menentukan kelas sentimen berdasarkan data uji. Adapun pada tahap *improved KNN* dibutuhkan beberapa *library* untuk mendukung implementasi tahap ini yang dapat dilihat pada Kode Program 5.29

Import Library	
1	import math
2	import numpy as np
3	import pandas as pd

Kode Program 5.29 Import Library

Penjelasan dari Kode Program 5.29 adalah sebagai berikut

1. Baris 1 digunakan untuk *import built-in library* bernama *math*.
2. Baris 2 digunakan untuk *import library* bernama *numpy*.
3. Baris 3 digunakan untuk *import library* bernama Pandas.

Salah satu tahapan pada *improved KNN* yaitu mendapatkan jumlah dokumen pada tiap kelas dengan implementasi pada Kode Program 5.30

Algoritme Get Number Doc

```

1 def get_number_doc(self, dataset):
2     list_num_docs_class = []
3     class_unique = dataset['sentimen'].unique()
4
5         for each_doc_class in class_unique:
6             docs_class = dataset[dataset['sentimen'] == each_doc_class]
7             number_docs_class = len(docs_class)
8             list_num_docs_class.append((each_doc_class,
number_docs_class))
9
10    return list_num_docs_class

```

Kode Program 5.30 Implementasi Get Number Doc

Penjelasan dari Kode Program 5.30 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *get_number_doc* dengan dataset sebagai parameter.
2. Baris 2 digunakan untuk membuat *list* kosong yang disimpan dalam variabel *list_num_docs_class*.
3. Baris 3 digunakan untuk mendapatkan nilai unik dari dataset pada kolom 'sentimen'.
4. Baris 4-7 digunakan untuk mendapatkan jumlah dokumen pada tiap *class*, kemudian ditambah ke dalam *list_num_docs_class* berupa *tuple* disertai nama *class*.
5. Baris 8 digunakan untuk mengembalikan nilai *list_num_docs_class*.

Berikutnya, nilai *k* untuk masing-masing kelas dicari dengan implementasi sesuai pada Kode Program 5.31.

Algoritme Get New K

```

1 def get_new_k(self, initial_k, list_num_docs_class: []): #return
new K value for each class: positive and negative
2
3     num_of_docs_each_class = [each_doc_class[1] for each_doc_class
in list_num_docs_class]
4     max_number_of_docs = np.max(num_of_docs_each_class)
5     new_k_list = []
6
7         for each_class_docs in list_num_docs_class:
8             new_k = math.floor((each_class_docs[1] *
initial_k)/max_number_of_docs)
9             new_k_list.append((each_class_docs[0], new_k))
10
11     new_k_df = pd.DataFrame(new_k_list, columns=['sentimen',
'jumlah dokumen'])
12
13     return new_k_df

```

Kode Program 5.31 Implementasi Get New K

Penjelasan dari Kode Program 5.31 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *get_new_k* dengan parameter *initial_k*, dan *list_num_docs_class*.

2. Baris 2 digunakan untuk mendefinisikan *list* yang berisi jumlah dokumen pada tiap *class*.
3. Baris 3 digunakan untuk mendapatkan jumlah dokumen terbanyak dari seluruh *class* yang disimpan dalam variabel *max_number_of_docs*.
4. Baris 4 digunakan untuk mendefinisikan *list* kosong yang disimpan dalam variabel *new_k_list*.
5. Baris 5 hingga 7 digunakan untuk menghitung nilai k baru didapat dari pembagian banyak dokumen pada *class* dikali dengan nilai k awal dibagi dengan dokumen terbanyak yang disimpan dalam variabel bernama *new_k*. Kemudian hasil tersebut ditambahkan ke *new_k_list* berupa *tuple*.
6. Baris 8 digunakan untuk membuat *dataframe* berisi *new_k_list* dengan kolom ‘sentimen’ dan ‘jumlah dokumen’.
7. Baris 9 digunakan untuk mengembalikan nilai *new_k_df*.

Selanjutnya, mencari nilai base KNN yang tidak memperhatikan nilai k pada tiap kelas dengan implementasi pada Kode Program 5.32.

Algoritme Sum Nearest Base KNN

```

1 def sum_nearest_base_knn(self, sorted_cosine_similarity,
2                             initial_k):
3     nearest_base_knn =
4         sorted_cosine_similarity.iloc[:initial_k] #retrieve top docs based
5         on k value
6         sum_cossim_nearest_base_knn =
7             np.sum(nearest_base_knn['nilai cossim'])
8
9         return sum_cossim_nearest_base_knn

```

Kode Program 5.32 Implementasi Sum Nearest Base KNN

Penjelasan dari Kode Program 5.32 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *sum_nearest_base_knn* dengan *sorted_cosine_similarity* dan *initial_k* sebagai parameter.
2. Baris 2 digunakan untuk mendapatkan nilai *cosine similarity* yang telah diurutkan sebanyak *initial_k* yang disimpan dalam variabel *nearest_base_knn*.
3. Baris 3 digunakan untuk mendapatkan nilai jumlah dari *nearest_base_knn* yang disimpan dalam *sum_cossim_nearest_base_knn*.
4. Baris 4 digunakan untuk mengembalikan nilai *sum_cossim_nearest_base_knn*.

Tahap untuk menentukan kelas dari data uji yaitu pada perhitungan peluang data tersebut masuk ke masing-masing kelas yang ditulis pada Kode Program 5.33.

Algoritme Probability Each Class

```

1 def objective_function(self, current_data_class, sentimen_class):
2     if current_data_class == sentimen_class:
3         return 1
4     else:
5         return 0

```

```

6  def probability_each_class(self, dataset, cosine_similarity,
7      initial_k):
8      list_number_doc_each_class = self.get_number_doc(dataset)
9      sorted_cosine_similarity =
cosine_similarity.sort_values(by='nilai cossim',
ascending=False).reset_index(drop=True)
9      list_prob_class = []
10
10     for each_class in
sorted_cosine_similarity['sentimen'].unique():
11         sum_nearest_each_class = 0
12         new_k_df = self.get_new_k(initial_k,
list_number_doc_each_class)
13         new_k = new_k_df[new_k_df['sentimen'] ==
each_class]['jumlah dokumen'].iloc[0] #using 0 to returns the value
14
14         # looping nearest neigbhor based on new_k and the sentimen
class
15         for idx, each_row in
sorted_cosine_similarity.iloc[:new_k].iterrows():
15             sum_nearest_each_class += each_row['nilai cossim'] \
* self.objective_function(each_row['sentimen'], each_class)
16
16         prob_each_class =
sum_nearest_each_class/self.sum_nearest_base_knn(sorted_cosine_simi-
larity, initial_k)
17         list_prob_class.append((P(query), prob_each_class,
each_class))
18
18     prob_class_df = pd.DataFrame(list_prob_class,
columns=[P(query, dokumen), 'nilai peluang', 'sentimen'])
19
19     return prob_class df

```

Kode Program 5.33 Implementasi *Probability Each Class*

Penjelasan dari Kode Program 5.33 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *objective_function* dengan *current_data_class* dan *sentimen_class* sebagai parameter.
2. Baris 2 hingga 5 digunakan untuk mengecek jika nilai *current_data_class* sama dengan *sentimen_class* maka nilai 1 akan dikembalikan. Jika sebaliknya maka nilai 0 akan dikembalikan dari fungsi.
3. Baris 6 digunakan untuk mendefinisikan fungsi *probability_each_class* dengan dataset, *cosine_similarity* dan *initial_k* sebagai parameter.
4. Baris 7 digunakan untuk mendapatkan banyaknya dokumen dari dataset yang disimpan dalam *list_number_doc_each_class*.
5. Baris 8 digunakan untuk mengurutkan nilai *cosine similarity* secara menurun yang disimpan dalam variabel *sorted_cosine_similarity*.
6. Baris 9 digunakan untuk mendefinisikan *list* kosong yang disimpan dalam variabel *list_prob_class*.
7. Baris 10 hingga 13 digunakan untuk mendapatkan nilai k baru yang disimpan dalam variabel *new_k_df* kemudian disaring sesuai dengan nilai *each_class* dan mengambil nilai kolom ‘jumlah dokumen’ dengan memilih indeks 0.

8. Baris 14 hingga 15 digunakan untuk menghitung nilai *cosine similarity* dikalikan dengan fungsi objektif sesuai *each_class* berdasarkan nilai *k* baru yang disimpan dalam *sum_nearest_each_class*.
9. Baris 16 digunakan untuk menghitung peluang tiap *class* dengan membagi *sum_nearest_each_class* dengan *sum_nearest_base_knn*.
10. Baris 17 digunakan untuk menambah *tuple* nilai dari *prob_each_class* dengan *each_class* ke dalam *list_prob_class*.
11. Baris 18 digunakan untuk membuat *dataframe* dengan kolom ‘P(query, dokumen)’, ‘nilai peluang’ dan ‘sentimen’ dan nilai *list_prob_class* yang disimpan dalam variabel *prob_class_df*.
12. Baris 19 digunakan untuk mengembalikan *prob_class_df*.

Selanjutnya, fungsi *predict* dibuat untuk mengembalikan kelas hasil perhitungan peluang seperti pada Kode Program 5.34.

Algoritme Sum Nearest Base KNN	
1	def predict(self, prob_each_class):
2	prob_each_class_copy = prob_each_class.copy()
3	sorted_prob_each_class =
	prob_each_class_copy.sort_values(by='nilai peluang', ascending=False)
4	sentiment_class = sorted_prob_each_class.iloc[0]['sentimen']
5	return sentiment_class

Kode Program 5.34 Implementasi *Predict* pada Improved KNN

Penjelasan dari Kode Program 5.34 adalah sebagai berikut.

1. Baris 1 digunakan untuk mendefinisikan fungsi *predict* dengan *prob_each_class* sebagai parameter.
2. Baris 2 digunakan untuk melakukan duplikasi pada *dataframe* yang disimpan dalam variabel *prob_each_class_copy*.
3. Baris 3 digunakan untuk mengurutkan nilai peluang pada tiap *class* berdasarkan kolom ‘nilai peluang’ secara menurun yang disimpan dalam variabel *sorted_prob_each_class*.
4. Baris 4 digunakan untuk mendapatkan kelas sentimen berdasarkan nilai peluang tiap *class* yang telah diurutkan yang disimpan dalam variabel *sentiment_class*.
5. Baris 5 digunakan untuk mengembalikan nilai *sentiment_class*.

BAB 6 PENGUJIAN DAN ANALISIS

Pada bab ini dijelaskan hasil pengujian berdasarkan rancangan yang telah dibuat sebelumnya serta dijelaskan analisis dari pengujian yang dilakukan.

6.1 Pengujian Nilai k (Ketetanggaan) dengan 5-Fold Cross Validation

Pada pengujian ini dilakukan pembagian dataset menjadi 5 lipatan *fold* dimana 1 *fold* digunakan sebagai data uji sedangkan *fold* lainnya digunakan sebagai data latih. Setiap nilai k ketetanggaan diuji pada tiap *fold* untuk mendapatkan *precision*, *recall* dan *f-measure*. Kemudian untuk mendapatkan nilai k terbaik dilakukan dengan menghitung rata-rata *f-measure* dari setiap nilai k .

6.1.1 Fold ke-1

Pada fold ke-1, data uji yang digunakan yaitu data pada fold ke-1 yakni sejumlah 100 data pertama, sedangkan fold ke-2 hingga ke-5 digunakan sebagai data latih yakni sebanyak 400 data. Hasil pengujian pada fold ke-1 sebagaimana terdapat pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Nilai k pada Fold ke-1

Nilai k	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
3	78.65%	86.42%	82.35%
5	75.53%	92.21%	83.04%
7	76.04%	94.81%	84.39%
9	76.04%	94.81%	84.39%
11	73.96%	94.67%	83.04%
13	76.29%	96.10%	85.06%
15	76.29%	96.10%	85.06%
17	73.20%	95.95%	83.04%
19	78.95%	93.75%	85.71%
21	77.32%	96.15%	85.71%
100	68.37%	97.10%	80.24%
400	70.10%	95.77%	80.95%

Berdasarkan Tabel 6.1 terlihat bahwa nilai $k = 19$ dan $k = 21$ memiliki *f-measure* terbaik sebesar 85.71% dengan *recall* 93.75% untuk $k = 19$, *recall* 96.15% untuk $k = 21$ dan *precision* sebesar 78.95% untuk $k = 19$, *precision* 77.32% untuk $k = 21$.

6.1.2 Fold ke-2

Pada fold ke-2, data uji yang digunakan yaitu data pada fold ke-2 yakni data ke-101 hingga 200, sedangkan fold lainnya digunakan sebagai data latih. Hasil pengujian pada fold ke-2 sebagaimana terdapat pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Nilai k pada Fold ke-2

Nilai k	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
3	80.85%	92.68%	86.36%
5	76.04%	94.81%	84.39%
7	76.29%	96.10%	85.06%
9	75.76%	98.68%	85.71%
11	77.08%	94.87%	85.06%
13	74.23%	96.00%	83.72%
15	73.47%	97.30%	83.72%
17	72.45%	97.26%	83.04%
19	73.47%	97.30%	83.72%
21	72.45%	97.26%	83.04%
100	69.70%	98.57%	81.66%
400	70.71%	98.59%	82.35%

Berdasarkan Tabel 6.2, terlihat bahwa *f-measure* tertinggi dicapai ketika $k = 9$ yaitu sebesar 85.71% dengan *precision* sebesar 75.76% dan *recall* 98.66%.

6.1.3 Fold ke-3

Pada fold ke-3, data uji yang digunakan yaitu data pada fold ke-2 yakni data ke-201 hingga 300, sedangkan fold lainnya digunakan sebagai data latih. Hasil pengujian pada fold ke-3 sebagaimana terdapat pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Nilai k pada Fold ke-3

Nilai k	<i>Precision</i>	<i>Recall</i>	<i>F-Measure</i>
3	72.92%	94.59%	82.35%
5	70.10%	95.77%	80.95%
7	71.13%	95.83%	81.66%
9	70.10%	95.77%	80.95%
11	71.13%	95.83%	81.66%
13	70.41%	97.18%	81.66%
15	70.41%	97.18%	81.66%

Tabel 6.3 Hasil Pengujian Nilai k pada Fold ke-3 (Lanjutan)

17	67.35%	97.06%	79.52%
19	67.35%	97.06%	79.52%
21	68.37%	97.10%	80.24%
100	64.65%	98.46%	78.05%
400	65.66%	98.48%	78.79%

Berdasarkan Tabel 6.3, *f-measure* tertinggi diraih ketika $k = 3$ yaitu sebesar 82.35%, *precision* sebesar 72.92% dan *recall* sebesar 94.59%.

6.1.4 Fold ke-4

Pada fold ke-4, data uji yang digunakan yaitu data pada fold ke-4 yakni data ke-301 hingga 400, sedangkan fold lainnya digunakan sebagai data latih. Hasil pengujian pada fold ke-4 sebagaimana terdapat pada Tabel 6.4.

Tabel 6.4 Hasil Pengujian Nilai k pada Fold ke-4

Nilai k	Precision	Recall	F-Measure
3	81.82%	85.71%	83.72%
5	82.22%	88.10%	85.06%
7	83.70%	90.59%	87.01%
9	83.16%	94.05%	88.27%
11	84.38%	95.29%	89.50%
13	84.21%	94.12%	88.89%
15	84.04%	92.94%	88.27%
17	83.87%	91.76%	87.64%
19	82.11%	93.98%	87.64%
21	80.85%	92.68%	86.36%
100	73.68%	93.33%	82.35%
400	74.74%	93.42%	83.04%

Berdasarkan Tabel 6.4, *f-measure* tertinggi diraih ketika $k = 11$ yaitu sebesar 80.24%, dengan *precision* sebesar 84.38% dan *recall* sebesar 95.29%.

6.1.5 Fold ke-5

Pada fold ke-5, data uji yang digunakan yaitu data pada fold ke-5 yakni data ke-401 hingga 500, sedangkan fold lainnya digunakan sebagai data latih. Hasil pengujian pada fold ke-5 sebagaimana terdapat pada Tabel 6.5.

Tabel 6.5 Hasil Pengujian pada Fold ke-5

Nilai k	Precision	Recall	F-Measure
3	80.22%	89.02%	84.39%
5	79.38%	96.25%	87.01%
7	79.38%	96.25%	87.01%
9	79.59%	97.50%	87.64%
11	79.80%	98.75%	88.27%
13	78.79%	98.73%	87.64%
15	76.53%	97.40%	85.71%
17	77.00%	100.00%	87.01%
19	76.00%	100.00%	86.36%
21	77.00%	100.00%	87.01%
100	74.00%	100.00%	85.06%
400	75.00%	100.00%	85.71%

Berdasarkan Tabel 6.5, nilai *f-measure* tertinggi diraih oleh $k = 11$ sebesar 88.27%, *precision* sebesar 79.80% dan *recall* sebesar 98.75%.

6.1.6 Pemilihan Nilai k Terbaik dari Rata-rata Pengujian pada 5-fold

Setelah menghitung *precision*, *recall* dan *f-measure* dari setiap nilai ketetanggaan dengan 5-fold. Nilai tersebut dirata-ratakan untuk setiap k dengan tujuan untuk mendapatkan nilai k terbaik sebagaimana yang tercantum pada Tabel 6.6.

Tabel 6.6 Rata-rata Hasil Pengujian pada Setiap k

Nilai k	Rata-rata Precision	Rata-rata Recall	Rata-rata F-measure
3	78.89%	89.69%	83.84%
5	76.66%	93.43%	84.09%
7	77.31%	94.72%	85.02%
9	76.93%	96.16%	85.39%
11	77.27%	95.88%	85.51%
13	76.78%	96.43%	85.39%
15	76.15%	96.19%	84.88%
17	74.77%	96.41%	84.05%
19	75.57%	96.42%	84.59%

Tabel 6.6 Rata-rata Hasil Pengujian pada Setiap k (Lanjutan)

21	75.20%	96.64%	84.47%
100	70.08%	97.49%	81.47%
500	71.24%	97.25%	82.17%

Berdasarkan Tabel 6.6 didapatkan nilai k terbaik pada pengujian dengan 5-fold yaitu ketika $k = 11$ yang dibuktikan dengan rata-rata $f\text{-measure}$ tertinggi yaitu sebesar 85.51%. Nilai $k = 11$ akan digunakan pada pengujian berikutnya yaitu pengujian terhadap seleksi fitur.

6.2 Pengujian Seleksi Fitur *Information Gain* dengan 5-Fold Cross Validation

Pada pengujian *information gain* ini dilakukan perhitungan *precision*, *recall* dan *f-measure* menggunakan k terbaik yang didapat pada pengujian sebelumnya dengan menguji pada masing-masing *fold*. Kemudian untuk mendapatkan *threshold* terbaik dilakukan dengan menghitung rata-rata *f-measure* dan dibandingkan dengan perubahan rata-rata *running time*.

6.2.1 Fold ke-1

Pada fold ke-1, data uji yang digunakan yaitu data pada fold ke-1 yakni sejumlah 100 data pertama, sedangkan fold ke-2 hingga ke-5 digunakan sebagai data latih yakni sebanyak 400 data. Jumlah fitur yang digunakan sebanyak 1815. Hasil pengujian pada fold ke-1 sebagaimana terdapat pada Tabel 6.7.

Tabel 6.7 Hasil Pengujian *Information Gain* pada Fold ke-1

Nilai k	Fitur	Precision	Recall	F-Measure	Waktu Komputasi (Menit)
10%	181	85.00%	56.04%	67.55%	0.3675
20%	363	65.48%	77.46%	70.97%	0.3862
30%	544	61.96%	87.69%	72.61%	0.3875
40%	726	60.64%	90.48%	72.61%	0.4638
50%	907	61.43%	58.90%	60.14%	0.4070
60%	1089	75.00%	65.85%	70.13%	0.4177
70%	1270	68.29%	75.68%	71.79%	0.4258
80%	1452	70.11%	82.43%	75.78%	0.4411
90%	1633	68.18%	83.33%	75.00%	0.4709
median	1053	78.13%	58.14%	66.67%	0.4185
mean	1433	68.97%	82.19%	75.00%	0.4375

Tabel 6.7 Hasil Pengujian *Information Gain* pada Fold ke-1

100%	1815	73.96%	94.67%	83.04%	0.4659
------	------	--------	--------	--------	--------

Berdasarkan Tabel 6.7, nilai *f-measure* tertinggi diraih ketika seluruh fitur digunakan sebesar 83.04%, *precision* sebesar 73.96% dan *recall* sebesar 94.67% serta *running time* 0.4659 menit.

6.2.2 Fold ke-2

Pada fold ke-2, data uji yang digunakan yaitu data pada fold ke-2 yakni data ke-101 hingga 200, sedangkan fold lainnya digunakan sebagai data latih. Jumlah fitur yang digunakan sebanyak 1764. Hasil pengujian pada fold ke-2 sebagaimana terdapat pada Tabel 6.8.

Tabel 6.8 Hasil Pengujian *Information Gain* pada Fold ke-2

Nilai k	Fitur	Precision	Recall	F-Measure	Waktu Komputasi (Menit)
10%	176	72.86%	62.96%	67.55%	0.3599
20%	352	65.48%	77.46%	70.97%	0.3721
30%	529	60.67%	83.08%	70.13%	0.3826
40%	705	61.96%	87.69%	72.61%	0.3932
50%	882	76.71%	67.47%	71.79%	0.4005
60%	1058	81.69%	66.67%	73.42%	0.4089
70%	1234	72.15%	73.08%	72.61%	0.4258
80%	1411	68.82%	90.14%	78.05%	0.4370
90%	1587	70.83%	94.44%	80.95%	0.4521
median	1028	78.67%	70.24%	74.21%	0.4096
mean	1391	68.82%	90.14%	78.05%	0.4375
100%	1764	77.08%	94.87%	85.06%	0.4560

Berdasarkan Tabel 6.8 nilai *f-measure* tertinggi diraih ketika seluruh fitur digunakan sebesar 85.06%, *precision* sebesar 77.08% dan *recall* sebesar 94.87% *running time* 0.4560 menit.

6.2.3 Fold ke-3

Pada fold ke-3, data uji yang digunakan yaitu data pada fold ke-2 yakni data ke-201 hingga 300, sedangkan fold lainnya digunakan sebagai data latih. Jumlah fitur yang digunakan sebanyak 1844. Hasil pengujian pada fold ke-3 sebagaimana terdapat pada Tabel 6.9.

Tabel 6.9 Hasil Pengujian *Information Gain* pada Fold ke-3

Nilai k	Fitur	Precision	Recall	F-Measure	Waktu Komputasi (Menit)
10%	184	73.33%	52.38%	61.11%	0.3510
20%	368	60.24%	74.63%	66.67%	0.3633
30%	553	57.45%	90.00%	70.13%	0.3773
40%	737	56.84%	91.53%	70.13%	0.3914
50%	922	62.96%	72.86%	67.55%	0.3964
60%	1106	68.57%	61.54%	64.86%	0.4070
70%	1290	69.23%	71.05%	70.13%	0.4177
80%	1475	68.75%	73.33%	70.97%	0.4344
90%	1659	65.98%	95.52%	78.05%	0.4565
median	1100	70.77%	56.79%	63.01%	0.4151
mean	1457	70.59%	80.00%	75.00%	0.4346
100%	1844	71.13%	95.83%	81.66%	0.4508

Berdasarkan Tabel 6.9 nilai *f-measure* tertinggi diraih ketika seluruh fitur digunakan sebesar 81.66%, *precision* sebesar 71.13% dan *recall* sebesar 95.83% *running time* 0.4508 menit.

6.2.4 Fold ke-4

Pada fold ke-4, data uji yang digunakan yaitu data pada fold ke-4 yakni data ke-301 hingga 400, sedangkan fold lainnya digunakan sebagai data latih. Jumlah fitur yang digunakan sebanyak 1751. Hasil pengujian pada fold ke-4 sebagaimana terdapat pada Tabel 6.10.

Tabel 6.10 Hasil Pengujian *Information Gain* pada Fold ke-4

Nilai k	Fitur	Precision	Recall	F-Measure	Waktu Komputasi
10%	175	77.78%	76.83%	77.30%	0.3646
20%	350	75.27%	90.91%	82.35%	0.3753
30%	525	74.23%	96.00%	83.72%	0.3880
40%	700	72.45%	97.26%	83.04%	0.3956
50%	875	82.35%	82.35%	82.35%	0.4144
60%	1050	81.43%	65.52%	72.61%	0.4146
70%	1225	86.44%	55.43%	67.55%	0.4266

Tabel 6.10 Hasil Pengujian *Information Gain* pada Fold ke-4

80%	1400	72.73%	70.89%	71.79%	0.4391
90%	1575	78.08%	67.86%	72.61%	0.4500
median	989	85.45%	51.09%	63.95%	0.4135
mean	1380	72.60%	66.25%	69.28%	0.4354
100%	1751	84.38%	95.29%	89.50%	0.4901

Berdasarkan Tabel 6.10 nilai *f-measure* tertinggi diraih ketika seluruh fitur digunakan sebesar 89.50%, *precision* sebesar 84.38% dan *recall* sebesar 95.29% *running time* 0.4901 menit.

6.2.5 Fold ke-5

Pada fold ke-5, data uji yang digunakan yaitu data pada fold ke-5 yakni data ke-401 hingga 500, sedangkan fold lainnya digunakan sebagai data latih. Jumlah fitur yang digunakan sebanyak 1739. Hasil pengujian pada fold ke-5 sebagaimana terdapat pada Tabel 6.11.

Tabel 6.11 Hasil Pengujian *Information Gain* pada Fold ke-5

Nilai k	Fitur	Precision	Recall	F-Measure	Waktu Komputasi (Menit)
10%	173	69.62%	72.37%	70.97%	0.3698
20%	347	69.47%	92.96%	79.52%	0.3763
30%	521	70.71%	98.59%	82.35%	0.3888
40%	695	69.70%	98.57%	81.66%	0.3956
50%	869	70.65%	89.04%	78.79%	0.4081
60%	1043	72.09%	81.58%	76.54%	0.4052
70%	1217	69.41%	79.73%	74.21%	0.4117
80%	1391	72.41%	82.89%	77.30%	0.4221
90%	1565	75.82%	88.46%	81.66%	0.4510
median	1032	68.29%	75.68%	71.79%	0.4122
mean	1378	72.94%	80.52%	76.54%	0.4378
100%	1739	79.80%	98.75%	88.27%	0.4609

Berdasarkan Tabel 6.11 nilai *f-measure* tertinggi diraih ketika seluruh fitur digunakan sebesar 88.27%, *precision* sebesar 79.80% dan *recall* sebesar 98.75% *running time* 0.4609 menit.

6.2.6 Pemilihan *Threshold* Terbaik dari Rata-rata Pengujian pada 5-fold

Setelah menghitung *precision*, *recall* dan *f-measure* dari setiap *threshold* dari *information gain* dengan 5-fold. Nilai tersebut dirata-ratakan untuk setiap *threshold* dengan tujuan untuk mendapatkan nilai *threshold* terbaik namun juga melihat perubahan *running time* nya apakah signifikan atau tidak sebagaimana yang tercantum pada Tabel 6.12.

Tabel 6.12 Rata-rata Hasil Pengujian Seleksi Fitur *Information Gain* pada Setiap *Threshold*

<i>Threshold</i> yang digunakan	Rata-rata fitur	<i>Precision</i>	<i>Recall</i>	<i>F-measure</i>	Rata-rata Waktu Program (Menit)
10%	177	75.72%	64.12%	68.90%	0.3626
20%	356	67.19%	82.68%	74.09%	0.3746
30%	534	65.00%	91.07%	75.79%	0.3848
50%	891	70.82%	74.13%	72.13%	0.4053
40%	712	64.32%	93.11%	76.01%	0.4079
60%	1069	75.76%	68.23%	71.51%	0.4107
median	1040	76.26%	62.39%	67.93%	0.4138
70%	1247	73.11%	70.99%	71.26%	0.4215
80%	1425	70.56%	79.94%	74.78%	0.4347
mean	1407	70.78%	79.82%	74.77%	0.4366
90%	1603	71.78%	85.92%	77.65%	0.4561
100%	1782	77.27%	95.88%	85.51%	0.4647

Berdasarkan Tabel 6.12, perubahan *running time* atau waktu berjalannya program tidak signifikan seiring meningkatnya banyaknya fitur yang digunakan. Jika dilihat saat 90% fitur yang digunakan, *f-measure* yang diperoleh sebesar 77.65% memiliki perbedaan cukup besar yaitu 7.86% dibandingkan dengan *f-measure* ketika seluruh fitur digunakan. Namun perbedaan waktu program tidak jauh hanya 0.0086 menit atau 0.516 detik.

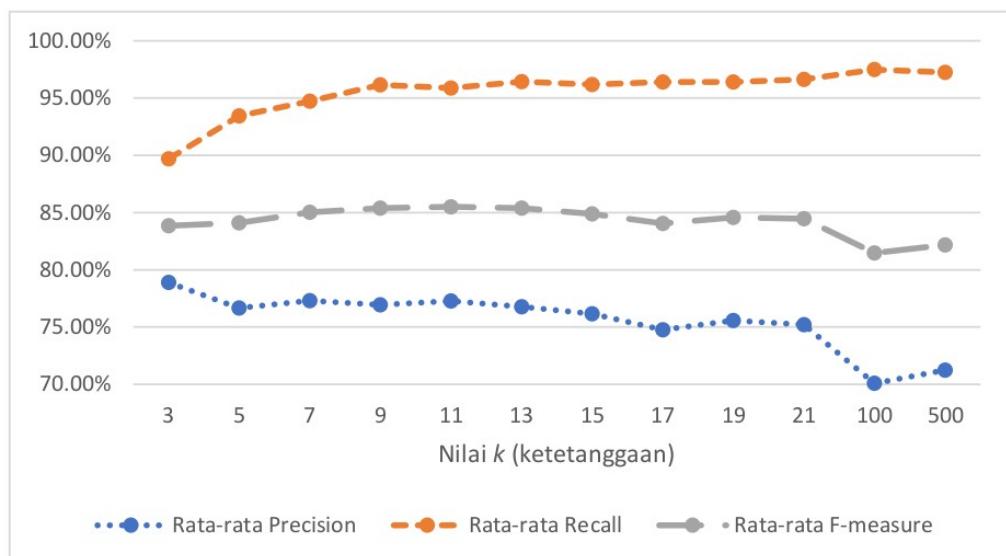
Sehingga nilai *precision*, *recall* dan *f-measure* terbaik diperoleh saat fitur yang digunakan yakni sebanyak 100% atau tidak menggunakan seleksi fitur dengan *precision* 77.27%, *recall* 95.88% dan *f-measure* 85.51% serta *running time* tertinggi yaitu 0.4647 menit.

6.3 Analisis

Pada bagian ini dijelaskan analisis terhadap hasil pengujian yang telah diperoleh dimulai dari pengujian nilai k dengan *5-fold cross validation* kemudian pengujian *threshold* dari *Information Gain* dengan *5-fold cross validation* menggunakan k terbaik yang kemudian dilihat berdasarkan nilai *precision*, *recall* dan *f-measure*.

6.3.1 Analisis Pengujian Nilai k dengan *5-fold*

Pada tahap ini akan dilakukan analisis pada pengaruh nilai k dengan *5-fold* untuk mendapatkan k terbaik sebagaimana digambarkan pada Gambar 6.1.

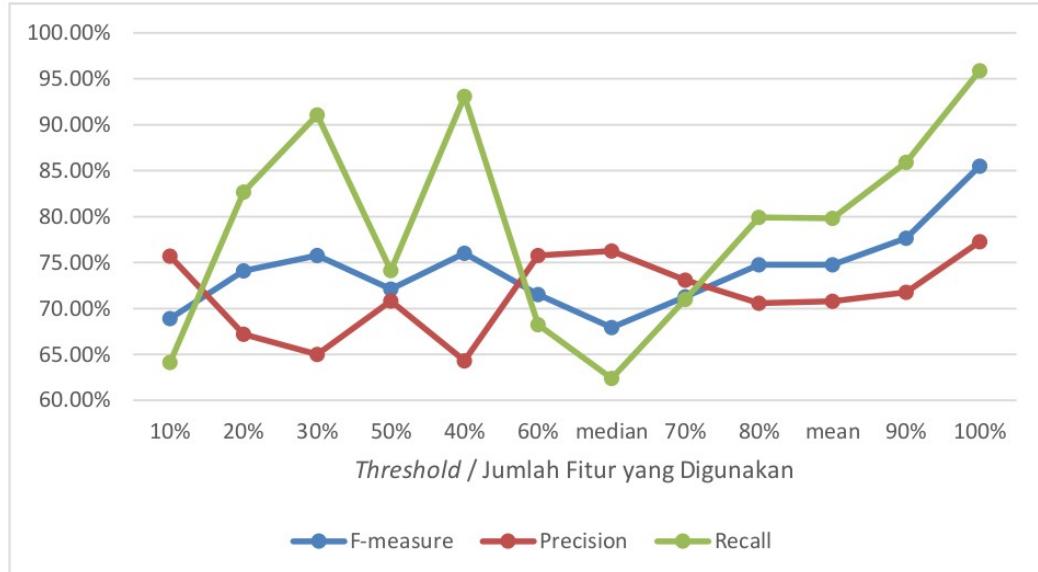


Gambar 6.1 Grafik Variasi Nilai k Terhadap *Precision*, *Recall* dan *F-measure*

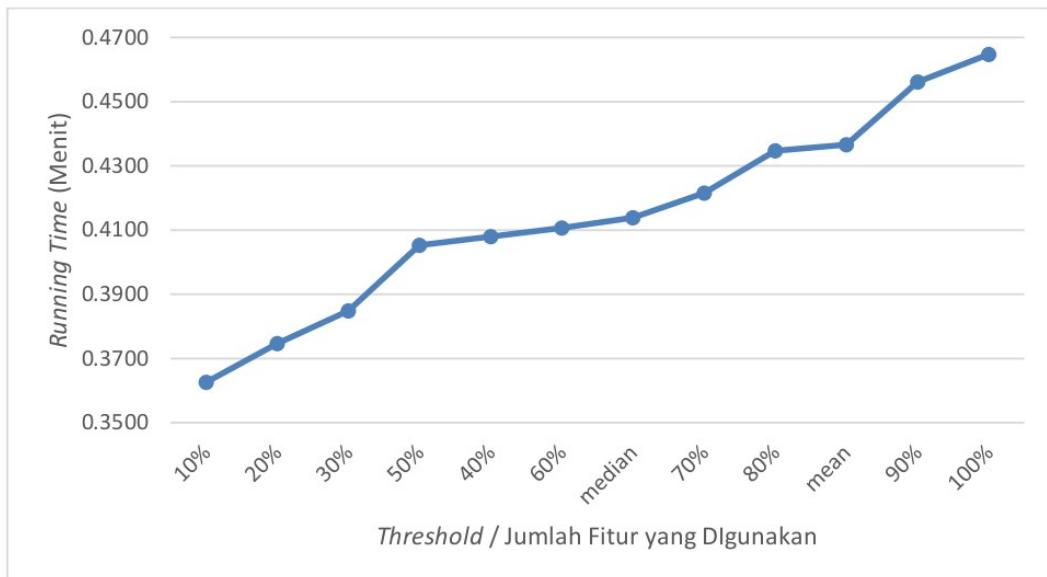
Berdasarkan Gambar 6.1, terlihat bahwa nilai *precision* tertinggi dicapai saat $k = 3$, sedangkan *recall* tertinggi dicapai ketika $k = 100$ dan *f-measure* saat $k = 11$. Selain itu terlihat juga bahwa pada *f-measure* tidak mengalami perubahan yang signifikan dari antar nilai k . Walaupun perubahan nilai k sangat besar dari 21 menuju 100 namun perubahan nilai evaluasinya tidak signifikan khususnya pada *recall* dan *f-measure*. Hal ini disebabkan karena metode *Improved K-Nearest Neighbor* ditujukan dan cocok untuk dataset dengan kelas yang tidak seimbang atau *imbalanced class*. Prinsip dari *Improved KNN* adalah dengan menentukan k pada setiap kelas sehingga kelas dengan anggota lebih sedikit atau *minority class* memiliki peluang yang lebih besar untuk masuk pada ketetanggaan terdekat. Pada penelitian ini dataset dengan kelas positif berjumlah 322 sedangkan kelas negatif berjumlah 178, jika menggunakan metode yang sensitif dengan *imbalanced class* maka evaluasi yang dihasilkan akan buruk. Dapat dikatakan bahwa pada kasus ini dengan *Improved KNN* sebagai metode klasifikasi menghasilkan evaluasi yang stabil pada nilai k yang kecil hingga besar.

6.3.2 Analisis Pengujian *Information Gain*

Pada bagian ini akan dijelaskan analisis pengujian terhadap *information gain* menggunakan berbagai variasi *threshold* serta persentase jumlah fitur yang digunakan yang divisualisasikan pada Gambar 6.2 dan Gambar 6.3.



Gambar 6.2 Hasil Pengujian *Information Gain* terhadap *Precision*, *Recall* dan *F-measure*



Gambar 6.3 Rata-rata Hasil Pengujian *Information Gain* dengan 5-fold terhadap Rata-rata *Running Time*

Berdasarkan Gambar 6.2 serta Gambar 6.3, terlihat bahwa tidak selamanya peningkatan jumlah fitur akan menghasilkan evaluasi yang lebih baik. Salah satu contohnya yaitu saat *threshold* yang digunakan 30% memiliki *f-measure* sebesar 75.79% sedangkan letika *threshold* 50% nilai *f-measure* nya sebesar 72.13% yang mana mengalami penurunan yaitu 3.66%. Terdapat keunikan ketika fitur yang

digunakan berkurang yakni sebelumnya 50% menjadi 40%, *running time* nya justru meningkat sebesar 0.0026 menit, jika ditelusuri lebih lanjut penyebabnya adalah adanya perbedaan signifikan pada *fold* ke-1 di mana *running time* pada *threshold* 40% lebih tinggi dibandingkan dengan 50% dengan perbedaan sebesar 0.0568 menit sedangkan pada *fold* lainnya *running time* ketika *threshold* 40% lebih singkat atau lebih kecil dibandingkan *threshold* 50%.

Adapun rata-rata *running time* tertinggi dari seluruh *fold* dicapai ketika semua fitur digunakan yaitu sebesar 0.4647 menit dengan rata-rata jumlah fitur terbanyak yaitu 1782. Serta, peningkatan *running time* mulai memberikan pengaruh terhadap *f-measure* saat *threshold* yang digunakan sebesar nilai median *information gain* dari seluruh *fold* dengan grafik *f-measure* yang ikut meningkat. Jika diamati berdasarkan naik turunnya nilai *f-measure* dibandingkan dengan *running time* maka terlihat bahwa evaluasi terbaik dicapai ketika semua fitur digunakan atau 100% dengan *f-measure* sebesar 85.51%. Sama seperti pengujian nilai *k* ketetanggaan yang belum menguji *threshold* *information gain* mendapat *f-measure* sebesar 85.51%. Hal ini disebabkan karena *information gain* akan bernilai rendah jika suatu term muncul pada kelas positif dan negatif. Akibatnya adalah term tersebut tidak mencirikan pada kelas tertentu karena tidak muncul pada satu kelas saja. Karena nilai *information gain* pada term tersebut rendah, akibatnya term tersebut dibuang karena dianggap kurang signifikan. Berikut merupakan 10 term dengan nilai *information gain* tertinggi dan 10 terendah beserta dengan nilai *document frequency* nya yang diambil dari *fold* ke-5 sebagai contoh sebagaimana dapat dilihat pada Tabel 6.13.

Tabel 6.13 Term beserta Nilai *Information Gain*, Kemunculan pada Dokumen dengan Kelas Positif dan Negatif

No	10 Term Tertinggi				10 Term Terendah			
	Term	IG	df positif	df negatif	Term	IG	df positif	df negatif
1	abang	0.2849	1	0	mrt	0.2773	37	17
2	abby	0.2849	1	0	jakarta	0.2771	48	12
3	abisini	0.2849	1	0	nya	0.2770	30	23
4	access	0.2849	1	0	ya	0.2767	23	30
5	acung	0.2849	1	0	ada	0.2761	40	22
6	aeon	0.2849	1	0	yg	0.2754	37	28
7	ahli	0.2849	1	0	ini	0.2751	39	28
8	ajaa	0.2849	1	0	dan	0.2749	55	20
9	akrab	0.2849	1	0	di	0.2713	70	34
10	aksen	0.2849	1	0	lrt	0.2665	127	57

Berdasarkan Tabel 6.13, terlihat bahwa term Irt dengan kemunculan yang tertinggi pada kedua kelas yaitu sebanyak 127 pada kelas positif dan 57 pada kelas negatif memiliki *information gain* paling rendah. Kemunculan Irt pada kedua kelas ini disebabkan karena kata ‘Irt’ saja tidak hanya dibahas ketika memberikan sentimen positif, namun juga negatif. Sebaliknya, term dengan kemunculan hanya pada satu kelas memiliki *information gain* yang tinggi karena term tersebut mencirikan bahwa ia memiliki makna signifikan pada kelas tersebut.

BAB 7 PENUTUP

Pada bab ini akan dijelaskan kesimpulan yang diperoleh pada penelitian ini berdasarkan pengujian maupun analisis yang telah dilakukan serta pemberian saran untuk penelitian berikutnya.

7.1 Kesimpulan

Berdasarkan penelitian, pengujian, dan analisis yang telah dilakukan dapat diambil kesimpulan sebagai berikut.

1. Perubahan nilai k yang beragam tidak memberi pengaruh signifikan terhadap hasil dari f -measure. Nilai dari f -measure cenderung stabil walaupun saat k bernilai besar. Penyebab dari hal tersebut yaitu kemampuan dari Improved KNN pada kasus *imbalanced class* dengan menentukan nilai k pada setiap kelas yang bertujuan agar *minority class* tidak terdominasi oleh *majority class*.
2. Peningkatan jumlah *threshold* atau jumlah fitur yang digunakan tidak selamanya berbanding lurus dengan rata-rata *running time* dari seluruh *fold*. Evaluasi terbaik diperoleh ketika seluruh fitur digunakan jika dilihat dari peningkatan *running time* dibandingkan dengan jumlah fitur yang digunakan. Disamping itu, meningkatnya *running time* mulai berpengaruh terhadap f -measure ketika *threshold* yang digunakan sebesar nilai median dari *information gain* di mana nilai f -measures juga meningkat.

7.2 Saran

Saran yang dapat diberikan pada penelitian ini untuk perkembangan pada penelitian berikutnya adalah sebagai berikut.

1. Menambah fitur N-gram agar dapat menangkap lebih dari satu kata seperti “Irt gembel” ataupun “bagusan Irt” pada kasus ini, dengan begitu dapat meminimalisir kemunculan satu kata “Irt” yang memiliki kemunculan tinggi di kedua kelas karena kata tersebut berdiri sendiri tidak disertai dengan kata keterangan yang mengikutinya.
2. Menggunakan *multilingual corpus* agar dapat menangani istilah asing yang terdapat pada dokumen. Pada kasus ini dapat diambil sampel yaitu term “announcementnya”, “announcer”, dan “announcernya” yang sebenarnya memiliki kata dasar yang sama yaitu “announcement”. Tujuan dari hal ini adalah untuk mengurangi jumlah fitur agar komputasi saat pembobotan maupun klasifikasi lebih cepat.
3. Melakukan penyamarataan pada kata dengan huruf berulang. Pada kasus ini dapat diambil sampel yaitu term “hehehehe” dan “hehe” yang sebenarnya dapat dianggap sebagai satu term yang sama. Namun dianggap berbeda karena terdapat pengulangan huruf.

4. Mencoba metode seleksi fitur lain salah satunya seperti *chi square* agar dapat membandingkan serta memastikan apakah seleksi fitur dapat meningkatkan hasil evaluasi ataukah tidak.

DAFTAR REFERENSI

- Al-Khafaji, D. H. K. & Habeeb, A. T., 2017. Efficient Algorithms for Preprocessing and Stemming of Tweets in. *Journal of Computer Engineering (IOSR-JCE)*, 19(3), pp. 44-50.
- Angiani, G. et al., 2016. A Comparison between Preprocessing. *KDWeb*, pp. 1-11.
- Baoli, L., Shiwen, Y. & Qin, L., 2003. *An Improved k-Nearest Neighbor Algorithm*. Beijing, Proceedings of the 20th International Conference on Computer Processing of Oriental Languages, pp. 2-3.
- Bhandari, S. & Ghosh, D. S., 2016. An Overview of Sentiment Analysis: Approaches and Applications. *International Journal of Research in Computer Science*, 3(4), pp. 4-7.
- Boyd, D. M. & Ellison, N. B., 2008. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, Volume 13, pp. 210-230.
- Fauziah, R., 2019. *Mengintip Pembangunan Stasiun LRT Ramah Lingkungan*. [Online] Available at: <https://economy.okezone.com/read/2019/01/27/320/2009982/mengintip-pembangunan-stasiun-lrt-ramah-lingkungan> [Accessed 15 Oktober 2019].
- Halwi, M. & Dewi H., C. M. T., 2019. *Hingga Juli 2019, 242 Ribu Penumpang Sudah Jajal LRT Jakarta*. [Online] Available at: <https://metro.tempo.co/read/1225192/hingga-juli-2019-242-ribu-penumpang-sudah-jajal-lrt-jakarta> [Accessed 15 Oktober 2019].
- Huq, M. R., Ali, A. & Rahman, A., 2017. Sentiment Analysis on Twitter Data using KNN and SVM. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, 8(6), p. 21.
- Jodha, R., BC, G. S., Chowdhary, K. & Mishra, A., 2018. Text Classification using KNN with different Features. *International Journal of Research Publications*, 8(1), pp. 6-7.
- Kapoor, K. K. et al., 2017. Advances in Social Media Research: Past, Present and Future. *Information Systems Frontiers*, pp. 1-28.
- Kietzmann, J. H., Hermkens, K., McCarthy, I. P. & Silvestre, B. S., 2011. Social media? Get serious! Understanding the functional building blocks of social media. *Business Horizons*, 54(3), pp. 241-251.
- Kohari, R. & Provost, F., 1998. Glossary of terms. *Editorial for the Special Issue on Applications of Machine and the Knowledge Discovery Process*.

- Kohavi, R., 1995. *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*. Montreal, International joint conference on Artificial intelligence.
- Liu, B., 2012. *Sentiment Analysis and Opinion Mining*. s.l.:Morgan & Claypool Publishers.
- Medhat, W., Hassan, A. & Korashy, H., 2014. Sentiment analysis algorithms and applications:. *Ain Shams Engineering Journal*, p. 1095.
- Nathania, D. Z., Indriati & Bachtiar, F. A., 2018. Klasifikasi Spam Pada Twitter Menggunakan Metode Improved K-Nearest Neighbor. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, Volume 2, pp. 3948-3956.
- 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> [Accessed 15 11 2019].
- Refaeilzadeh, P., Tang, L. & Liu, H., 2009. Cross-Validation. In: L. Liu & M. T. Özsü, eds. *Encyclopedia of Database Systems*. Boston: Springer US, pp. 532-538.
- Robbani, H. A., 2016. *Indonesian stemmer. Python port of PHP Sastrawi project..* [Online] Available at: <https://github.com/har07/PySastrawi> [Accessed 16 10 2019].
- Rudi, A., 2015. *Apa Saja Perbedaan LRT, MRT, dan KRL?*. [Online] Available at: <https://megapolitan.kompas.com/read/2015/07/03/07501531/Apa.Saja.Perbedaan.LRT.MRT.dan.KRL.?page=all> [Accessed 26 September 2019].
- Salton, G. & Buckley, C., 1988. Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5), pp. 513-523.
- Saputra, P. Y., 2017. IMPLEMENTASI TEKNIK CRAWLING UNTUK PENGUMPULAN DATA DARI MEDIA SOSIAL TWITTER. *Jurnal Dinamika Dotcom*, 8(2), pp. 160-168.
- Schouten, K., Frasincar, F. & Dekker, R., 2016. An Information Gain-Driven Feature Study for Aspect-Based Sentiment Analysis. In: *In Natural Language Processing and Information Systems*. s.l.:Springer International Publishing, pp. 48-59.
- Shang, L., 2012. A Feature Selection Method Based on Information Gain and Genetic Algorithm. *International Conference on Computer Science and Electronics Engineering*, pp. 355-358.
- Sparck Jones, K., 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1), pp. 11-21.

Sudheer, P. K. & Valarmathi, D. B., 2018. Real Time Sentiment Analysis of E-Commerce Websites using Machine Learning Algorithms. *International Journal of Mechanical Engineering and Technology (IJMET)*, 9(2), pp. 180-193.

Wang, Q., Guan, Y., Wang, X. & Xu, Z., 2006. A Novel Feature Selection Method Based on Category Information Analysis for Class Prejudging in Text Classification. *International Journal of Computer Science and Network Security*, Volume 6, pp. 113-119.

Widowati, H., 2019. *Ujicoba LRT Dimulai, Ini Beda LRT, MRT, dan KRL*. [Online] Available at: <https://katadata.co.id/berita/2019/06/12/ujicoba-lrt-dimulai-ini-beda-lrt-mrt-dan-krl> [Accessed 26 September 2019].