KLASIFIKASI ADUAN MASYARAKAT PADA SAMBAT *ONLINE* KOTA MALANG MENGGUNAKAN NW-KNN DAN SELEKSI FITUR *INFORMATION GAIN – GENETIC ALGORITHM*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Rosi Afiqo
NIM: 145150200111044

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

PENGESAHAN

KLASIFIKASI ADUAN MASYARAKAT PADA SAMBAT *ONLINE* KOTA MALANG MENGGUNAKAN NW-KNN DAN SELEKSI FITUR *INFORMATION GAIN – GENETIC ALGORITHM*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Rosi Afiqo
NIM: 145150200111044

Skripsi ini telah diuji dan dinyatakan lulus pada
1 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Agus Wahyu Widodo, S.T, M.Cs	Budi Darma Setiawan, S.Kom, M.Cs
NIP: 19740805 200112 1 001	NIP: 19841015 201404 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 8 Agustus 2018

Rosi Afiqo

NIM: 145150200111044

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberi taufik, hidayah, inayah serta kasih dan sayang-Nya, sehingga penulis dapat menyelesaikan skripsi ini dengan sebaik-baiknya. Sholawat dan salam tidak lupa penulis curahkan kepada utusan Allah SWT yang menjadi junjungan kita yaitu Nabi Agung Muhammad SAW, semoga kita semua diakui sebagai umat beliau dan mendapatkan syafa'at beliau di hari akhir nanti aamiin.

Skripsi ini berjudul "Klasifikasi Aduan Masyarakat pada SAMBAT *Online* Kota Malang dengan Menggunakan Seleksi fitur *Information Gain-Genetic Algorithm*". Tujuan penulisan skripsi ini adalah untuk memenuhi sebagian persyaratan untuk mendapatkan gelar sarjana komputer dari Fakultas Ilmu Komputer, Jurusan Teknik Informatika, Program Studi Teknik Informatika di Universitas Brawijaya. Terselesainya pengerjaan skripsi ini tentunya tidak lepas dari pihak-pihak yang telah banyak membantu dalam banyak hal. Penulis sampaikan terima kasih kepada:

1. Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D selaku Dekan Fakultas Ilmu Komputer yang telah memberikan fasilitas dalam pengerjaan skripsi ini.
2. Agus Wahyu Widodo, S.T., M.Cs yang telah memberikan bimbingan dan pengarahan dalam pengerjaan skripsi ini.
3. Budi Darma Setiawan, S.Kom., M.Cs yang telah memberikan bimbingan dan pengarahan dalam pengerjaan skripsi ini.
4. Faizatul Amalia S.Pd., M.Pd, pembimbing *tahfidz* putri, yang telah memberi motivasi dan dukungan dalam pengerjaan skripsi.
5. Zumrotus Sholihah, ibu penulis, yang telah memberikan doa yang luar biasa kepada penulis.
6. Masruri Syuhadak, Siti Ulfah, orang tua penulis, yang telah memberikan fasilitas, doa, dan dukungan dalam pengerjaan skripsi.
7. Miftakhul Huda, Dandong Prastyawan, Niska Shofia, Ana Mufida, Faris Humami, Muchammad Farhan Anda, Nabil Mazaputra, Ahmad Faiq Hakiim, Azifa Bilqis Sania Firdausi, Aisha Zaafirah Ameeranada, kakak, adik, serta keponakan penulis, yang telah memberi fasilitas, dukungan, pengarahan, semangat, motivasi, dan hiburan dalam pengerjaan skripsi.
8. Dwi Wahyu Puji Lestari, teman seperjuangan penulis, yang telah memberi motivasi, semangat, dukungannya dalam pengerjaan skripsi.
9. Riesma Rahman Nia, Siti Nur Wahidah Abroriyah, Uswatun Hasanah, Karmila Dewi Sulistyowati, Dhiyaan Khansa, Rosy Indah Permatasi, Dyah Ayu Wahyuning Dewi, sahabat seperjuangan penulis yang tergabung dalam Grup Pejuang Ayam Krispi, yang telah memberi motivasi, semangat, dan dukungan dalam pengerjaan skripsi.

10. Irene Wahyu Khairunnisa, Istin Fitriana Aziza, dan teman-teman kost 139 yang telah memberikan dukungan dan semangat dalam pengerjaan skripsi.
11. Novia Agusvina, S.Kom, teman sekota rantau dari asal kota yang sama, yang telah memberi fasilitas, semangat dan dukungan dalam pengerjaan skripsi.
12. Siti Sarah Ramadhani, *partner* menggapai cita-cita, yang telah memberi semangat dalam pengerjaan skripsi.
13. Firsta Ni'mah, *partner* bimbingan, yang selalu direpotkan oleh penulis dalam proses bimbingan serta yang telah memberi semangat dalam pengerjaan skripsi.
14. Teman-teman TIF-C 2014 serta seluruh teman-teman seangkatan dan seperjuangan yang telah memberi semangat dan dukungan dalam pengerjaan skripsi.
15. Seluruh pihak yang terlibat dan yang telah menjadi saksi perjalanan penulis selama ini hingga selesainya masa studi penulis, yang tidak dapat penulis sebutkan satu-persatu.

Atas dukungan, semangat, motivasi dan doa yang telah diberikan, penulis mengucapkan terima kasih. Semoga skripsi ini dapat memberi manfaat bagi seluruh pihak, khususnya bagi pembaca dan ilmu pengetahuan.

Malang, 18 Juli 2018

Penulis

afi.afiqo@gmail.com

ABSTRAK

Rosi Afiqo, Klasifikasi Aduan Masyarakat pada SAMBAT *Online* Kota Malang Menggunakan NW-KNN dan Seleksi Fitur *Information Gain – Genetic Algorithm*

Pembimbing: Agus Wahyu Widodo, S.T., M.Cs dan Budi Darma Setiawan, S.Kom., M.Cs

SAMBAT *Online* merupakan sistem aplikasi yang digunakan untuk menampung aduan dari masyarakat terhadap pemerintah Kota Malang. Tidak dilengkapinya fitur pemilihan SKPD terkait pada sistem tersebut menyulitkan diskominfo Kota Malang dalam melakukan pelaporan aduan tersebut kepada SKPD terkait. Hal ini dikarenakan pengelompokan aduan masyarakat berdasarkan SKPD terkait masih dilakukan secara manual. Oleh karena itu, diperlukan sistem yang dapat mengelompokkan aduan berdasarkan SKPD terkait secara otomatis untuk efisiensi waktu. NW-KNN merupakan metode klasifikasi yang dapat digunakan untuk menangani masalah data tidak seimbang yang bekerja dengan melibatkan seluruh *data training* dalam prosesnya. Teknik seleksi fitur yang akan digunakan adalah *information gain* dan *genetic algorithm* untuk mendapatkan jumlah fitur yang sedikit dan *f-measure* yang tinggi. Tahapan yang dilakukan sistem dalam mendapatkan fitur terbaik yaitu pertama *pre-processing data*, kedua seleksi fitur dengan *information gain*, dan yang ketiga seleksi fitur dengan *genetic algorithm*. Hasil pengujian yang dilakukan menghasilkan rata-rata *f-measure* sebesar 0,22 untuk data tidak seimbang dan 0,39 untuk data seimbang. Hasil tersebut telah mengalami peningkatan hingga 0,04 untuk data tidak seimbang dan 0,22 untuk data seimbang dari hasil klasifikasi tanpa menggunakan proses seleksi fitur. Berdasarkan hasil tersebut, dapat disimpulkan bahwa klasifikasi menggunakan NW-KNN dapat mengatasi permasalahan keseimbangan data dan penggunaan seleksi fitur *information gain – genetic algorithm* dapat digunakan untuk meningkatkan hasil klasifikasi.

Kata kunci: klasifikasi, NW-KNN, seleksi fitur, *information gain*, algoritme genetika

ABSTRACT

Rosi Afiqo, Classification of Public Complaints on SAMBAT Online of Malang City by Using NW-KNN and Information Gain – Genetic Algorithm Selection Features

Supervisors: Agus Wahyu Widodo, S.T., M.Cs and Budi Darma Setiawan, S.Kom., M.Cs.

SAMBAT Online is an application system used to accommodate complaints from the public against to the government of Malang. The incomplete features of SKPD selection related to the system made it difficult for Diskominfo of Malang City to report the complaint to the related SKPD. This is because the complaint grouping based on related SKPD is still done manually. Therefore, a system that can group complaints based on the relevant SKPD is required for time efficiency. NW-KNN is classification method which can be used to handle balanced issues that work by involving all training data in the process. The feature selection techniques that will be used are information gain and genetic algorithm to get a small number of features and high f-measure. Stages performed in the system get the best features of the first is pre-processing data, second is feature selection by using information gain, and the third is selection features by using genetic algorithm. The results of the tests performed resulted 0.22 in average of f-measure for unbalanced data and 0.39 for balanced data. These results have increased up to 0.04 for unbalanced data and 0.22 for balanced data from classification results without using feature selection process. Based on these results, it can be concluded that the classification using NW-KNN and information gain-genetic algorithm feature selection can be used to improve the classification results.

Keywords : classification, NW-KNN, feature selection, information gain, genetic algorithm

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

SAMBAT *Online* merupakan akronim dari Sistem Aplikasi Masyarakat Bertanya Terpadu *Online*. Sistem aplikasi ini dikelola langsung oleh Dinas Komunikasi dan Informatika (Diskominfo) Kota Malang sebagai salah satu upaya dalam mengembangkan sistem pelayanan publik. Dengan adanya sistem aplikasi ini, masyarakat Kota Malang dapat menyalurkan baik kritik, saran, pertanyaan maupun pengaduan mengenai pemerintah Kota Malang.

Saat ini, aduan masyarakat Kota Malang dapat disalurkan melalui dua cara yaitu *Short Message Service* (SMS) dan *website* SAMBAT *online* itu sendiri. Pengaduan yang masuk baik melalui SMS maupun *website* akan tersimpan dalam *database* sehingga masyarakat yang mengajukan aduan tidak perlu khawatir aduannya tidak sampai pada Diskominfo Kota Malang. Pengaduan yang sesuai dengan peraturan dan ketentuan akan ditindaklanjuti dengan meneruskannya pada jajaran Satuan Kerja Perangkat Daerah (SKPD) terkait. Akan tetapi, pada SAMBAT *online* tidak terdapat pilihan SKPD pemerintah Kota Malang, sehingga masyarakat Kota Malang hanya menuliskan aduan mereka tanpa menyertakan SKPD yang dituju. Sedangkan masyarakat sendiri baru dapat mengetahui pada SKPD mana aduan mereka akan ditindaklanjuti yaitu berasal dari balasan atau tanggapan SKPD yang terkait dari aduan masyarakat tersebut.

Kondisi tersebut menimbulkan kesulitan bagi Diskominfo Kota Malang untuk meneruskannya pada SKPD terkait karena setiap pengaduan yang masuk dapat berisi aduan terhadap SKPD yang berbeda-beda dan saat ini, penyortiran yang dilakukan masih bersifat manual yaitu dengan memanfaatkan sumber daya manusia. Oleh karena itu, diperlukan sebuah sistem yang dapat melakukan penyortiran aduan masyarakat pada SAMBAT *Online* Kota Malang berdasarkan SKPD terkait secara otomatis, sehingga dalam pelaporan yang dilakukan oleh Diskominfo Kota Malang kepada SKPD dapat dilakukan dengan lebih mudah dan cepat.

Salah satu cara pernyortiran aduan masyarakat berdasarakan SKPD terkait dapat dilakukan dengan klasifikasi. Menurut Pramudiono (2003), *classification* adalah sebuah proses penemuan model untuk membedakan antar kelas, sehingga dokumen yang belum diketahui kelasnya dapat digolongkan pada kelas yang sesuai. Beberapa peneliti sebelumnya (Suharno, 2017; Sari, 2018; Prasanti, 2018) telah mengangkat SAMBAT *Online* sebagai objek dalam penelitiannya untuk dikelompokkan menggunakan teknik klasifikasi yaitu menggunakan metode *K-Nearest Neighbor*. Akan tetapi, masih diperlukan adanya penelitian lain yang mengangkat objek yang serupa agar dapat dibandingkan ketepatan hasil dalam proses pengelompokan (klasifikasi) aduan masyarakat pada SAMBAT *Online* Kota Malang dengan memadukan metode KNN tersebut dengan metode lain.

K-Nearest Neighbor (KNN) adalah salah satu metode klasifikasi, dimana dalam pemrosesannya melibatkan seluruh *data training* dengan memperhitungkan jarak terdekat atau kemiripan data (Riany, Fajar, & Lukman, 2016). Metode K-NN ini memiliki kelemahan salah satunya adalah menggunakan seluruh atribut untuk proses pengklasifikasian (Kustiyahningsih & Syafa'ah, 2015). Sehingga perlu menerapkan *feature selection* sebelum dilakukan pengklasifikasian. *Feature selection* merupakan teknik untuk mereduksi jumlah fitur, dimana fitur terbaik dari sekumpulan fitur yang nantinya akan digunakan. Menurut Yang & Pedersen (1997, disitasi dalam Sari & Puspanigrum, 2013, p.27) *information gain* merupakan teknik seleksi fitur yang memiliki akurasi yang paling bagus dibandingkan seleksi fitur yang lain seperti *document frequency thresholding*, *mutual information*, *chi-square*, dan *Term Strength*. Akan tetapi penelitian yang dilakukan oleh Uguz (2011) menunjukkan bahwa klasifikasi dokumen dengan menggunakan seleksi fitur *information gain* yang dipadukan dengan *genetic algorithm* menghasilkan *f-measure* yang lebih tinggi dibandingkan dengan menggunakan perpaduan *information gain* dengan *principal component analysis. Genetic algorithm* merupakan algoritme *stochastic operator* yang bersifat *probabilistic* yang memiliki salah satu kelebihan yaitu dapat ditemukannya berbagai macam jenis solusi dari solusi yang lain. Sehingga hal ini dapat memberi kemungkinan terdapatnya jumlah fitur yang sedikit yang berpeluang menghasilkan tingkat akurasi yang tinggi. Untuk dapat mengatasi adanya data yang tidak seimbang, diperlukan perpaduan antara *Neighbor Weighted* dengan KNN (NW-KNN) agar dapat menghasilkan hasil klasifikasi yang lebih baik dari pada KNN (Ridok & Latifah, 2015).

Berdasarkan uraian tersebut, penulis mengangkat objek yang sejenis yaitu aduan masyarakat pada SAMBAT *Online* Kota Malang untuk dikelompokkan dengan teknik klasifikasi menggunakan metode NW-KNN sebagai proses pengklasifikasian dan *information gain-genetic algorithm* sebagai seleksi fitur untuk mendapatkan tingkat akurasi yang lebih tinggi dari hasil penelitian sebelumnya.

1.2 Rumusan masalah

Dari latar belakang yang telah dipaparkan, dapat dirumuskan rumusan masalah yang akan dikaji pada penelitian yang akan dilakukan yaitu:

1. Berapa jumlah fitur berdasarkan nilai *information gain* yang dapat menghasilkan tingkat akurasi yang tinggi?
2. Berapa besar *genetic algorithm* dapat meningkatkan hasil akurasi dari pengklasifikasian aduan masyarakat apabila dibandingkan dengan tanpa menggunakan *genetic algorithm*?

1.3 Tujuan

Tujuan yang menjadi target pencapaian dari penelitian ini adalah sebagai berikut:

1. Mengetahui banyaknya jumlah fitur yang dapat menghasilkan tingkat akurasi yang tinggi berdasarkan nilai *information gain* yang dimiliki.
2. Mengetahui besar kenaikan akurasi yang dihasilkan *genetic algorithm* dalam pengklasifikasian aduan masyarakat berdasakan SKPD terkait jika dibandingkan tanpa menggunakan *genetic algorithm*.

1.4 Manfaat

Manfaat bagi pihak-pihak terkait yang menjadi dampak dilakukannya penelitian ini adalah :

1.4.1 Bagi peneliti

1. Sebagai media untuk mengimplementasikan ilmu mengenai Teknologi Informasi (TI) khususnya dalam bidang komputasi cerdas.
2. Penulis mendapatkan pemahaman mengenai metode *Nearest Weighted K-Nearest Neighnor* (NW-KNN) dan seleksi fitur *information gain - genetic algorithm* dalam mengklasifikasikan aduan masyarakat pada SAMBAT *Online* Kota Malang.

1.4.2 Bagi Diskominfo Kota Malang

1. Pengguna dapat mengetahui aduan masyarakat yang sesuai dengan SKPD tertentu secara otomatis tanpa membaca isi aduan terlebih dahulu.
2. Pengguna dapat melaporkan aduan masyarakat kepada SKPD terkait secara lebih mudah dan cepat.

1.4.3 Bagi masyarakat Kota Malang

Masyarakat dapat memperoleh tanggapan atas aduan yang diajukan pada waktu yang lebih cepat

1.5 Batasan masalah

Batasan masalah dibuat dengan tujuan pembahasan penelitian tidak menyimpang dari yang dirumuskan sebelumnya oleh peneliti. Batasan masalah penelitian ini yaitu:

1. Metode klasifikasi yang diterapkan adalah NW-KNN (*Nearest Weighted K-Nearest Neighbor*)
2. Metode seleksi fitur yang diterapkan adalah *information gain* dan *genetic algorithm*
3. Obyek yang diklasifikasikan adalah aduan masyarakat pada SAMBAT *Online* Kota Malang
4. Obyek yang diklasifikasikan menggunakan Bahasa Indonesia

5. Keluaran yang dihasilkan adalah perhitungan dari metode yang digunakan.
6. Aduan masyarakat yang digunakan adalah aduan masyarakat yang ditujukan pada 3 SKPD yaitu DKP, DPUPPB, dan Dishub

1.6 Sistematika pembahasan

Sistematika penulisan penelitian ini adalah:

1.6.1 Bab 1 pendahuluan

Bab ini meliputi latar belakang, rumusan, batasan masalah, tujuan, manfaat dilakukannya penelitian serta sistematika dalam penulisan mengenai klasifikasi aduan masyarakat pada SAMBAT *Online* Kota Malang menggunakan metode NW-KNN dan seleksi fitur *information gain-genetic algorithm.*

1.6.2 Bab 2 landasan kepustakaan

Bab ini meliputi kajian kepustakaan serta dasar teori terkait dengan penelitian dalam mengklasifikasikan aduan masyarakat pada SAMBAT *Online* Kota Malang menggunakan metode NW-KNN dan seleksi fitur *information gain-ganetic algorithm*.

1.6.3 Bab 3 metodologi penelitian

Bab ini membahas metode atau langkah-langkah yang akan dijalankan oleh peneliti dalam mengklasifikasikan aduan masyarakat pada SAMBAT *Online* Kota Malang menggunakan metode NW-KNN dan seleksi fitur *information gain-ganetic algorithm*.

1.6.4 Bab 4 perancangan

Bab ini berisi perancangan alur penyelesaian masalah yang akan dilakukan pada setiap proses dari setiap teknik/metode yang digunakan.

1.6.5 Bab 5 implementasi

Pada bab ini berisi impelemtasi dari proses metode/teknik yang digunakan pada penelitian ini.

1.6.6 Bab 6 pengujian dan analisis

Bab ini berisi hasil pengujian serta analisisnya terhadap parameter-parameter yang mempengaruhi teknik/metode yang digunakan pada pengklasifikasian aduan masyarakat yang akan dilakukan.

1.6.7 Bab 7 penutup

Bab ini memuat kesimpulan serta saran atas kekurangan dalam penelitian mengenai klasifikasi aduan masyarakat pada SAMBAT *Online* Kota Malang menggunakan metode NW-KNN dan seleksi fitur *information gain-ganetic algorithm*.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian pustaka

Sebelumnya sudah ada yang meneliti tentang masalah yang serupa tetapi belum ada yang menerapkan metode NW-KNN dimana *information gain* dan *genetic algorithm* juga diterapkan sebagai teknik seleksi fitur untuk mengklasifikasikan aduan masyarakat pada SAMBAT *Online* Kota Malang. Sehingga pada kajian kepustakaan ini, peneliti akan memaparkan tentang hasil penelitian sebelumnya mengenai pengaruh nilai k, jumlah fitur yang digunakan serta pemilihan seleksi fitur terhadap kinerja algoritme NW-KNN pada proses pengklasifikasian.

Pada penelitian yang telah dilakukan oleh Maulida, Suyatno, & Hatta (2016) mengenai penggunaan metode *information gain* untuk seleksi fitur menunjukkan hasil bahwa *information gain* mampu mereduksi fitur hingga 89%. Pada penelitian yang telah dilakukan oleh Sari, B. N. (2016) mengenai penggunakan teknik seleksi fitur *information gain* untuk prediksi peforma akademik siswa, menunjukkan bahwa *information gain* menghasilkan hasil yang terbaik, dimana teknik ini bisa meningkatkan tingkat akurasi sistem karena teknik ini berhasil mereduksi fitur yang tidak relevan dengan target klasifikasi. Pada penelitian yang telah dilakukan oleh Darmawan (2015) mengenai penggunaan seleksi fitur *genetic algorithm* menunjukkan bahwa algoritme *genetic algorithm* dapat meningkatkan hasil akurasi sistem dibandingkan tanpa menggunakan *genetic algorithm*. Pada penelitian yang telah dilakukan oleh Somantri & Khambali (2017) mengenai penggunakan *genetic algorithm* sebagai *feature selection* menunjukkan bahwa *genetic algorithm* sebagai *feature selection* dapat meningkatkan hasil klasifikasi sebesar 5.7%. Pada penelitian yang telah dilakukan oleh Muthia (2016) mengenai penggunaan seleksi fitur IG-GA dapat meningkatkan hasil akurasi hingga 6%. Berdasarkan penelitian-penelitian tersebut dapat disimpulkan bahwa seleksi fitur *information gain* dan *genetic algorithm* dapat mereduksi fitur dan meningkatkan hasil akurasi.

Pada penelitian yang telah dilakukan oleh Suharno, Fauzi, & Perdana (2017) mengenai pengklasifikasian dokumen sambat *online* menggunakan metode KNN yang dipadukan dengan seleksi fitur *chi-square* menunjukkan bahwa banyaknya jumlah fitur yang digunakan memengaruhi nilai *f-measure* yang dihasilkan. Selain itu, hasil penelitian yang dilakukan peneliti menunjukkan bahwa nilai k tidak memengaruhi nilai *f-measure* yang dihasilkan. Pada penelitian yang telah dilakukan oleh Sari, Fauzi, & Adikara (2018) mengenai pengklasifikasian dokumen sambat *online* menggunakan metode KNN yang dipadukan dengan seleksi fitur *categorical proportional difference* menunjukkan bahwa perpaduan metode KNN dengan seleksi fitur *categorical proportional difference* jika dibandingkan dengan hanya menggunakan metode KNN menghasilkan tingkat akurasi yang lebih rendah. Sedangkan nilai k yang dihasilkan mempengaruhi tingkat akurasi yang dihasilkan. Pada penelitian yang telah dilakukan oleh Prasanti, Fauzi, & Furqon

(2018) mengenai pengklasifikasian dokumen sambat *online* menggunakan metode n-gram dan NW-KNN menunjukkan hasil bahwa jumlah nilai tetangga k berpengaruh terhadap tingkat kesalahan pada hasil klasifikasi yang dihasilkan, dan NW-KNN dapat mengatasi data yang tidak seimbang ketika nilai k besar. Berdasarkan penelitian-penelitian tersebut dapat disimpulkan bahwa nilai k berpengaruh terhadap hasil klasifikasi yang dihasilkan dan jumlah fitur yang digunakan yang diseleksi dengan teknik seleksi fitur dapat memberikan hasil yang lebih baik apabila seleksi fitur yang digunakan tepat.

Persamaan dan perbedaan dari penelitian yang telah dilakukan dengan penelitian yang penulis ajukan berdasarkan studi literatur yang telah dilakukan terdapat pada Tabel 2.1.

Tabel 2.1 Perbedaan penelitian sebelumnya dengan usulan

Peneliti	Judul	Objek	Metode	Keluaran
			Proses	**Hasil Penelitian**
Suharno, Fauzi, & Perdana (2017)	Klasifikasi Teks Bahasa Indonesia pada Dokumen Pengaduan Sambat *Online* Menggunakan Metode *K-Nearest Neighbor* dan *Chi-Square*	Dokumen Pengaduan Sambat *Online*	- *K-Nearest Neighbor* (KNN) - *Chi-square*	- Rasio fitur - *Precision* - *Recall* - *F-measure*
			- *Preprocessing* - Seleksi Fitur - *Term Weighting* - Klasifikasi Dokumen Uji	- Besarnya jumlah tetangga terdekat k tidak memengaruhi hasil klasifikasi - Banyaknya jumlah fitur memengaruhi hasil klasifikasi.
Sari, Fauzi, & Adikara (2018)	Klasifikasi Dokumen Sambat *Online* Menggunakan Metode *K-Nearest Neighbor* dan *FeaturesSelection Berbasis Categorical Proportional Difference*	Dokumen Sambat *Online*	- *K-Nearest Neighbor* (KNN) - *Categorical Proportional Defference*	- Rasio jumlah fitur - *Accuracy* - *Precision* - *Recall* - *F-measure*
			- *Preprocessing* - *Feature Selection* - *Term Weighting* - Klasifikasi	- Semakin besar nilai k tingkat akurasi yang dihasilkan semakin kecil. - Semakin kecil jumlah fitur yang digunakan semakin rendah tingkat akurasi yang dihasilkan - Metode KNN dengan *categorical proportional difference* menghasilkan tingkat akurasi yang lebih rendah dibandingkan dengan menggunakan metode KNN itu sendiri.
Prasanti, Fauzi, & Furqon	Klasifikasi Teks Pengaduan pada Sambat *Online*	Teks pengaduan pada	- *N-gram* - *Neighbor* Weighted *K*-	- *Precision* - *Recall*

Tabel 2.1 Perbedaan penelitian sebelumnya dengan usulan

Peneliti	Judul	Objek	Metode	Keluaran
			Proses	**Hasil Penelitian**
(2018)	Menggunakan Metode *N-Gram* dan *Neighbor* Weighted *K-Nearest Neighbor* (NW-KNN)	sambat *online*	*Nearest Neighbor* (NW-KNN)	- *F-measure*
			- *Preprocessing* - *Term Weighting* - Klasifikasi teks dengan NW-KNN	- Metode N-gram tidak berpengaruh besar terhadap hasil klasifikasi - Penggunaan NW-KNN dapat mengatasi data yang tidak seimbang ketika nilai k besar
Muthia (2016)	*Opinion Mining* pada *Review* Buku Menggunakan Algoritme *Naive Bayes*	*Review* buku *best seller* yang akan difilmkan	- *Information gain* - *Genetic algorithm* - *Naive bayes*	- Akurasi
			- *Preprocessing* - *Feature selection (information gain-genetic algorithm*) - Klasifikasi (*Naive bayes*)	- Penggabungan metode seleksi fitur dapat meningkatkan hasil akurasi klasifikasi
Afiqo, Widodo, & Setiawan (2018)	Klasifikasi Aduan Masyarakat pada SAMBAT *Online* Kota Malang Menggunakan Metode NW-KNN dan Seleksi Fitur *Information Gain-Genetic Algorithm*	Aduan masyarakat pada SAMBAT *Online* Kota Malang	- *Information gain* - *Genetic algorithm* - *Neighbor Weighted K-Nearest Neighbor* (NW-KNN)	- Rasio jumlah fitur - *Precision* - *Recall* - *F-measure*
			- *Preprocessing* - Seleksi Fitur (*information gain – genetic algorithm)* - *Term Weighting* - Klasifikasi dengan NW-KNN	- Dengan rasio jumlah fitur yang dihasilkan dari seleksi fitur *information gain* dan *genetic algorithm* serta dengan nilai k yang besar dapat menghasilkan estimasi hasil klasifikasi yang lebih akurat

2.2 *Pre-processing*

Pada salah satu bidang ilmu yang mempelajari tentang pengolahan data tekstual yang tidak terstruktur yang biasa disebut bidang *text mining*, terdapat teknik *pre-processing* yang biasa digunakan untuk mengomputasikan data

tekstual. Data tekstual dapat diolah untuk dikomputasi dengan cara mengkonversikannya dalam bentuk data numerik. Menurut Sari & Puspaningrum (2013) dalam penelitiannya, tahap *pre-processing* terdiri dari *case folding*, *stopword removal*, *tokenization*, dan *stemming*. Menurut Manning, Prabhakar, & Hinrich (2009, disitasi dalam Handoyo, M, & Nasution, 2014, p.74-75), proses *case folding* pada umumnya adalah proses yang merubah semua huruf kapital menjadi huruf kecil (*lowercase*) dan *Stopword removal* adalah proses penghapusan kata yang tidak penting dalam artian adalah kata yang tidak mempunyai makna yang berarti. Kata yang tidak penting misalnya adalah kata sambung, kata depan, dll (Hariri, Utami, & Amborowati, 2015). Dalam penelitian ini, daftar *stopword* yang digunakan adalah *stopword* dari Tala (2003). *Tokenization* adalah proses yang memecah kalimat menjadi kata/*token* sedangkan *Stemming* adalah suatu proses pengembalian semua kata menjadi kata asalnya (kata dasar). Pada penelitian kali ini, *stemming* dilakukan dengan menerapkan algoritme Nazief dan Adriani.

2.3 Algoritme nazief dan adriani

Menurut Asian (2007) dalam penelitiannya, algoritme Nazief dan Andriani adalah algoritme yang paling efektif, dibandingkan algoritme *stemming* lainnya seperti algoritme Arifin dan Setiono, algoritme Vega, algoritme Ahmad, Yusoff, dan Sembok, serta algoritme Idris. Dalam melakukan proses *stemming* dengan menggunakan algoritme nazief dan adriani diperlukan kamus kata dasar. Dalam penelitian ini, kamus kata dasar yang digunakan diambil dari https://bahtera.org yang dilakukan oleh Rahmady Liyantanto. Bahtera adalah kamus Bahasa Indonesia yang digunakan sebagai rujukan sesuai Kamus Besar Bahasa Indonesia (KBBI) yang memiliki 28.256 kata dasar (Liyantanto, 2011). Berikut adalah tahapan algoritme *stemming* nazief dan adriani berdasarkan penelitian yang telah dilakukan oleh Nazief & Adriani (1996, disitasi dalam Pramudita, 2014, p.16-17) :

1. Mencocokkan kata dengan kamus yang berisi kata dasar (*stem*), apabila cocok (kata tersebut terdapat dalam kamus) maka kata itu merupakan *root word* dan algoritme tidak dilanjutkan.
2. Menghapus *inflection suffixes* yaitu “-ku”, “-lah”, “-mu”, “-kah”, atau “-nya”. Akan tetapi, apabila yang menjadi *inflection suffixes* nya adalah “-lah”, “-kah”, “-pun”, atau “-tah” maka langkah 2 dikerjakan kembali untuk menghilangkan “-ku”, “-mu” dan “-nya” (*possesive pronouns*).
3. Membuang *derivation suffixes* yaitu “-an”, “-i”, atau “-kan”. Apabila hasilnya ditemukan dalam kamus, maka hentikan algoritme, apabila tidak ditemukan, maka lakukan langkah 3 poin a:
    a. Apabila akhiran “-an” telah dihilangkan, dan huruf terakhir dari kata tersebut adalah huruf “-k”, maka hapus huruf tersebut. Kemudian cocokkan kembali kata tersebut dengan kamus, dan apabila tidak ditemukan, lakukanlah langkah 3 poin b.

b. Mengembalikan akhiran "-i", "-an", atau "-kan" yang tadi telah dihilangkan, dan kemudian lakukan langkah 4.

4. Menghapus *derivation prefixes* yaitu "di-", "se-", "ke-", "be-", "me-", "te-", dan "pe-". Apabila pada langkah 3 tadi terdapat *suffix* yang dihapus maka menjalankan langkah 4 poin a, sedangkan apabila tidak terdapat suffix yang dihapus maka menjalankan 4 poin b.

   a. Melakukan pengecekan terhadap pasangan awalan dan akhiran yang tidak diperbolehkan yaitu yang terdapat pada Tabel 2.2. Apabila ditemukan terjadi, maka algoritme berhenti. Akan tetapi, apabila tidak terdapat pasangan awalan dan akhiran yang tidak diperbolehkan maka langkah 4 poin b dijalankan.

Tabel 2.2 Pasangan awalan dan akhiran yang tidak diperbolehkan

Awalan	Akhiran
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan

Sumber : Nazief & Adriani (1996) disitasi dalam Pramudita (2014, p.16)

   b. Menentukan tipe awalan kemudian menghilangkannya sebanyak 3 kali. Algoritme akan dihentikan apabila :

      1) Sudah ditemukan dalam kamus
      2) Jika awalan kedua (awalan yang sedang terdeteksi) sama dengan awalan pertama

      Akan tetapi, apabila masih tidak terdapat dalam kamus maka langkah 5 dijalankan. Langkah-langkah dalam menentukan tipe awalan yaitu sebagai berikut:

      1) Apabila berawalan di-, ke-, atau se-, maka tipe awalannya secara beturut-turut adalah di-, ke-, atau se-.
      2) Apabila berawalan be-, me-, te-, atau –pe, maka diperlukan proses tambahan untuk menentukan tipe awalannya.
      3) Apabila awalannya bukan di-, ke-, se-, be-, me-, te-, atau pe-, maka berhenti.
      4) Apabila tipe awalannya adalah none, maka berhenti, dan apabila bukan none maka dapat ditentukan dengan mengacu seperti contoh pada Tabel 2.3.

Tabel 2.3 Cara menentukan tipe awalan untuk kata yang berawalan te-

Following Character				Tipe Awalan
Set 1	*Set* 2	*Set* 3	*Set* 4	
"-r-"	"-r-"	-	-	*none*
"-r-"	*Vowel*	-	-	ter-luluh
"-r-"	*Not* (*vowel or* "-r")	"-er-"	*Vowel*	ter
"-r-"	*Not* (*vowel or* "-r")	"-er-"	*Not vowel*	ter-
"-r-"	*Not* (*vowel or* "-r")	*Not "-er-"*	-	ter
Not (*vowel or* "-r")	"-er-"	*Vowel*	-	*none*
Not (*vowel or* "-r")	"-er-"	*Not vowel*	-	te

Sumber : Nazief & Adriani (1996) disitasi dalam Pramudita (2014, p.16)

5. Apabila hasil dari langkah 4 masih tidak menghasilkan kecocokan kata antara kata yang diidentifikasi dengan kata-kata dalam kamus, maka proses *recording* dijalankan. Proses *recording* adalah proses penambahan karakter pada awal kata, misalnya adalah kata 'menari', apabila kata tersebut melakukan proses pemenggalan maka, kata yang dihasilkan adalah 'nari'. Kata ini dianggap tidak valid, sehingga proses penambahan karakter dilakukan untuk menghasilkan kata 'tari' (Nazief & Adriani, 1996 disitasi dalam Hariri, Utami, & Amborowati, 2015, p.138).
6. Apabila langkah-langkah tersebut telah dijalankan, akan tetapi tetap tidak berhasil, maka kata asli (kata sebelum dilakukan algoritme ini) dianggap sudah menjadi *root word*.

2.4 *Term weighting*

Term weighting merupakan pembobotan fitur hasil dari proses seleksi fitur yang telah dilakukan. Menurut Indriati & Ridok (2016) pembobotan fitur dapat dilakukan dengan 2 cara yaitu pembobotan dengan menggunakan *term frequency* (TF) dan pembobotan dengan menggunakan *term frequency - invers document frequency* (TF-IDF). TF merupakan jumlah atau frekuensi kemunculan setiap term pada setiap dokumen. Sedangkan IDF merupakan pembobotan fitur dengan melihat banyaknya dokumen yang mengandung fitur tersebut. Pembobotan menggunakan TF dan TF-IDF dihitung menggunakan persamaan 2.1 dan 2.2.

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

Dimana :

$W_{tf_{(t,d)}}$ = bobot *term frequency* pada *term* t di dokumen d

$TF_{(t,d)}$ = *term frequency* (jumlah kemunculan) *term* t pada dokumen d

$$W_{(t,d)} = TF_{(t,d)} \times \log\left(\frac{D}{Dt}\right) \quad (2.2)$$

Dimana :

$W_{(t,d)}$ = bobot *term* t pada dokumen d

$TF_{(t,d)}$ = *term frequency* (jumlah kemunculan) *term* t pada dokumen d

D = jumlah dokumen keseluruhan yang digunakan

Dt = jumlah dokumen yang mengandung term t

Menurut Taufik (2017) rumus 2.2 kurang tepat digunakan ketika terjadi munculnya suatu term di seluruh dokumen, dengan kata lain nilai D sama dengan nilai Dt karena dapat menyebabkan nilai 0 pada hasil perhitungan $\log\left(\frac{D}{Dt}\right)$ pada rumus pembobotan menggunakan TF-IDF, sehingga perlu adanya perubahan rumus perhitungan pembobotan term menggunakan TF-IDF seperti pada persamaan 2.3.

$$W_{(t,d)} = TF_{(t,d)} \times (1 + \log\left(\frac{D}{Dt}\right)) \tag{2.3}$$

Rumus perhitungan pembobotan *term* menggunakan TF-IDF dapat dinormalisasikan kedalam interval [0,1] dengan menggunakan persamaan 2.4 (Intan & Defeng, 2006 disitasi dalam Taufik, 2017, p. 42).

$$W_{(t,d)} = \frac{TF_{(t,d)} \times (1+\log\left(\frac{D}{Dt}\right))}{\sqrt{\sum_{k=1}^{t}(TF)^2 \times (1+\log\left(\frac{D}{Dt}\right))^2}} \tag{2.4}$$

2.5 *Neighbor Weighted K-Nearest Neighbor* (NW-KNN)

Neighbor Weighted K-Nearest Neighbor (NW-KNN) merupakan metode yang digunakan untuk mengatasi terjadinya data yang tidak seimbang, dimana metode ini diusulkan oleh Tan (Patel & Thakur, 2016). Pada dasarnya metode NW-KNN ini sama dengan KNN biasa yaitu dengan memperhitungkan jarak terhadap data uji dengan sejumlah k data latih yang digunakan, dimana jarak yang digunakan dapat berupa *euclidean distance* maupun *cosine similarity* (Indriati & Ridok, 2016). Pada data yang tidak terstuktur yaitu berupa data teks perhitungan jarak yang digunakan adalah *cosine similarity*. Nilai *cosine similarity* didapatkan dari hasil perhitungan menggunakan persamaan 2.5 (Suharno, Fauzi, & Perdana, 2017).

$$Sim_{cosine}(D_i, D_j) = \frac{\sum_{k=1}^{m}(w_{ik} \times w_{jk})}{\sqrt{\sum_{k=1}^{m}(w_{ik})^2 \times \sum_{k=1}^{m}(w_{jk})^2}} \tag{2.5}$$

Dimana m merupakan banyaknya fitur yang digunakan, k merupakan indeks fitur, i merupakan indeks dokumen uji, dan j merupakan indeks dokumen latih.

Pada algoritme NW-KNN dilakukan perhitungan bobot untuk setiap kategori. Bobot tiap kategori pada algoritme NW-KNN didapatkan dengan menggunakan persamaan 2.6 (Indriati & Ridok, 2016).

$$Weight_i = \frac{1}{\left(\frac{Num\left(C_i^d\right)}{Min\left\{Num\left(C_j^d\right)|j=1,\ldots,k\right\}}\right)^{\frac{1}{exp}}} \quad (2.6)$$

Dimana :

$Weight_i$ = bobot pada kategori i

$Num\left(C_i^d\right)$ = jumlah data latih d pada kategori i

$Num\left(C_j^d\right)$ = jumlah data latih d pada kategori j, dimana j merpakan *element* dari k tetangga terdekat

exp = exponen dengan nilai lebih dari 1

Setelah dilakukan pembobotan dilakukan perhitungan *score* seperti pada KNN biasa yaitu dengan menggunakan persamaan 2.7 (Indriati & Ridok, 2016).

$$Score\ (q, C_i) = Weight_i(\sum_{d_j \in KNN(q)} Sim(q, d_j)\ \delta(d_j, C_i)\ ) \quad (2.7)$$

Dimana :

$Score\ (q, C_i)$ = *score* data uji q pada kategori C_i

$Weight_i$ = bobot pada kategori i

$d_j \in KNN(q)$ = data latih d_j pada tetangga terdekat dari data uji q

$Sim(q, d_j)$ = kemiripan data uji q dengan data latih d_j

$\delta(d_j, C_i)$ = $\begin{cases} 1, & d_j \in C_i \\ 0, & d_j \notin C_i \end{cases}$

2.6 *Information gain*

Information gain adalah teknik dalam seleksi fitur dimana dalam metodenya menggunakan metode *scoring* untuk pembobotan dari hasil pendiskretan atribut kontinyu dengan memanfaatkan nilai *entropy* yang maksimum dimana nilai *entropy* yang dihasilkan dilibatkan untuk perhitungan nilai dari *information gain* (Maulida, Suyatno, & Hatta, 2016). Nilai *entropy* yang didapatkan mampu memberikan gambaran mengenai jumlah informasi yang dibutuhkan dalam pengkodean kelas (Abadi, 2013). Penerapan *information gain* ini sendiri dalam mereduksi *term* dalam suatu dokumen dilakukan dengan menghitung jumlah informasi dari kategori tertentu, dimana ada atau tidaknya *term* dalam suatu dokumen sangat dipertimbangkan (Maulida, Suyatno, & Hatta, 2016). Persamaan matematis dari *entropy* dan *information gain* ditunjukkan pada persamaan 2.8 dan 2.9.

$$Entropy(S) = -\sum \frac{|S_i|}{S} \log \frac{S_i}{S} \quad (2.8)$$

$$InfoGain(S, A) = Entropy(S) - \sum_{v\ \epsilon\ Value(A)} \frac{|S_v|}{S} Entropy(S_v) \quad (2.9)$$

Keterangan :

S	= data sampel untuk proses *training*
A	= atribut/fitur
S_v	= data sampel untuk nilai v
v	= nilai dari atribut/fitur A
S_i	= data sampel kriteria tertentu
Value(A)	= himpunan untuk atribut *A*

2.7 *Genetic algorithm*

Genetic algorithm merupakan algoritme optimasi yang menggunakan prinsip-prinsip seleksi alam dan genetika alami yang dapat digunakan pada masalah *mechine learning* (Wati, 2016). Untuk dapat memecahkan masalah dengan menggunakan algoritme genetika maka solusi dari masalah harus dipetakan (*encoding*) menjadi *string chromosom*. Menurut Mahmudy (2015) siklus dari algoritme genetika secara umum yaitu:

1. Inisialisasi populasi awal
2. Reproduksi (*crossover* dan *mutation*)
3. Evaluasi
4. Seleksi

Proses *genetic algorithm* ini akan berhenti apabila terjadi salah satu kondisi dari tiga kondisi berikut (Mahmudy, 2015):

1. Generasi yang terjadi telah sama dengan maksimum generasi yang telah ditentukan sebelumnya.
2. Telah terjadi konvergensi dimana solusi yang dihasilkan tidak menghasilkan solusi yang lebih baik.
3. Generasi yang dilakukan telah mencapai durasi waktu yang telah ditentukan sebelumnya.

2.7.2 Representasi kromosom

Representasi kromosom desebut juga dengan *encoding*. *Encoding* dalam algoritme genetika merupakan sebuah proses pemetaan dari suatu solusi dari permasalahan menjadi *string* kromosom dimana s*tring* kromosom itu sendiri merupakan kumpulan gen yang menggambarkan variabel-variabel keputusan dalam pencarian solusi (Mahmudy, 2013 disitasi dalam Mu'asyaroh & Mahmudy, 2016, p.213). Menurut Imbar & Jayanti (2011, disitasi dalam Mu'asyaroh & Mahmudy, 2016, p.213) macam-macam representasi kromosom dalam algoritme genetika yaitu:

1. Representasi biner, representasi kromosom yang paling sederhana dibandingkan representasi kromosom yang lain karena setiap gen yang ada pada kromosom hanya memiliki 2 kemungkinan nilai yaitu 0 atau 1, seperti 10111011, 10001011, 11101101, dst.

2. Representasi integer, representasi kromosom dimana gen dalam kromosomnya hanya berisi bilangan bulat, seperti 17, 24, 10, dst.
3. Representasi *real code*, representasi kromosom dimana gen dalam kromosomnya berisi bilangan *real*, seperti 27.19, 10.6, 9,8, dst.
4. Representasi permutasi, representasi kromosom yang biasa digunakan dalam permasalahan *scheduling*, TSP (*Travel Salesman Problem*), atau permasalahan yang tidak dapat menggunakan representasi biner, integer, dan *real code*.

2.7.3 Inisialisasi

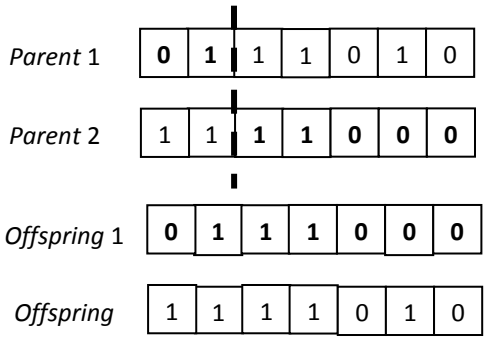
Menurut Mu'asyaroh & Mahmudy (2016) inisialisasi merupakan proses untuk membangkitkan solusi baru secara *random* dalam sejumlah *string* kromosom dalam satu popolasi. Ukuran dari populasi atau yang biasa disebut dengan *popSize* harus ditentukan terlebih dahulu, sedangkan panjang string kromosom atau biasa disebut dengan *stringLen* adalah sejumlah presisi variable solusi yang dicari (Mahmudy, 2013, disitasi dalam Mu'asyaroh & Mahmudy, 2016, p. 213).

2.7.4 *Crossover*

Crossover merupakan proses reproduksi pada algoritme genetika yang melibatkan 2 induk untuk dapat menghasilkan individu baru yang biasa disebut dengan *offspring*. Jumlah *offspring* yang harus dihasilkan dalam proses reproduksi dengan *crossover* adalah sejumlah hasil perkalian antara *crossover rate* dengan *popsize* yang telah ditentukan sebelumnya (Fitri & Mahmudy, 2017). Menurut BariCkly (2013) terdapat beberapa cara *crossover* yaitu *one cut point crossover*, *two cut point crossover*, *crossover uniform* dan *crossover aritmatik*.

1. *One cut point crossover*

One cut point crossover adalah salah satu cara *crossover* yang dilakukan dengan memilih secara acak satu titik potong pada kromosom induk yang akan dilakukan proses *crossover* sehingga titik potong tersebut akan membagi 2 daerah pada masing-masing kromosom induk dan ndividu baru (*offspring*) didapatkan dari hasil pertukaran daerah antar 2 induk (BariCkly, 2013). Proses *one cut point crossover* diilustrasikan pada Gambar 2.1.

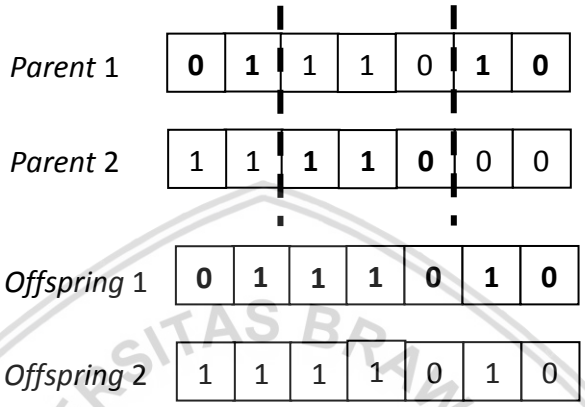


Gambar 2.1 Ilustrasi proses *one cut point crossover*

Sumber : Anon (2016)

2. *Two cut point crossover*

Two cut point crossover merupakan teknik *crossover* yang dilakukan dengan memilih secara acak 2 titik potong dari pada kromosom induk yang akan dilakukan proses *crossover* dan *offspring* (individu baru) didapatkan dari pertukaran darerah yang terletak diantara dua titik potong antar induk yang dilakukan *crossover* (BariCkly, 2013). Proses *two cut point crossover* diilustrasikan pada Gambar 2.2.

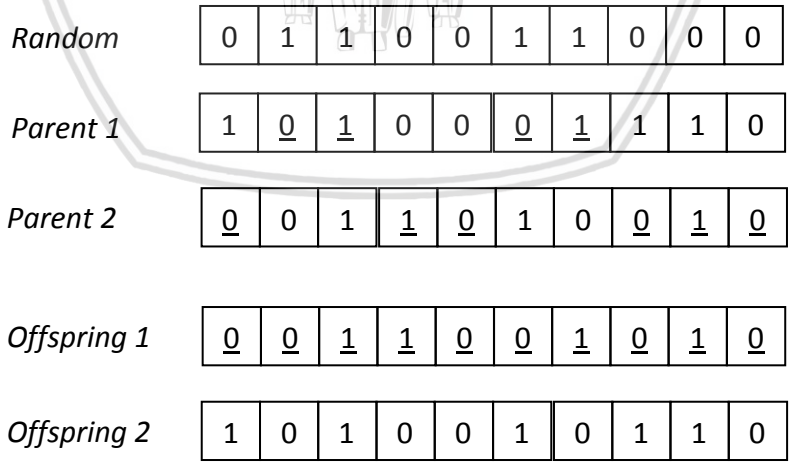


Gambar 2.2 Ilustrasi proses *two cut point crossover*

Sumber : Anon (2016)

3. *Crossover uniform*

Crossover uniform merupakan salah satu teknik *crossover* dimana individu baru (*offspring*) didapatkan dengan memilih nilai biner secara acak dari dua induk yang akan dilakukan proses *crossover* (BariCkly, 2013). Ilustrasi proses *crossover uniform* dapat dilihat pada Gambar 2.3.

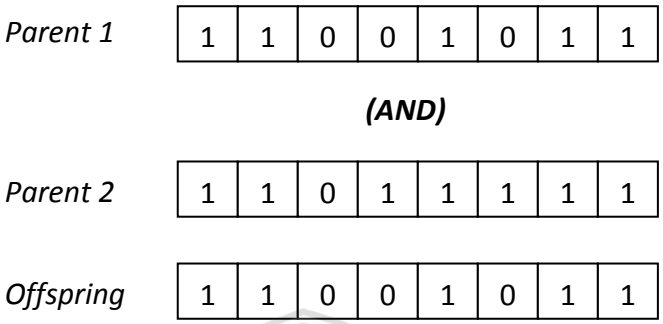


Gambar 2.3 Ilustrasi proses *crossover uniform*

Sumber : Diadaptasi dari Anon (2016)

4. *Crossover* aritmatik

Crossover aritmatik merupakan salah satu teknik *crossover* yang memanfaatkan operasi aritmatik untuk menghasilkan individu baru (*offspring*). Ilustrasi proses *crossover* aritmatik digambarkan pada Gambar 2.4.



Gambar 2.4 Ilustrasi proses *crossover* aritmatik

Sumber : BariCkly (2013)

2.7.5 Mutasi

Mutasi merupakan salah satu cara reproduksi pada algoritme genetika untuk menghasilkan individu baru (*offspring*) dengan cara memilih salah satu *parent* secarak acak pada populasi agar keragaman populasi tetap terjaga dengan jumlah *offspring* yang harus dihasilkan sebesar hasil perkalian *mutation rate* dengan *popsize* dimana nilai *mutation rate* ditentukan di awal. (Mu'asyaroh & Mahmudy, 2016). Menurut BariCkly (2013) terdapat beberapa teknik mutasi berdasarkan representasi kromosomnya seperti berikut:

1. Representasi kromosom biner, melakukan inversi pada bit biner yang terpilih secara acak, seperti 1 1 0 0 1 0 0 1 apabila digit kedua yang terpilih maka *offspring* yang dihasilkan adalah 1 0 0 0 1 0 0 1.
2. Representasi kromosom permutasi, melakukan penukaran terhadap dua gen yang telah terpilih secara acak, seperti 1 2 3 4 5 8 9 7 apabila gen yang terpilih adalah gen kedua dan gen ke-enam *maka offspring* yang dihasilkan adalah 1 8 3 4 5 2 9 7. Selain itu terdapat operator mutasi lain yang telah dibuat untuk representasi permutasi seperti metode *inversion*, *insertion*, *displacement*, dan *reciprocal excange mutation*.
    a. *Inversion mutation*, memilih dua buah posisi dari kromosom induk yang akan dilakukan mutasi secara acak untuk dilakukan inversi *substring* yang berada diantara kedua posisi tersebut.
    b. *Insertion mutation*, memilih secara *random* satu gen dari satu induk yang akan dilakukan mutasi, kemudian dimasukkan pada satu kromosom secara acak.
    c. *Displacement mutation*, memilih secara *random* sekelompo gen kemudian memasukkannya kedalam kromosom secara acak.

d. *Reciprocal exchange mutation*, dua buah posisi dipilih secara *random*, kemudian menukar gen yang berada dalam posisi tersebut.

2.7.6 Evaluasi

Evaluasi merupakan proses perhitungan nilai *fitness* atau kebugaran setiap solusi (Mahmudy, 2015). Nilai *fitness* itu sendiri merupakan nilai yang akan digunakan sebagai acuan keoptimalan suatu solusi, dimana individu yang memiliki nilai fitness lebih tinggi maka individu tersebut lebih mendekati daripada solusi optimal (Uguz, 2011). Dalam penelitian ini, nilai *fitness* setiap individu didapatkan berdasarkan hasil rata-rata nilai *f-measure* dari sekumpulan data uji. *F-measure* dihitung menggunakan persamaan 2.10 (Riany, Fajar, & Lukman, 2016).

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.10)$$

$Precision$ merupakan nilai yang digunakan untuk mengukur tingkat ketepatan sistem, sedangkan $Recall$ merupakan nilai yang digunakan untuk mengukur tingkat keberhasilan sistem. Nilai *precision* dan *recall* didapatkan dengan menggunakan persamaan 2.11 dan 2.12 (Riany, Fajar, & Lukman, 2016).

$$Precision = \frac{TP}{TP + FP} \quad (2.11)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.12)$$

Dimana TP (*True* Positive) adalah jumlah data yang berhasil diklasifikasikan dengan benar, sedangkan FP (*False Positive*) adalah jumlah data yang salah pengklasifikasian tetapi ditemukan, dan FN (*False Negative*) adalah jumlah data yang benar pengklasifikasian tapi tidak ditemukan. Sedangkan rata-rata *precision* dan *recall* dihitung menggunakan persamaan 2.13 dan 2.14 (Uguz, 2011).

$$Rata - rata\ Precision = \frac{\sum_{i=1}^{N} d_i \times Precision_i}{\sum_{i=1}^{N} d_i} \quad (2.13)$$

$$Rata - rata\ Recall = \frac{\sum_{i=1}^{N} d_i \times Recall_i}{\sum_{i=1}^{N} d_i} \quad (2.14)$$

Sehingga dari rata-rata *precision* dan recall, rata-rata *F-measure* bisa didapatkan dengan menggunakan persamaan 2.15 (Uguz, 2011).

$$Rata - rata\ F - measure = \frac{\sum_{i=1}^{N} d_i \times F - measure_i}{\sum_{i=1}^{N} d_i} \quad (2.15)$$

Dimana i pada persamaan-persamaan diatas i menyatakan kategori i, dan d_i merupakan jumlah dokumen pada kategori i, sedangkan N sendiri merupakan jumlah kategori.

2.7.7 Seleksi

Seleksi merupakan proses pemilihan individu dalam populasi sebagai individu bertahan yang akan digunakan pada proses generasi selanjutnya (Mahmudy, 2013 disitasi dalam Sulistiyorini & Mahmudy, 2015, p.5). Menurut

Mu'asyaroh & Mahmudy (2016) terdapat beberapa cara penyeleksian dalam algoritme genetika yang biasa digunakan yaitu:

1. Seleksi *Eltism*

Menurut Mu'asyaroh & Mahmudy (2016) seleksi *eltism* ini menyeleksi individu dalam populasi dengan mengumpulkan seluruh individu baik *parent* maupun *offsping* dalam satu populasi dimana individu yang memiliki nilai *fitness* tertinggi sejumlah populasi menandakan bahwa individu-individu tersebut merupakan individu terbaik dan individu terbaiklah yang akan lolos dan digunakan pada generasi selanjutnya. Proses seleksi menggunakan *eltism* memberi kepastian bahwa individu terbaik akan selalu lolos dan digunakan dalam generasi berikutnya (Mahmudy, 2013 disitasi dalam Mu'asyaroh & Mahmudy, 2016, p.215).

2. *Roulette whell*

Menurut Mu'asyaroh & Mahmudy (2016) *roulette whell* ini bekerja dengan menggunakan nilai peluang atau probabilitas individu berdasarkan nilai kebugaran (*fitness*) yang telah dihitung sebelumnya dimana nilai probabilitas tersebut nantinya akan menghasilkan nilai probabilitas kumulatif setiap individu dalam satu populasi untuk digunakan dalam proses penyeleksian individu. Menurut Mahmudy (2013 disitasi dalam Mu'asyaroh & Mahmudy, 2016, p.215) tahapan proses penyeleksian menggunakan *roulette whell* adalah menghitung total *fitness* dari seluruh individu baik *parent* maupun *offspring* kemudian menghitung peluang (probabilitas) seleksi dari masing-masing individu dan yang terakhir adalah menghitung probabilitas kumulatif setiap individu setelah itu individu dipilih secara *random* berdasarkan probabilitas kumulatif yang telah dihitung sebelumnya.

3. *Binary Tournament Selection*

Menurut Mu'asyaroh & Mahmudy (2016) *binary tournament selection* ini bekerja dengan membandingkan *fitness* setiap individu dalam populasi dengan individu yang dipilih secara *random* dimana individu yang memiliki nilai *fitness* lebih tinggi akan lolos pada generasi berikutnya. Sebagai contoh adalah terdapat P1 sebagai individu yang akan dibandingkan, dan P3 adalah individu pembanding yang diperoleh secara acak. P1 memiliki *fitness* sebesar 14.778, sedangkan P3 memiliki *fitness* sebesar 14.897 maka P3 akan digunakan pada generasi selanjutnya dikarenakan memiliki *fitness* yang lebih tinggi dibandingkan dengan P1 (Mu'asyaroh & Mahmudy, 2016).

4. *Replacement Selection*

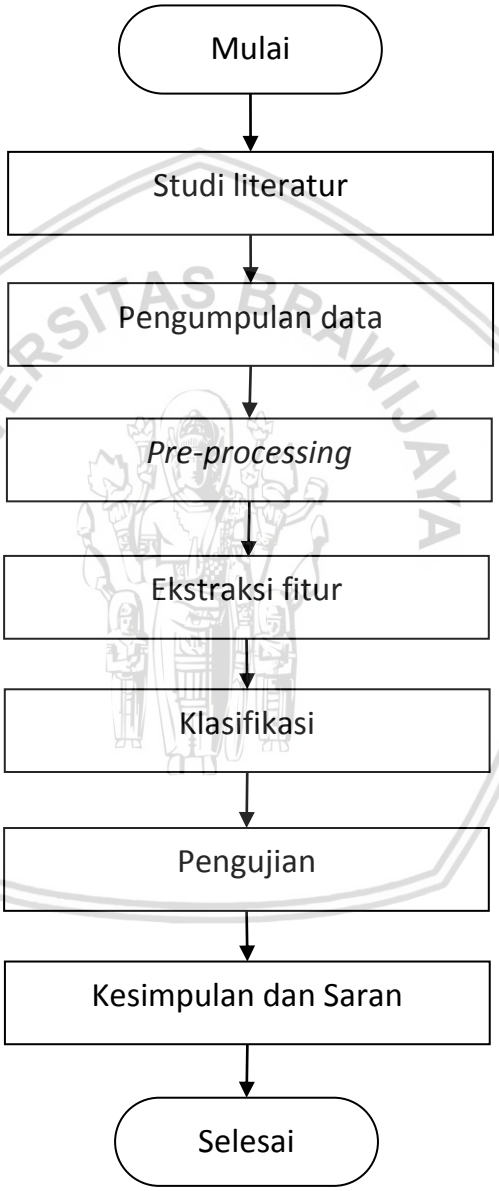
Menurut Mu'asyaroh & Mahmudy (2016) *replacement selection* bekerja dengan membandingkan *parent* dengan *offspring* dimana apabila *fitness* yang dimiliki oleh *offspring* lebih besar dari *parent* maka *offspring* akan menggantikan *parent* dan akan diproses pada generasi selanjutnya. Aturan-aturan yang dimiliki oleh metode iniberdasarkan reproduksinya yaitu (Mahmudy, 2013 disitasi dalam Mu'asyaroh & Mahmudy, 2016, p.216):

a. Pada proses mutasi, *offspring* yang dihasilkan akan menggantikan posisi induknya dalam populasi apabila nilai *fitness* yang dimiliki *offspring* lebih besar dari induknya.

b. Pada proses *crossover*, satu *offspring* yang dihasilkan akan menggantikan posisi satu dari 2 induknya yang memiliki nilai *fitness* terkecil dengan catatan nilai *fitness* yang dimiliki *offspring* tersebut lebih besar dari induk yang memiliki *fitness* terkecil.

BAB 3 METODOLOGI

Pengklasifikasian otomatis dari aduan masyarakat ini akan dibuat dalam bahasa pemrograman Java dengan menggunakan aplikasi NetBeans. Tahapan penelitian yang dilakukan peneliti dalam pengklasifikasian aduan masyarakat pada SAMBAT *Online* Kota Malang menggunakan algoritme NW-KNN (*Neighbor Weighted K-Nearest Neighbor*) dengan seleksi fitur *information gain* dan *genetic algorithm* digambarkan pada Gambar 3.1.



Gambar 3.1 Tahap metodologi penelitian

Gambar 3.1 adalah tahap yang akan dijalankan peneliti sebagai metodologi dalam penelitian mulai dari tahap studi literatur, kemudian pengumpulan data, *pre-processing*, *feature extraction*, *classification* dan *evaluation* hingga pada

pembuatan laporan yang menyertakan kesimpulan dan saran dari hasil penelitian.

3.1 Studi literatur

Sebelum masuk tahap pemrosesan data, tentunya studi literatur perlu dilaksanakan untuk memperkuat dasar dari penelitian yang akan dilakukan. Literatur yang diperlukan adalah literatur yang berkaitan dengan:

1. Klasifikasi dokumen
2. Algoritme NW-KNN
3. Seleksi fitur *information gain* dan *genetic algorithm*

Literatur-literatur tersebut berupa jurnal, *paper* dan *internet*.

3.2 Pengumpulan data

Data yang akan digunakan dalam proses penelitian ini adalah data sekunder dan data primer yang berupa aduan masyarakat pada SAMBAT *Online* Kota Malang disimpan dalam *file* dengan ekstensi .txt. Data ini dikatakan primer dikarenakan data ini diperoleh melalui media perantara yaitu melalui halaman *website* sambat.malangkota.go.id dan dikatakan sekunder karena data yang akan digunakan juga diambil dari peneliti sebelumnya yaitu Prasanti, Fauzi, & Furqon (2018) untuk mendapatkan data tahun lalu. Jumlah data aduan masyarakat yang digunakan adalah sebesar 300 aduan untuk data tidak seimbang dan 114 untuk data seimbang, dimana 70% diantaranya digunakan sebagai data latih dan 30% diantaranya sebagai data uji. Data tersebut terdiri dari 3 SKPD yaitu SKPD Dinas Perhubungan (Dishub) Kota Malang, SKPD Dinas Pekerjaan Umum, Perumahan, dan Pengawasan Bangunan (DPUPPB) Kota Malang, dan SKPD Dinas Kebersihan dan Pertamanan (DKP) Kota Malang. Pada data tidak seimbang terdapat 38 aduan yang ditujukan pada SKPD DKP, 111 aduan pada SKPD DPUPPB dan 151 aduan pada SKPD Dishub. Sedangkan pada data seimbang setiap SKPD baik DKP, DPUPPB, maupun Dishub masing-masing terdapat 38 aduan. Contoh data yang akan digunakan dalam proses pengklasifikasian dapat dilihat pada Tabel 3.1.

Table 3.1 Contoh aduan masyarakat

No	Aduan Masyarakat	SKPD Terkait
1	Mohon traffic light penyeberangan diperbaiki karena tidak berfungsi	DKP
2	Di Pasar Gadang banyak jalan berlubang	DPUPPB
3	Jukir tidak pernah memberikan karcis secara sukarela di SEMUA titik parkir kota Malang	Dishub

3.3 Pre-processing

Pre-processing merupakan proses untuk menghilangkan noise agar dapat diambil fitur yang menarik dari keseluruhan data. *Pre-processing* ini dilakukan terhadap seluruh data yang akan di gunakan dalam penelitian, dimana dalam prosesnya setiap data yang akan digunakan diubah dalam bentuk huruf kecil (*lowercase*) dan semua *stopword* akandihapus. Setelah itu, akan dilakukan pemecahan kalimat menjadi *token-token*, dimana setiap *token* tersebut akan dilakukan proses *stemming* untuk dihasilkan *stem* dari token tadi.

3.4 Ekstraksi fitur

Proses *feature extraction* hanya melibatkan data latih. *Feature extraction* ini digunakan untuk mendapatkan ciri khas data dari segi fitur yaitu dengan melakukan penyeleksian fitur dengan menggunakan teknik *information gain* dan *genetic algorithm*. Namun, sebelum dilakukan proses *feature extraction*, diperlukan adanya pembobotan *term* untuk mendapatkan frekuensi dari masing masing kata.

3.5 Klasifikasi

Pada tahap ini akan dilakukan proses pengelompokan data berdasarkan SKPD terkait menggunakan perpadudan metode *Neighbor Weighted* dan *K-Nearest Neighbor* (NW-KNN). Dalam proses klasifikasi ini menggunakan fitur hasil dari proses *feature extraction* untuk menetapkan data pada kelas tertentu yang berupa SKPD terkait.

3.6 Pengujian

Pada tahap ini akan dilakukan evaluasi terhadap tingkat keberhasilan sistem. Evaluasi yang akan dilakukan pada penelitian ini menggunakan metode pengujian kuantitatif untuk menghitung keoptimalan metode seleksi fitur *information gain* dan *genetic algorithm* dalam membantu metode NW-KNN dalam melakukan pengklasifikasian aduan masyarakat pada SAMBAT *Online* Kota Malang. Keoptimalan metode *information gain* dapat dilihat dari sedikitnya jumlah ftur yang digunakan akan tetapi hasil akurasi yang dihasilkan tinggi.

Fitur yang dihasilkan dari proses *information gain* akan dilakukan optimasi fitur yang digunakan dengan menggunakan *genetic algorithm*. Keoptimalan *genetic algorithm* dalam menyeleksi fitur yang dihasilkan dari perhitungan *information gain* dilihat dari nilai *fitness* yang dihasilkan. Nilai fitness yang digunakan sebagai acuan dalam keoptimalan hasil seleksi merukan nilai *f-measure* dari proses pengklasifikasian. Dalam proses *genetic algorithm* akan diuji nilai *crossover rate, mutation rate*, jumlah populasi yang optimal untuk digunakan dalam proses menyeleksi fitur dengan *genetic algorithm*. Tingkat akurasi yang dihasilkan dengan menggunakan hasil fitur dari proses *feature selection* didapatkan dengan melakukan pengujian *data testing* dengan sejumlah

k tetangga terdekat. Tingkat akurasi yang dihasilkan dapat dilihat dari nilai *f-measure* yang dihasilkan, dimana *f-measure* yang dihasilkan memperhitungkan nilai *precision* dan *recall* dari hasil proses klasifikasi. *Precision* digunakan untuk mengukur tingkat ketepatan hasil klasifikasi dan *recall* digunakan untuk mengukur tingkat keberhasilan sistem dalam pengklasifikasian. Sedangkan keseimbangan dari *precision* dan *recall* dilihat dari nilai *f-measure*. Pengujian pengaruh persentase jumlah fitur juga akan dilakukan pada penelitian ini dengan persentase jumlah fitur yang digunakan adalah mulai 5% hingga 100% dari jumlah fitur dengan kelipatan 5 pada proses pengklasifikasian. Hasil pengujian tersebut akan disimpan pada Tabel 3.2.

Table 3.2 Rancangan pengujian persentase jumlah fitur pada *information gain*

Persentase Jumlah Fitur (%)	Banyak Fitur	Rata-rata		
		Precision	*Recall*	*F-Measure*
5				
10				
15				
20				
25				
30				
35				
40				
45				
50				
55				
60				
65				
70				
75				
80				
85				
90				
95				
100				

Kemudian pengujian kedua adalah pengujian pengaruh metode NW-KNN dengan menguji nilai K, dimana nilai K yang akan diuji adalah 3, 6, 9, 12, 15, 18, dan 21. Pengujian ini dilakukan dua kali dengan menggunakan metode yang berbeda yaitu NW-KNN dan KNN standar. Nilai rata-rata *precision, recall* dan *f-measure*

yang dihasilkan akan disimpan kedalam tabel hasil pengujian. Rancangan tabel pengujian nilai K ditunjukkan pada Tabel 3.3.

Table 3.3 Rancangan hasil pengujian nilai K

Nilai K	Rata-rata		
	Precision	***Recall***	***F-Measure***
3			
6			
9			
12			
15			
18			
21			

Selanjutnya adalah pengujian ukuran populasi dengan menggunakan nilai k dan persentase jumlah fitur terbaik dari hasil pengujian sebelumnya. Pada pengujian ukuran populasi, ukuran populasi yang akan diujikan adalah 10 hingga 50 populais dengan rentang 10. Pengujian setiap ukuran populasi dilakukan sebanyak 3 kali percobaan yang kemudian akan dilakukan perhitungan rata-rata. Setiap hasil dari proses pengujian akan disimpan pada tabel dengan rancangan seperti pada Tabel 3.4.

Table 3.4 Rancangan hasil pengujian ukuran populasi pada *genetic algorithm*

Popsize	**Percobaan ke-**			**Rata-rata *fitness***
	1	**2**	**3**	
10				
20				
30				
40				
50				

Selanjutnya adalah pengujian nilai *crossover rate* (CR) dan *mutation rate* (MR) dengan menggunakan nilai k dan persentase jumlah fitur serta ukuran populasi terbaik dari hasil pengujian yang talah dilakukan sebelumnya. Nilai CR dan MR yang akan dilakukan pengujian adalah 0.1 dan 0.9, 0.2 dan 0.8, 0.3 dan 0.7, 0.4 dan 0.6, 0.5 dan 0.5, 0.6 dan 0.4, 0.7 dan 0.3, 0.8 dan 0.2 serta 0.9 dan 0.1. Setiap nilai tersebut dilakukan sebanyak 3 kali percobaan yang akan dilakukan perhitungan rata-rata. Hasil pengujian setiap prosesnya akan disimpan pada tabel dengan rancangan tabel seperti padab Tabel 3.5.

Table 3.5 Rancangan pengujian nilai CR dan MR pada *genetic algorithm*

CR	MR	Percobaan ke-			Rata-rata *fitness*
		1	2	3	
0,1	0,9				
0,2	0,8				
0,3	0,7				
0,4	0,6				
0,5	0,5				
0,6	0,4				
0,7	0,3				
0,8	0,2				
0,9	0,1				

Pengujian terakhir yang akan dilakukan adalah pengujian banyaknya generasi yang dilakukan dengan menggunakan nilai k dan persentase jumlah fitur serta ukuran populasi, nilai CR dan MR terbaik dari hasil pengujian yang telah dilakukan sebelumnya. Pada pengujian generasi ini nilai maksimum generasi yang digunakan adalah sebanyak 150 generasi. Pengujian ini dilakukan sebanyak 3 kali percobaan untuk menguji konvergensi sehingga dapat didapatkan banyaknya generasi optimal sebagai hasil terbaik. Rancangan pengujian banyaknya generasi ditunjukkan pada Tabel 3.6.

Table 3.6 Rancangan pengujian banyaknya generasi pada *genetic algorithm*

Banyaknya generasi	Percobaan ke-			Rata-rata *fitness*
	1	2	3	
1				
2				
3				
4				
⋮				
146				
147				
148				
149				
150				

3.7 Kesimpulan dan saran

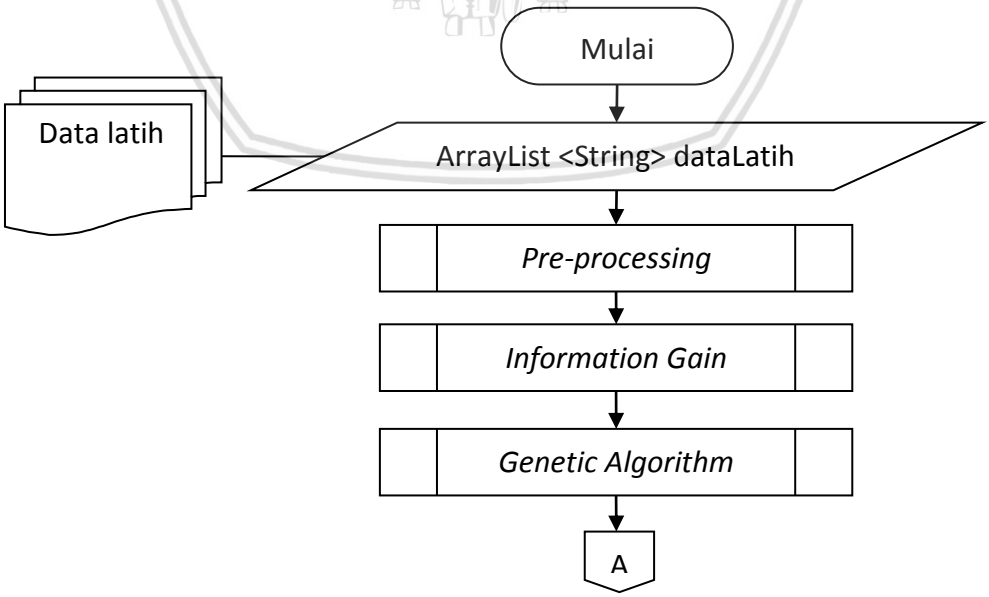
Pengambilan kesimpulan akan dilakukan ketika semua tahapan pada metodologi penelitian telah dilaksanakan dari tahap perancangan, implementasi hingga pengujian sistem, dimana pengambilann kesimpulan ini digunakan untuk menjawab setiap poin rumusan masalah dari masalah yang telah dipaparkan sebelumnya. Tahap ini juga melakukan pemberian saran untuk penelitian selanjutnya mengenai hasil penelitian, sehingga penelitian ini dapat diperbaiki setiap kekurangannya.

BAB 4 PERANCANGAN

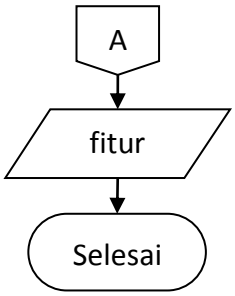
4.1 Gambaran umum sistem

Perancangan merupakan kegiatan yang menggambarkan secara detail mengenai bagaimana sistem berjalan. Perancangan adalah bagian dari sebuah perencanaan yang diperlukan untuk memastikan setiap langkah yang akan dilakukan dapat mencapai tujuan dari dilakukannya penelitian ini. Selain itu perancangan dilakukan agar dapat memecahkan rumusan masalah dari penelitian ini, maka setiap langkah yang akan dilakukan harus terencana dengan baik. Berdasarkan tinjauan pustaka pada Bab 2, langkah awal dari penilitian ini yang harus dilakukan adalah mengambil ciri (fitur) dari data yang telah terkumpul. Fitur yang telah terkumpul akan diseleksi berdasarkan pentingnya fitur tersebut terhadap hasil klasifikasi. Acuan dari fitur tersebut dianggap penting adalah dari besarnya hasil perhitungan *information gain* fitur tersebut. Sejumlah fitur diambil berdasarkan nilai *information gain* yang tinggi dan akan dilakukan kombinasi fitur. Kombinasi fitur ini dilakukan menggunakan *genetic algorithm* untuk mendapatkan hasil akurasi yang tinggi dengan waktu yang lebih singkat daripada dilakukannya kombinasi fitur secara manual.

Berdasarkan uraian tersebut, rancangan yang diperlukan pada penelitian ini adalah rancangan dalam pengambilan fitur dan rancangan pengujian. Rancangan pengambilan fitur diperlukan untuk direncanakan karena fitur menjadi hal yang sangat dibutuhkan dalam pengklasifikasian. Setiap dokumen yang akan dilakukan pengklasifikasian harus diambil fiturnya sebagai karakteristik dari dokumen tersebut. Sedangkan rancangan pengujian perlu dilakukan untuk mendapatkan parameter terbaik dan menghasilkan akurasi yang tinggi. Gambaran umum sistem dalam melakukan pengambilan fitur digambarkan pada Gambar 4.1.



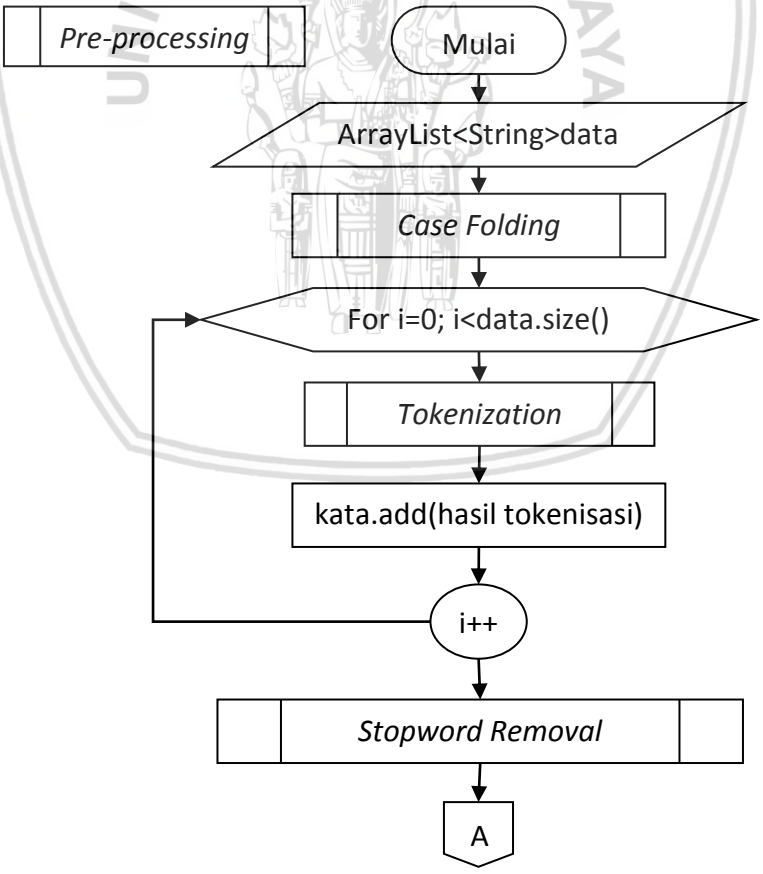
Gambar 4.1 Gambaran umum sistem



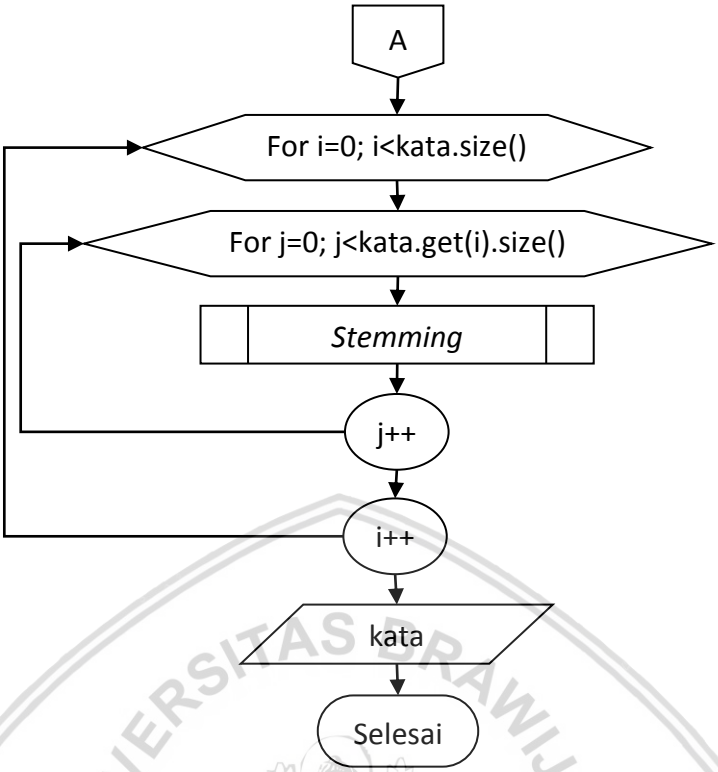
Gambar 4.1 Gambaran umum sistem

4.1.1 Pre-processing

Pre-processing merupakan langkah pertama yang harus dijalankan untuk mengambil fitur yang terstruktur dari dokumen. Berdasarkan tinjauan pustaka pada bab 2, pada *pre-processing* ini *data input* dari *pre-processing* dilakukan proses *case folding*. Kemudian masing-masing dokumen dari *data input* dilakukan proses tokenisasi dimana kumpulan hasil tokenisasi tersebut dihilangkan *stopword* (kata yang tidak penting). Setelah dilakukan penghaspusan *stopword*, proses *stemming* dilakukan terhadap setiap kata di setiap dokumen untuk mengubah kata tersebut menjadi kata dasar. Alur dari proses *pre-processing* dijelaskan pada Gambar 4.2.



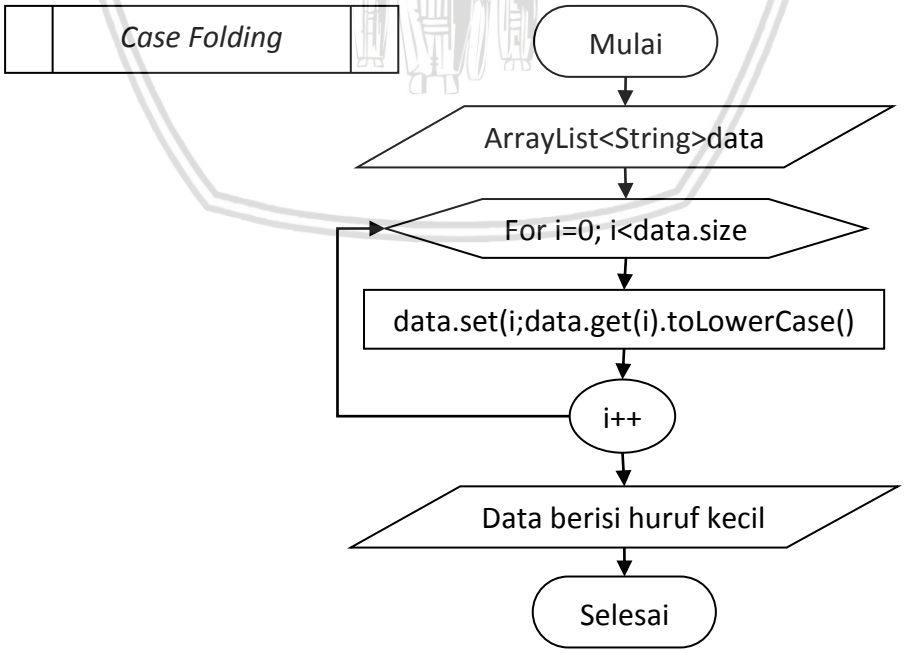
Gambar 4.2 Diagram alir *pre-processing*



Gambar 4.2 Diagram alir *pre-processing*

1. *Case folding*

Case folding merupakan fungsi yang berisi proses pengubahan kalimat menjadi bentuk *lowercase* atau huruf kecil. Alur dari proses *case folding* dijelaskan pada Gambar 4.3.

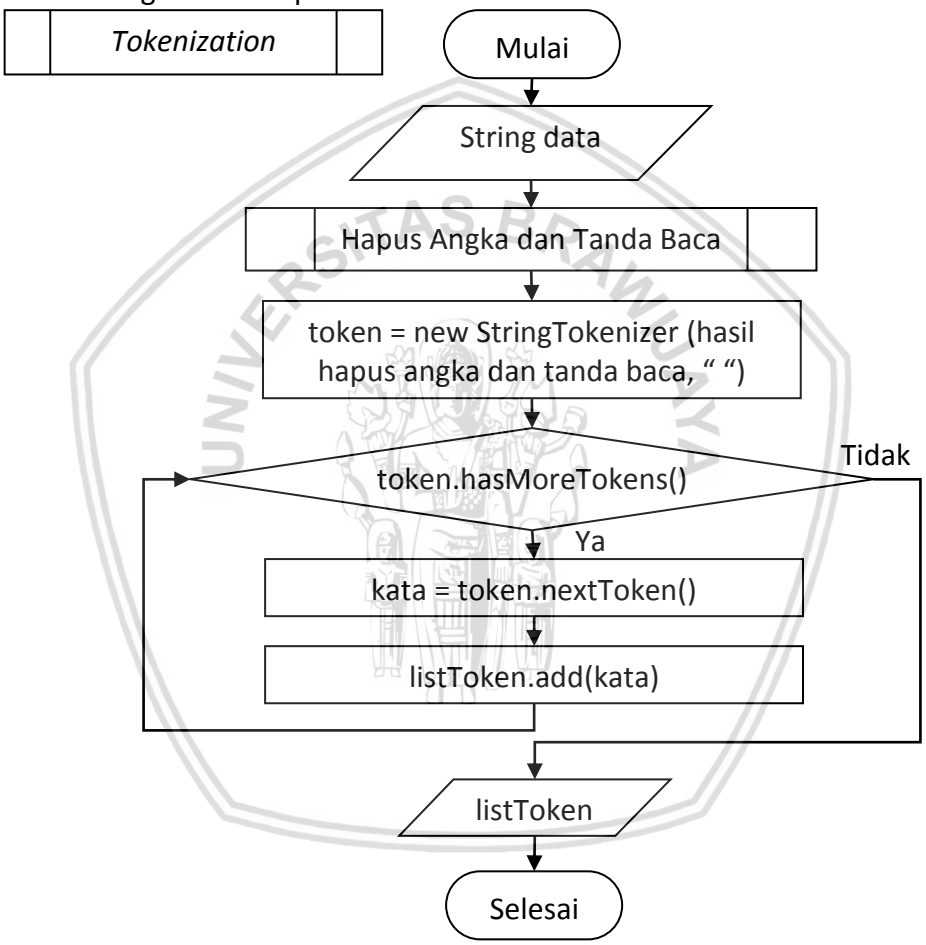


Gambar 4.3 Diagram alir *case folding*

Berdasarkan Gambar 4.3, pada proses *case folding* menggungnakan fungsi .toLowerCase() yang diterapkan pada seluruh data input. toLowerCase() merupakan sebuah fungsi yang digunakan untuk mengubah semua huruf maupun karakter menjadi huruf kecil semua.

1. *Tokenization*

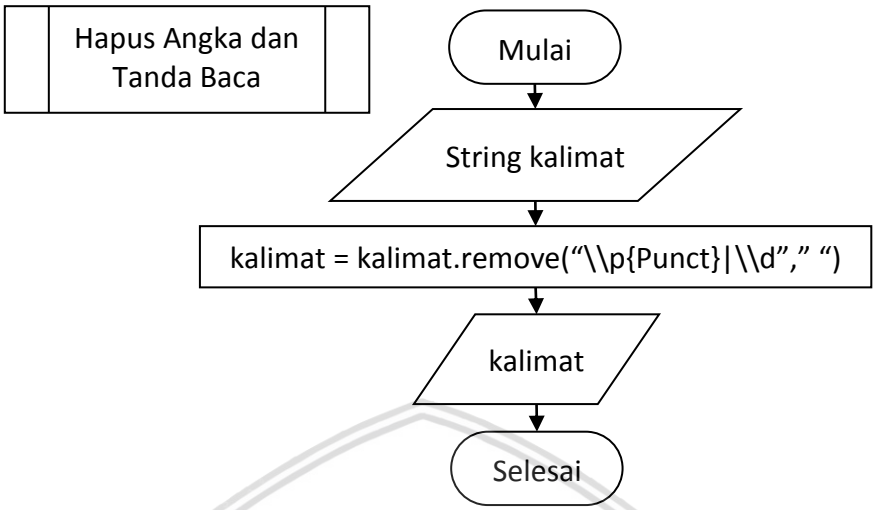
Tokenization merupakan proses untuk mendapatkan token. Token dalam hal ini merupakan kata dari *data input* yang berupa kalimat. Kata yang diambil atau yang menjadi keluaran dari proses tokenisasi adalah seluruh kata tidak termasuk angka maupun tanda baca yang dipisahkan dengan tanda spasi. Alur proses *tokenization* digambarkan pada Gambar 4.4.



Gambar 4.4 Diagram alir *tokenization*

Pada gambar tersebut proses tokenisasi dilakukan dengan memanfaatkan *class* StringTokenizer. StringTokenizer merupakan kelas yang digunakan untuk memanipulasi karakter yang tersimpan dalam bentuk String dengan memecahnya menjadi beberapa bagian karakter sesuai dengan *delimiter* yang digunakan. Semua kata/*token* yang telah tersimpan dalam objek *token* akan disimpan kembali ke dalam *array list*, dimana apabila masih terdapat *token* pada objek *token* maka *token* yang sedang diidentifikasi menggunakan fungsi nextToken() sebagai *pointer* pada obyek token akan disimpan pada penyimpanan yang baru yaitu dalam *array list*. Selain itu pada proses tokenisasi terdapat

proses penghapusan angka dan tanda baca. Alur penghapusan angka dan tanda baca ini digambarkan pada Gambar 4.5.

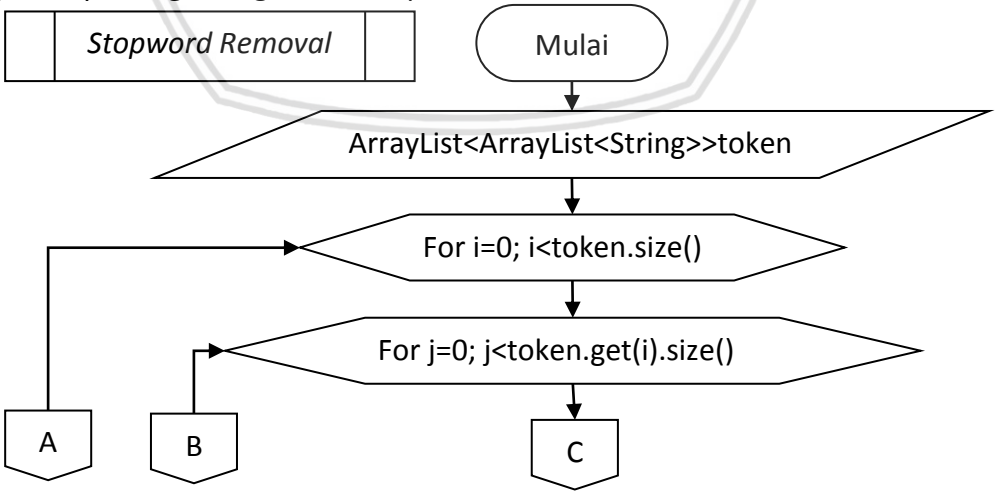


Gambar 4.5 Diagram alir penghapusan angka dan tanda baca

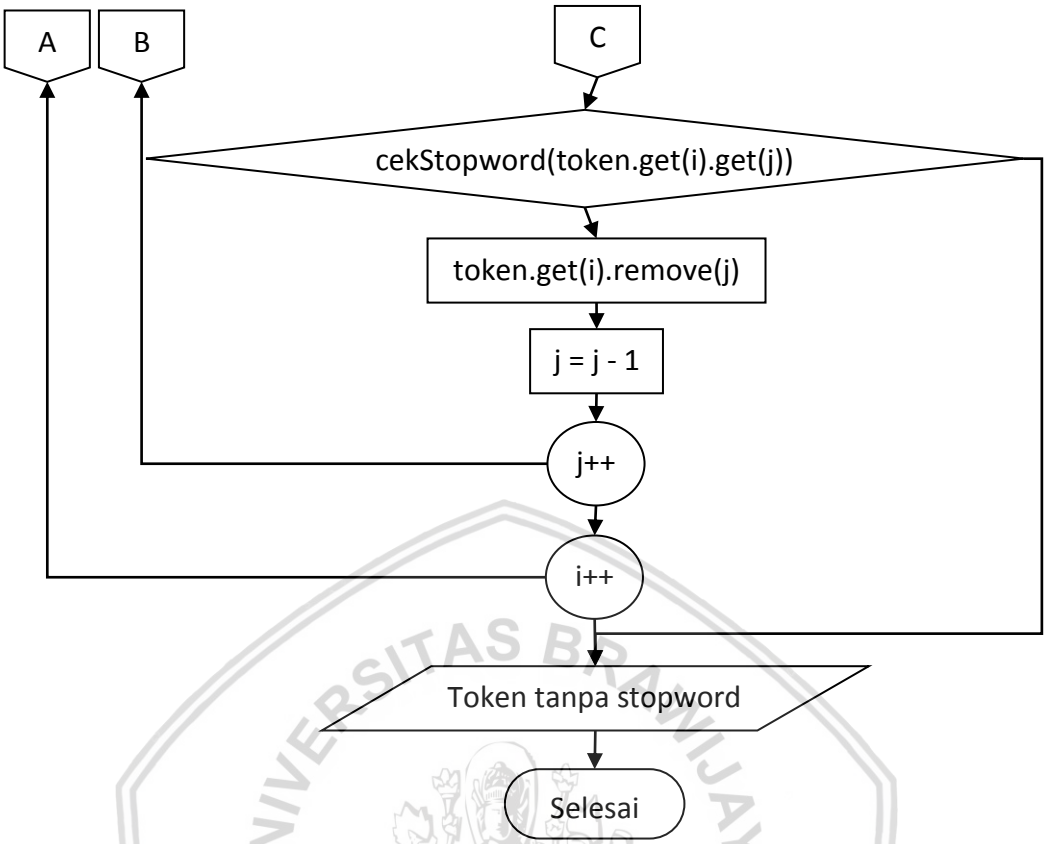
Pada Gambar 4.5, penghapusan angka dan tanda baca dilakukan dengan menggunakan fungsi remove() dengan *regular expression* \\p{Punct}|\\d dan angka dan tanda baca akan diganti dengan spasi. Simbol \\p{Punct} pada *regular expresion* merupakan simbol untuk pengecekan kesesuaian karakter yang berupa tanda baca dimana punct itu sendiri adalah kependekan dari *punctuation*. Sedangkan simbol //d pada *regular expression* merupakan simbol untuk pengecekan kesesuaian karakter yang berupa *digit* angka.

2. *Stopword removal*

Stopword removal merupakan langkah untuk menghilangkan (menghapus) *stopword* (kata/token yang dianggap tidak penting). Proses penghapusan kata yang tidak penting ini digambarkan pada Gambar 4.6.

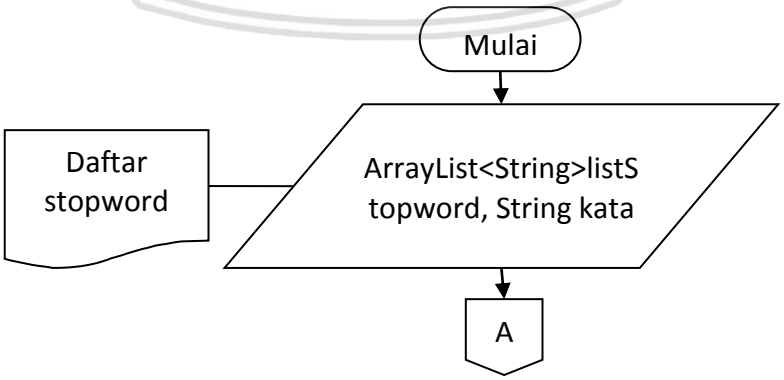


Gambar 4.6 Diagram alir *stopword removal*

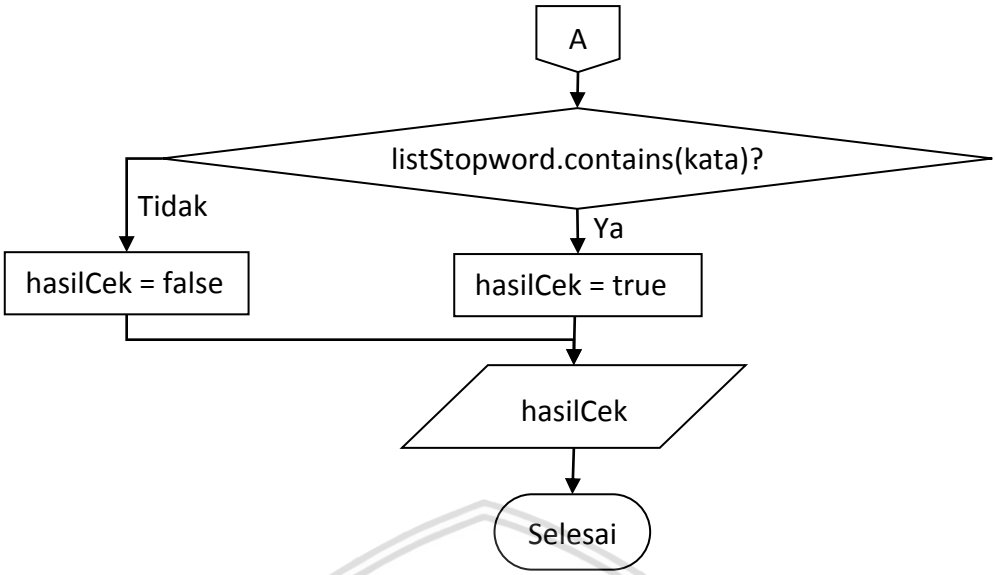


Gambar 4.6 Diagram alir *stopword removal*

Pada gambar 4.6, proses penghapusan *stopword* dilakukan dengan mengecek setiap token apakah token tersebut termasuk *stopword* atau bukan. Apabila token tersebut merupakan *stopword* maka token tersebut akan dihapus dari daftar token dengan menggunakan fungsi remove. Alur proses pengecekan yang terjadi pada saat proses penghapusan *stopword* berjalan digambarkan pada Gambar 4.7. Pada gambar tersebut, pengecekan dilakukan dengan menggunakan fungsi *contains* dimana akan bernilai benar apabila kata (token) tersebut merupakan *stopword*, apabila sebaliknya maka akan bernilai salah.



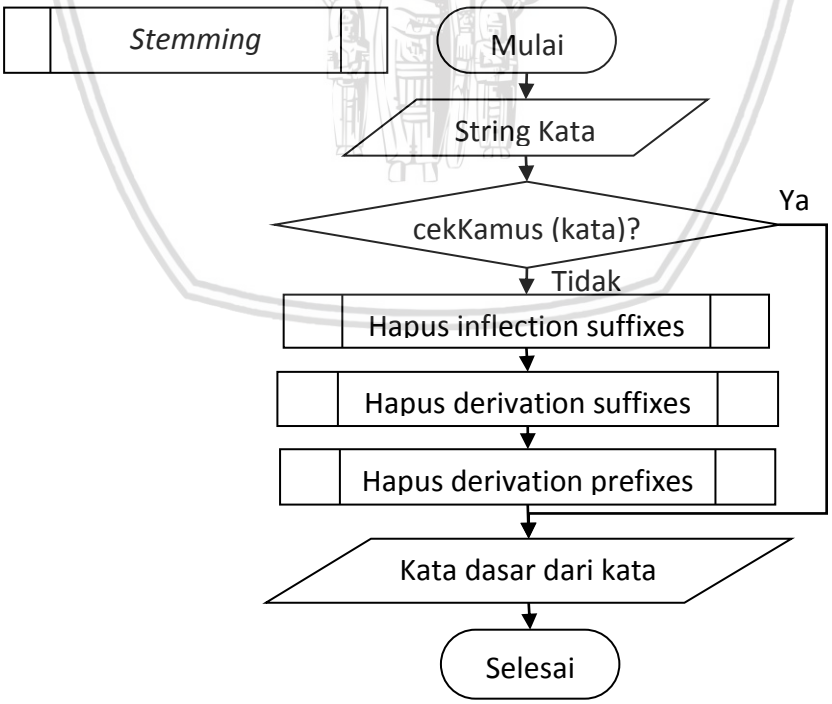
Gambar 4.7 Diagram alir pengecekan *stopword*



Gambar 4.7 Diagram alir pengecekan *stopword*

3. *Stemming*

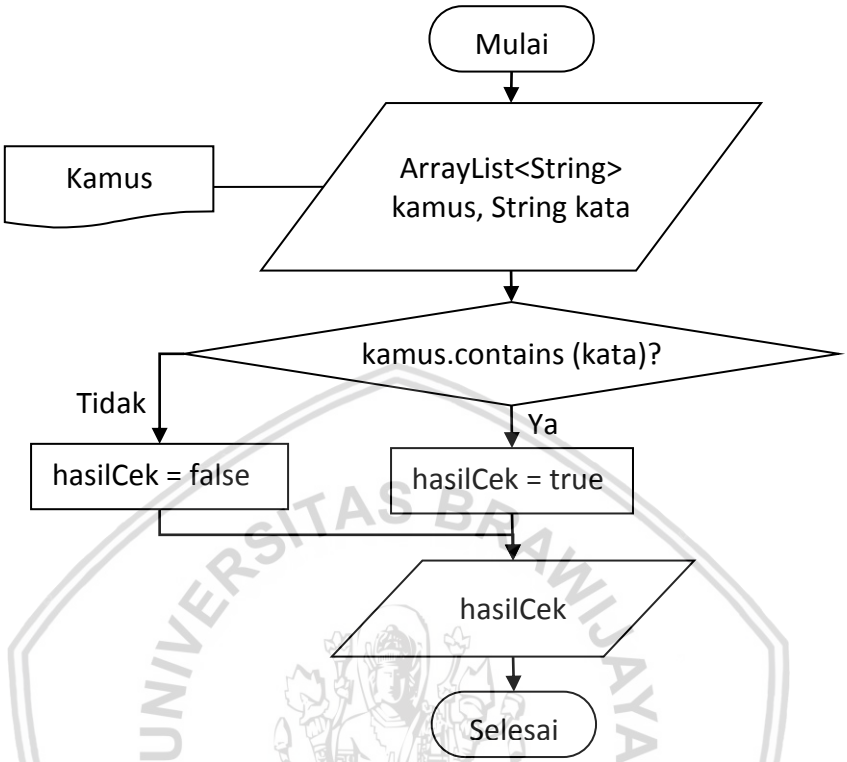
Stemming merupakan proses pengubahan kata menjadi bentuk dasarnya dengan menghapus imbuhan baik berupa awalan maupun akhiran. Setiap proses penghapusan yang dijalankan akan mengeceknya pada kamus Bahasa Indonesia. Apabila kata hasil penghapusan terdapat pada kamus, maka kata tersebut dianggap sebagai kata dasar dan proses *stemming* selesai. Diagram alir dari proses *stemming* digambarkan pada Gambar 4.8.



Gambar 4.8 Diagram alir *stemming*

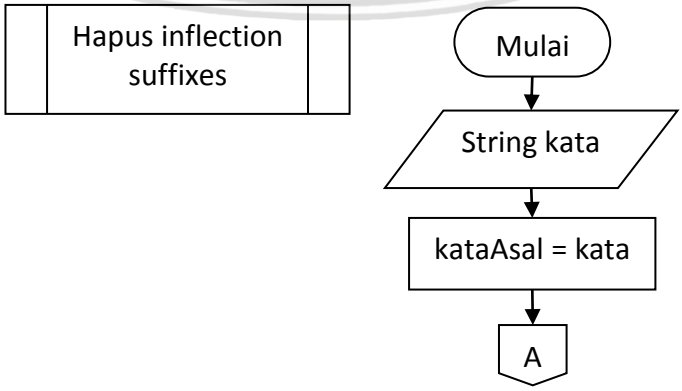
Berdasarkan diagram alir proses *stemming* pada Gambar 4.9, terdapat kondisi pengecekan kamus. Alur proses pengecekan kamus tersebut

digambarkan secara detail pada diagram alir Gambar 4.9. Seperti pada pengecekan *stopword* pada Gambar 4.7, pengecekan kamus juga menggunakan fungsi *contains* dalam pengecekannya.

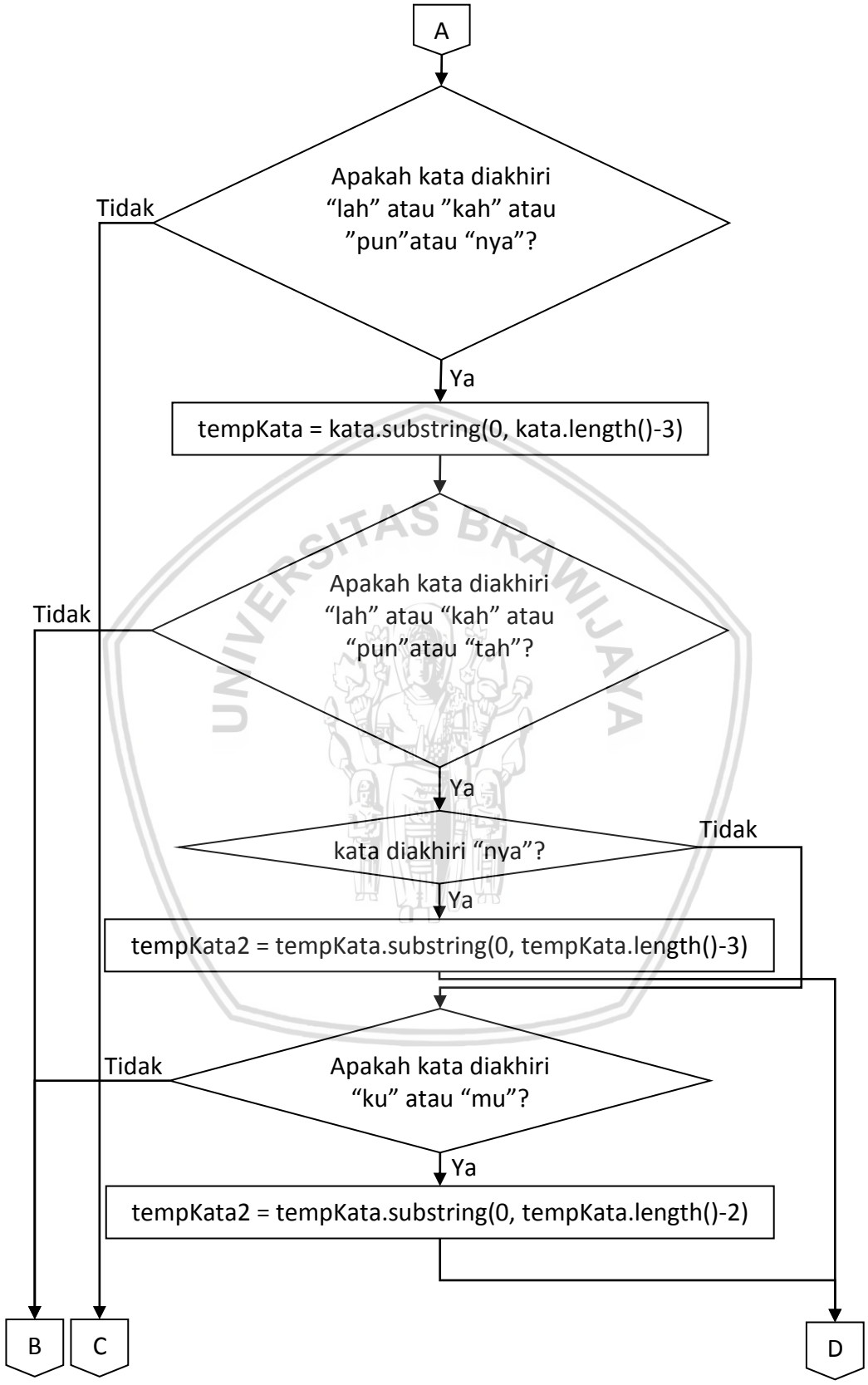


Gambar 4.9 Diagram alir pengecekan pada kamus

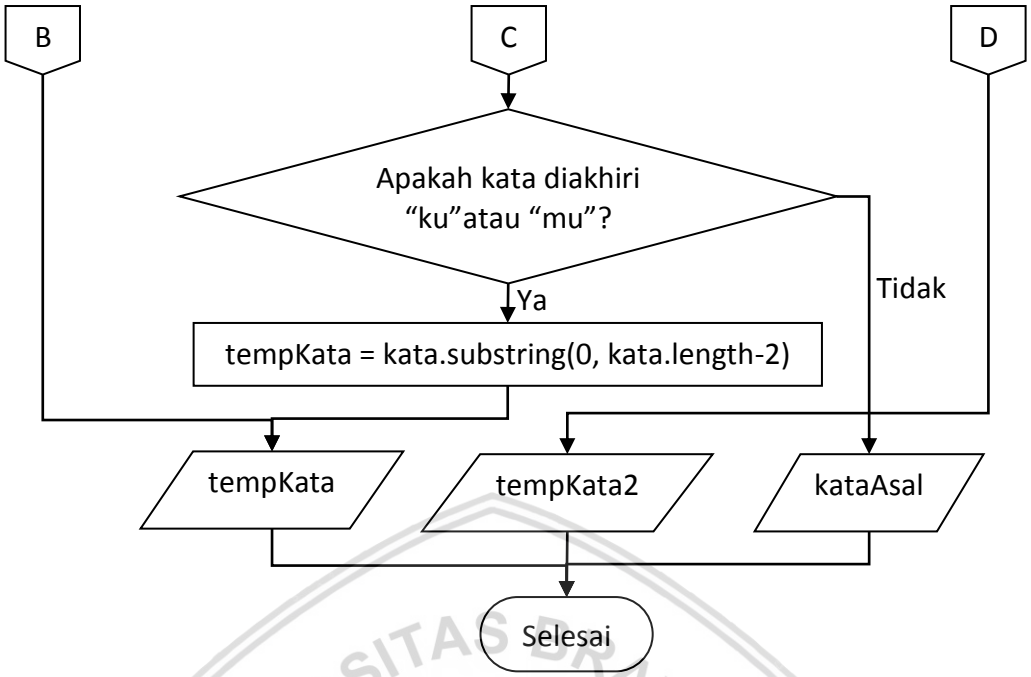
Selain itu pada proses *stemming* juga terdapat proses penghapusan *inflection suffixes*. *Inflection suffixes* yang akan dihapus adalah –lah, -kah, -tah, -pun, -ku, -mu, dan –nya. Imbuhan-imbuhan ini secara otomatis akan dihilangkan tanpa melakukan pengecekan terhadap kamus setelah penghapusan dilakukan. Diagram alir penghapusan *inflection suffixes* ini digambarkan pada Gambar 4.10. pada gambar tersebut, penghapusan imbuhan dilakukan dengan menggunakan fungsi *substring*.



Gambar 4.10 Diagram alir penghapusan *inflection suffixes*

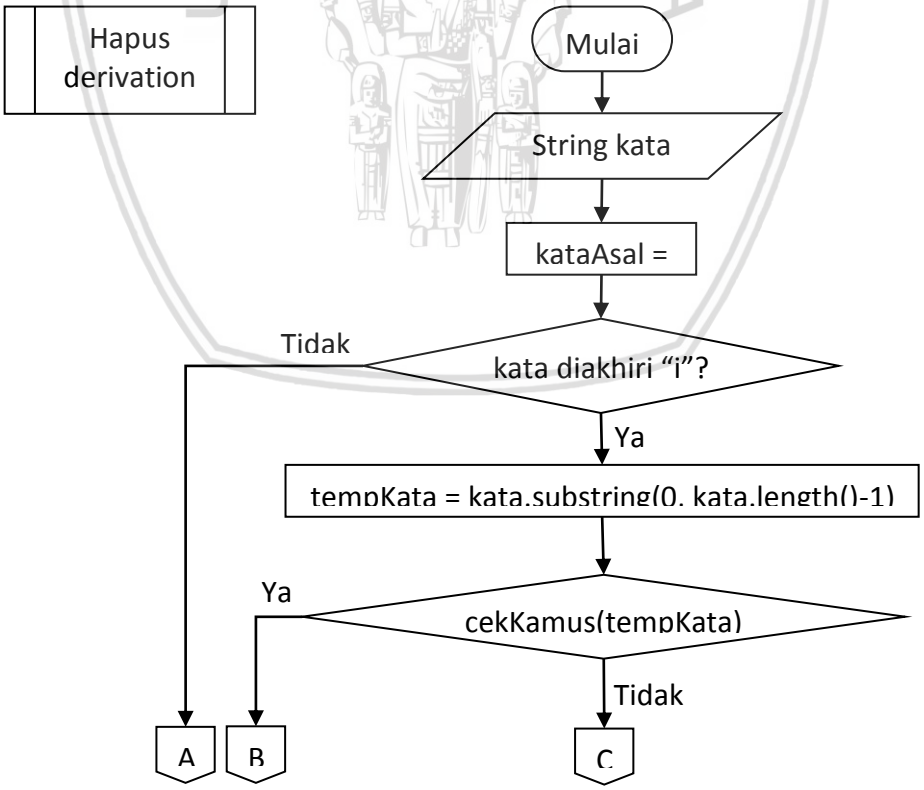


Gambar 4.10 Diagram alir penghapusan *inflection suffixes*

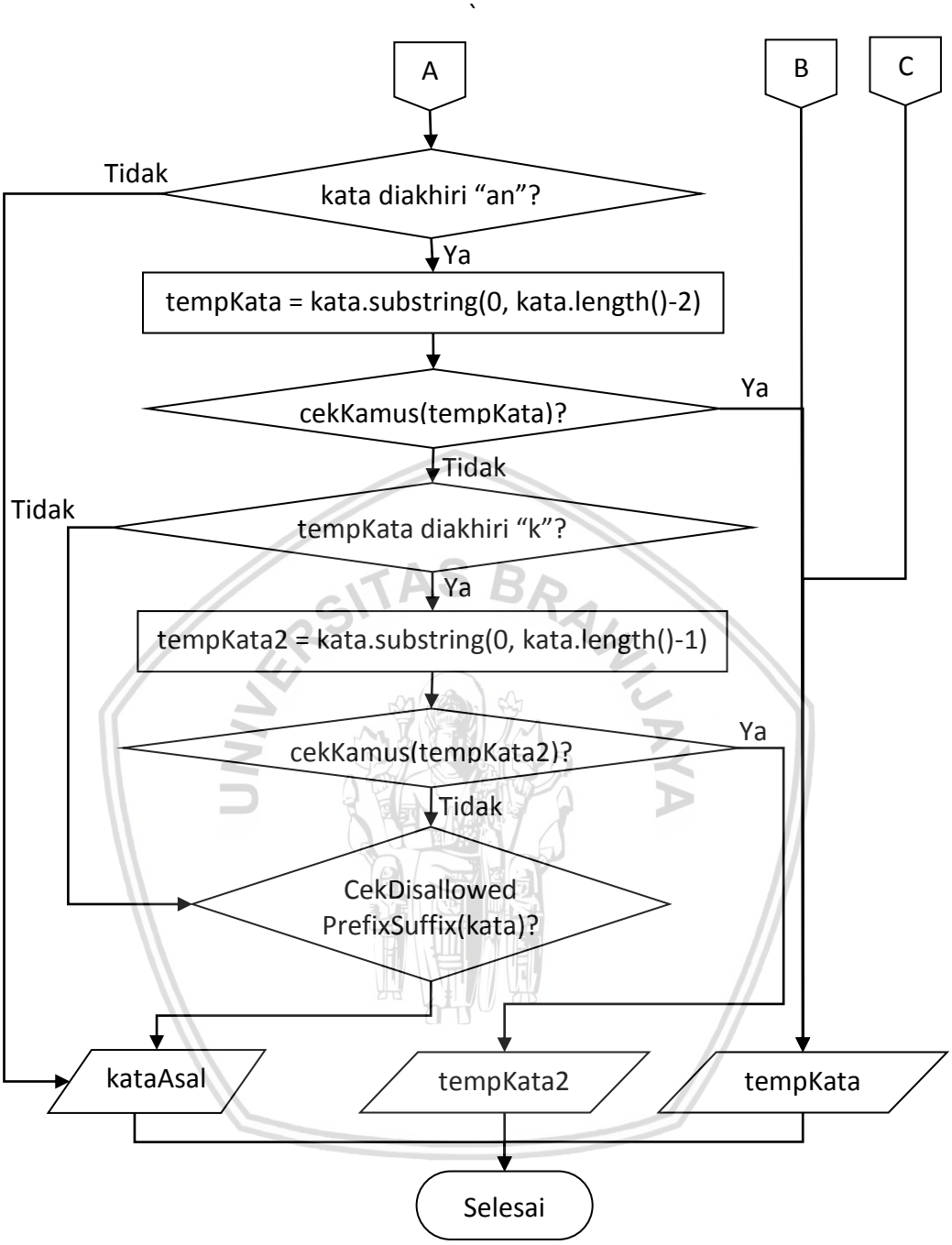


Gambar 4.10 Diagram alir penghapusan *inflection suffixes*

Setelah dilakukannya penghapusan *inflection suffixes*,maka proses hapus *derivation suffixes* dijalankan. Proses ini digambarkan dengan diagram alir pada Gambar 4.11.

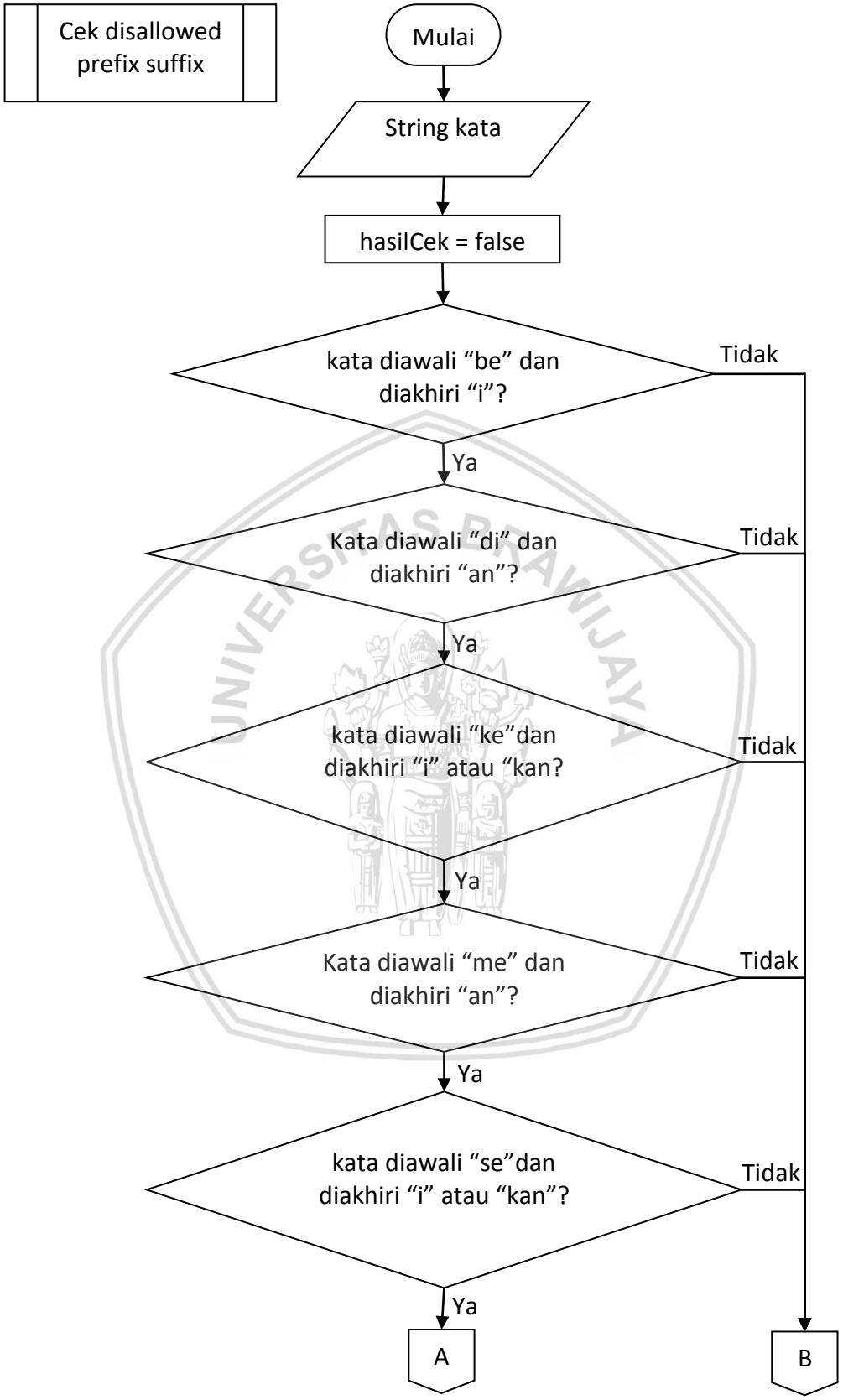


Gambar 4.11 Diagram alir penghapusan *derivation suffixes*

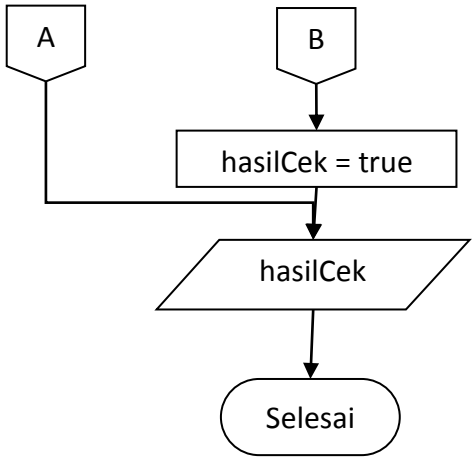


Gambar 4.11 Diagram alir penghapusan *derivation suffixes*

Pada proses penghapusan *derivation suffixes* terdapat pengecekan pasangan imbuhan yang tidak diperbolehkan. Apabila terjadi pasangan awalan dan akhiran yang dilarang maka hasil pengecekan akan bernilai *true*, sedangkan apabila sebaliknya maka hasil pengecekan bernilai *false.* Proses pengecekan tersebut digambarkan pada Gambar 4.12.

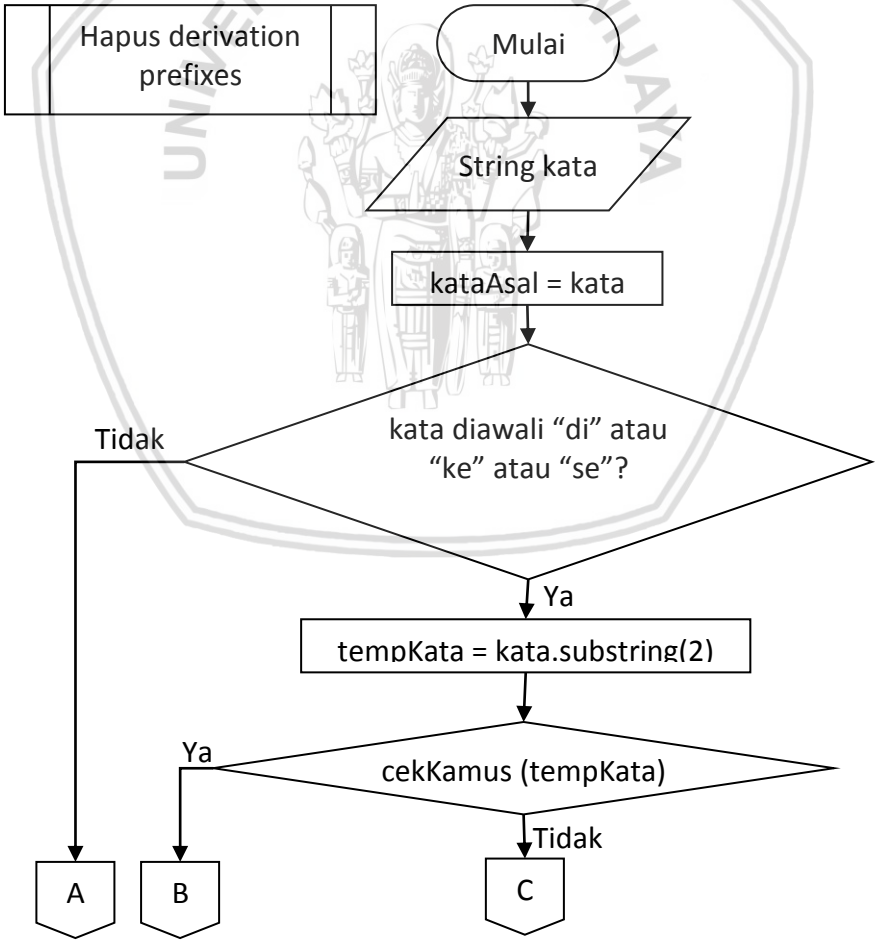


Gambar 4.12 Diagram alir proses pengecekan *disallowed preffix-suffix*

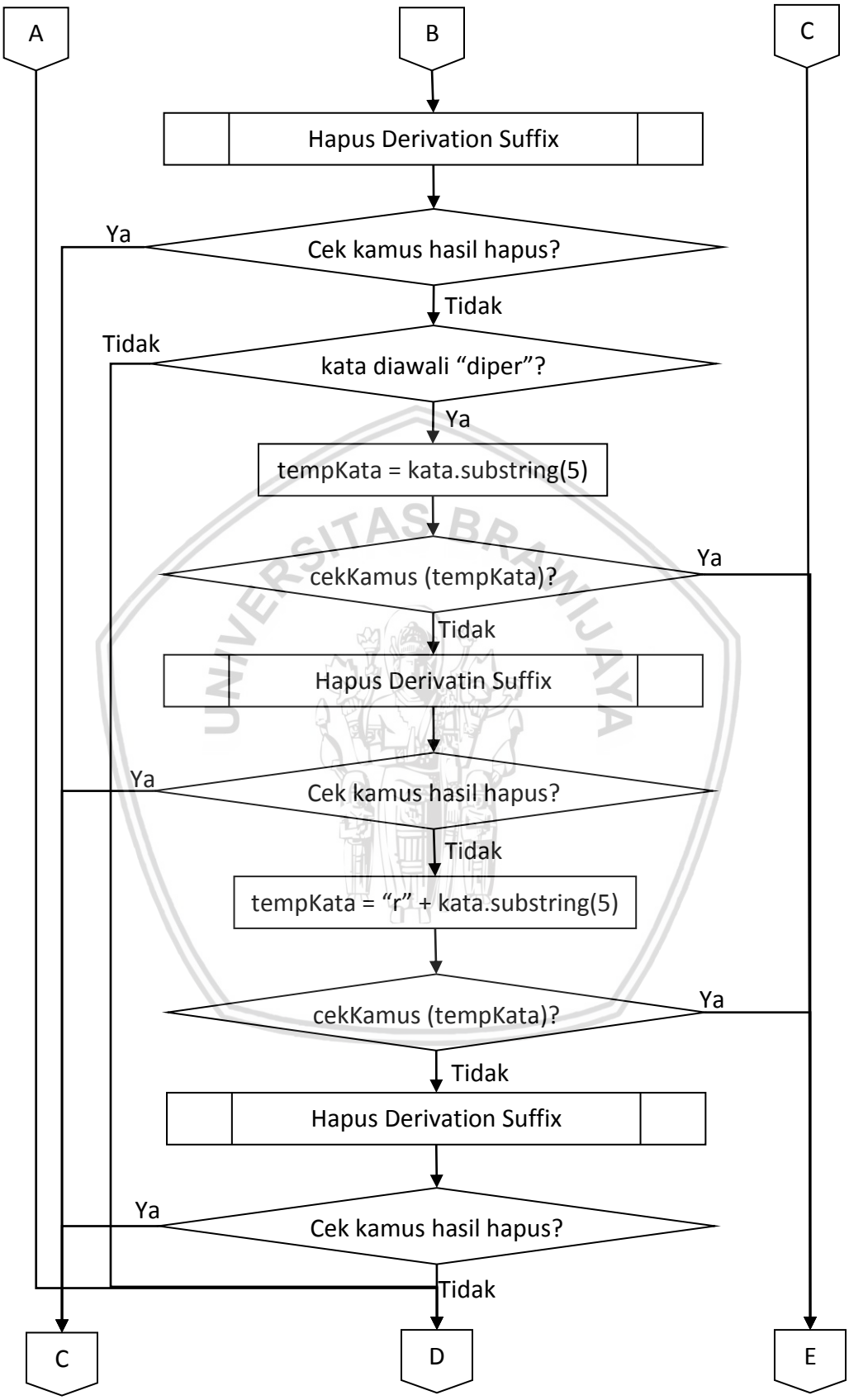


Gambar 4.12 Diagram alir proses pengecekan *disallowed preffix-suffix*

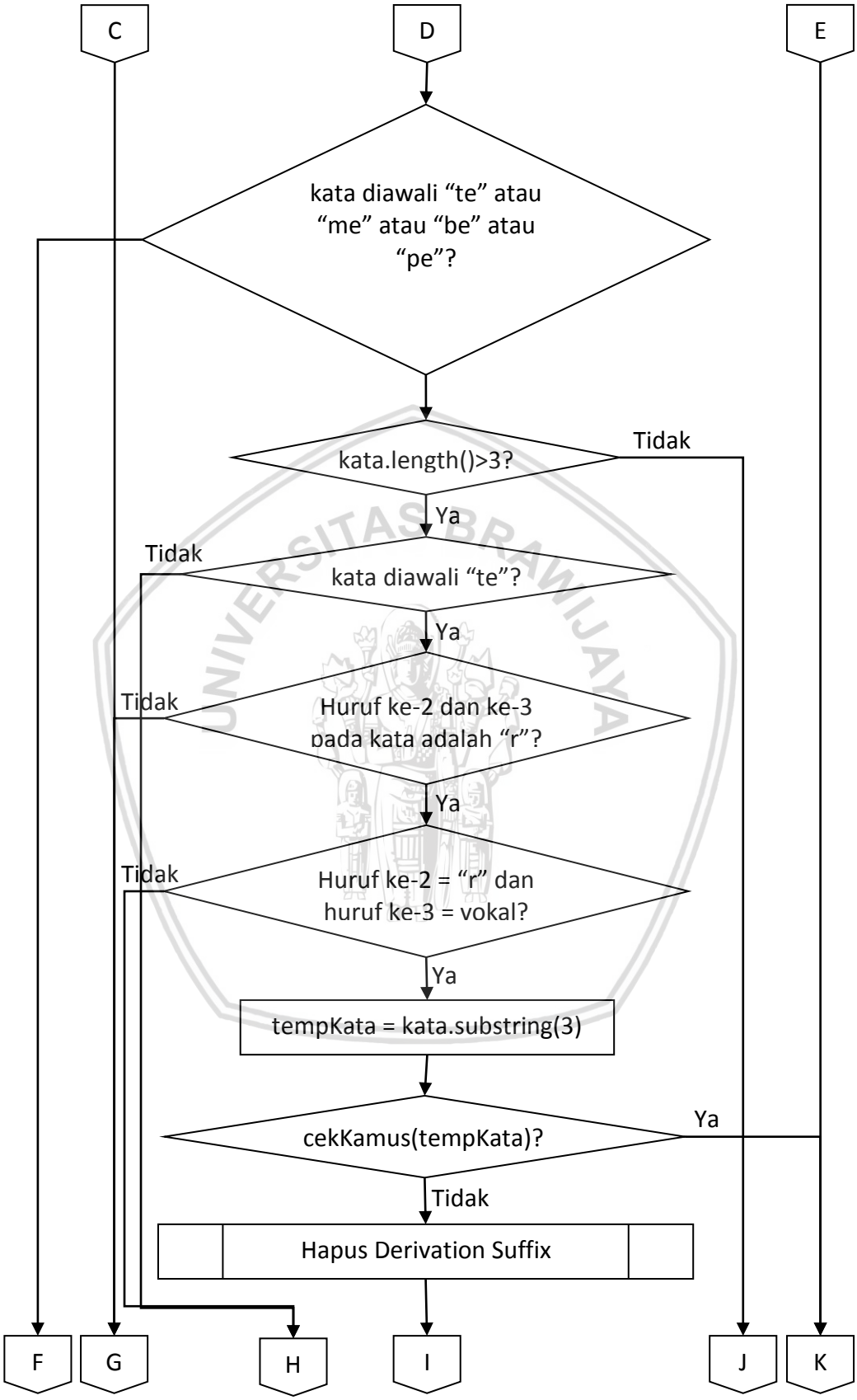
Langkah terakhir pada proses *stemming* adalah penghapusan *derivation prefixes*. Pengecekan ini dilakukan dengan mengecek setiap karakter pada kata. Diagram alirdari proses penghapusan *derivation prefixes* ini digambarkan pada Gambar 4.13.



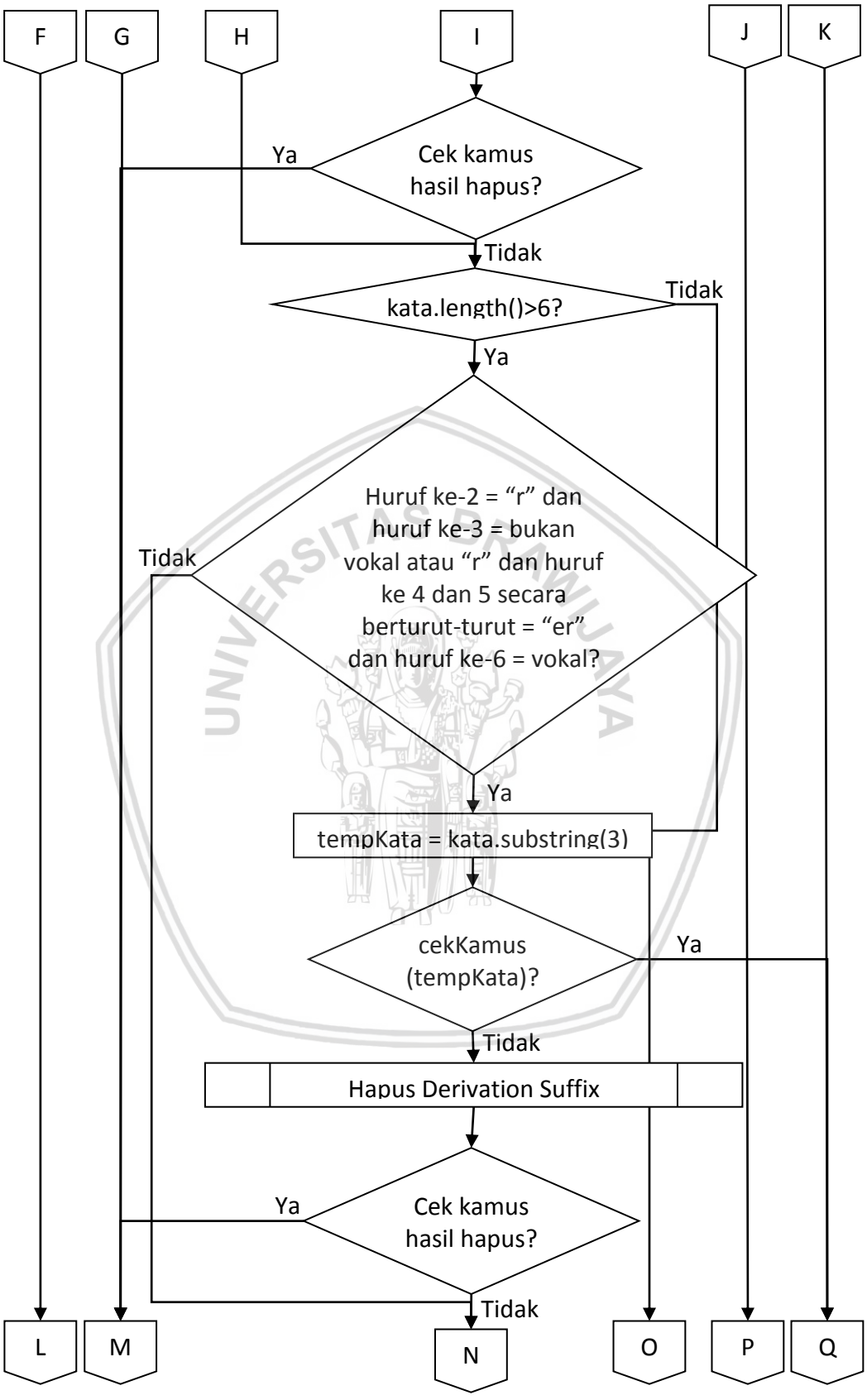
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



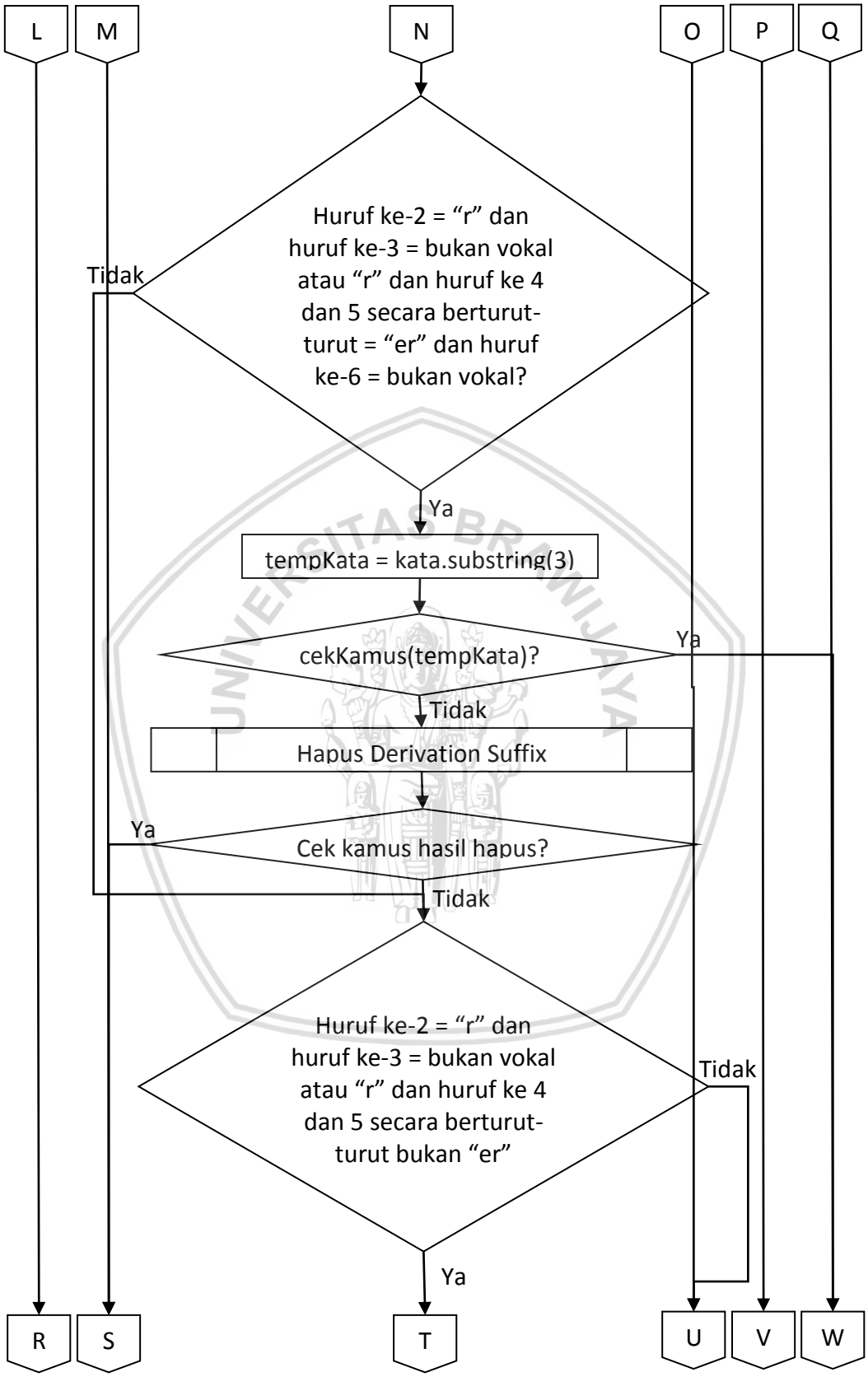
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



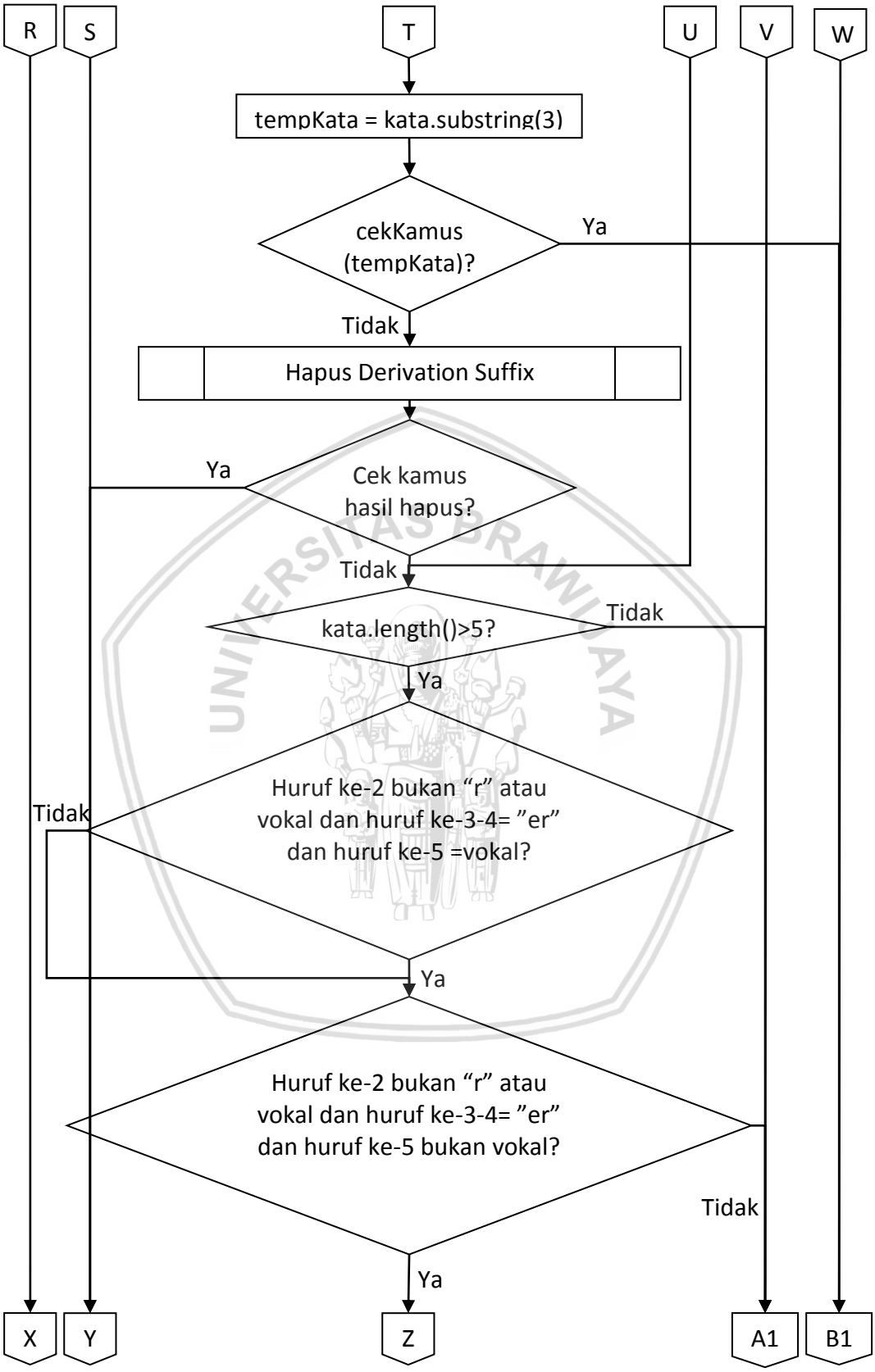
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



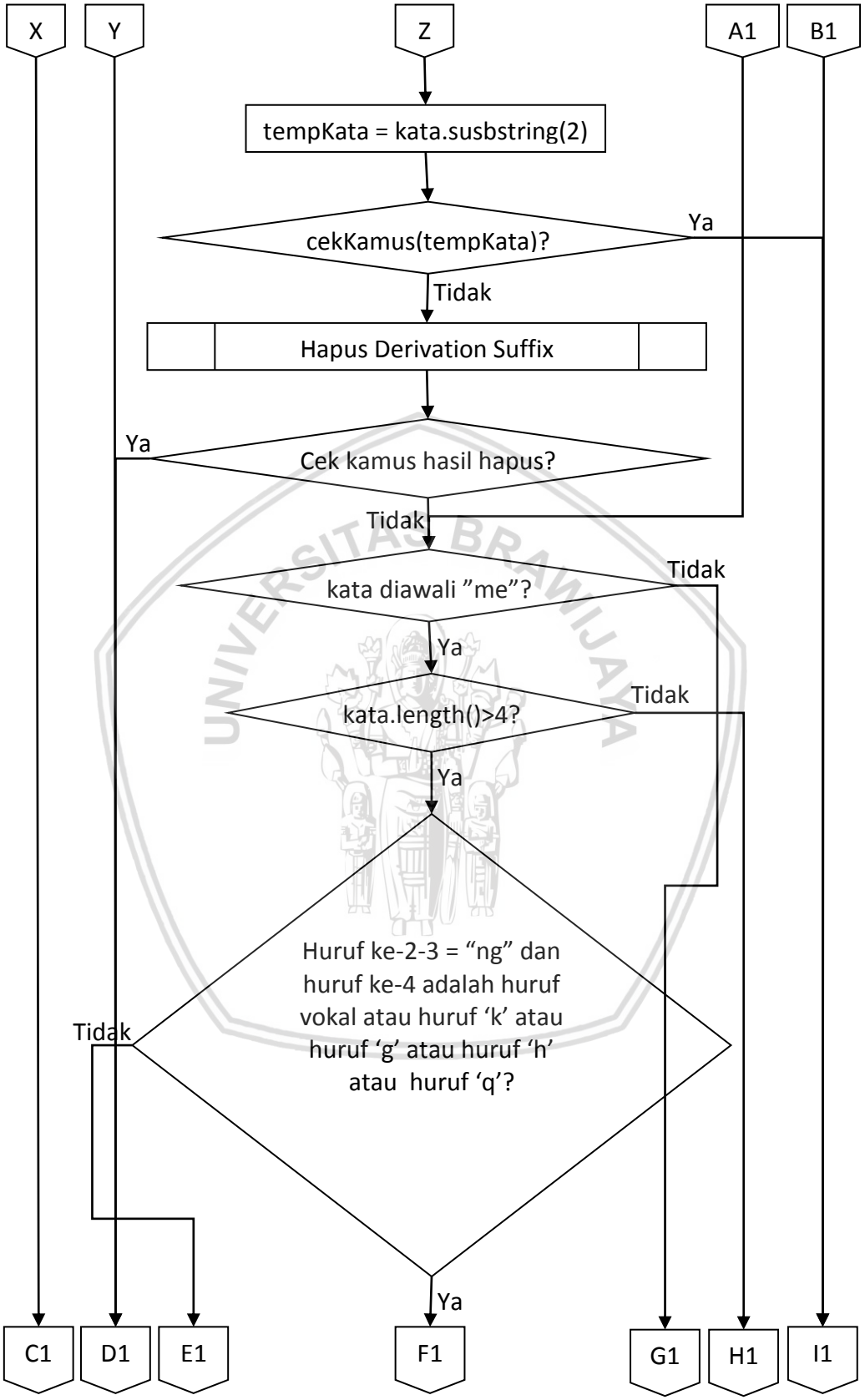
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



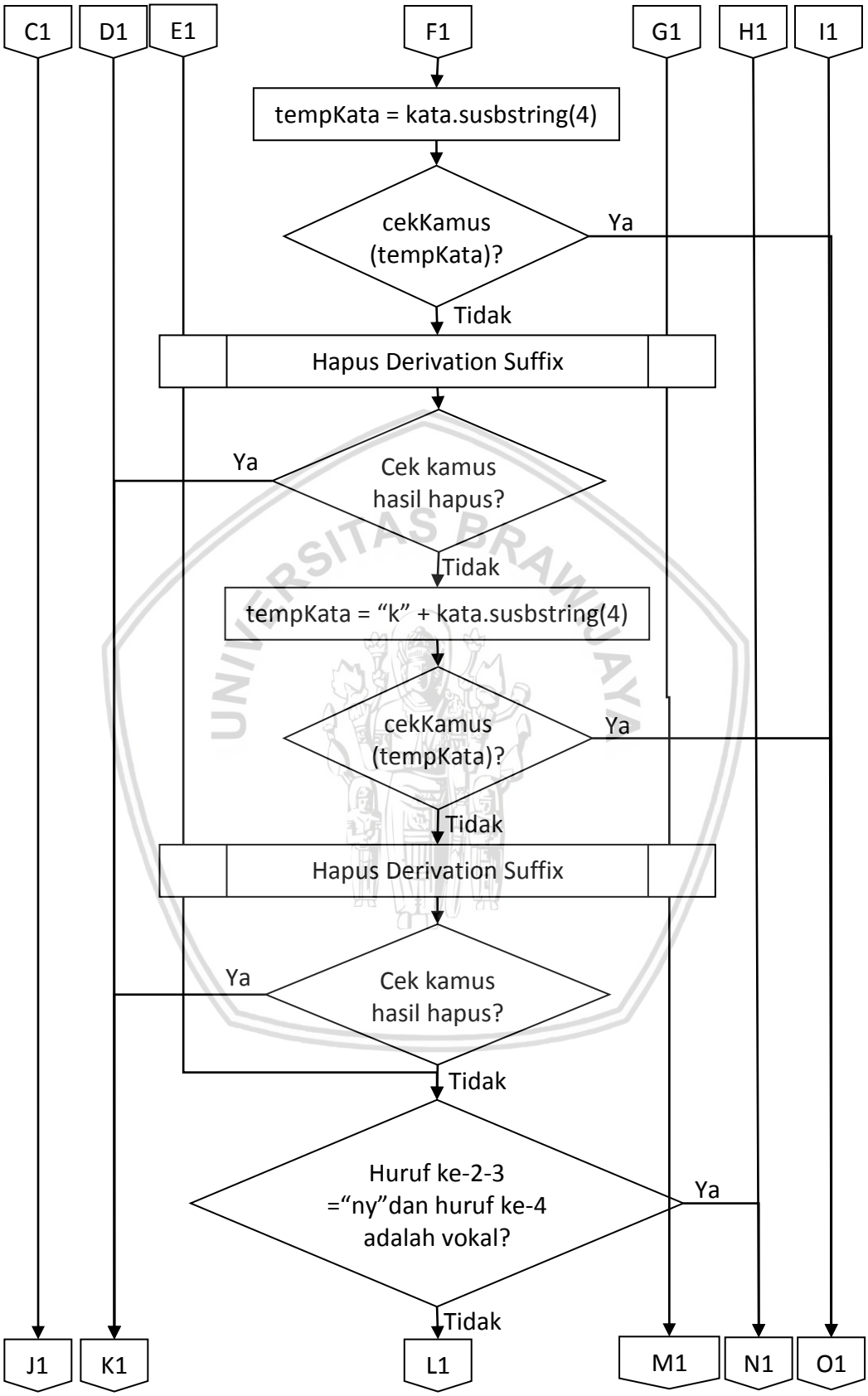
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



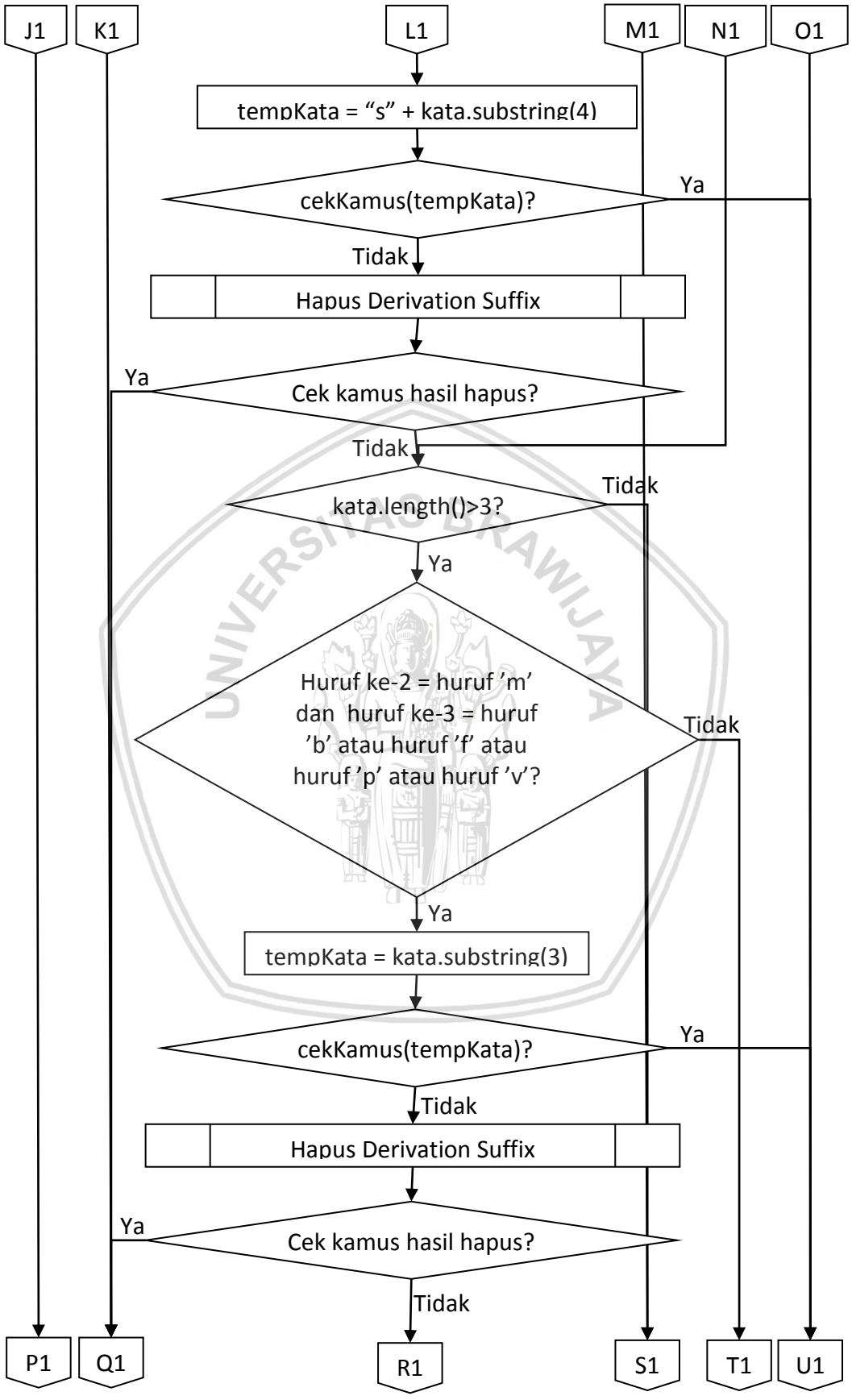
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



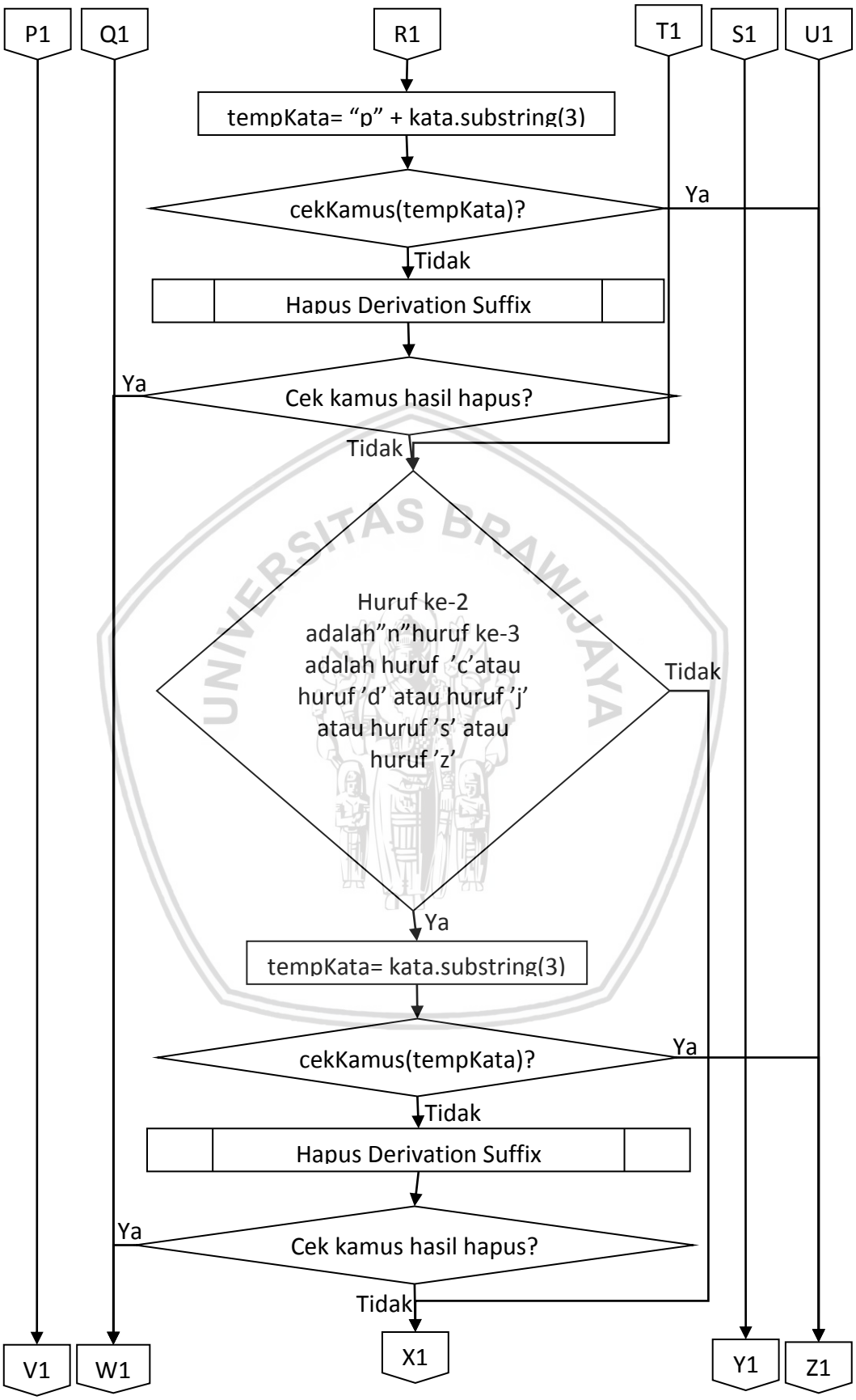
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



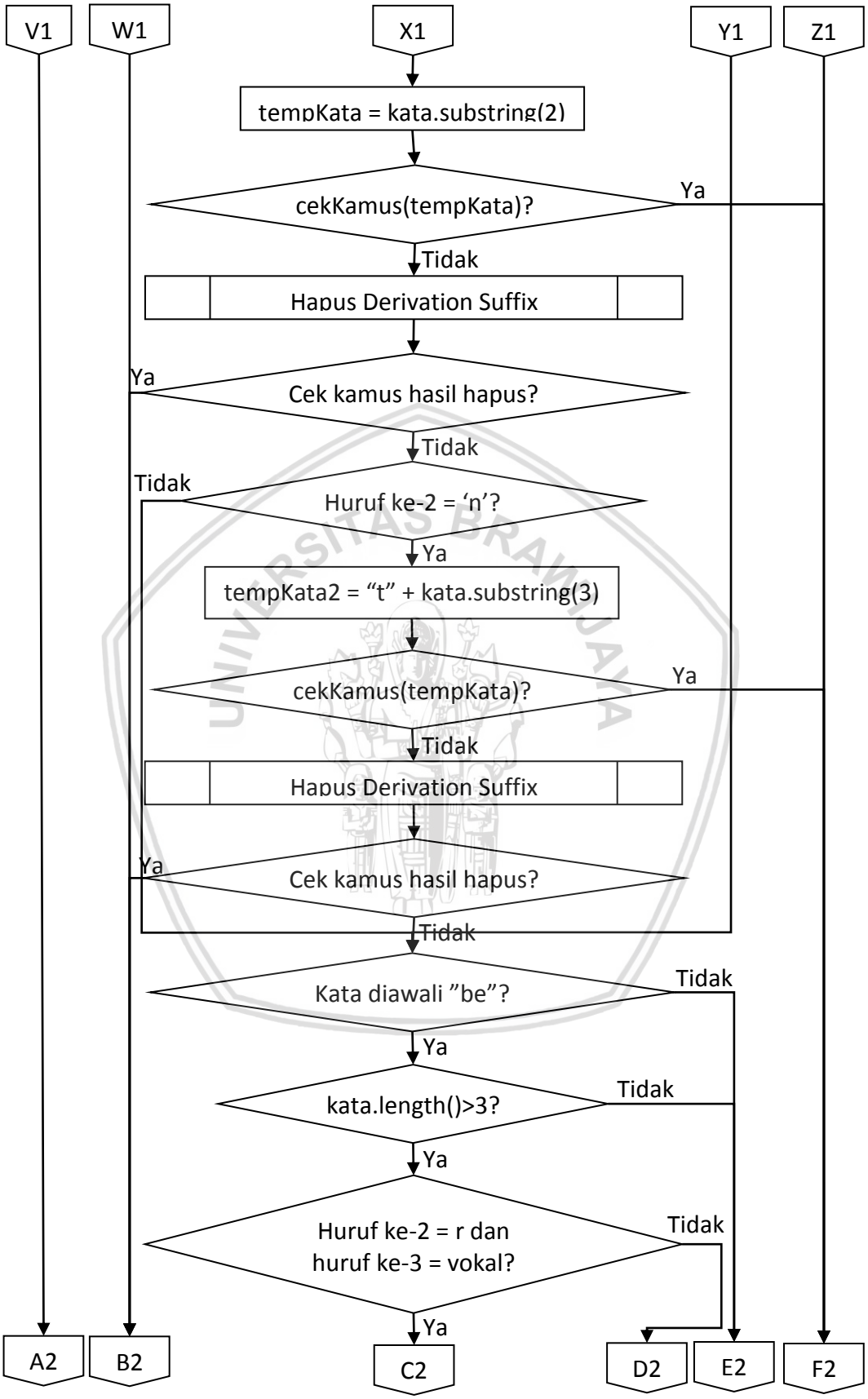
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



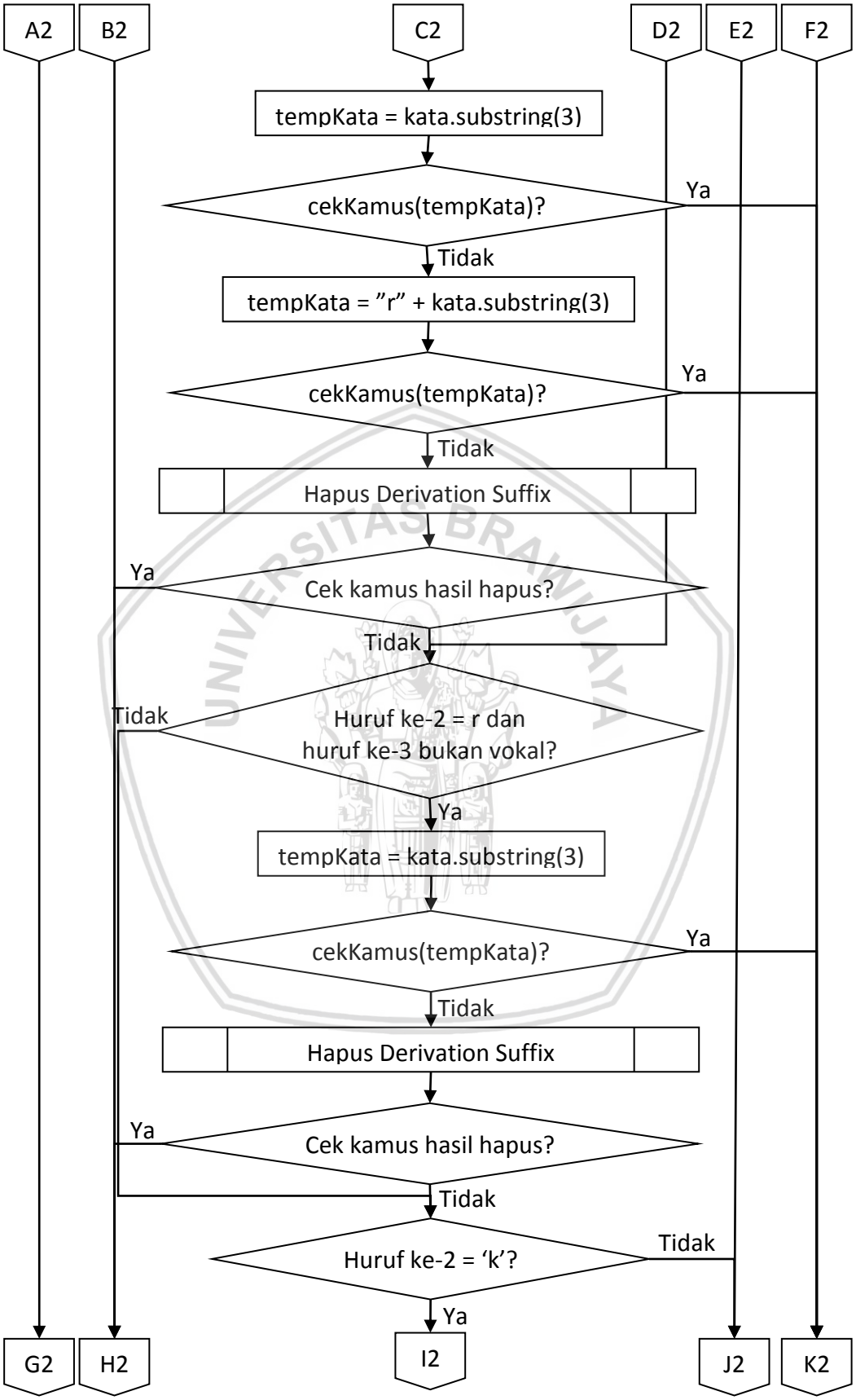
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



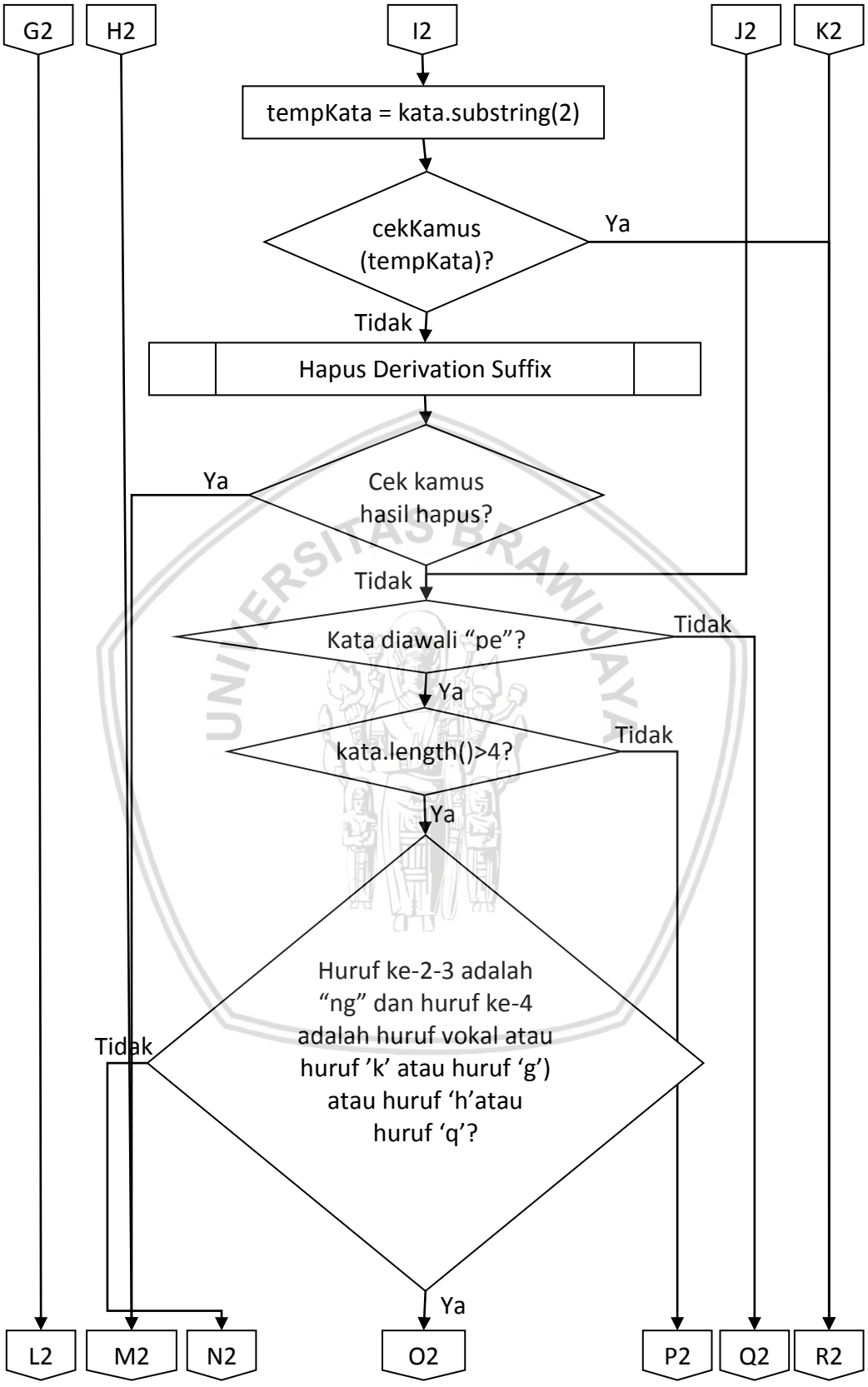
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



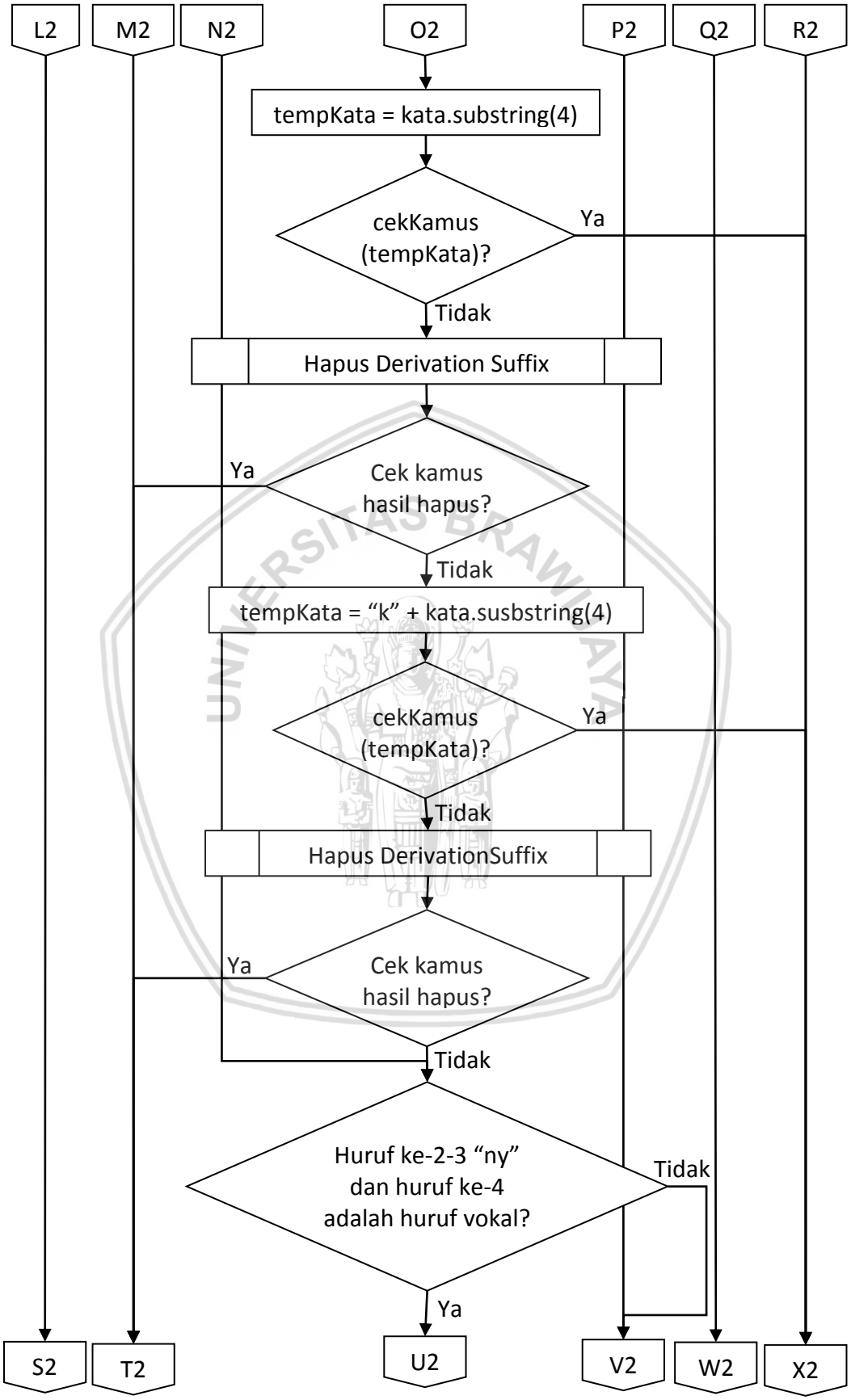
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



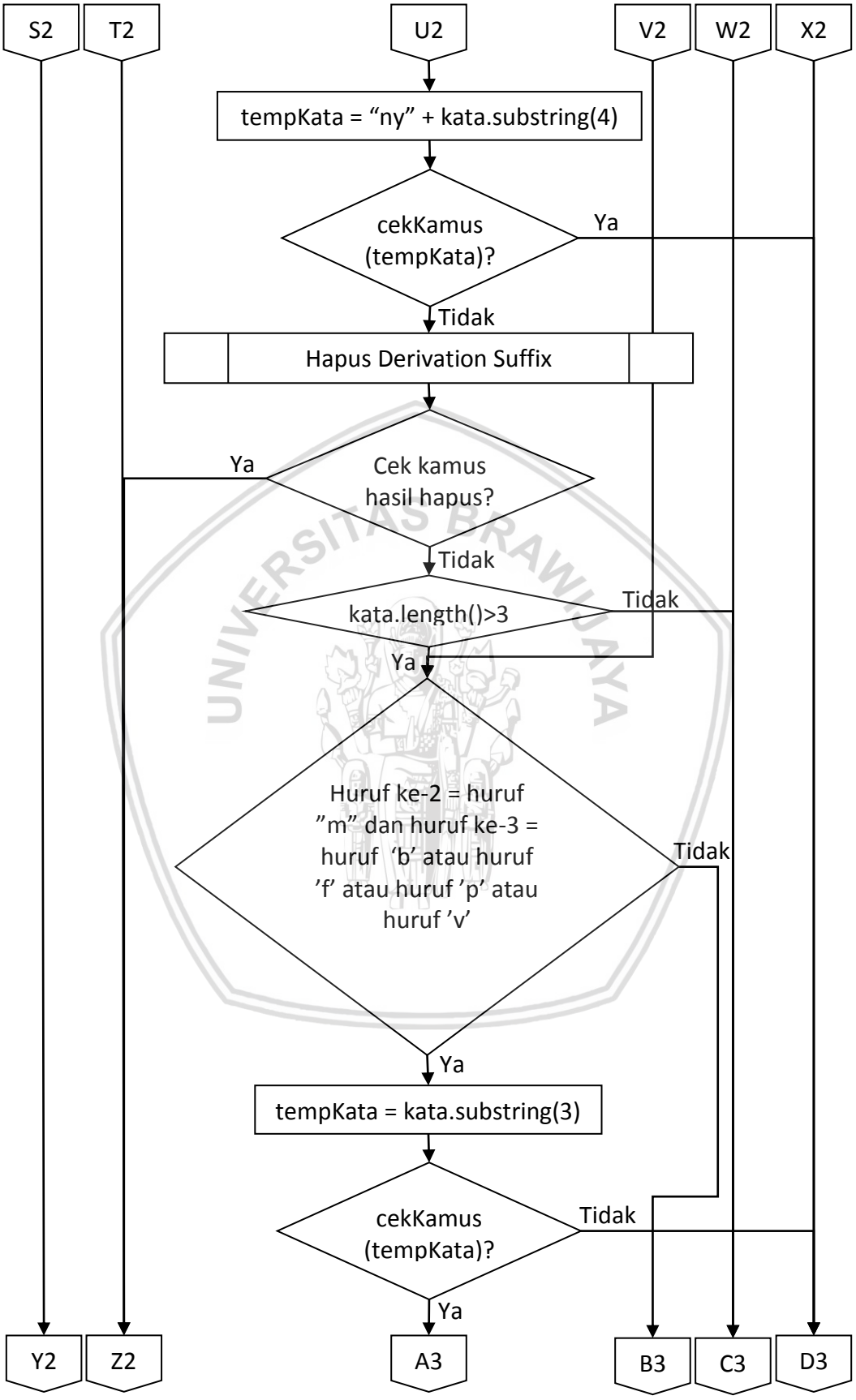
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



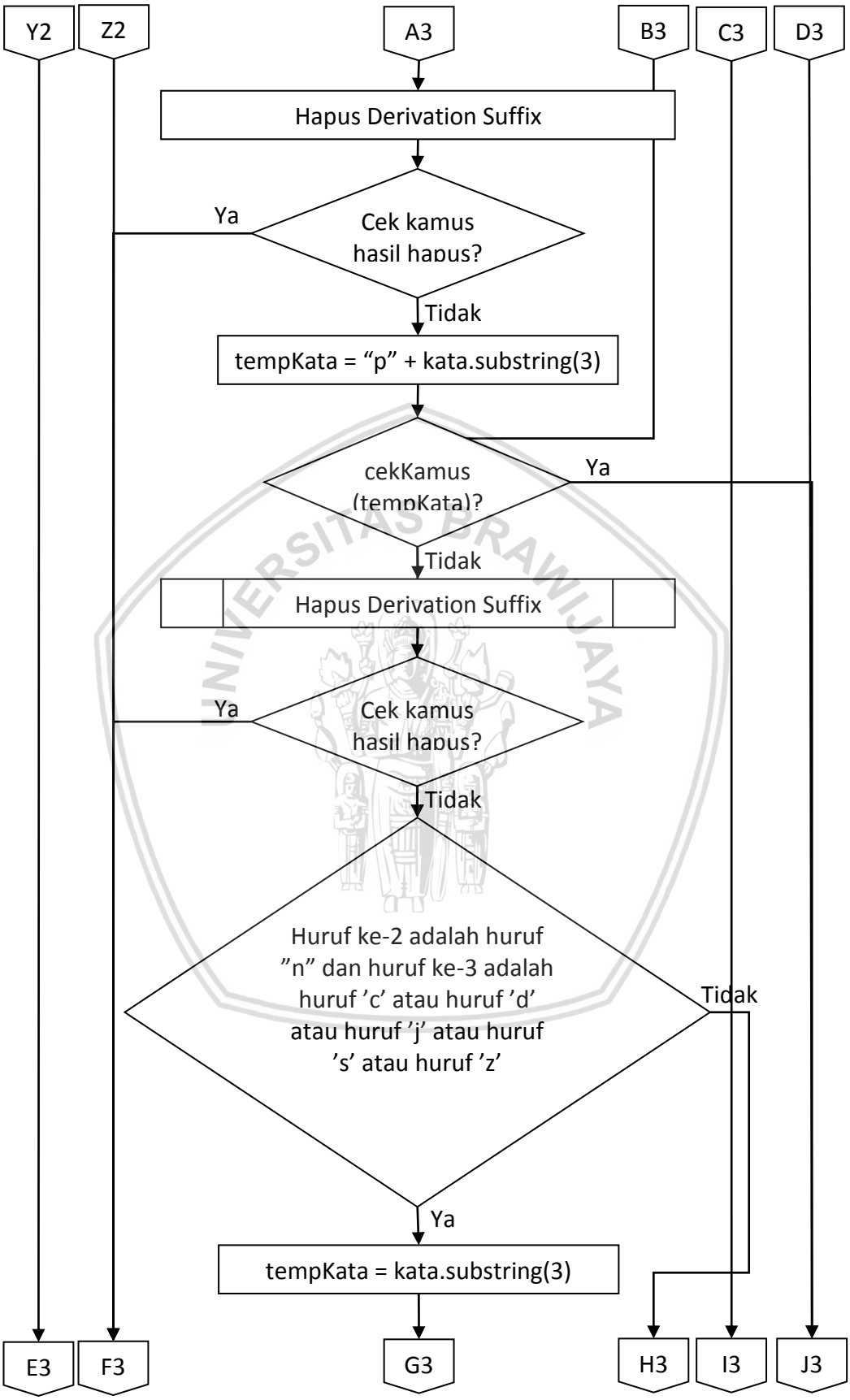
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



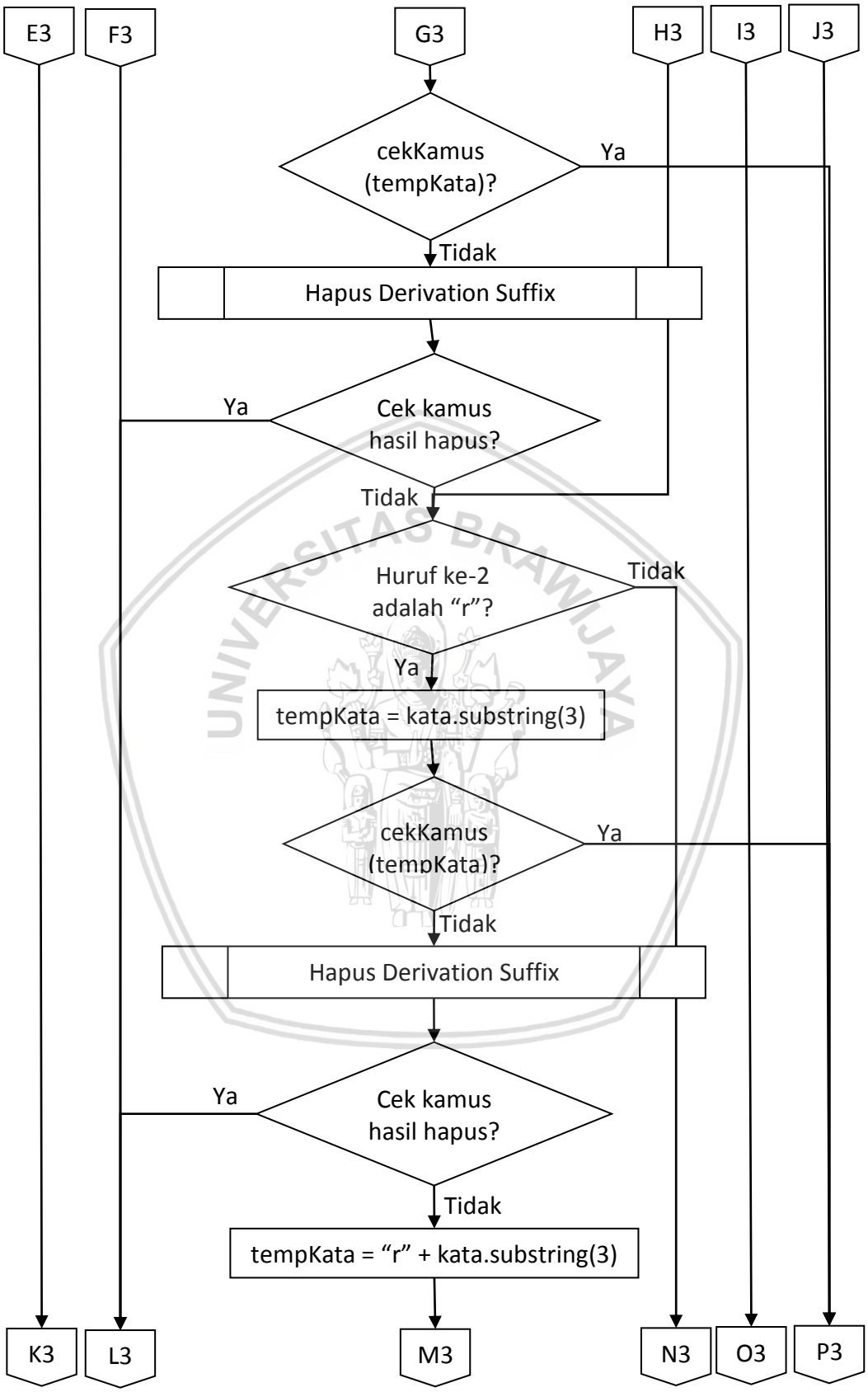
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



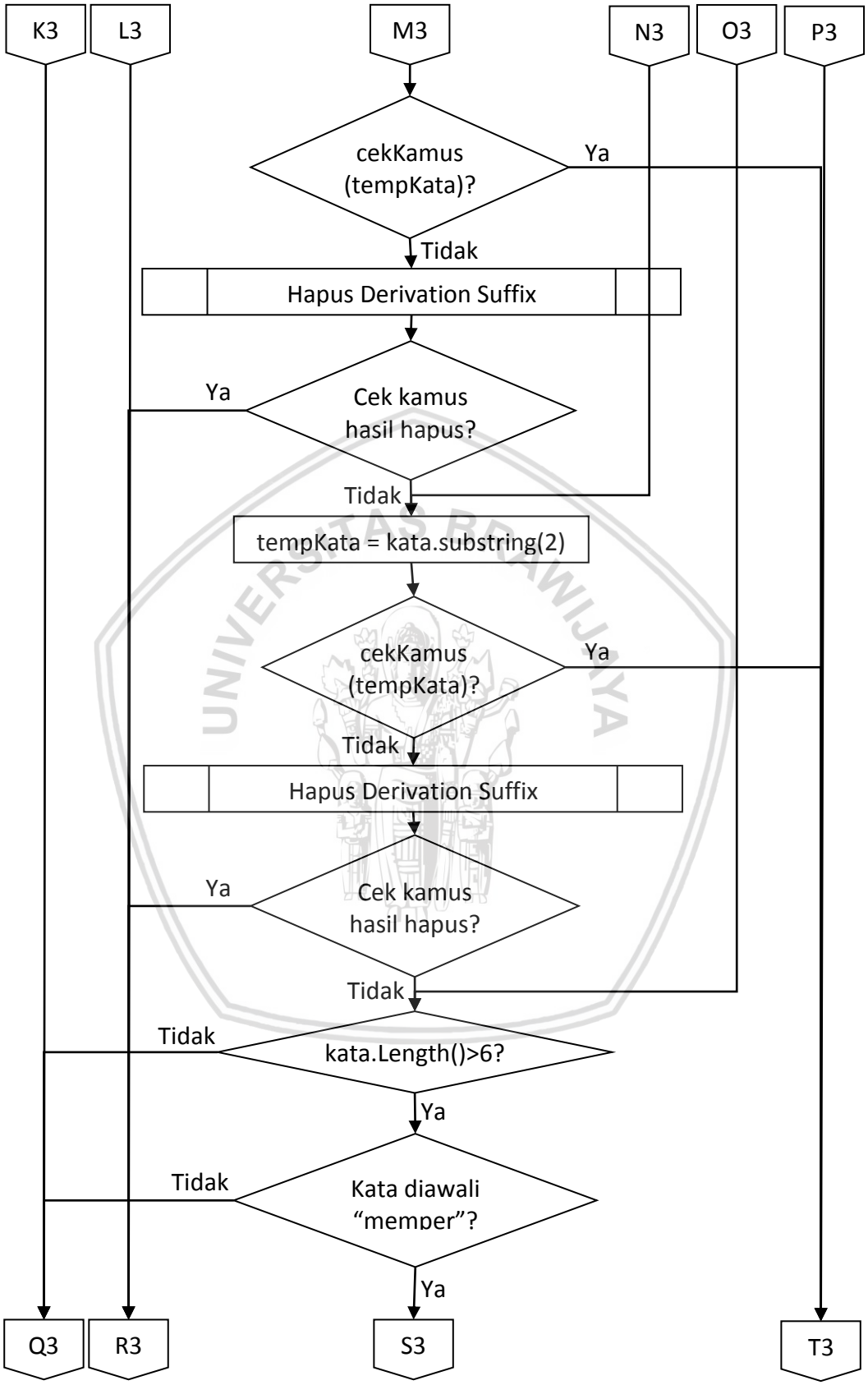
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



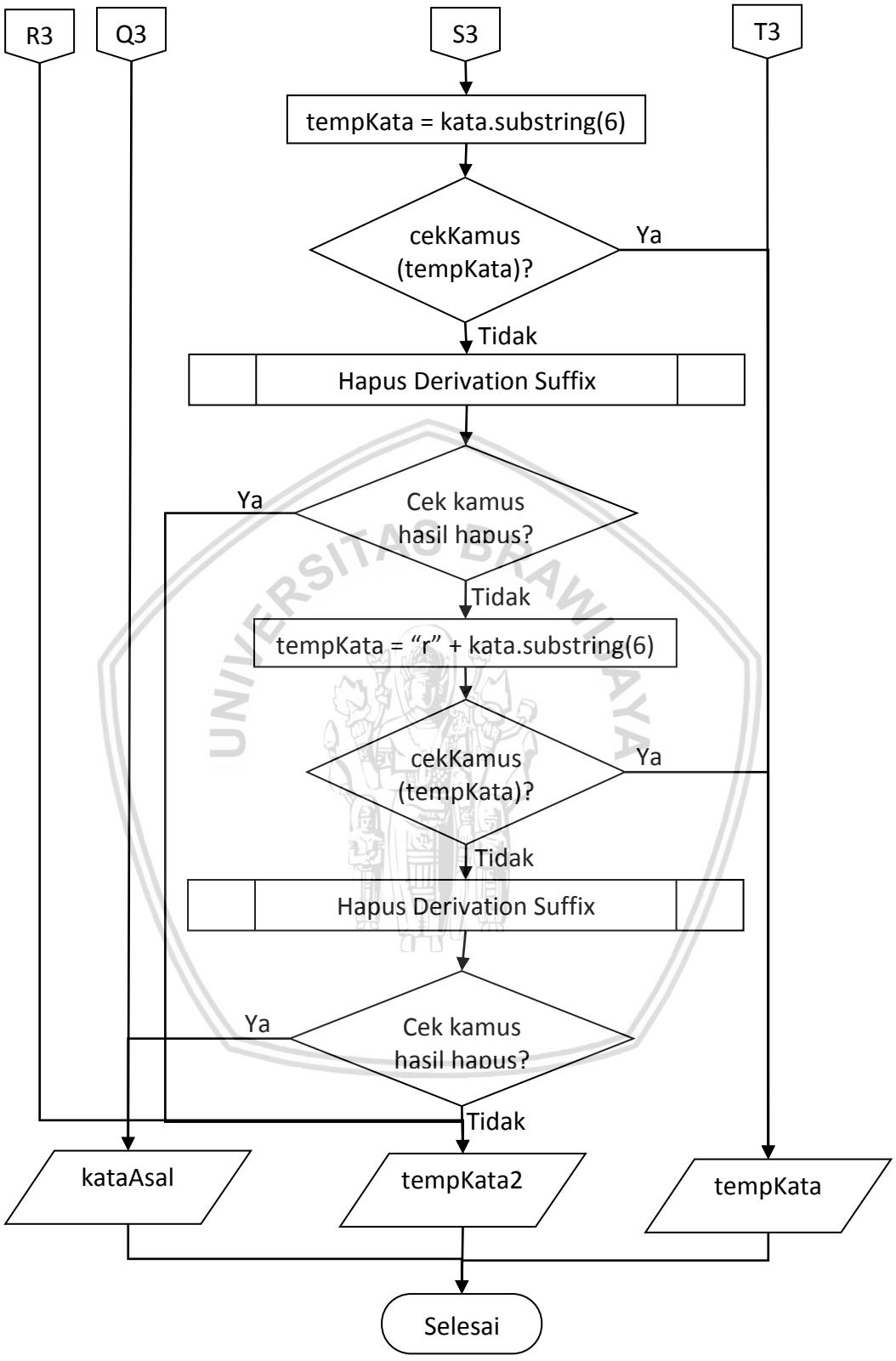
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



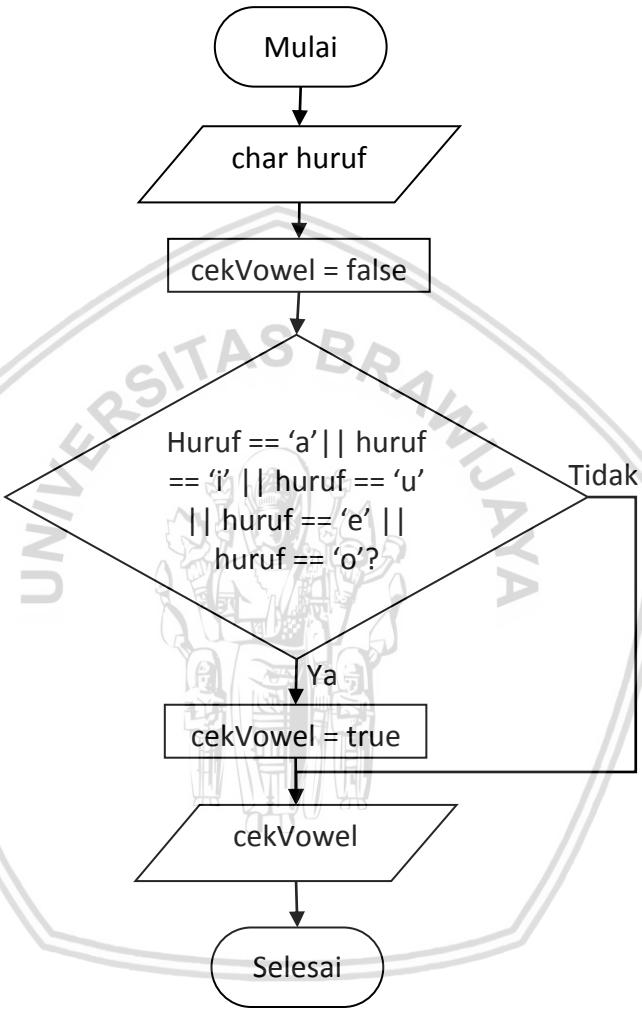
Gambar 4.13 Diagram alir penghapusan *derivation preffixes*



Gambar 4.13 Diagram alir penghapusan *derivation preffixes*

Berdasarkan Gambar 4.36 banyak digunakan fungsi substring(). Fungsi ini digunakan untuk mengambil bagian kata dengan menghilangkan penggalan

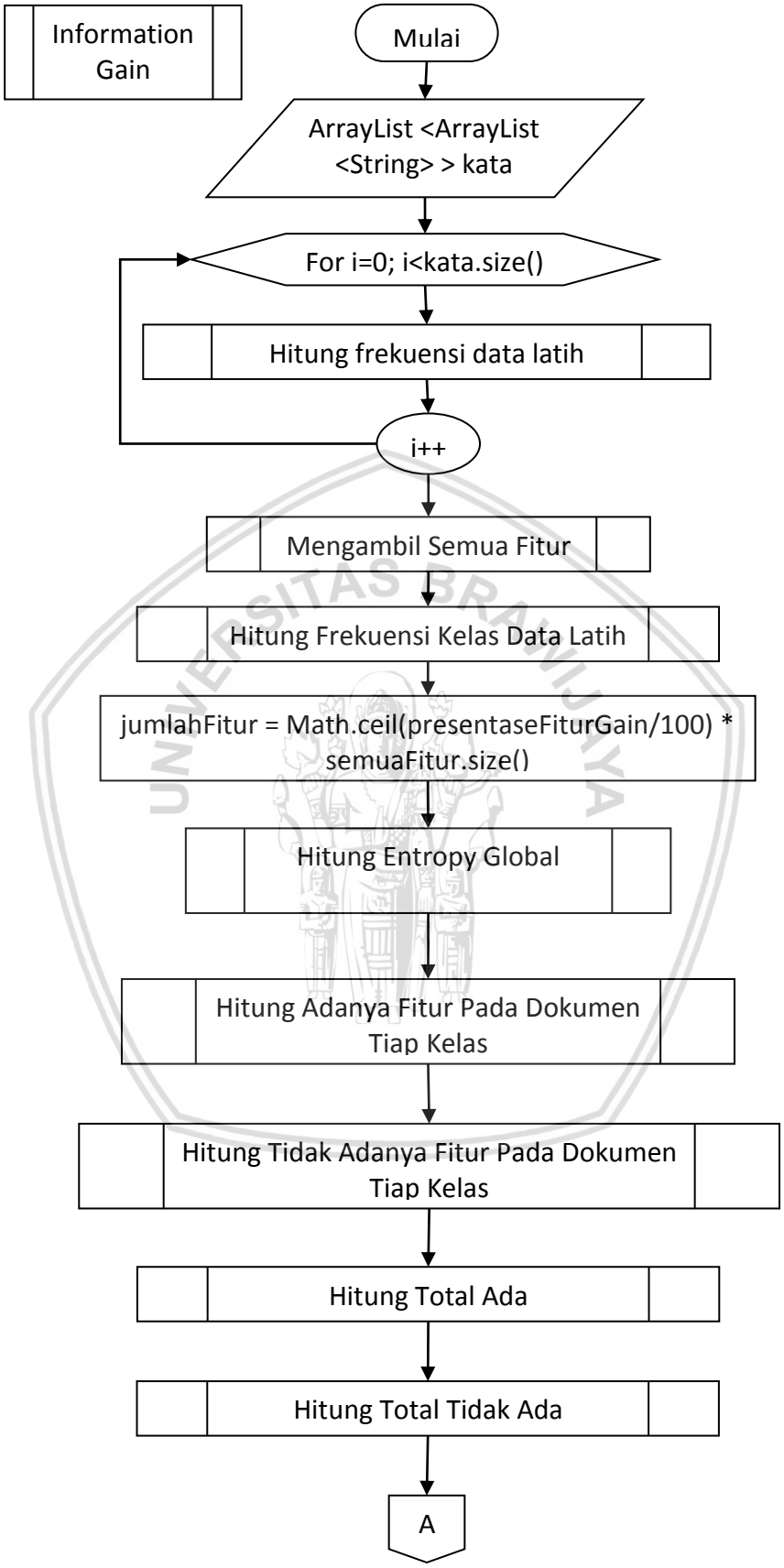
karakter yang dianggap sebagai imbuhan. Selain itu juga digunakan fungsi subSequence(). Fungsi ini digunakan untuk mendapatkan beberapa karakter huruf dengan parameter indeks mulai dan indeks batas. Pada proses penghapusan derivationprefix ini juga terdapat pengecekan huruf vokal yaitu a, i, u, e, o yang dilakukan dengan melakukan pemanggilan fungsi vowel() dengan parameter berupa huruf bertipe char. Proses pengecekan huruf vokal ini digambarkan pada Gambar 4.14.



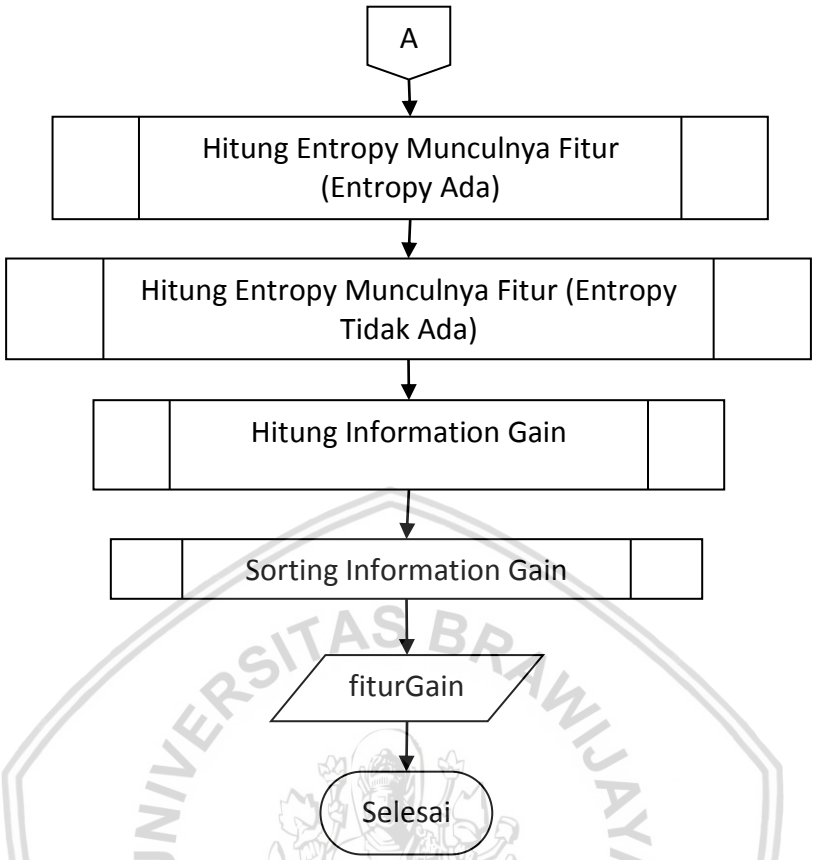
Gambar 4.14 Diagram alir pengecekan huruf vokal

4.1.2 Information Gain

Information gain pada penelitian ini menggunakan perhitungan ada dan tidak adanya fitur dalam dokumen, baik perhitungan pada masing-masing kategori maupun secara keseluruhan. Secara umum, alur proses perhitungan information gain ini digambarkan pada Gambar 4.15.

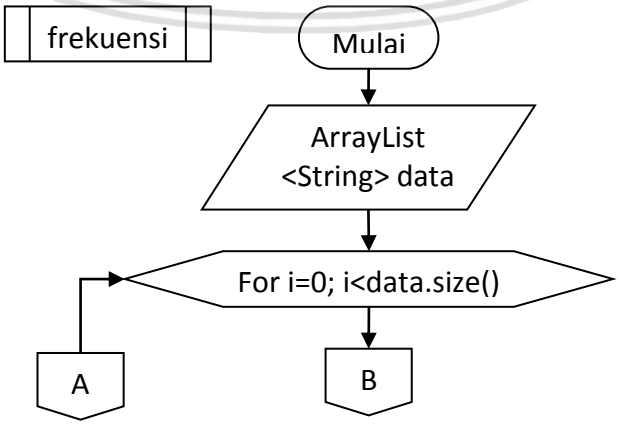


Gambar 4.15 Diagram alir proses *information gain*

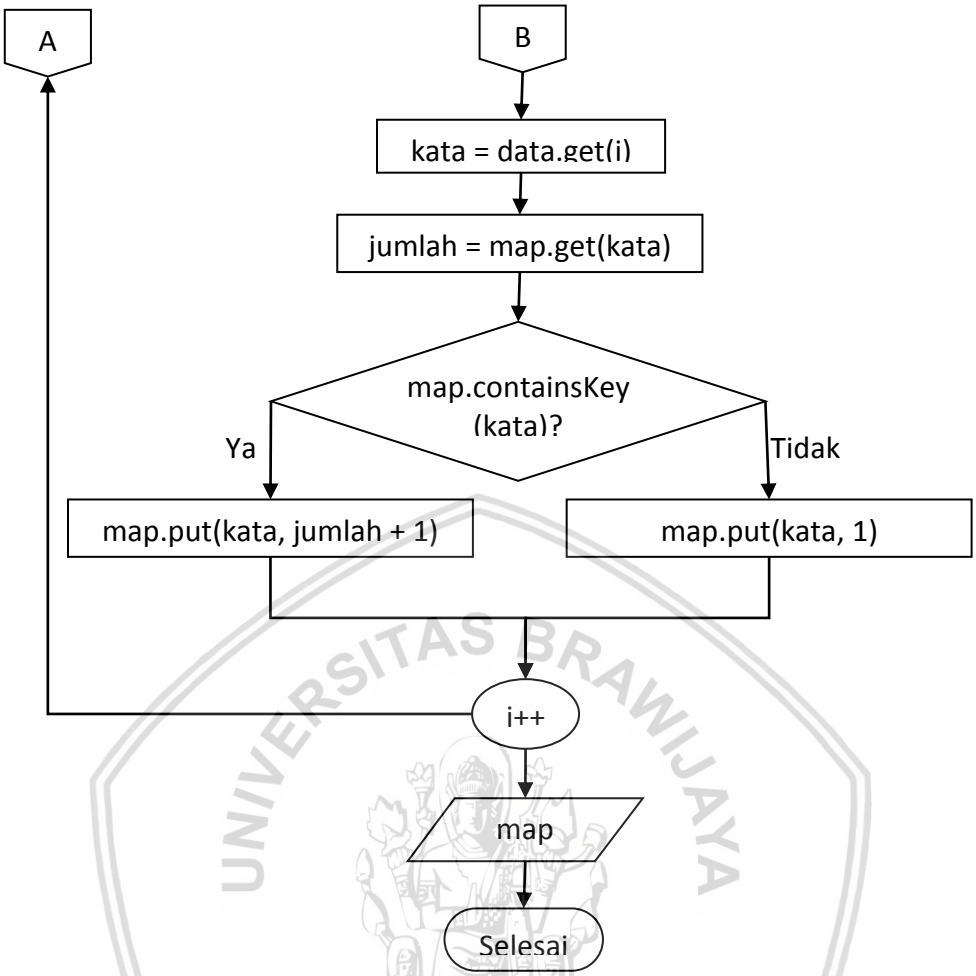


Gambar 4.15 Diagram alir proses *information gain*

Pada gambar 4.15 diatas, pada proses *information gain,* terdapat proses menghitung *frekuensi* data baik data latih maupun kelas/kategori dari data latih. Perhitungan frekuensi ini dilakukan dengan melakukan pemanggilan fungsi frekuensi untuk setiap dokumen dari data yang akan dilakukan perhitungan. Fungsi frekuensi ini berisi proses perhitungan jumlah elemen setiap data yang terdapat dalam dokumen latih. Proses perhitungan *frequency* ini digambarkan pada Gambar 4.16.

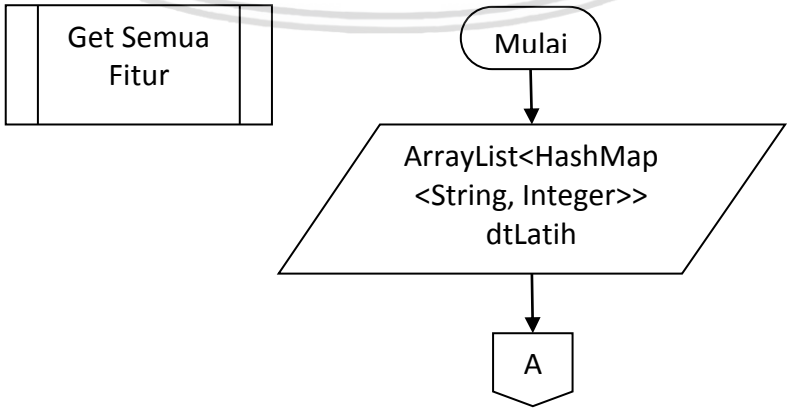


Gambar 4.16 Diagram alir proses perhitungan frekuensi

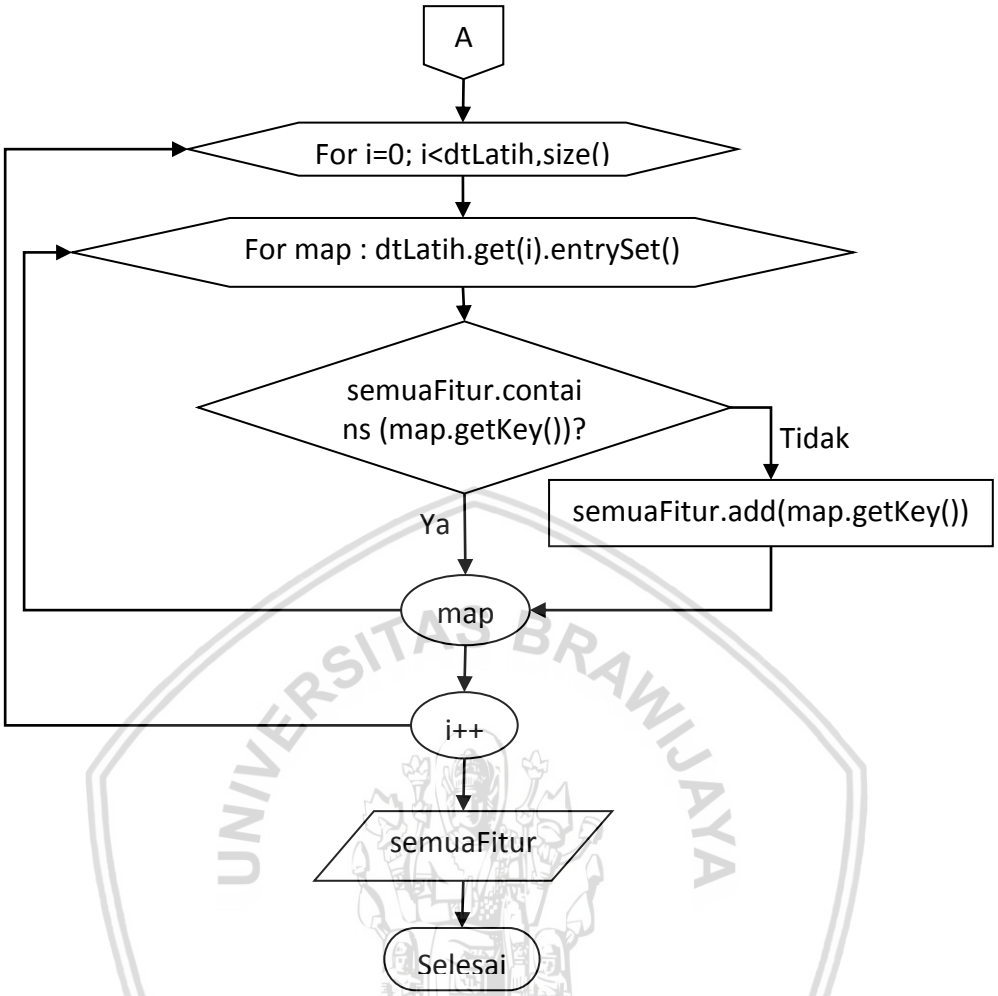


Gambar 4.16 Diagram alir proses perhitungan frekuensi

Selain perhitungan frekuensi, pada proses *information gain* terdapat proses yang digunakan untuk mengambil fitur (ciri)dari seluruh dokumen. Proses ini dilakukan dengan memanggil fungsi getSemuaFitur(). Alur proses ini digambarkan pada Gambar 4.17.

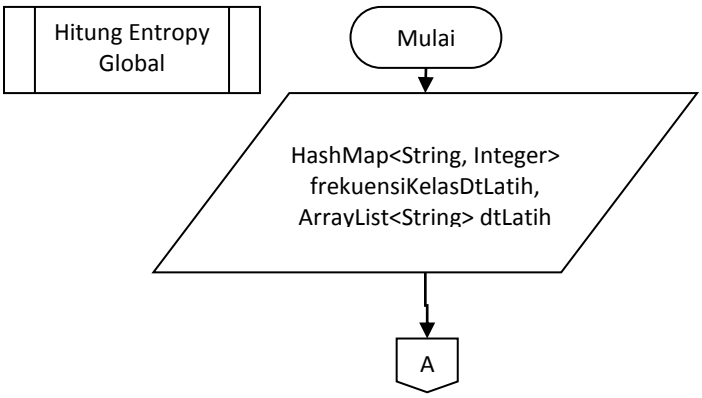


Gambar 4.17 Diagram alir proses pengambilan semua fitur

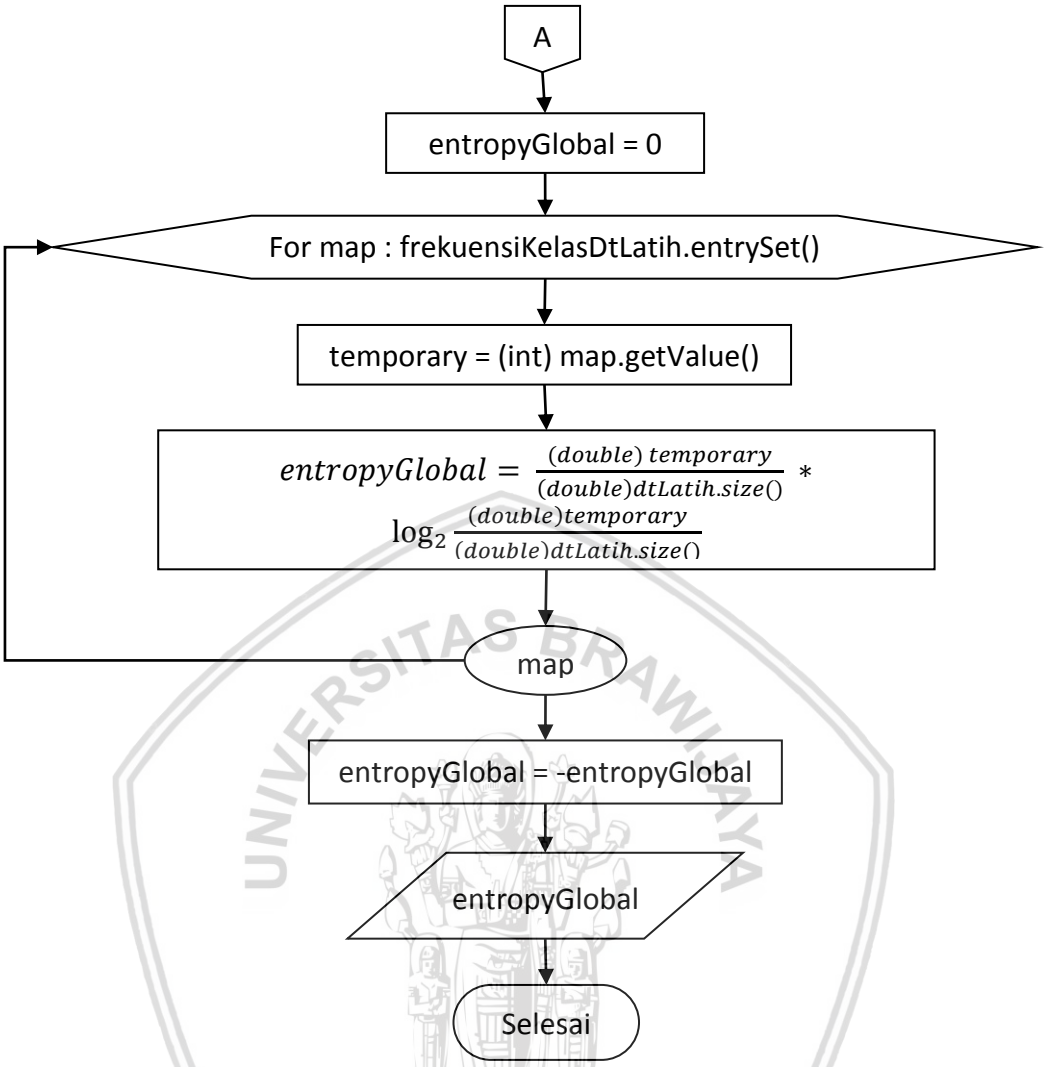


Gambar 4.17 Diagram alir proses pengambilan semua fitur

Pada proses seleksi fitur dengan *information gain* terdapat proses perhitungan entropy global yang menjadi langkah awal pencarian *information gain* setiap fitur. Proses ini dilakukan dengan melakukan pemanggilan fungsi hitungEntropyGlobal(). Fungsi ini memiliki detail proses seperti pada diagram alir Gambar 4.18.

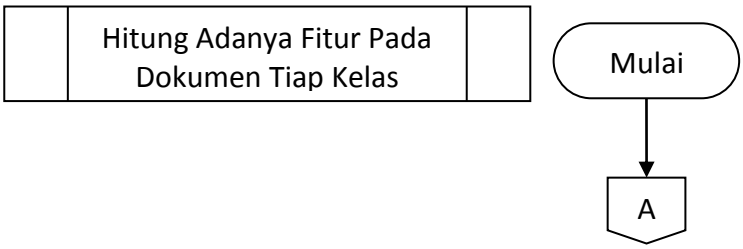


Gambar 4.18 Diagram alir proses perhitungan *entropy global*

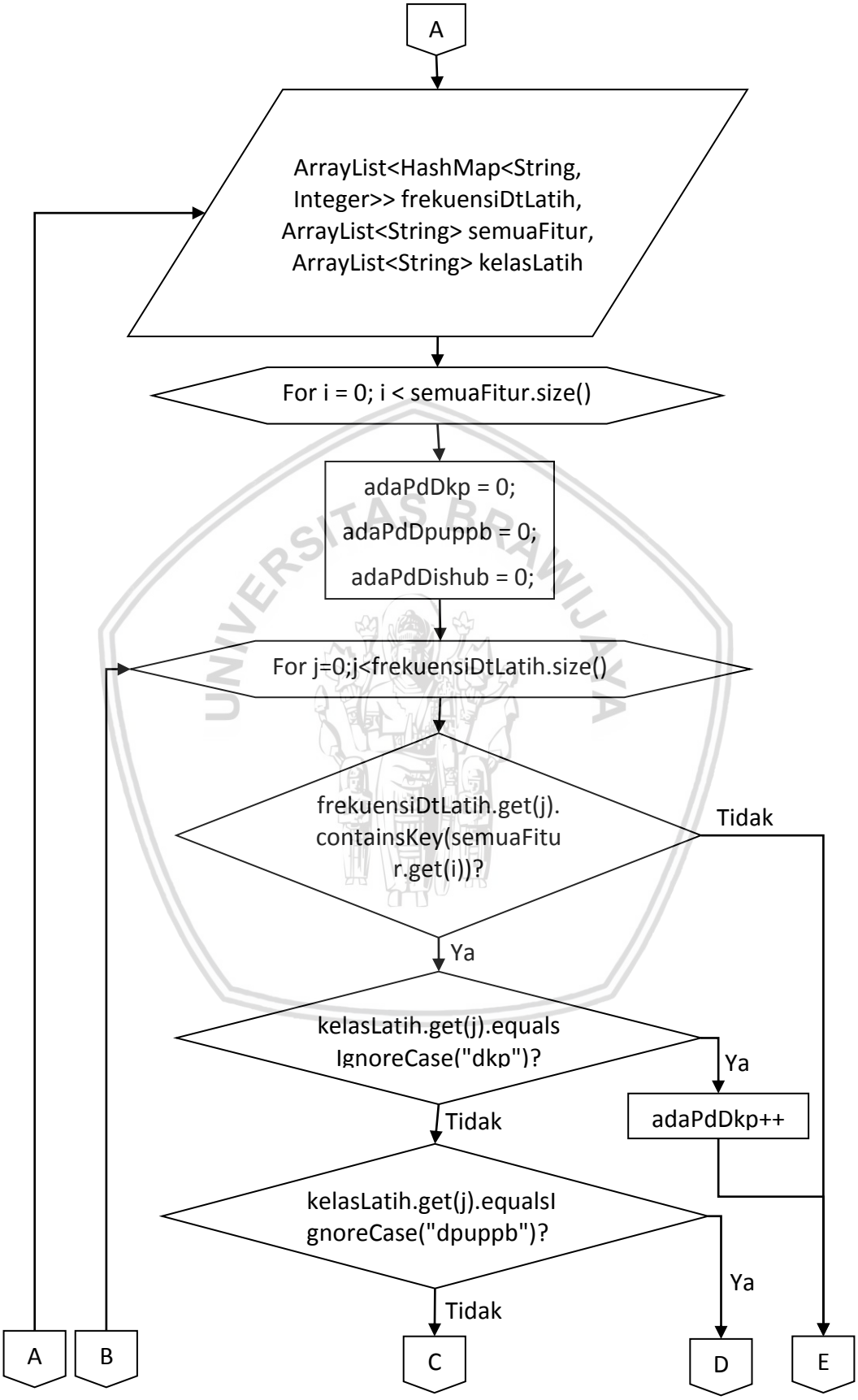


Gambar 4.18 Diagram alir proses perhitungan *entropy global*

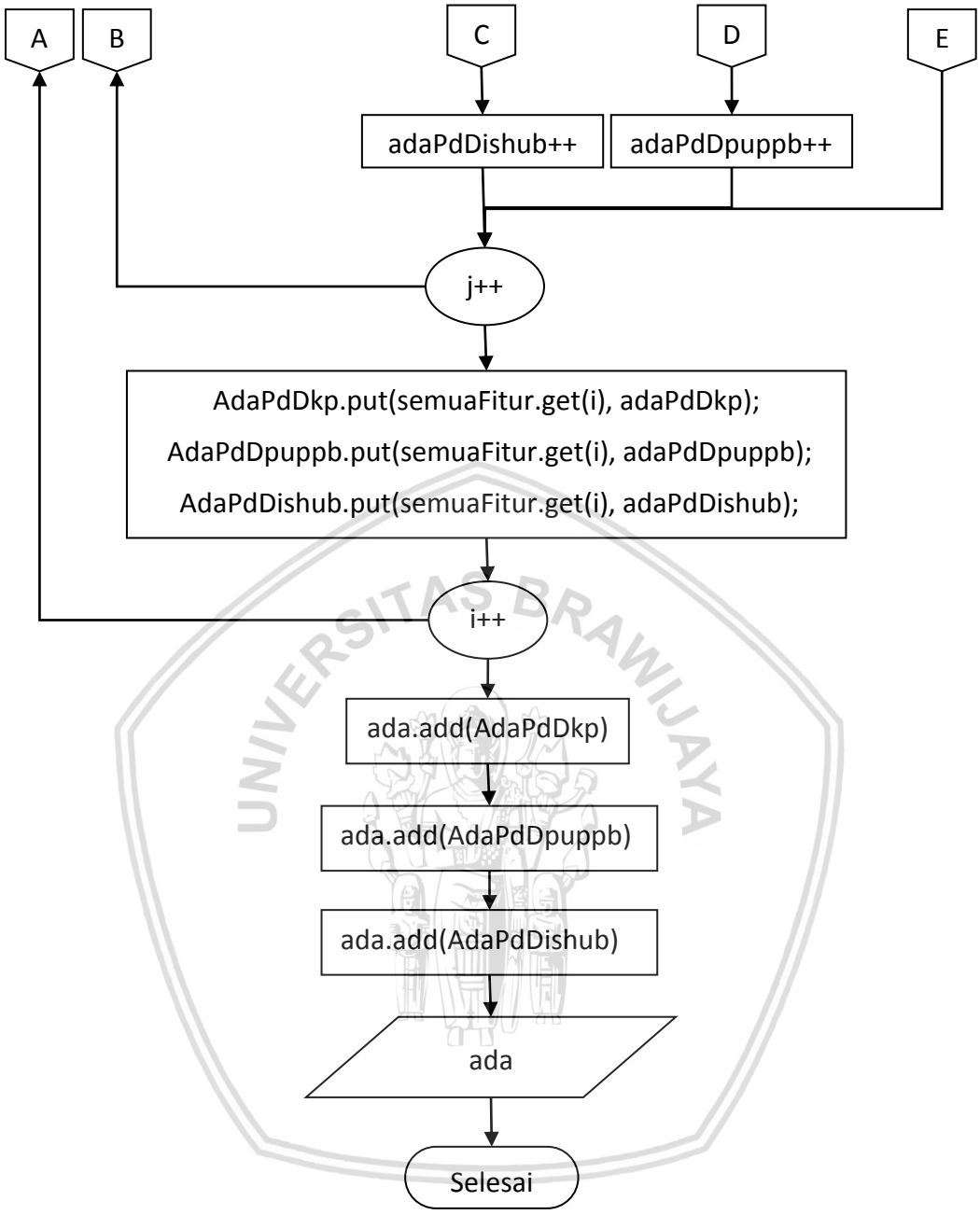
Seteleh dilakukan perhitungan *entropy global*, maka akan dilakukan perhitungan jumlah dokumen yang mengandung fitur tertentu pada masing-masing kategori. Pada proses information gain ini, perhitungan tersebut dilakukan dengan memanggil fungsi hitungAdanyaFiturPdDokumenTiapKelas(). Proses yang terdapat pada fungsi tersebut digambarkan seperti pada diagram alir Gambar 4.19.



Gambar 4.19 Diagram alir perhitungan adanya fitur pada dokumen tiap kelas

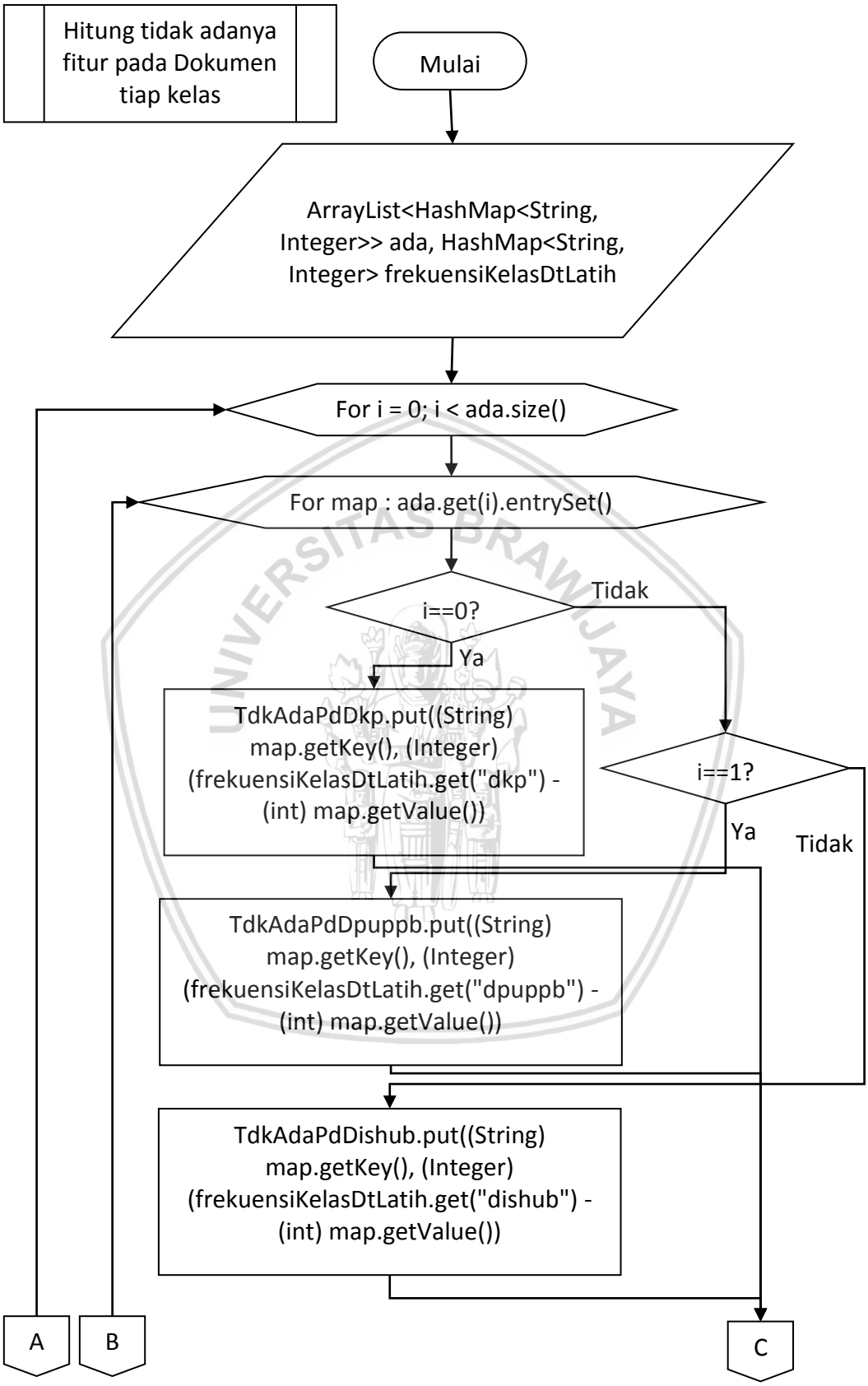


Gambar 4.19 Diagram alir perhitungan adanya fitur pada dokumen tiap kelas

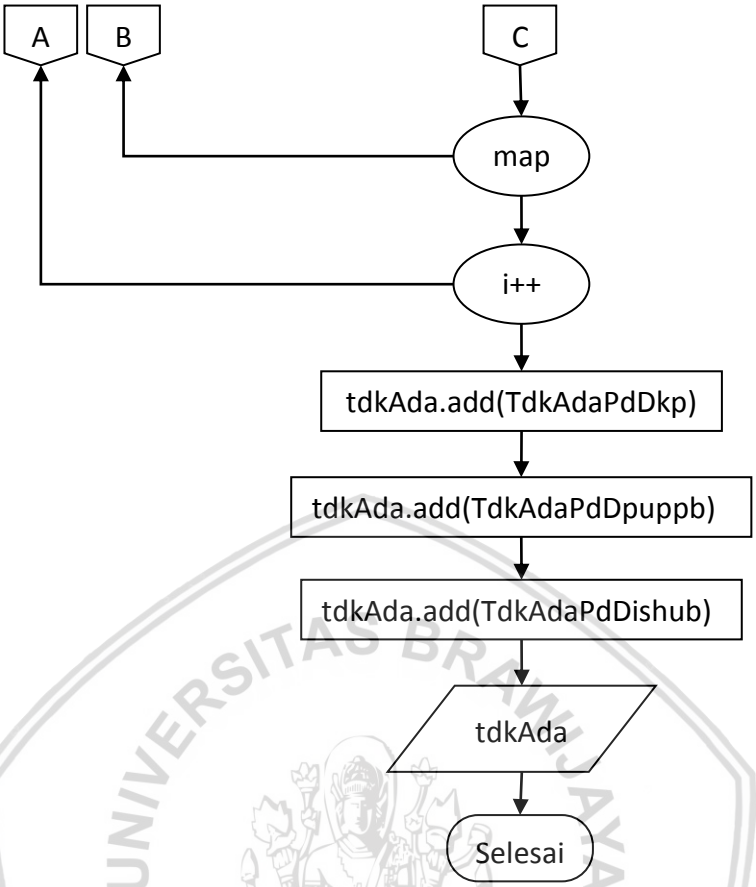


Gambar 4.19 Diagram alir perhitungan adanya fitur pada dokumen tiap kelas

Selain menghitung jumlah dokumen yang mengandung fitur tertentu pada setiap kategori, perhitungan *information gain* juga memerlukan perhitungan jumlah dokumen yang tidak mengandung fitur tertentu pada setiap kategori. Perhitungan ini dilakukan dengan memanggil fungsi hitungTdkAdanyaFiturPdDokumenTiapKelas(). Proses yang terjadi pada fungsi ini digambarkan pada Gambar 4.20.

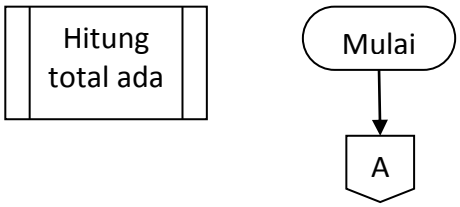


Gambar 4.20 Diagram alir perhitungan tidak adanya fitur pada dokumen tiap kelas

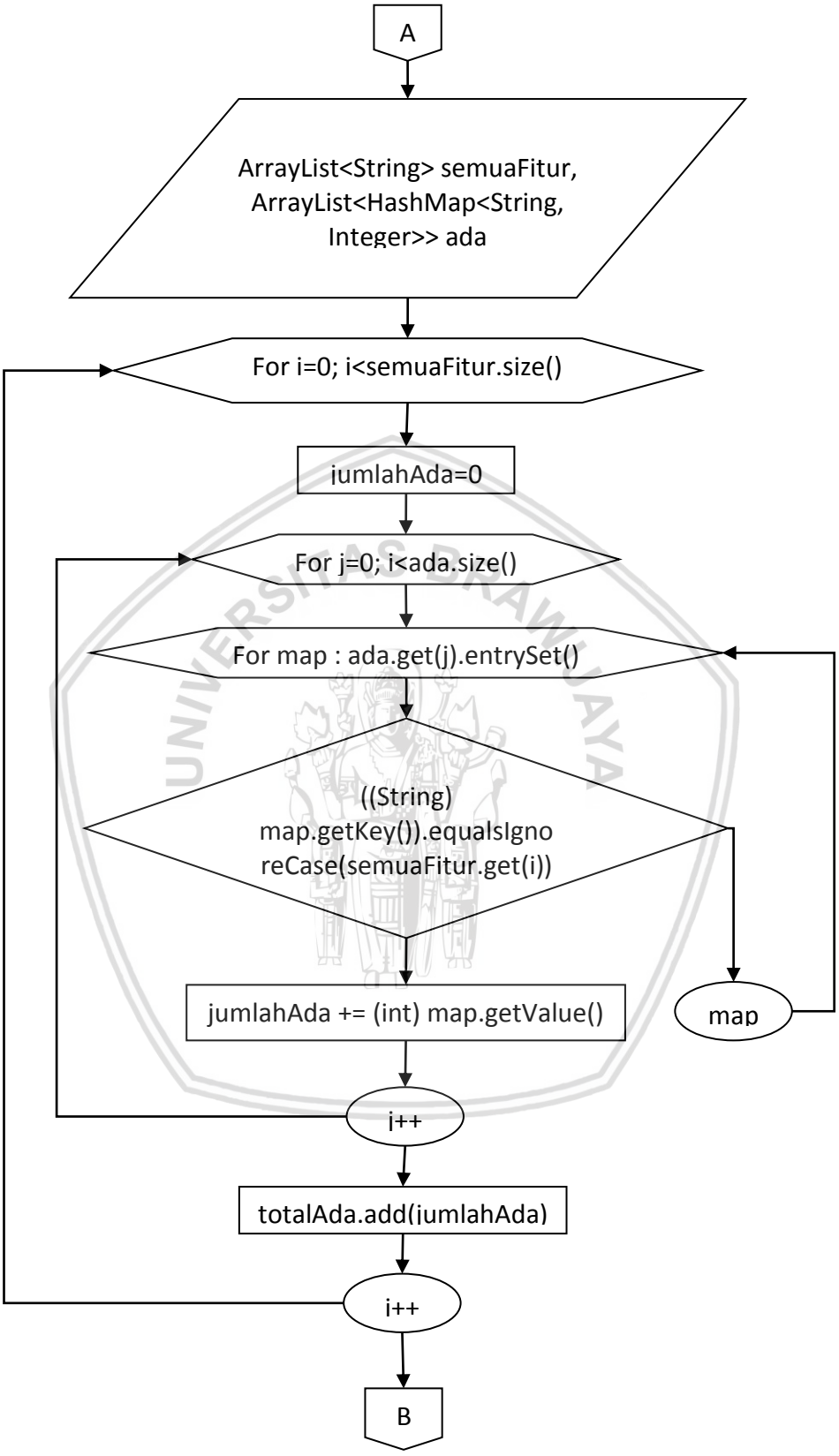


Gambar 4.20 Diagram alir perhitungan tidak adanya fitur pada dokumen tiap kelas

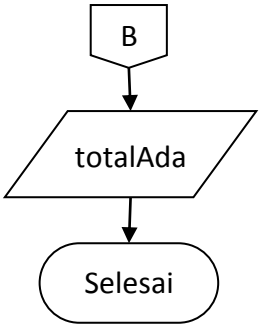
Pada proses perhitungan *information gain* juga dilakukan perhitungan total dokumen yang mendandung dan tidakmengandung fitur tertentu. Proses ini dijalankan setelah perhitungan ada dan tidak adanya fitur pada dokumen masing-masing kategori selesai dijalankan. Perhitungan total ada dan tidak adanya fitur pada seluruh dokumen ini dilakukan dengan memanggil fungsi hitungTotalAda() untuk menghitung jumlah dokumen yang mengandung fitur tertentu, dan hitungTotalTdkAda() untuk menghitung jumlah dokumen yang tidak mengandung fitur tertentu yang dijalankan setelah proses pada fungsi hitungTotalAda selesai dijalankan. Proses yang terdapat pada fungsi hitungTotalAda() dan hitungTotalTdkAda () secara berturut-turut digambarkan pada Diagram alir Gambar 4.21 dan 4.22.



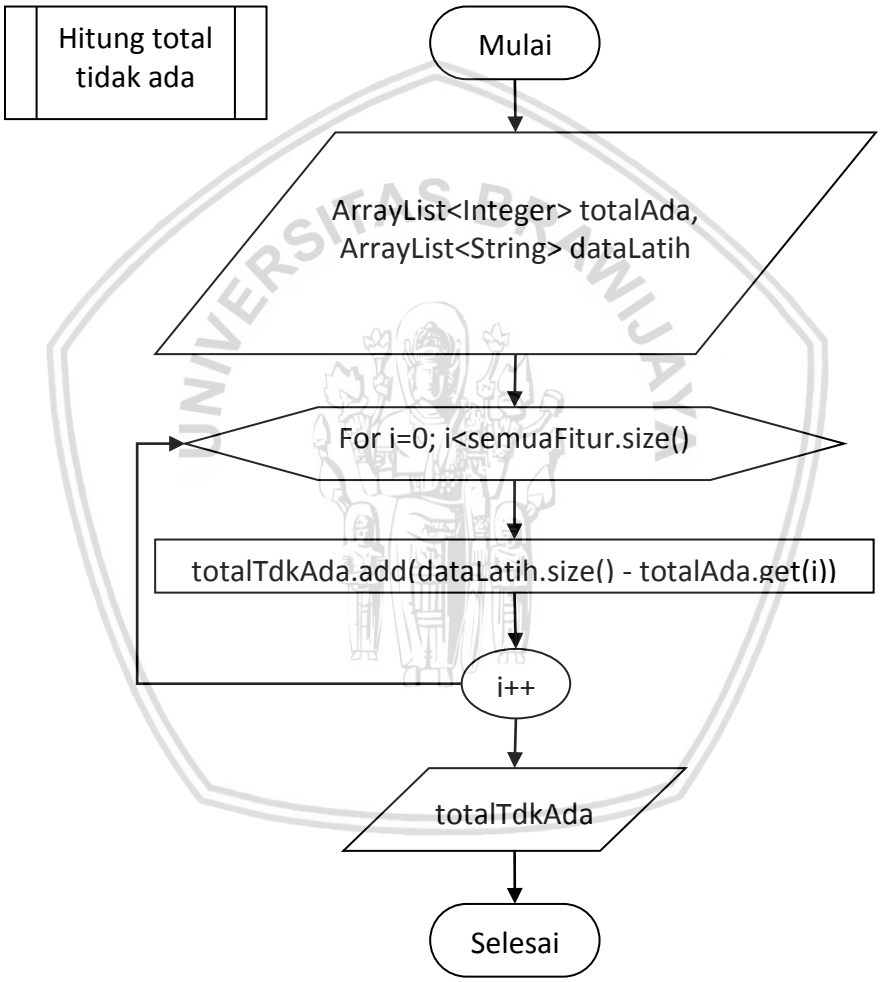
Gambar 4.21 Diagram alir perhitungan total ada



Gambar 4.21 Diagram alir perhitungan total ada

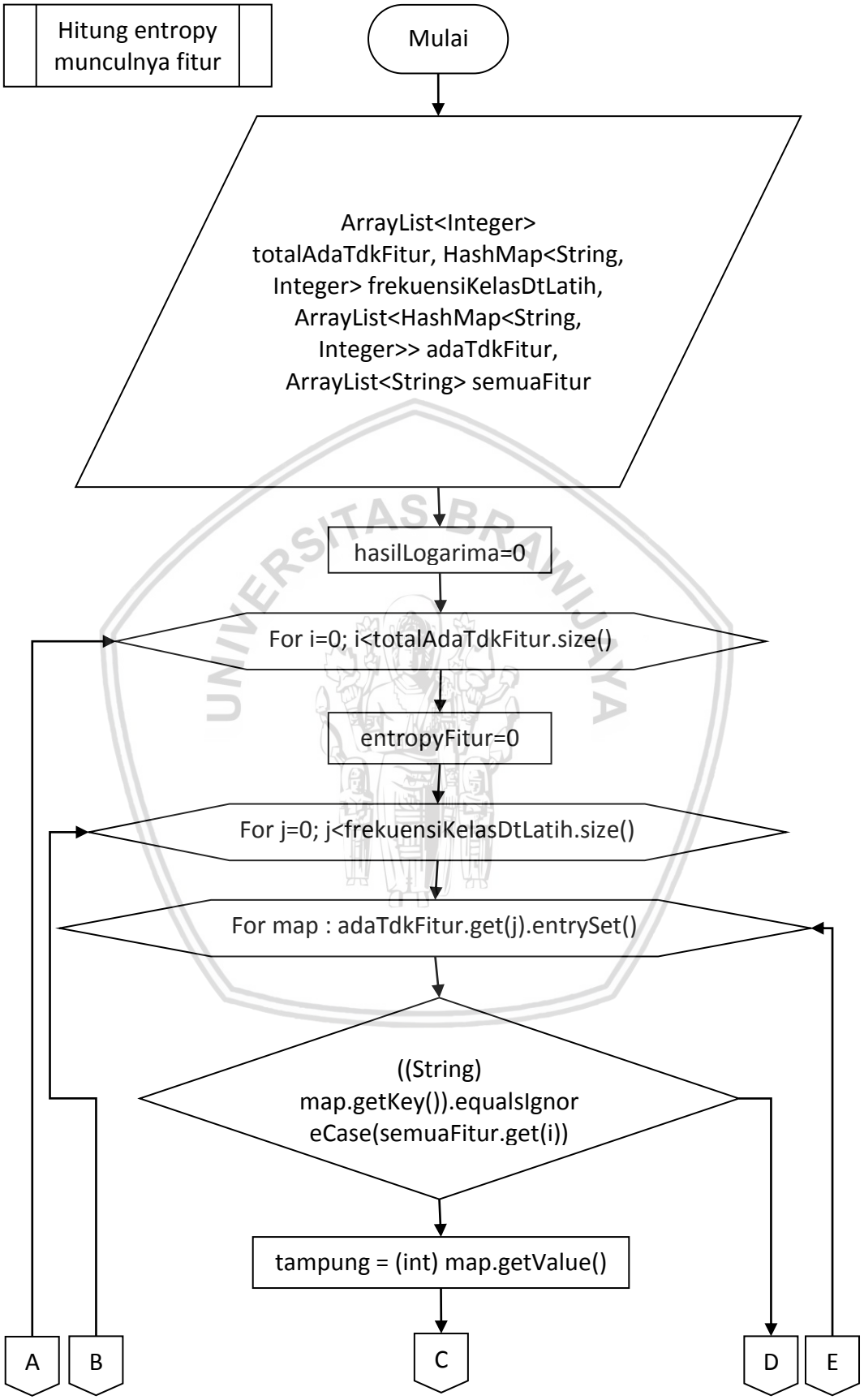


Gambar 4.21 Diagram alir perhitungan total ada

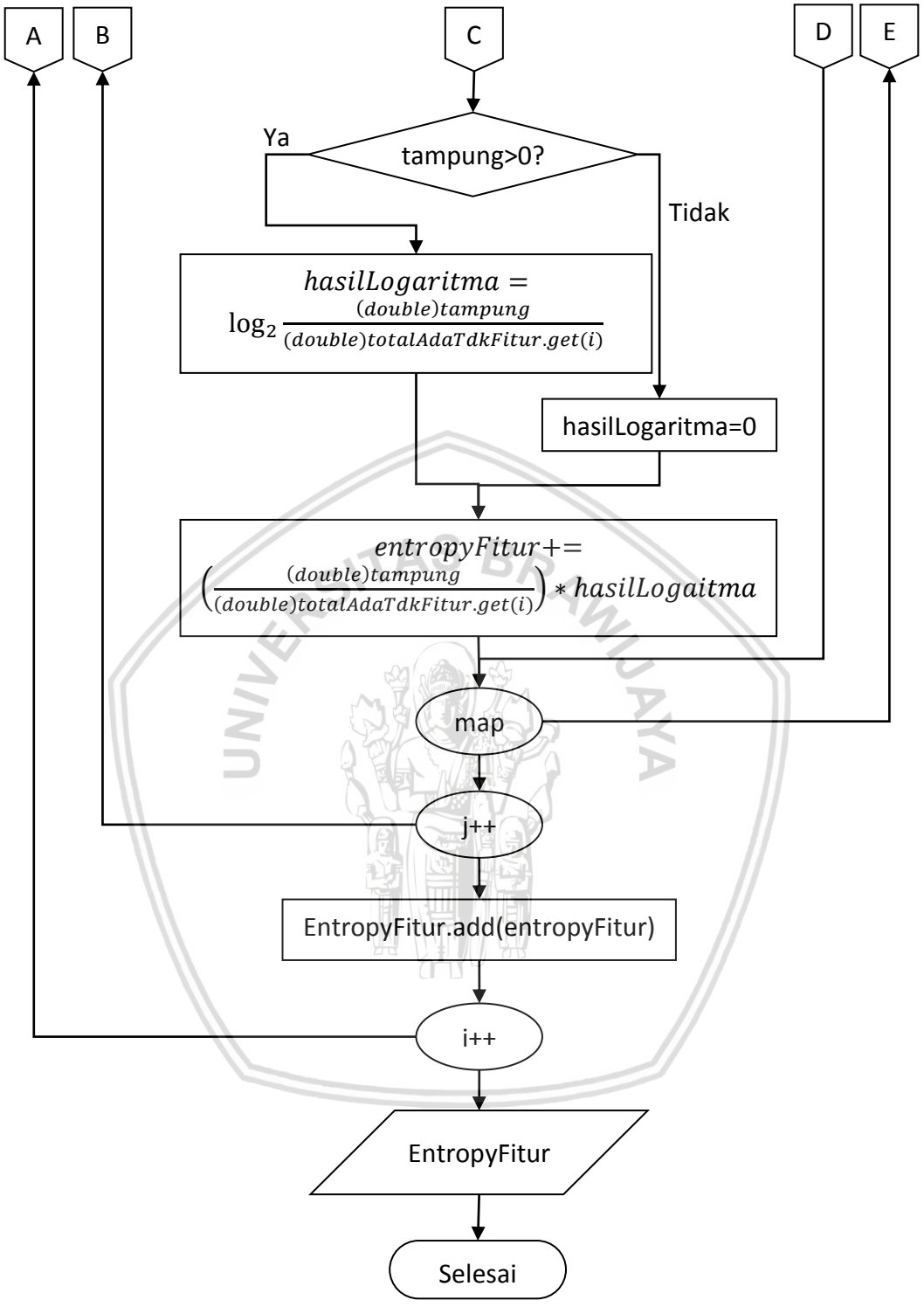


Gambar 4.22 Diagram alir perhitungan total tidak ada

Fungsi hitungMunculnya fitur dijalankan ketika proses perhitungan total tidak ada pada perhitungan *information gain* selesai dijalankan. Fungsi ini digunakan untuk menghitung *entropy* ada dan tidak adanya setiap fitur pada data latih yang digunakan, sehingga pemanggilan fungsi ini perlu dilakukan sebanyak dua kali dimana pemanggilan pertama untuk menghitung entropy ada, dan pemanggilan kedua untuk menghitung *entropy* tidak ada. Detail proses dari fungsi ini digambarkan pada Gambar 4.23.

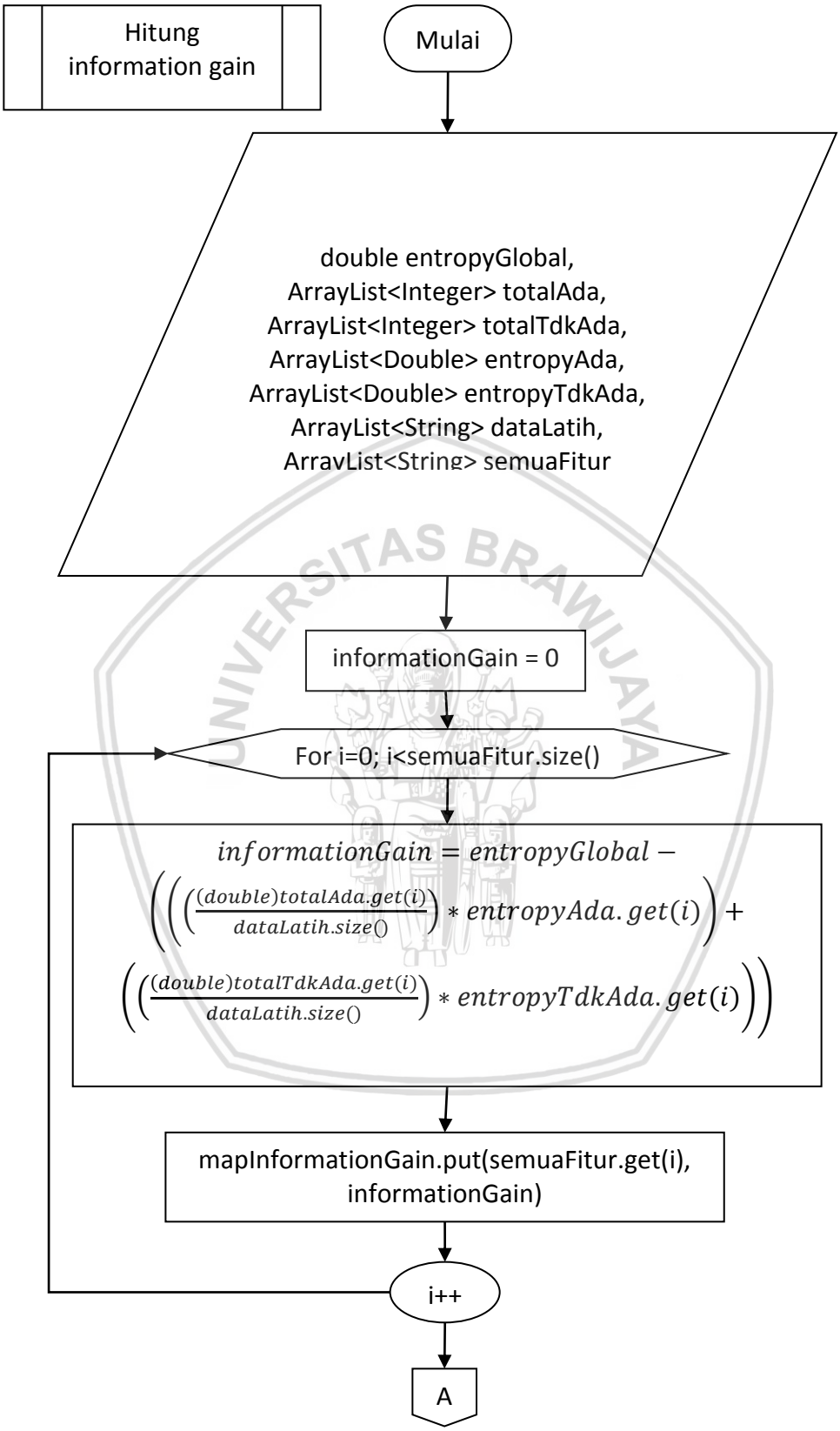


Gambar 4.23 Diagram alir perhitungan *entropy* munculnya fitur

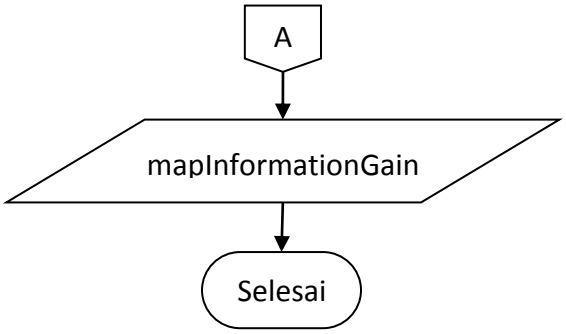


Gambar 4.23 Diagram alir perhitungan *entropy* munculnya fitur

Setelah dijalankan proses-proses tersebut, perhitungan *information gain* dari setiap fitur baru dapat dilakukan. Proses perhitungan information gain dilakukan dengan memanggil fungsi hitungInformationGain(), dimana fungsi tersebut memilikidetail proses seperti pada Gmabar 4.24.

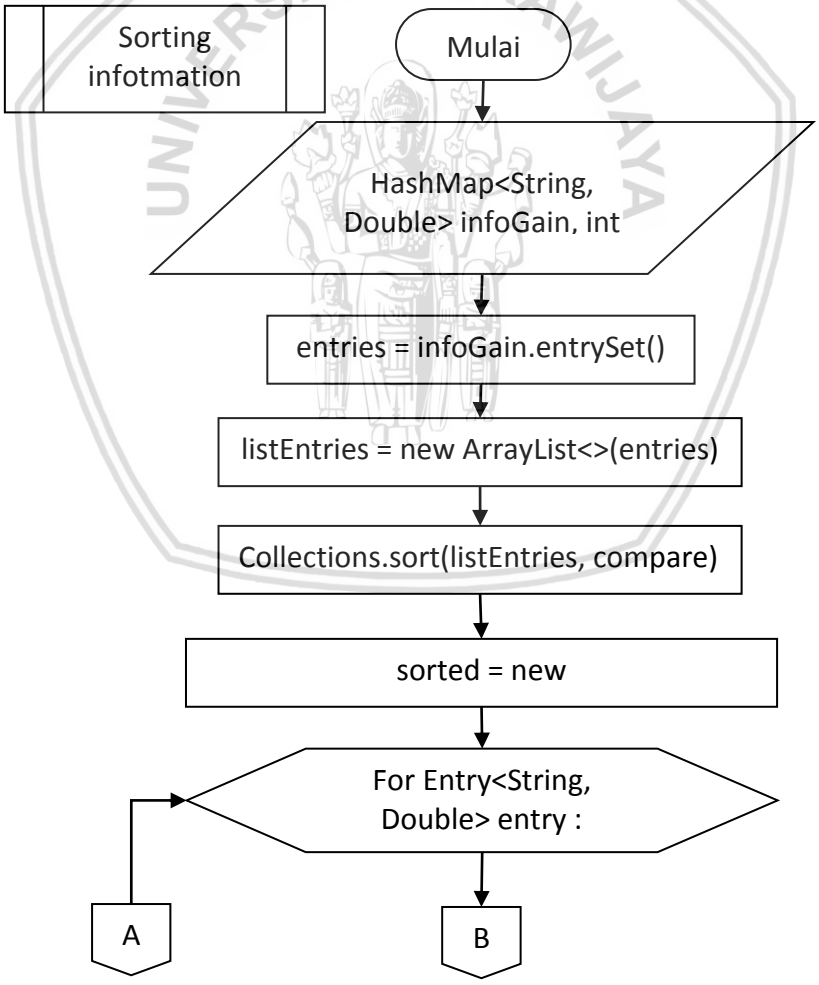


Gambar 4.24 Diagram alir perhitungan *information gain* setiap fitur

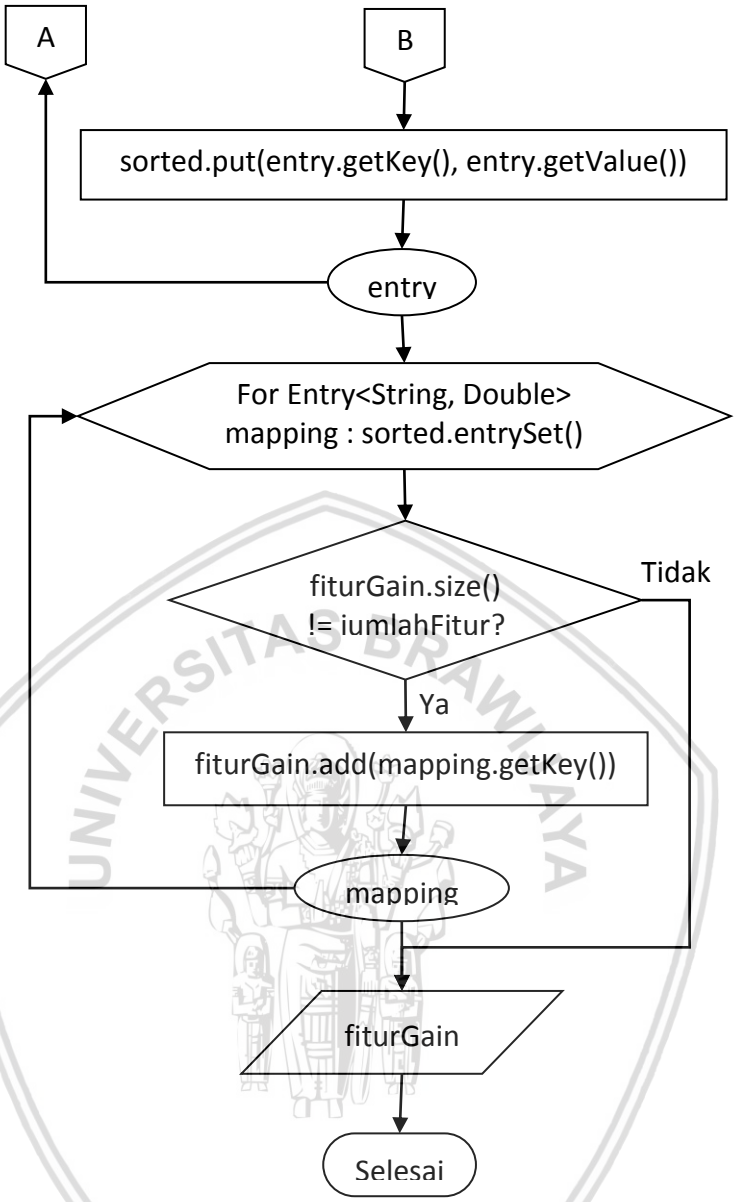


Gambar 4.24 Diagram alir perhitungan *information gain* setiap fitur

Setelah mendapatkan *information gain* setiap fitur, maka perlu dilakukan pengurutan berdasarkan besaranya *information gain* dari besar ke kecil untuk diambil sebesar nilai variable jumlahFitur fitur yang memiliki *information gain* tertinggi. Pengurutan ini dilakukan dengan memanggil fungsi sortingInformationGain(), dimana fungsi tersebut memiliki detail proses seperti pada Gambar 4.25.

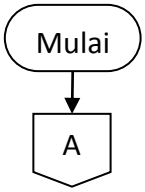


Gambar 4.25 Diagram alir pengurutan fitur berdasarkan *information gain*-nya

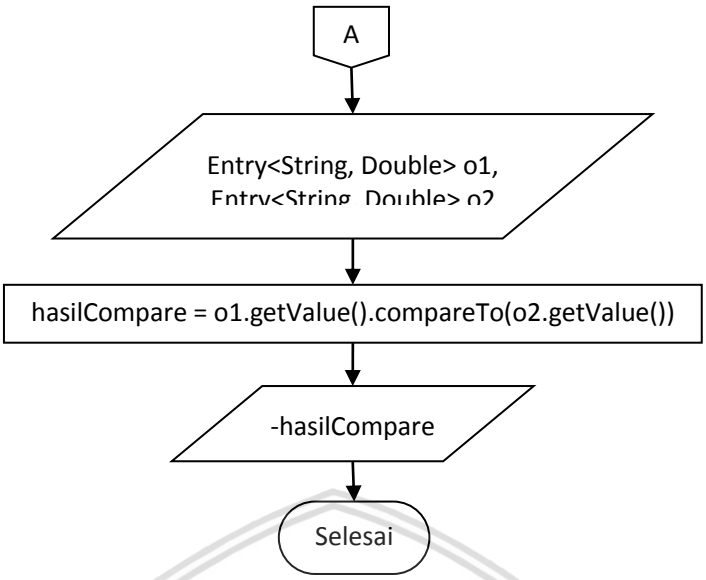


Gambar 4.25 Diagram alir pengurutan fitur berdasarkan *information gain*-nya

Pada gambar 4.25 terdapat pemanggilan *object compare*, dimana pemanggilan itu akan memproses perbandingan 2 object. Detail proses yang terjadi dari pemanggilan tersebut adalah seperti pada Gambar 4.26.



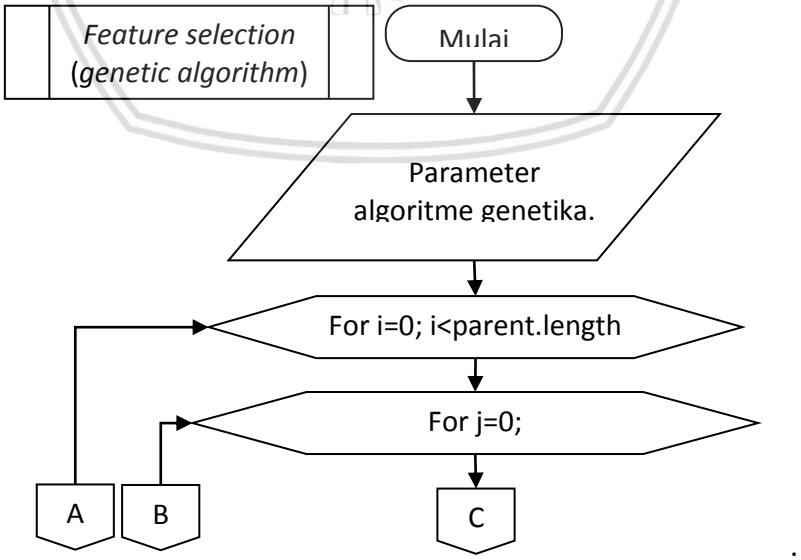
Gambar 4.26 Diagram alir proses pembandingan nilai *information gain*



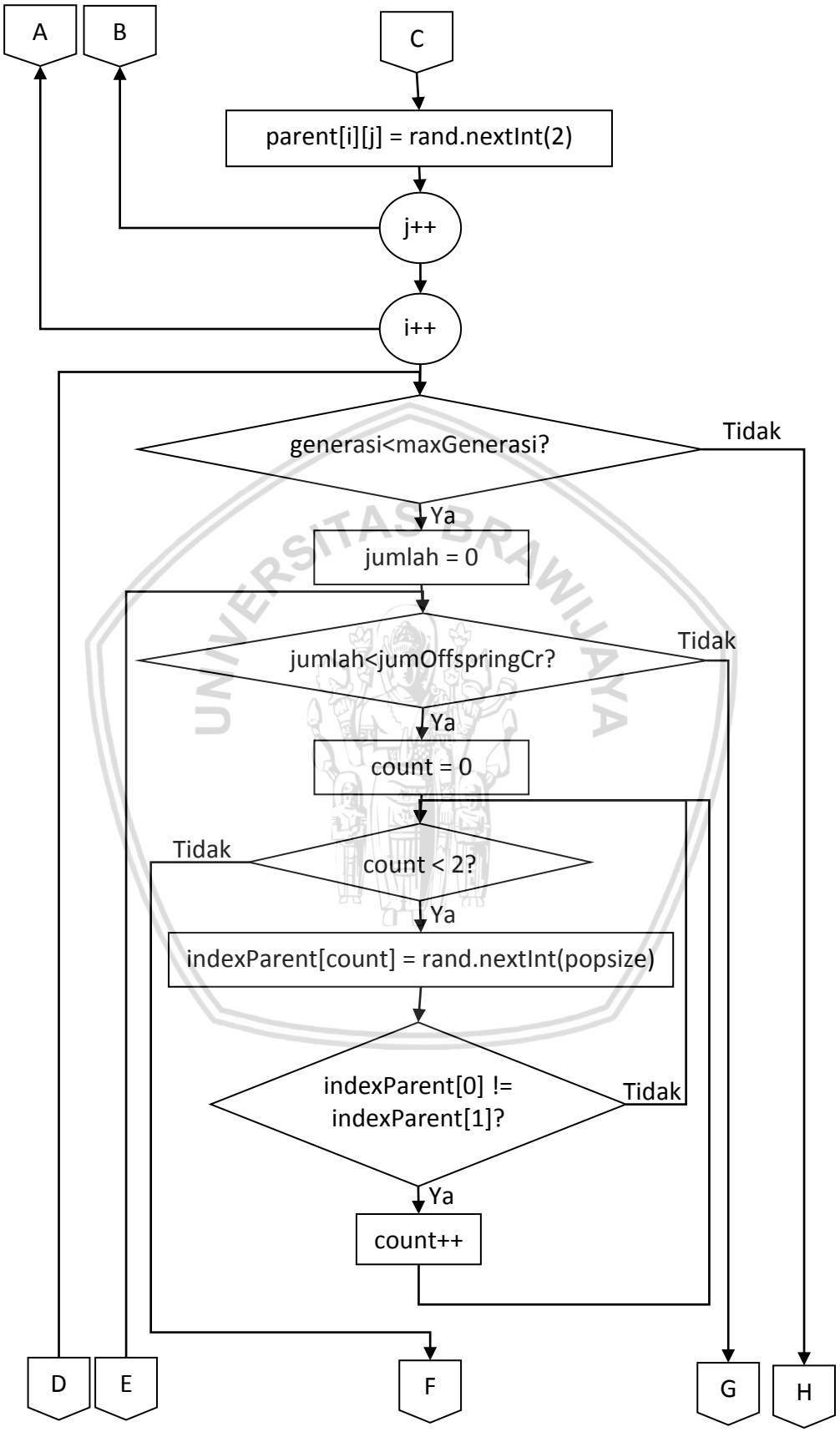
Gambar 4.26 Diagram alir proses pembandingan nilai *information gain*

4.1.3 Genetic Algorithm

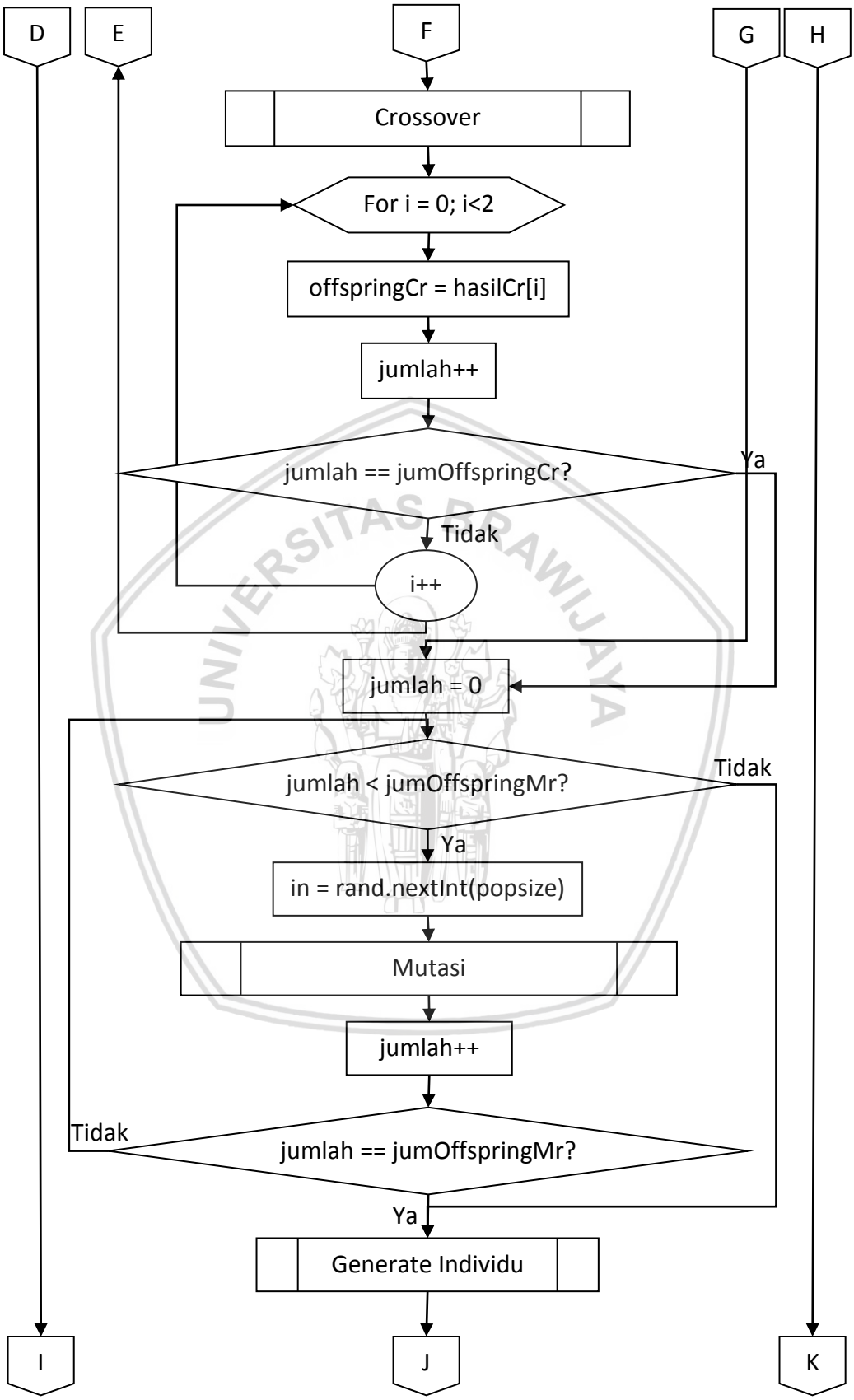
Pada penelitian ini, representasi kromosom yang digunakan adalah representasi kromosom biner. Setiap gen pada kromosom merepresentasikan terpakai atau tidaknya fitur. Nilai setiap gen dari kromosom bernilai 1 atau 0, dimana gen yang bernilai 1 menunjukkan bahwa fitur tersebut digunakan, dan gen bernilai 0 menunjukkan bahwa fitur tersebut tidak digunakan. Panjang kromosom yang digunakan adalah dinamis yaitu sesuai dengan jumlah fitur hasil seleksi dari *information gain* yaitu sejumlah persentase jumlah fitur yang digunakan yang bisa berubah. Tahapan proses seleksi fitur menggunakan algoritme genetika ini digambarkan pada Gambar 4.27.



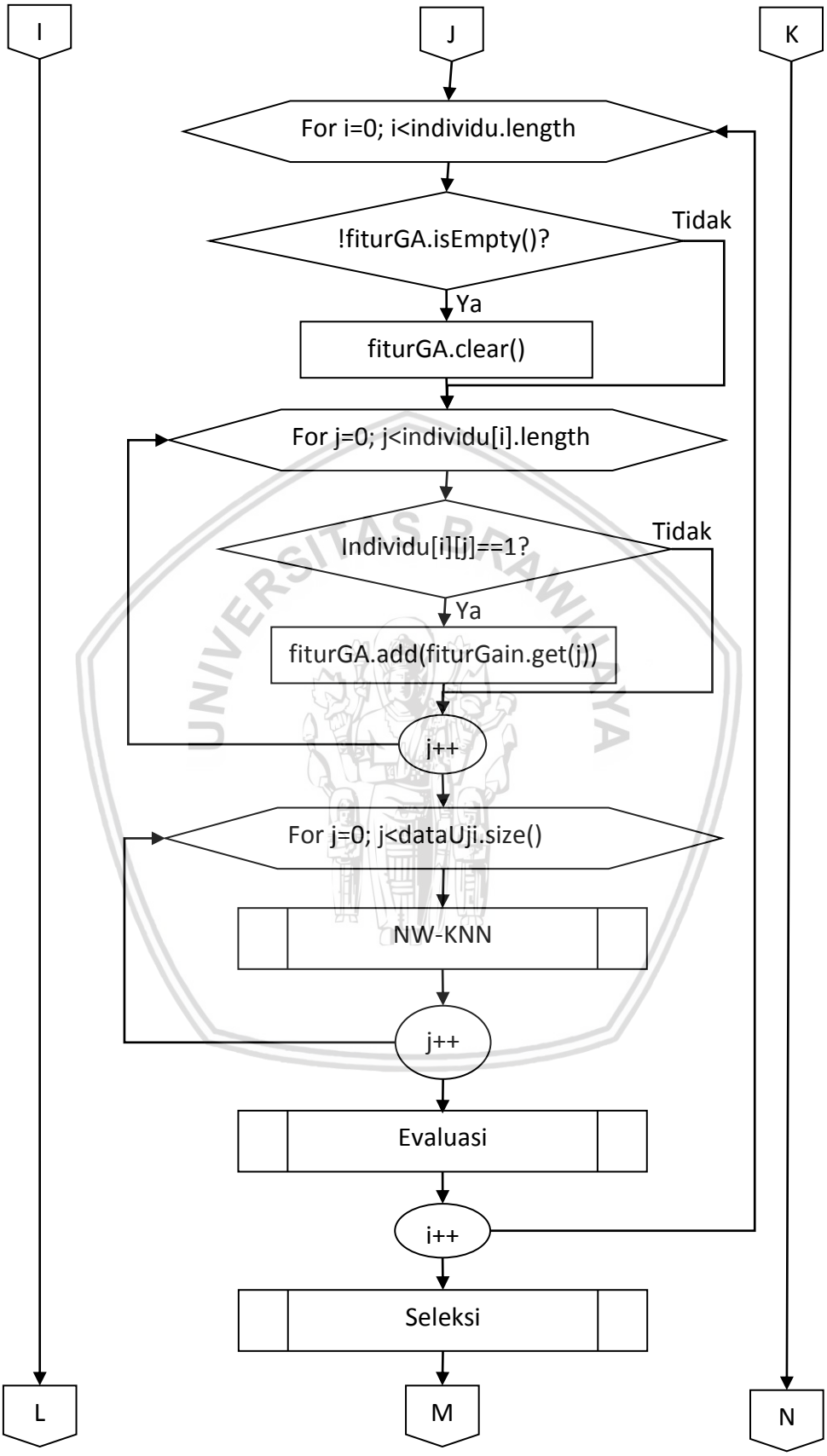
Gambar 4.27 Diagram alir proses *genetic algorithm*



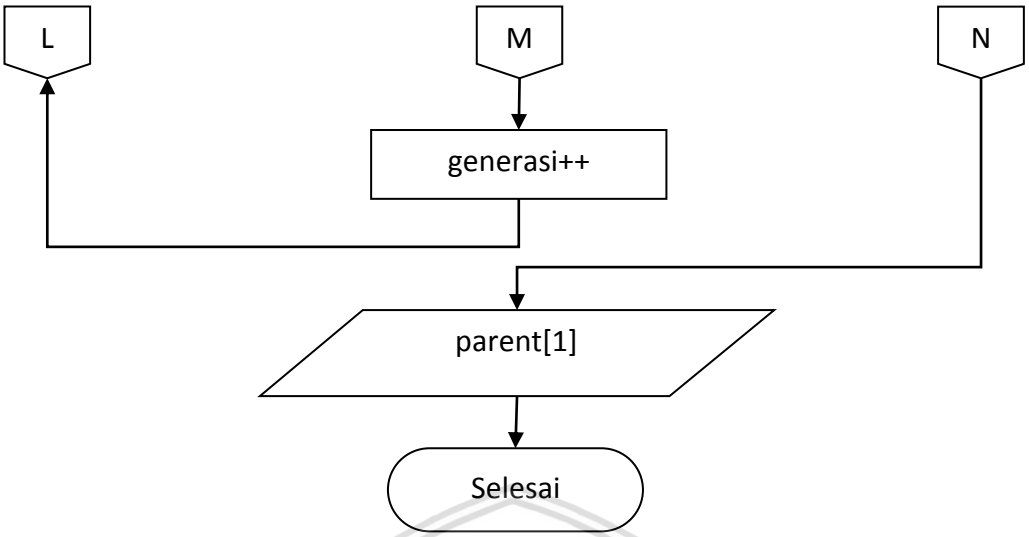
Gambar 4.27 Diagram alir proses *genetic algorithm*



Gambar 4.27 Diagram alir proses *genetic algorithm*



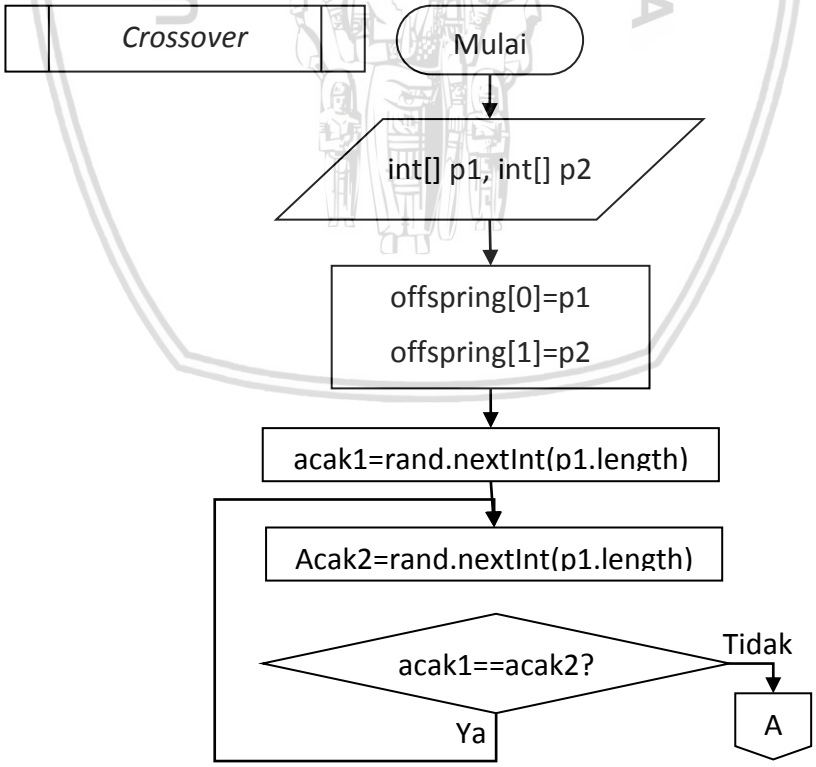
Gambar 4.27 Diagram alir proses *genetic algorithm*



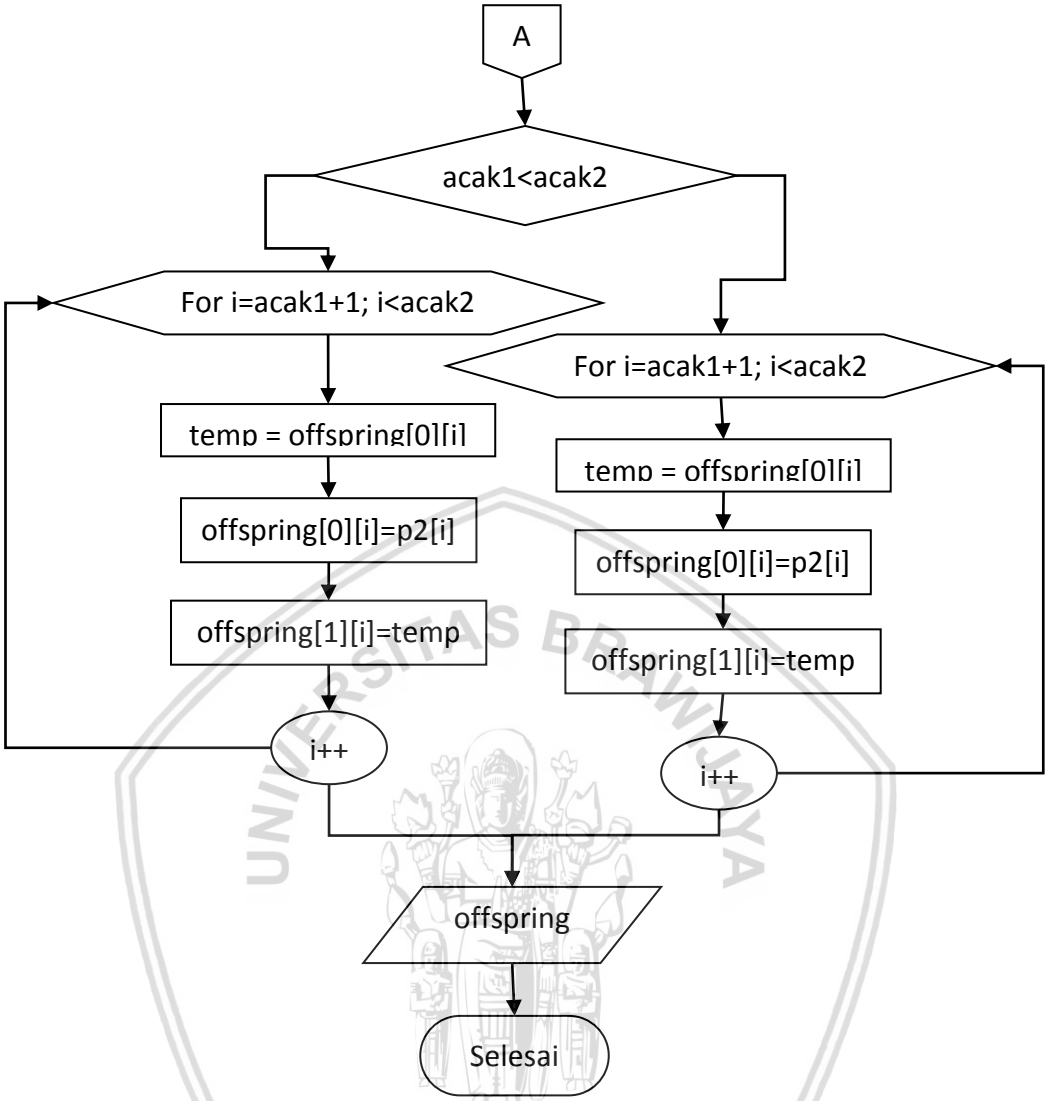
Gambar 4.27 Diagram alir proses *genetic algorithm*

4.1.3.1 *Crossover*

Crossover merupakan salah satu cara reproduksi dalam algoritme genetika yang melibatkan dua *parent* secara acak. Pada penelitian ini teknik crossover yang digunakan adalah *two cut point crossover*. Alur proses crossover ini digambarkan pada gambar 4.28.



Gambar 4.28 Diagram alir proses *crossover*

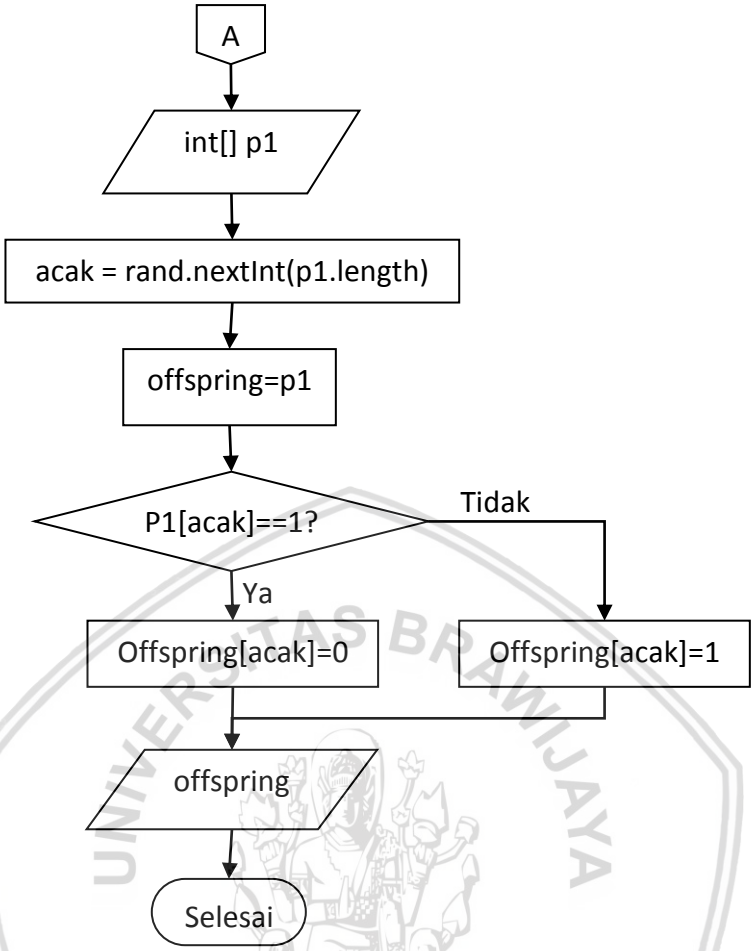


Gambar 4.28 Diagram alir proses *crossover*

4.1.3.2 Mutasi

Pada penelitian ini mutasi dilakukan dengan mengubah nilai bit biner yang ditentukan secara *random* menjadi nilai sebaliknya dengan kata lain melakukan inversi bit biner pada gen yang ditentukan secara random. Proses inversi yang akan terjadi pada proses mutasi ini adalah nilai 0 menjadi 1 dan nilai 1 menjadi 0. Alur proses mutasi pada penelitian ini digambarkan pada Gambar 4.29.

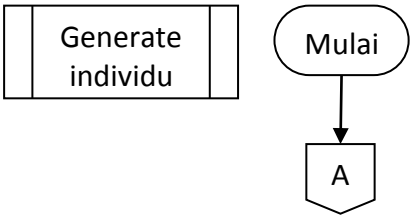
Gambar 4.29 Diagram alir proses mutasi



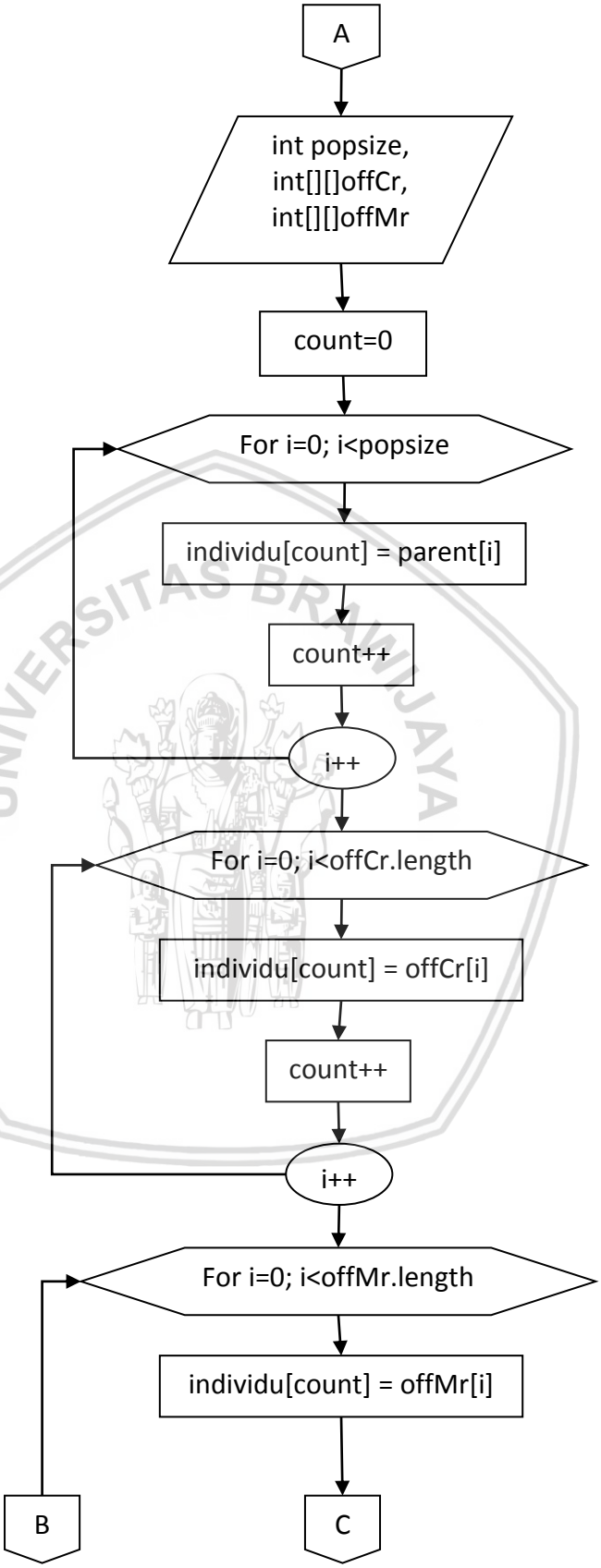
Gambar 4.29 Diagram alir proses mutasi

4.1.3.3 Evaluasi

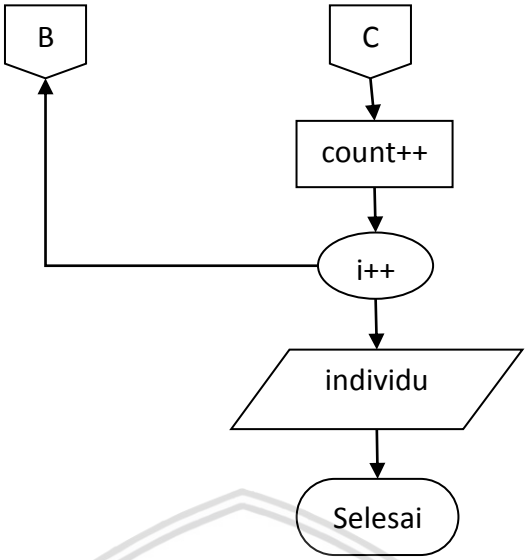
Evaluasi dilakukan dengan menghitung *fitness.* Nilai fitness didapatkan dari perhitungan *f-measure*. Nilai *f-measure* diperolah dari proses pengklasifikasian menggunakan NW-KNN. Sebelum melakukan evaluasi, perlu dilakukan pengumpulan seluruh individu dalam satu kelompok sebagai persiapan evaluasi. Proses pengumpulan ini dilakukandengan memanggil fungsi generateIndividu() yang memiliki prosesdetailseperti yang digambarkan pada Gambar 4.30.



Gambar 4.30 Diagram alir proses *generate individu*

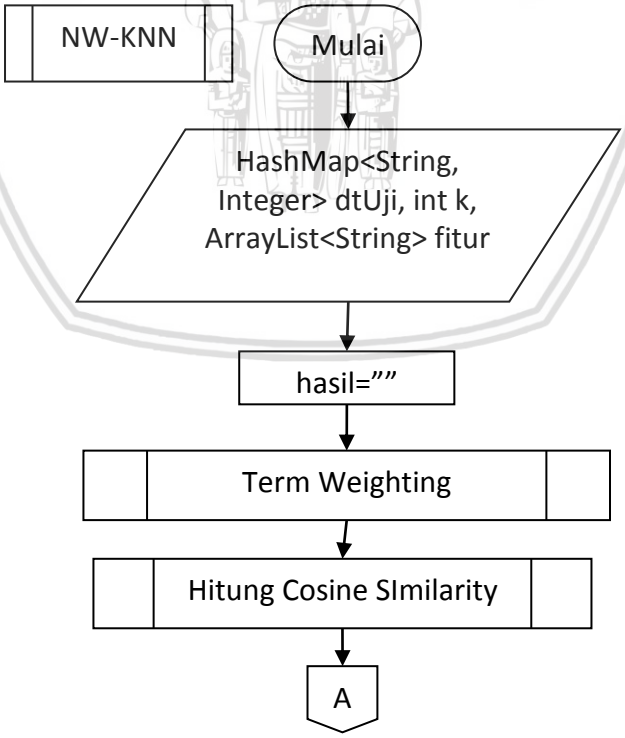


Gambar 4.30 Diagram alir proses *generate individu*

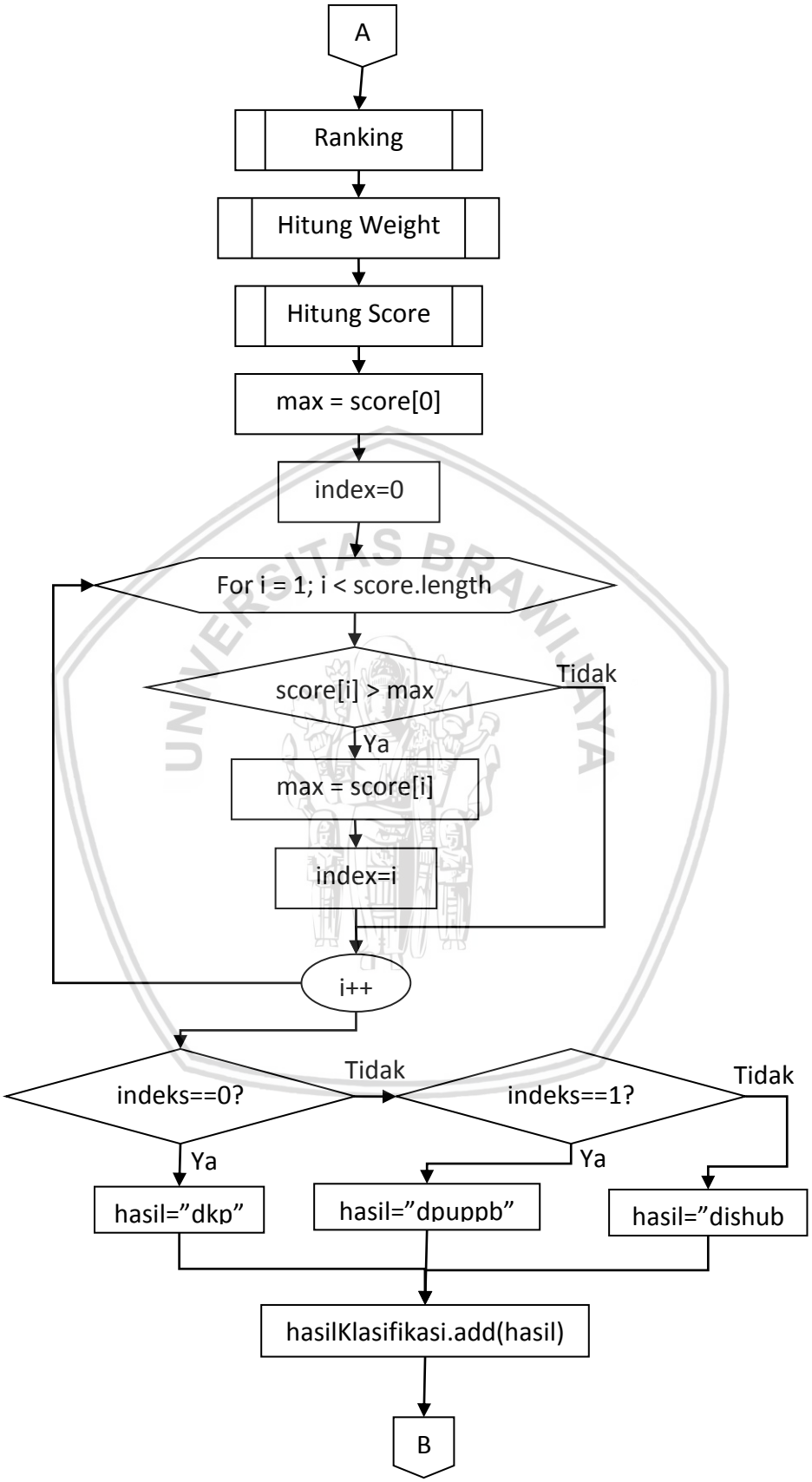


Gambar 4.30 Diagram alir proses *generate individu*

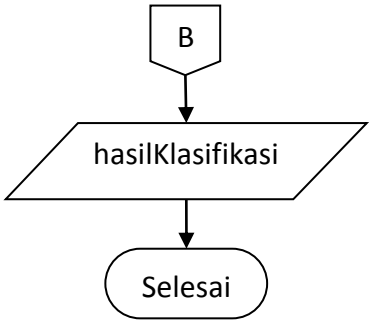
Pada gambar 4.30, terdapat offCr dan OffMr yang merupakan sebuah *array* yang terdiri dari individu-individu hasil proses *crossover* dan mutasi secara berturut-turut. Setelah seluruh individu dikumpulkan dalam satu kelompok,maka akan dilakukan proses klasifikasi dengan menggunakan NW-KNN. Proses klasifikasi ini melibatkan fitur yang memiliki nilai gen 1. Secara detail proses NW-KNN digambarkan seperti pada Gambar 4.31.



Gambar 4.31 Diagram alir proses klasifikasi dengan NW-KNN

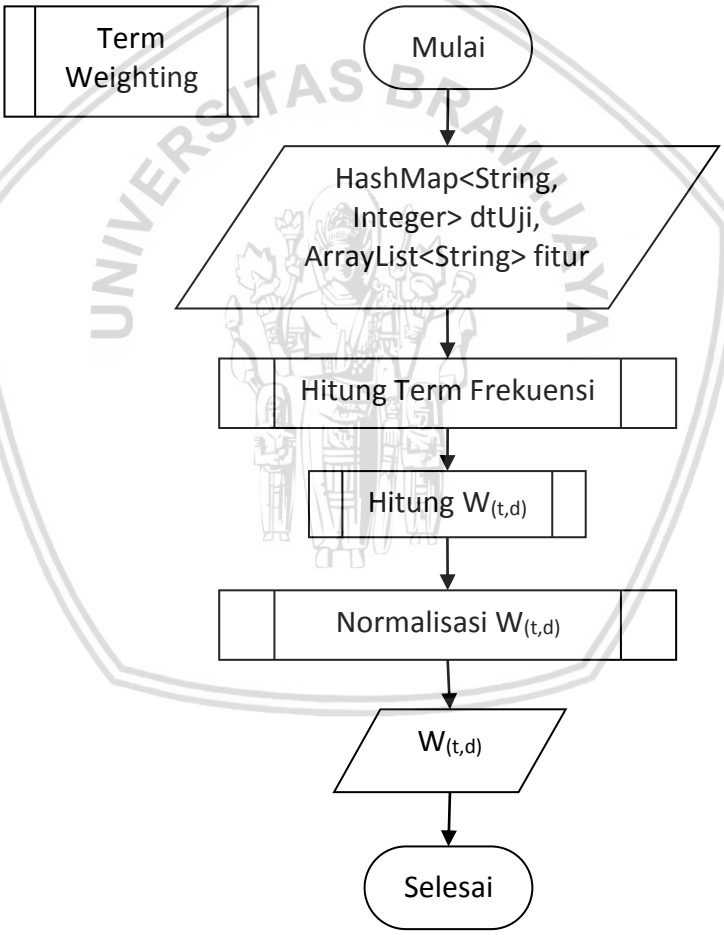


Gambar 4.31 Diagram alir proses klasifikasi dengan NW-KNN



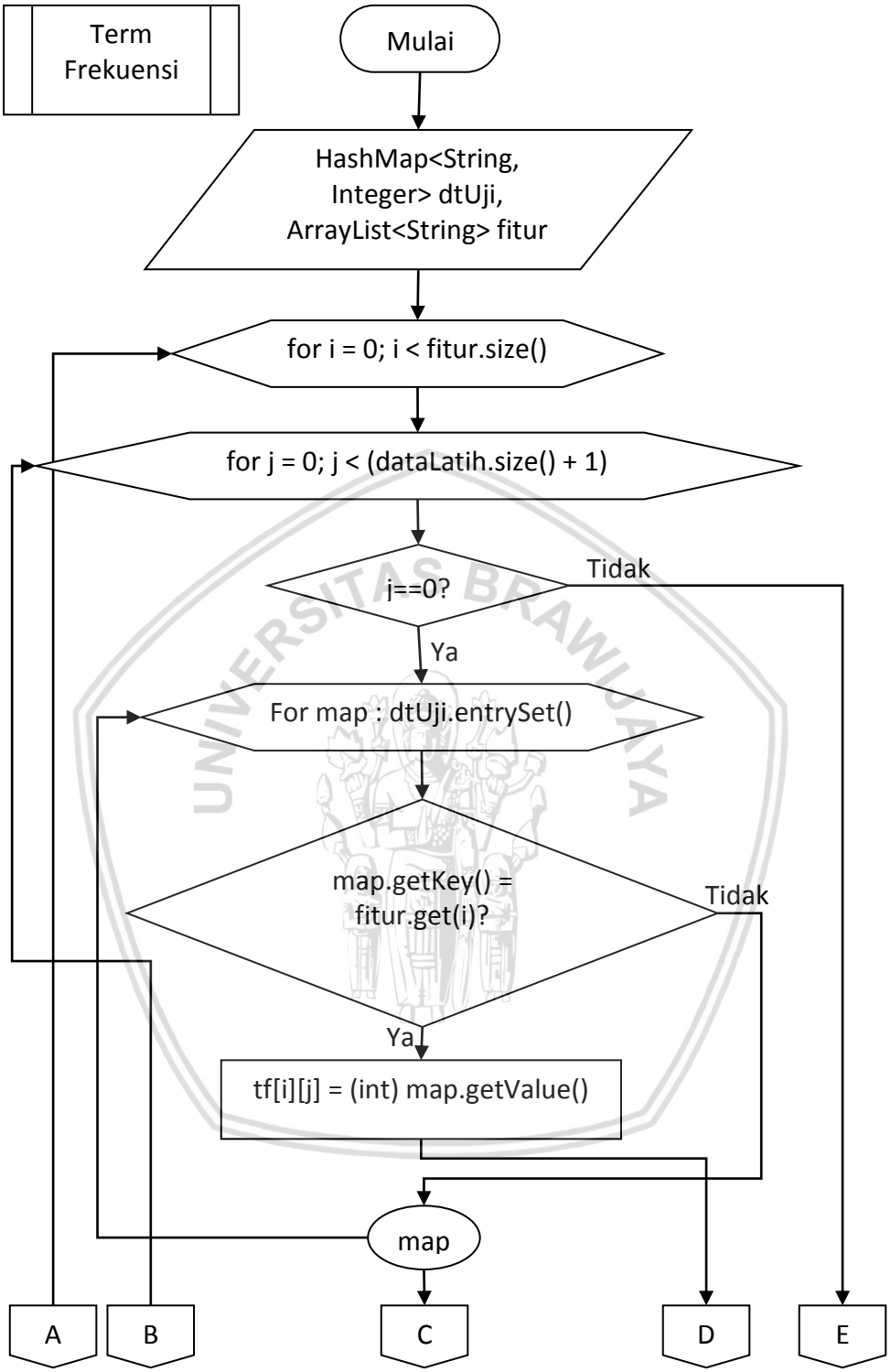
Gambar 4.31 Diagram alir proses klasifikasi dengan NW-KNN

Pada proses klasifikasi diatas, terdapat proses *term weighting* (pembobotan term). Proses *term weighting* tersebut secara detail digambarkan pada Gambar 4.32.

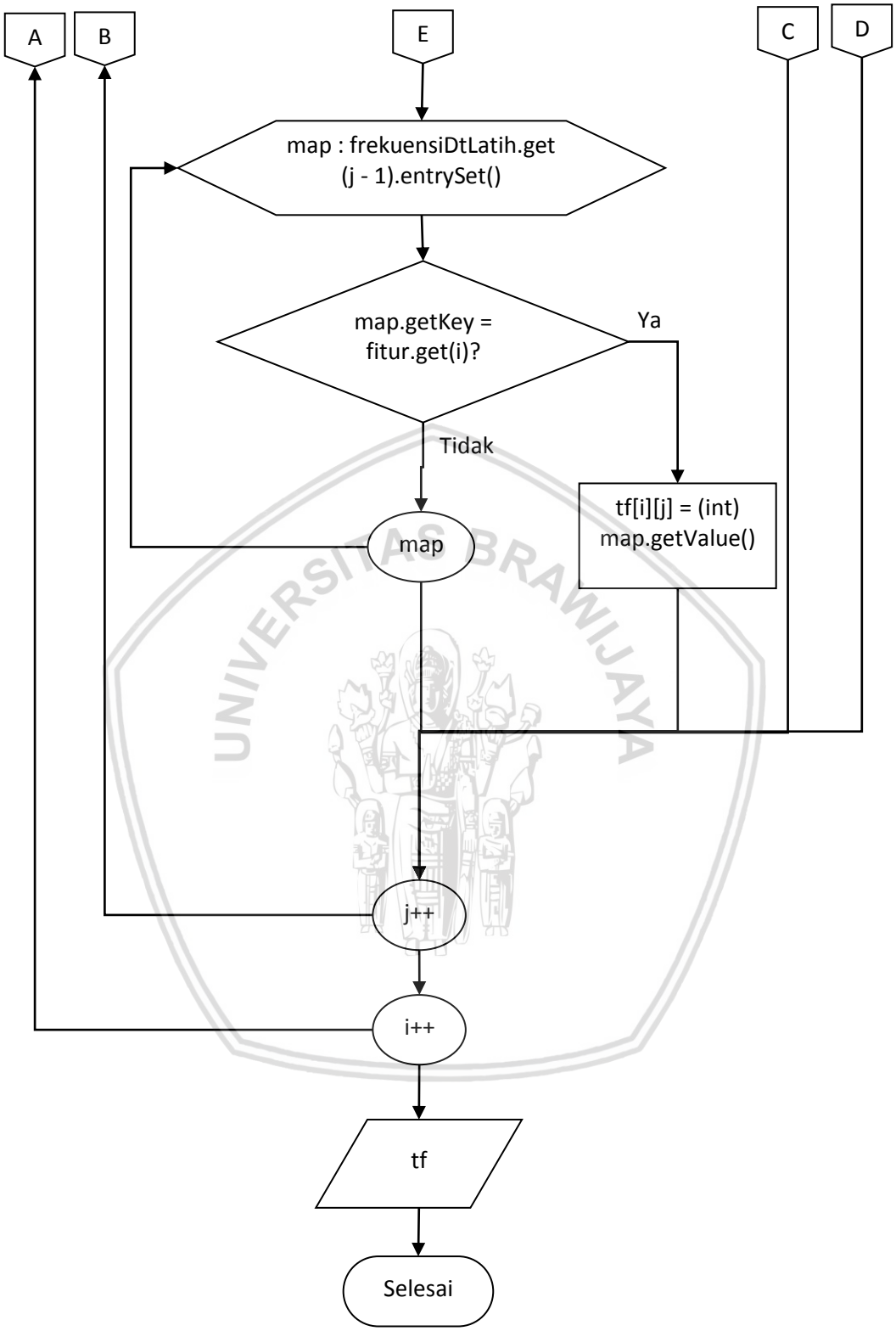


Gambar 4.32 Diagram alir proses *term weighting*

Pada proses *term weighting* pada Gambar 4.31 terdapat proses term frekuensi yang digunakan untuk menghitung frekuensi fitur dari data uji. Selain itu juga terdapat perhitungan $W_{(t,d)}$ yang merupakan perhitungan bobot term terhadap dokumen yang kemudian akan dinormalisasi dengan pemanggilan fungsi WtdNormal(). Detail proses *term frequency* digambarkan pada Gambar 4.33.

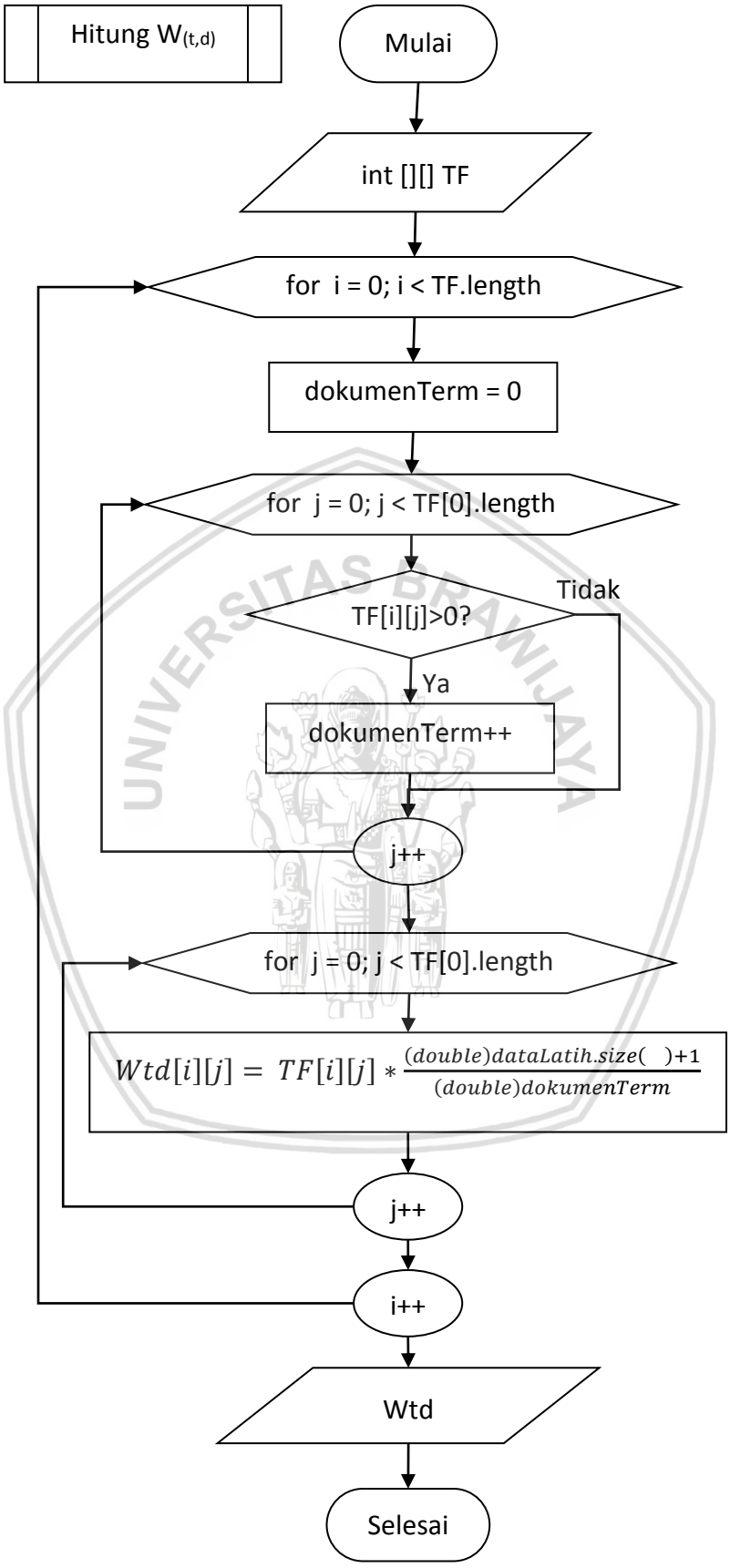


Gambar 4.33 Diagram alir proses *term frequency*



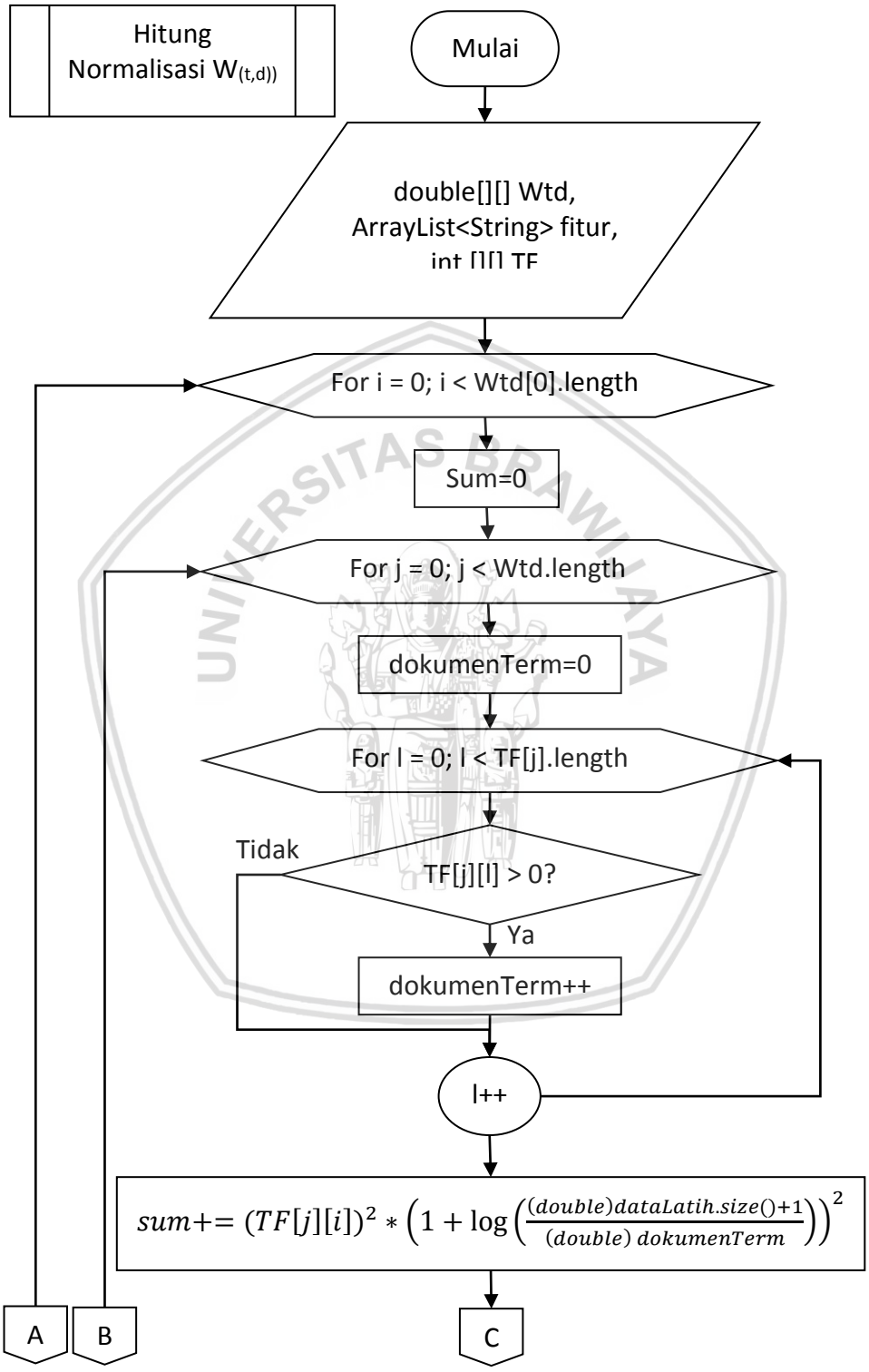
Gambar 4.33 Diagram alir proses *term frequency*

Hasil dari Term Frekuensi tersebut digunakan untuk menghitung bobot term terhadap dokumen $W_{(t,d)}$. Detail proses yang terjadi dalam perhitungan $W_{(t,d)}$ digambarkan pada Gambar 4.34.

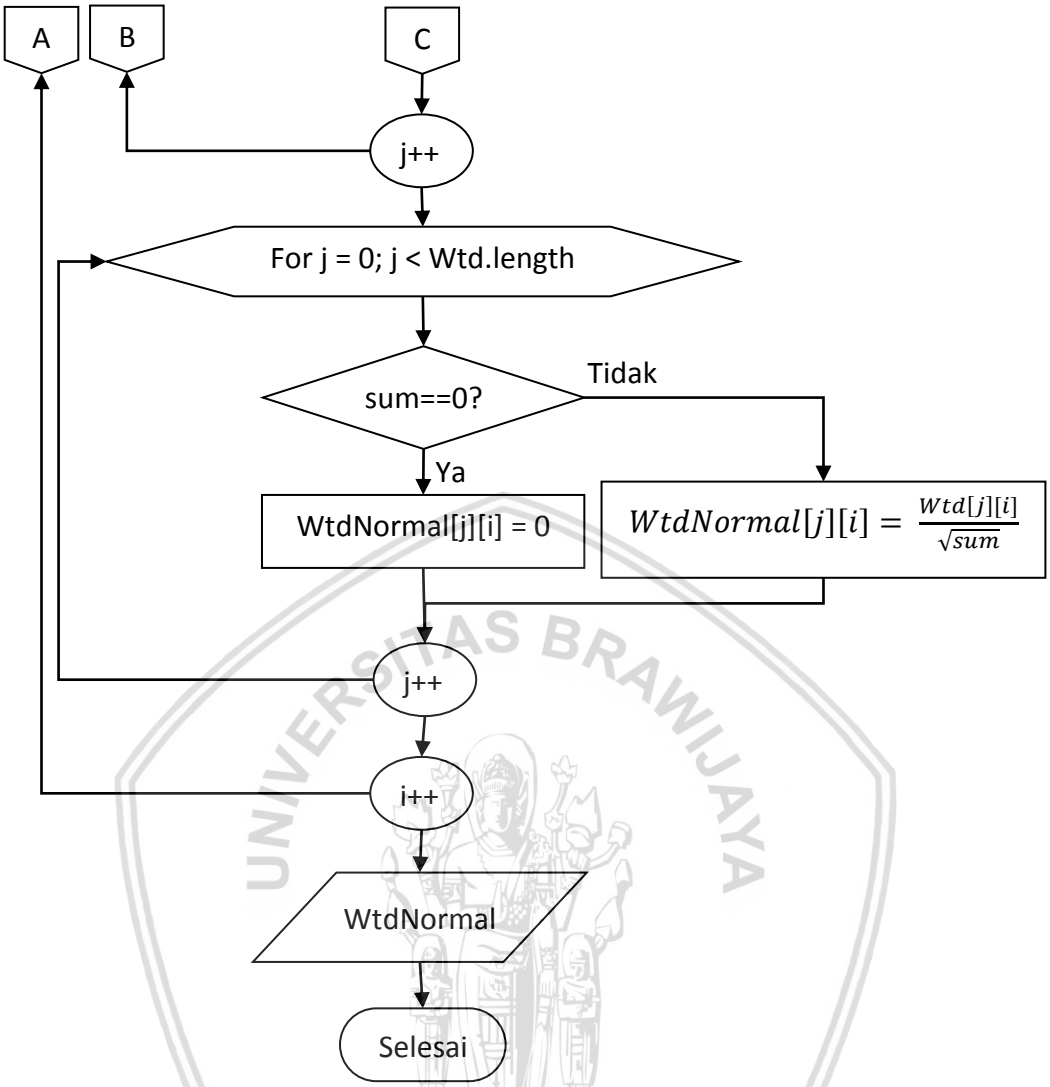


Gambar 4.34 Diagram alir proses perhitungan $W_{(t,d)}$

Hasil $W_{(t,d)}$) yang telahdidapatkan dariproses hitung $W_{(t,d)}$ keludiandilakukan normalisasi untukmendapatkan nilai $_{(t,d)}$ dalamrentang 0 hingga 1. Proses normalisasi yang dilakukan digambarkanpada Gambar 4.35.

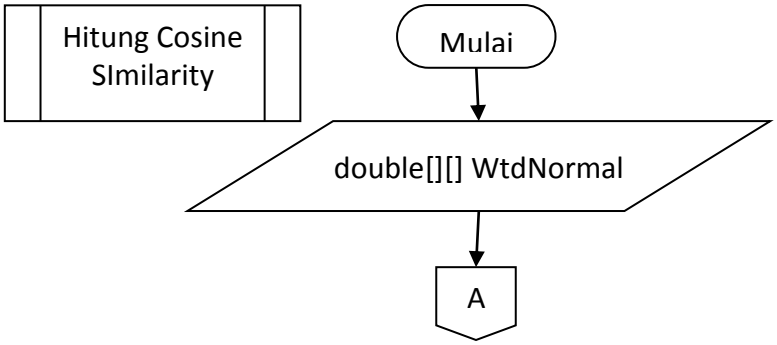


Gambar 4.35 Diagram alir proses normalisasi $W_{(t,d)}$

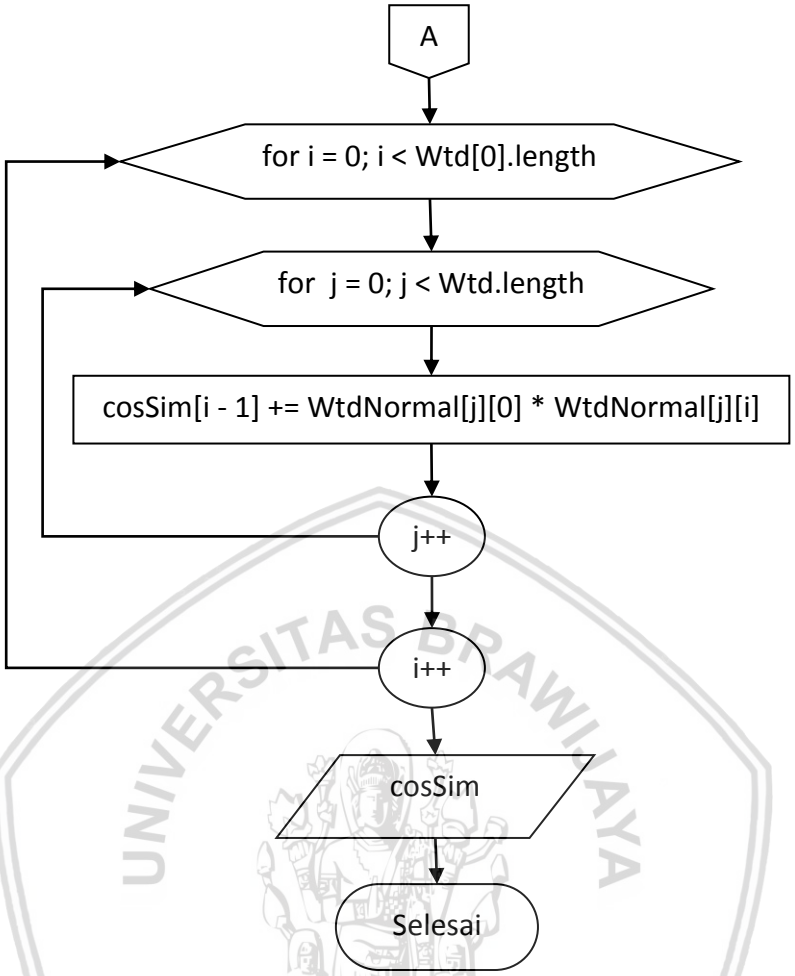


Gambar 4.35 Diagram alir proses normalisasi $W_{(t,d)}$

Setelah dilakukan proses term weighting, dengan nilai $W_{(t,d)}$ yang dihasilkandapat dihitung nilai *cosine similarity*. Perhitungan *cosine similariry* digambarkan pada Gambar 4.36.

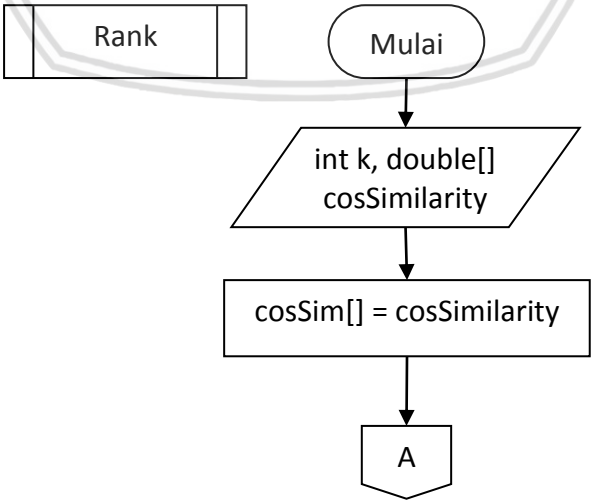


Gambar 4.36 Diagram alir proses perhitungan *cosine similarity*

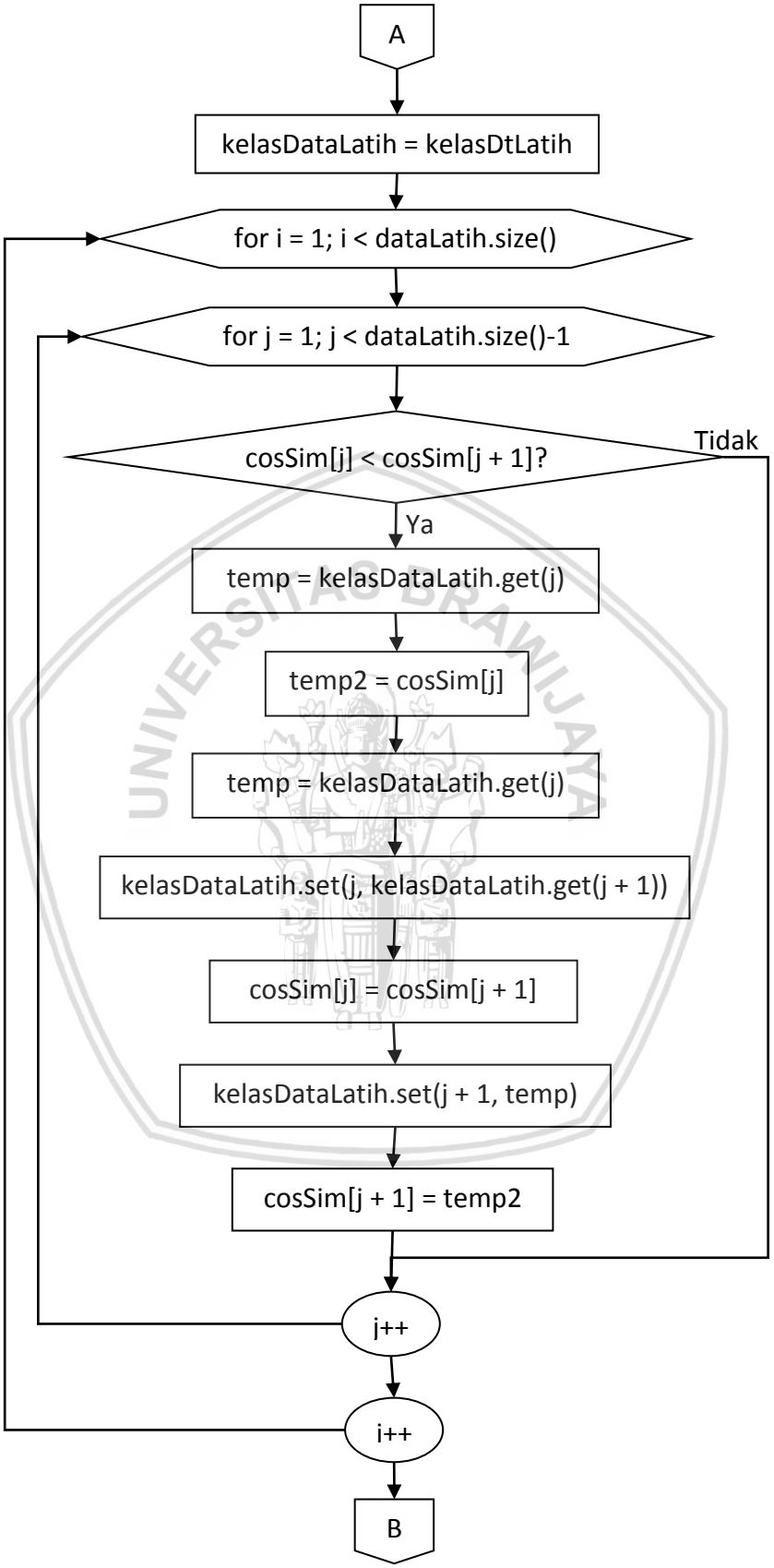


Gambar 4.36 Diagram alir proses perhitungan *cosine similarity*

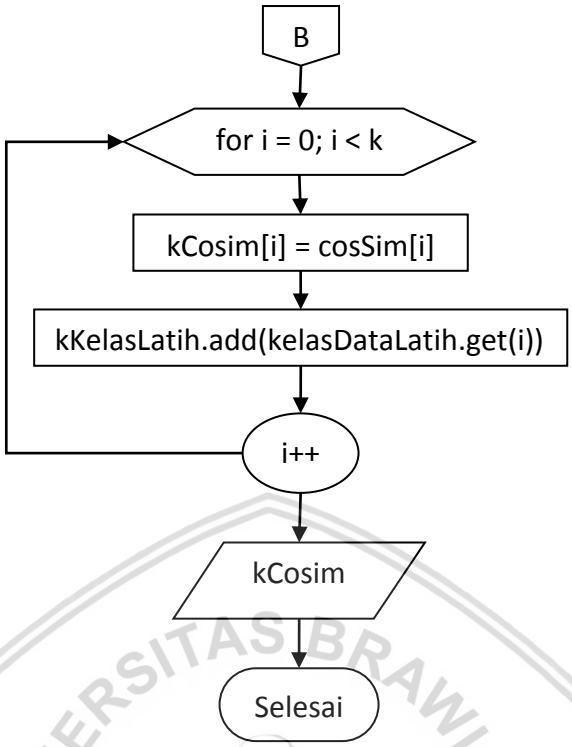
Hasil *cosine similarity* akan dilakukan perangkingan. Proses perangkingan *cosine similarity* digambarkan pada Gambar 4.37.



Gambar 4.37 Diagram alir proses perangkingan nilai *cosine similarity*

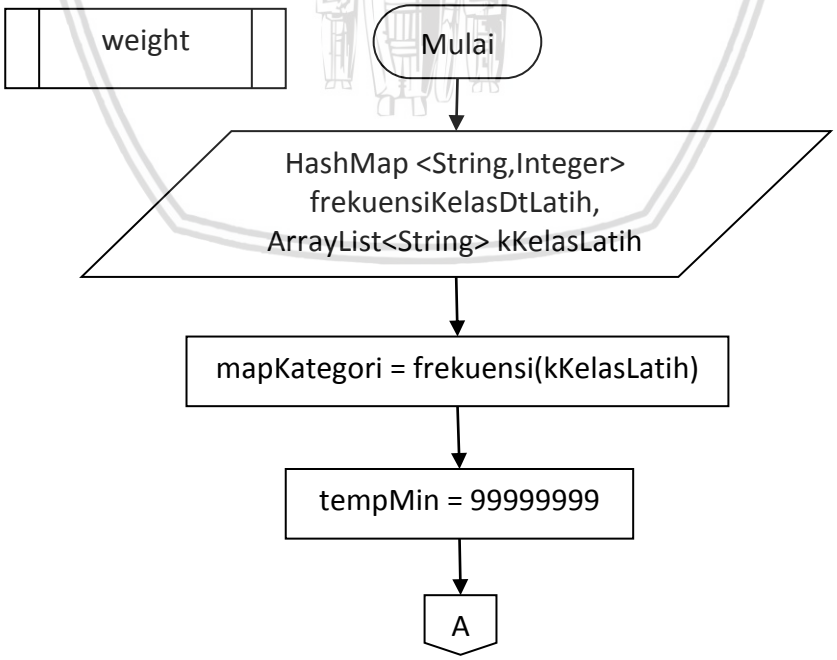


Gambar 4.37 Diagram alir proses perangkingan nilai *cosine similarity*

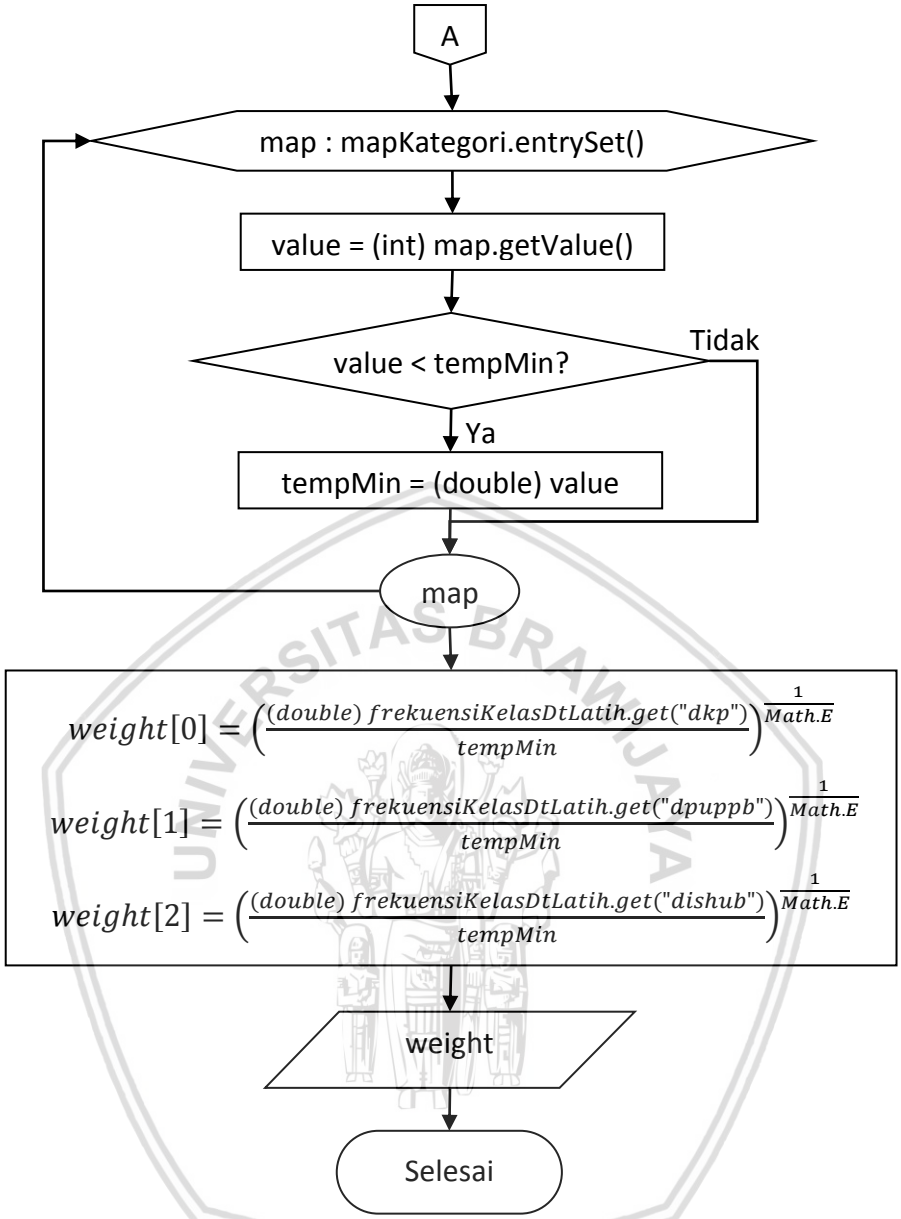


Gambar 4.37 Diagram alir proses perangkingan nilai *cosine similarity*

Setelah dilakukan perhitungan *cosine similarity* akan dihitung pembobotan tiap kelas yang digunakan dengan memanfaatkan hasil perhitungan cosine similarity sejumlah k tertinggi. Proses perhitungan bobot kelas digambarkan pada Gambar 4.38.

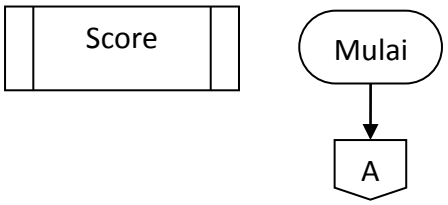


Gambar 4.38 Diagram alir perhitungan bobot kelas

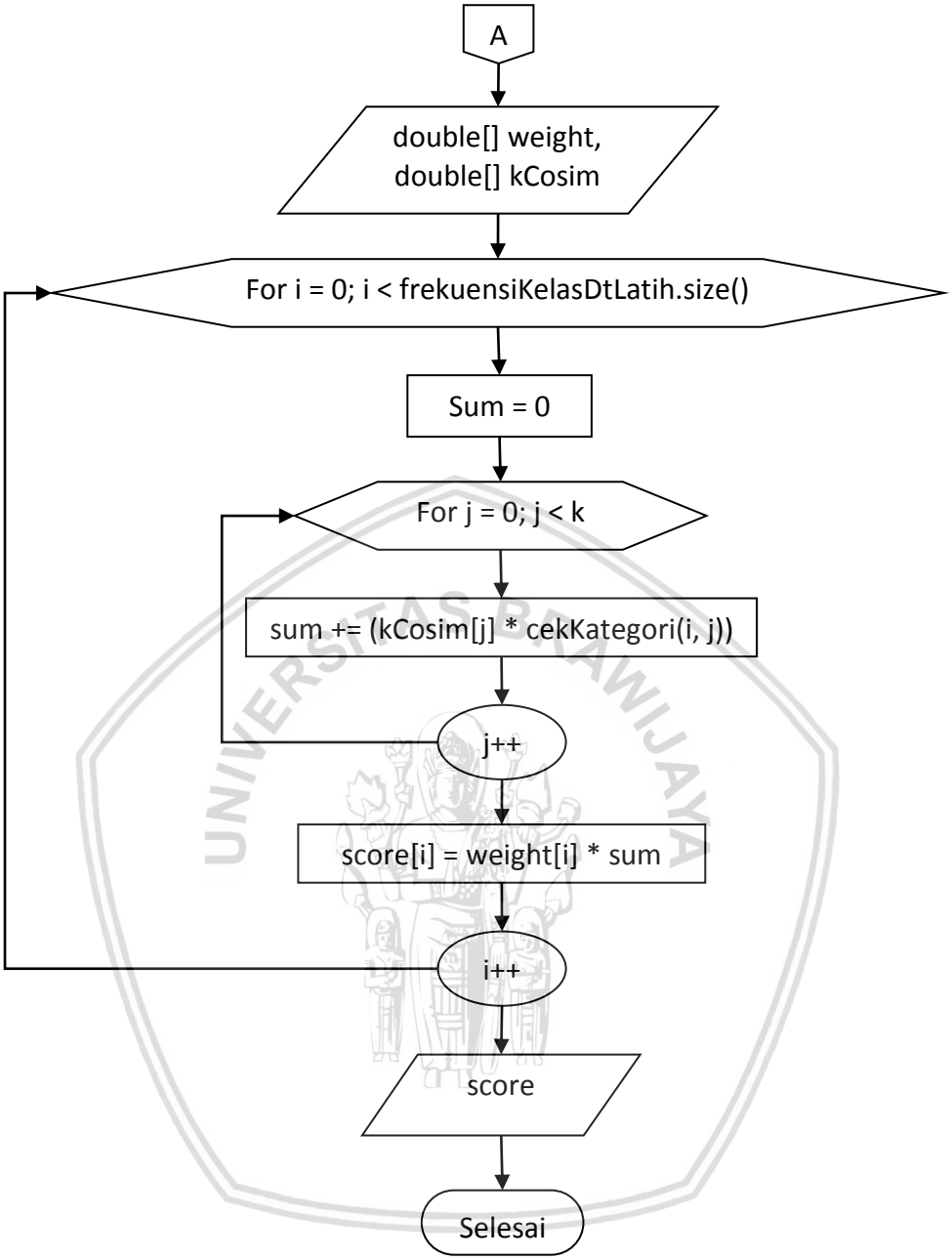


Gambar 4.38 Diagram alir perhitungan bobot kelas

Setelah dilakukan perhitungan bobot kelas,maka langkah selanjutnya dalam proses NW-KNN adalah menghitung *score* dari data uji terhadap setiap kelas. Proses perhitungan *score* dilakukan dengan proses seperti pada Gambar 4.39.

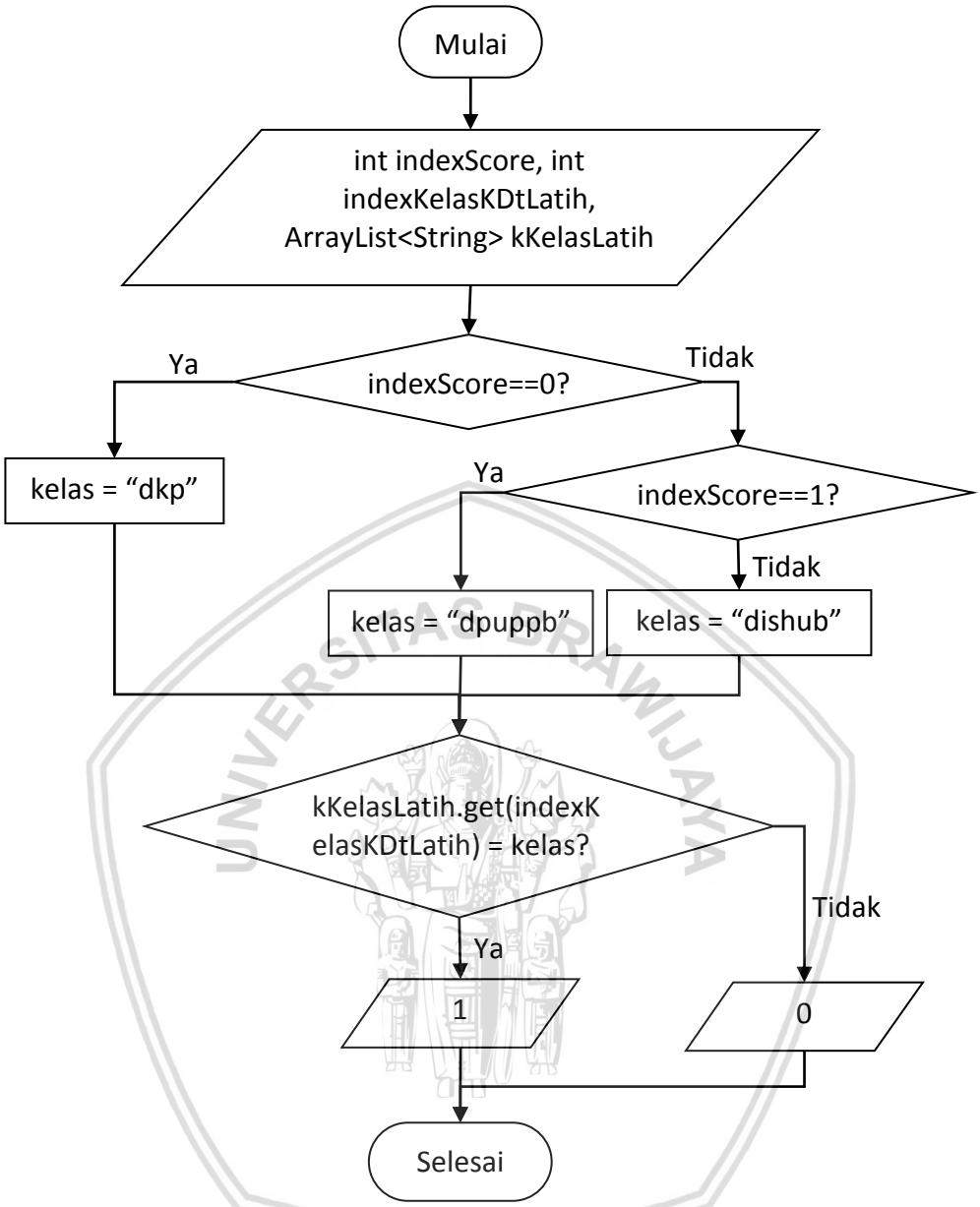


Gambar 4.39 Diagram alir perhitungan *score* data uji



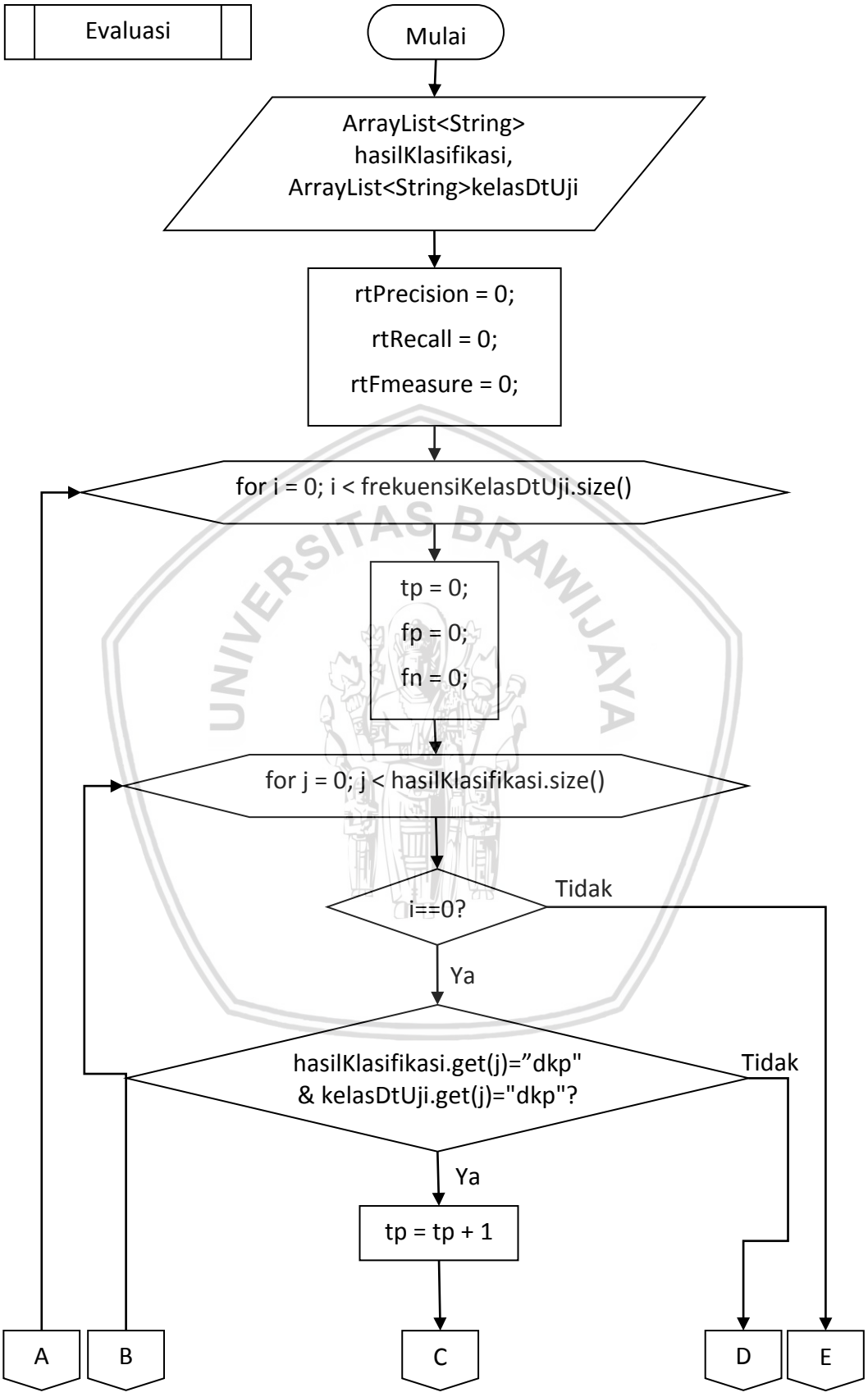
Gambar 4.39 Diagram alir perhitungan *score* data uji

Pada proses perhitungan *score* data uji terdapat hasil cek kategori. Hasil cek kategori merupakan hasil dari menjalankan method cekKategori() yang berisi proses dimana hasil akan bernilai 1 apabila kelas latih pada *range* k sesuai dengan kategori yang sedang dilakukan perhitungan *score*. Sedangkan apabila yang terjadi adalah sebaliknya, maka hasil cekKategori() bernilai 0. Secara detail proses pengecekan kategori dalam hal ini digambarkan pada Gambar 4.40.

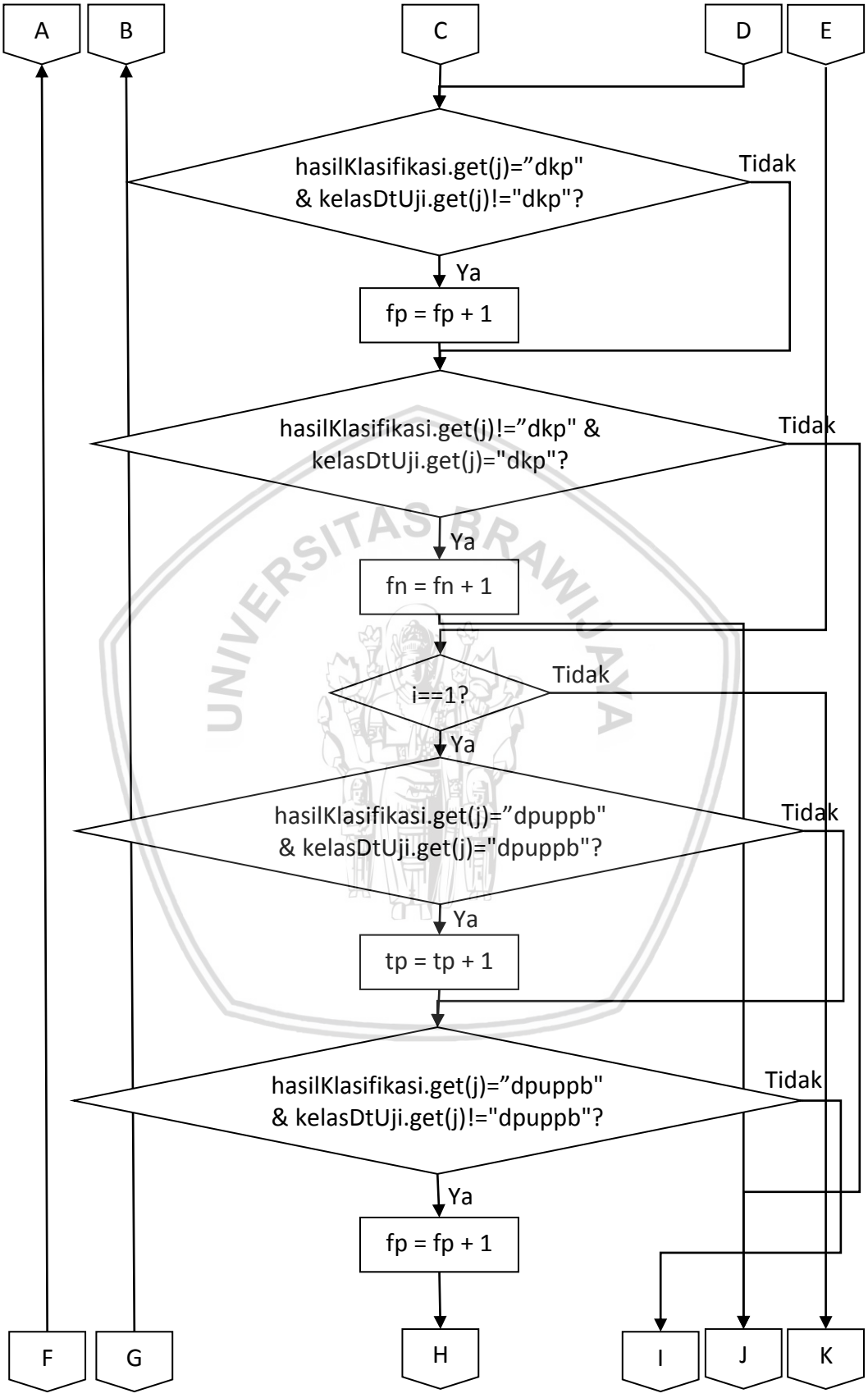


Gambar 4.40 Diagram alir proses pengecekan kategori

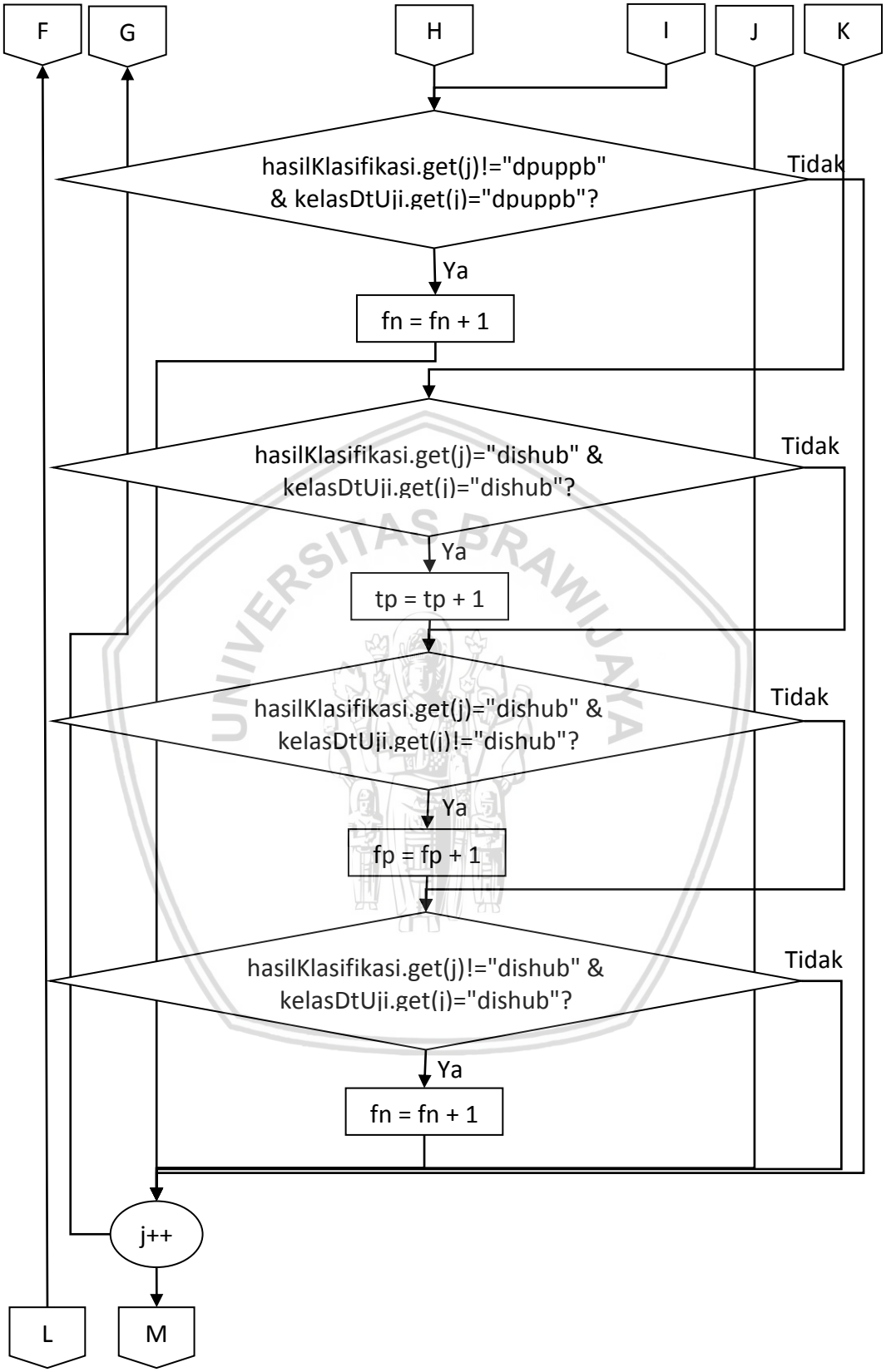
Dari hasil klasifikasi yang dihasilkan dari proses klasifikasi dengan NW-KNN pada Gambar 4.31 dapat dilakukan perhitungan *f-measure* sebagai proses evaluasi dari algoritme genetika ini. Proses evaluasi tersebut secara detail digambarkan pada Gambar 4.41.



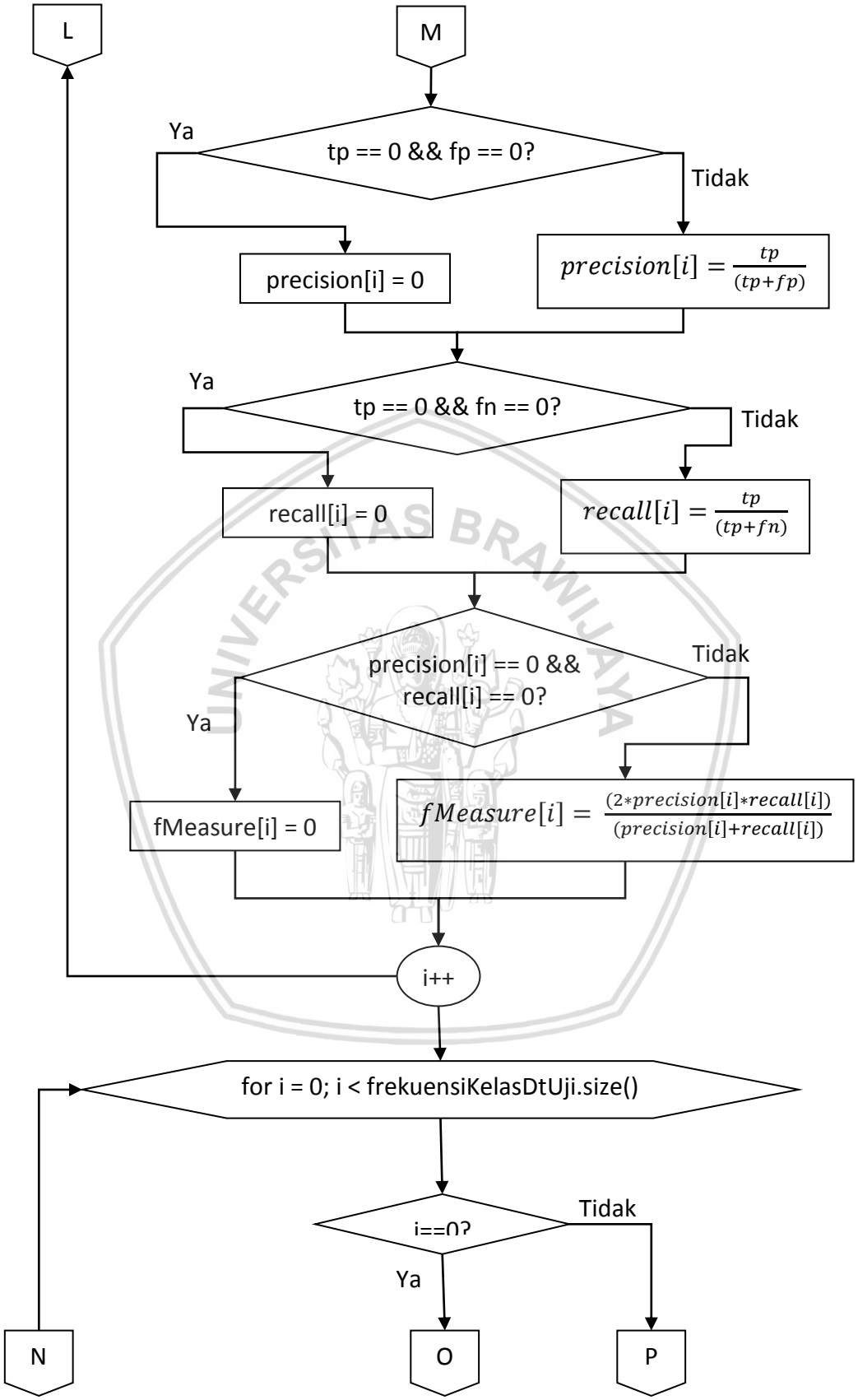
Gambar 4.41 Diagram alir proses evaluasi



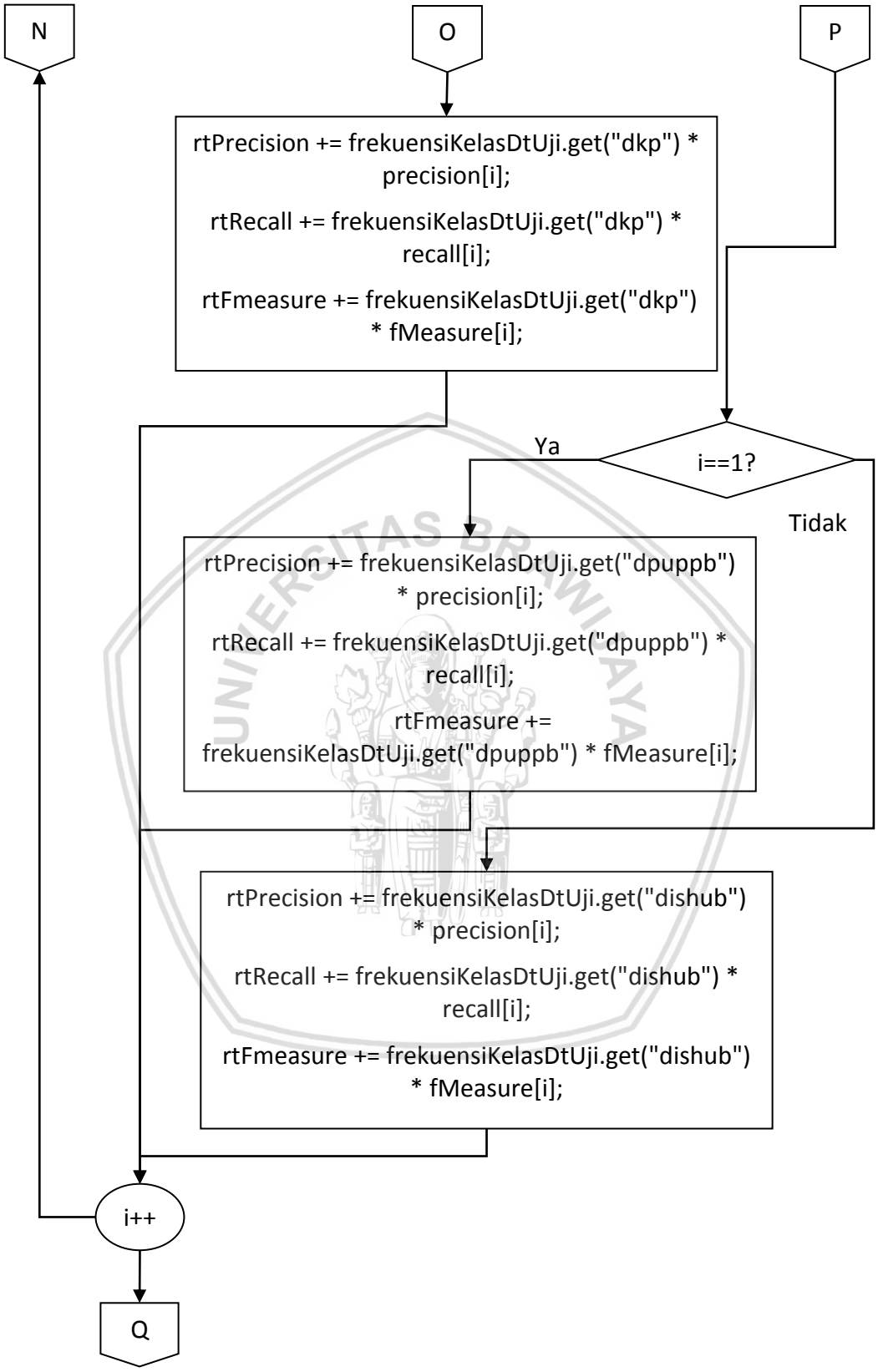
Gambar 4.41 Diagram alir proses evaluasi



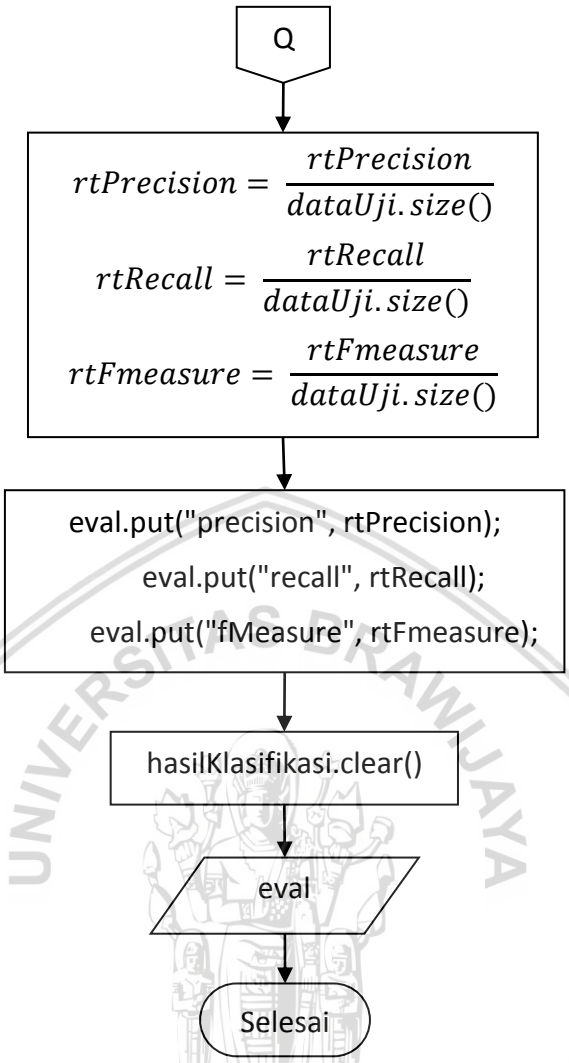
Gambar 4.41 Diagram alir proses evaluasi



Gambar 4.41 Diagram alir proses evaluasi



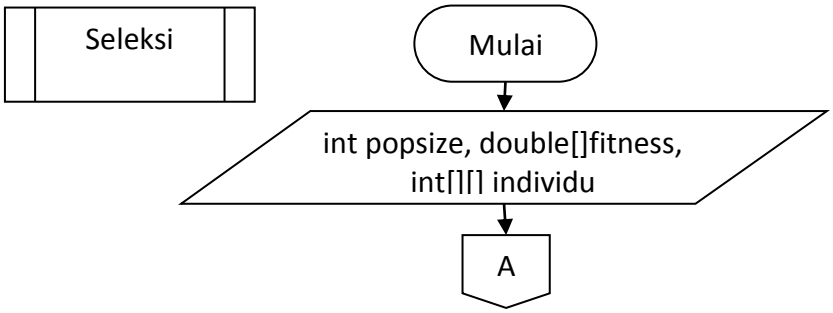
Gambar 4.41 Diagram alir proses evaluasi



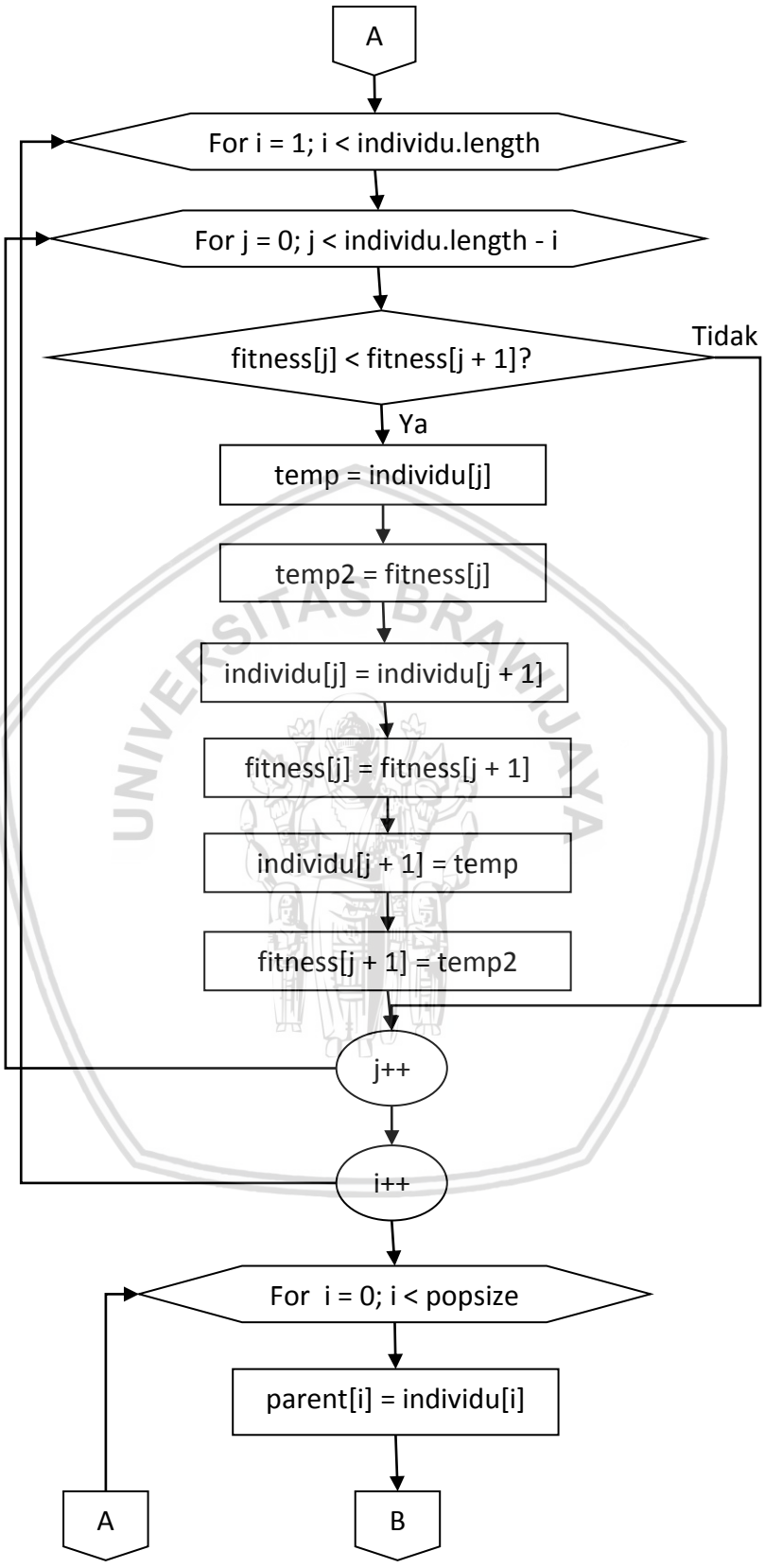
Gambar 4.41 Diagram alir proses evaluasi

4.1.3.4 Seleksi

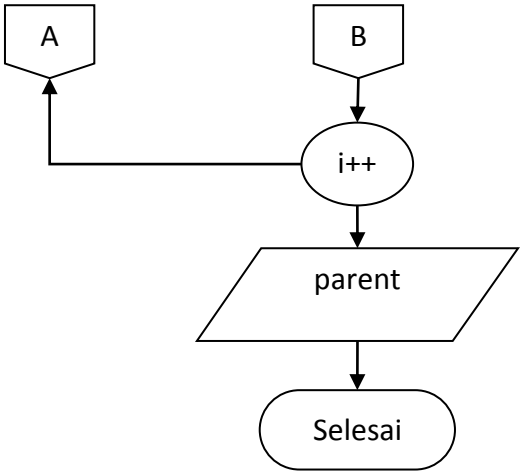
Proses seleksi dilakukan untuk mendapatkan individu yang memiliki *f-measure* tertinggi sebanyak *popsize*. Teknik seleksi yang digunakan adalah *eltism selection*, dimana proses teknik tersebut ditunjukkan pada Gambar 4.42.



Gambar 4.42 Diagram alir proses seleksi



Gambar 4.42 Diagram alir proses seleksi



Gambar 4.42 Diagram alir proses seleksi

4.2 Manualisasi

Manualisasi merupakan proses perhitungan yang dilakukan secara manual yang diperlukan untuk mendapatkan gambaran dalam penyelesaian masalah. Manualisasi ini dilakukan hanya terhadap 5 data latih dan 3 data uji, dimana data yang digunakan hanyalah aduan masyarakat yang ditujukan pada 3 SKPD yaitu Dinas Kebersihan dan Pertamanan (DKP), Dinas Pekerjaan Umum,Perumahan dan Pengawasan Bangunan (DPUPPB), dan Dinas Perhubungan (Dishub). Sampel data latih dan data uji beserta kelasnya dalam manualisasi ini ditunjukkan pada Tabel 4.1 dan 4.2 secara berturut-turut. Sampel data latih dan uji tersebut nantinya akan menjadi masukan (inputan) pada *pre-processing* data.

Tabel 4.1 Sampel data latih

No	Aduan	Kelas
1	Mohon traffic light penyeberangan diperbaiki karena tidak berfungsi	DKP
2	Di Pasar Gadang banyak jalan berlubang	DPUPPB
3	Jembatan Lori RW.05 Kel. Bumiayu, sangat memperihatinkan banyak jalan berlubang dan tiang penyangga kropos	DPUPPB
4	Jukir tidak pernah memberikan karcis secara sukarela di SEMUA titik parkir kota Malang	DISHUB
5	Kepada yth walikota bpk HM. ANTON di jln sulfat tepatnya dalam areal puskesmas kendalkerep harus membayar parkir padahal berobatnya gratis parkir tetep mbayar ini puskesmas milik pemerinta masak parkir mbayar saya berharap di semua tempat puskesmas atau kelurahan atau kecamatan di gratiskan selama masuk areal parkir dalam	DISHUB

Tabel 4.2 Sampel data uji

No	Aduan	Kelas
1	Assalamu'alaikum wr.wb. Yth. Pak Walikota Malang, , untuk keamanan dan kenyamanan pejalan kaki / wisatawan , apakah pemkota malang mempunyai rencana program perbaikan fasilitas untuk pejalan kaki :1) Kayutangan - Alun-alun - Pecinan; 2). Alun-alun bunder - Kahuripan - Semeru - Ijen; 3). jalan. Kawi - Alun-alun. ? Fasilitas untuk pekalan kaki jangan/tidak boleh untuk parkir mobil dan sepeda motor ! . Terima kasih, wassalam	DKP
2	Ada rencana gak? Membenahi bedak2 ilegal sepanjang jalan halmahera, nusakambangan, irian jaya. Kumuh makan bdn jalan	DPUPPB
3	selamat pagi, saya ingin konfirmasi parkir di atm bca cabang borobudur apakah resmi, tolong kalau resmi petugas parkirnya diwajibkan memberi karcis parkir resmi kepada kami.. kalau tidak resmi mohon ditertibkan saja.. sekian terima kasih..	DISHUB

4.2.1 Pre-processing

Langkah awal dari *pre-processing* adalah *case folding*. Sampel data yang berupa aduan masyarakat akan menjadi masukan pada proses *case folding* ini. Sehingga hasil dari proses case folding ditunjukkan pada Tabel 4.3 untuk *case folding* dari sampel data latih dan Tabel 4.4 untuk *case folding* dari sampel data uji. Pada tabel-tabel tersebut hasil *case folding* adalah masukan (inputan) yang telah dirubah setiap huruf besarnya menjadi huruf kecil (*lowercase*).

Tabel 4.3 Hasil proses *case folding* sampel data latih

No	Hasil case folding
1	mohon traffic light penyeberangan diperbaiki karena tidak berfungsi
2	di pasar gadang banyak jalan berlubang
3	jembatan lori rw.05 kel. bumiayu, sangat memperihatinkan banyak jalan berlubang dan tiang penyangga kropos
4	jukir tidak pernah memberikan karcis secara sukarela di semua titik parkir kota malang
5	kepada yth walikota bpk hm. anton di jln sulfat tepatnya dalam areal puskesmas kendalkerep harus membayar parkir padahal berobatnya gratis parkir tetep mbayar ini puskesmas milik pemerinta masak parkir mbayar saya berharap di semua tempat puskesmas atau kelurahan atau kecamatan di gratiskan selama masuk areal parkir dalam

Tabel 4.4 Hasil proses *case folding* sampel data uji

No	Hasil Case Folding
1	assalamu'alaikum wr.wb. yth. pak walikota malang, , untuk keamanan dan kenyamanan pejalan kaki / wisatawan , apakah pemkota malang mempunyai rencana program perbaikan fasilitas untuk pejalan kaki :1) kayutangan - alun-alun - pecinan; 2). alun-alun bunder - kahuripan - semeru - ijen; 3). jalan. kawi - alun-alun. ? fasilitas untuk pekalan kaki jangan/tidak boleh untuk parkir mobil dan sepeda motor ! . terima kasih, wassalam

Tabel 4.4 Hasil proses *case folding* sampel data uji

No	Hasil Case Folding
2	ada rencana gak? membenahi bedak2 ilegal sepanjang jalan halmahera, nusakambangan, irian jaya. kumuh makan bdn jalan
3	selamat pagi, saya ingin konfirmasi parkir di atm bca cabang borobudur apakah resmi, tolong kalau resmi petugas parkirnya diwajibkan memberi karcis parkir resmi kepada kami.. kalau tidak resmi mohon ditertibkan saja.. sekian terima kasih..

Setelah dilakukan proses *case folding*, makaakan dilakukan proses tokenisasi. Pada proses tokenisasi ini yang menjadi masukannya (inputannya) adalah *output* (keluaran/hasil) dari *case folding*.Berdasarkan diagram alir tokenisasi yang terdapat pada Gambar 4.4, proses tokenisasi diawali dengan proses penghapusan angka dan tanda baca. Proses penghapusan angka dan tanda baca pada manualisasi ini terdapat pada Tabel 4.5 untuk penghapusan angka dan tanda baca pada data latih, dan Tabel 4.6 untuk penghapusan angka dan tanda baca pada data uji.

Tabel 4.5 Hasil penghapusan angka dan tanda baca pada data latih

No	Hasil hapus angka dan tanda baca
1	mohon traffic light penyeberangan diperbaiki karena tidak berfungsi
2	di pasar gadang banyak jalan berlubang
3	jembatan lori rw kel bumiayu sangat memperihatinkan banyak jalan berlubang dan tiang penyangga kropos
4	jukir tidak pernah memberikan karcis secara sukarela di semua titik parkir kota malang
5	kepada yth walikota bpk hm anton di jln sulfat tepatnya dalam areal puskesmas kendalkerep harus membayar parkir padahal berobatnya gratis parkir tetep mbayar ini puskesmas milik pemerinta masak parkir mbayar saya berharap di semua tempat puskesmas atau kelurahan atau kecamatan di gratiskan selama masuk areal parkir dalam

Tabel 4.6 Hasil penghapusan angka dan tanda baca pada data uji

No	Hasil hapus angka dan tanda baca
1	assalamu alaikum wr wb yth pak walikota malang untuk keamanan dan kenyamanan pejalan kaki wisatawan apakah pemkota malang mempunyai rencana program perbaikan fasilitas untuk pejalan kaki kayutangan alun alun pecinan alun alun bunder kahuripan semeru ijen jalan kawi alun alun fasilitas untuk pekalan kaki jangan tidak boleh untuk parkir mobil dan sepeda motor terima kasih wassalam
2	ada rencana gak membenahi bedak ilegal sepanjang jalan halmahera nusakambangan irian jaya kumuh makan bdn jalan
3	selamat pagi, saya ingin konfirmasi parkir di atm bca cabang borobudur apakah resmi tolong kalau resmi petugas parkirnya diwajibkan memberi karcis parkir resmi kepada kami kalau tidak resmi mohon ditertibkan saja sekian terima kasih

Setelah melakukan penghapusan angka dan tanda baca pada setiap dokumen, maka proses tokenisasi dilakukan untuk mendapatkan token (karakteristik) dari setiap dokumen. Proses tokenisasi ini dilakukan dengan

pengambilan setiap kata yang dipisahkan dengan spasi. Proses tokenisasi pada manualisasi ini ditunjukkan pada Tabel 4.7 untuk tokenisasi terhadap data latih dan 4.8 untuk tokenisasi terhadap data uji.

Tabel 4.7 Hasil tokenisasi data latih

No	Hasil Tokenisasi	
1	mohon	diperbaiki
	traffic	karena
	light	tidak
	penyeberangan	berfungsi
2	di	banyak
	pasar	jalan
	gadang	berlubang
3	jembatan	banyak
	lori	jalan
	rw	berlubang
	kel	dan
	bumiayu	tiang
	sangat	penyangga
	memperihatinkan	kropos
4	jukir	di
	tidak	semua
	pernah	titik
	memberikan	parkir
	karcis	kota
	secara	malang
	sukarela	
5	kepada	puskesmas
	yth	milik
	walikota	pemerinta
	bpk	masak
	hm	parkir
	anton	mbayar
	di	saya
	jln	berharap
	sulfat	di
	tepatnya	semua

Tabel 4.7 Hasil tokenisasi data latih

No	Hasil Tokenisasi	
	dalam	tempat
	areal	puskesmas
	puskesmas	atau
	kendalkerep	kelurahan
	harus	atau
	membayar	kecamatan
	parkir	di
	padahal	gratiskan
	berobatnya	selama
	gratis	masuk
	parkir	areal
	tetep	parkir
	mbayar	dalam
	ini	

Tabel 4.8 Hasil tokenisasi data uji

No	Hasil Tokenisasi	
1	assalamu	alun
	alaikum	pecinan
	wr	alun
	wr	alun
	yth	bunder
	pak	kahuripan
	walikota	semeru
	malang	ijen
	untuk	jalan
	keamanan	kawi
	dan	alun
	kenyamanan	alun
	pejalan	fasilitas
	kaki	untuk
	wisatawan	pejalan
	apakah	kaki

Tabel 4.8 Hasil tokenisasi data uji

No	Hasil Tokenisasi	
	pemkota	jangan
	malang	tidak
	mempunyai	boleh
	rencana	untuk
	program	parkir
	perbaikan	mobil
	fasilitas	dan
	untuk	sepeda
	pejalan	motor
	kaki	terima
	kayutungan	kasih
	alun	wassalam
2	ada	halmahera
	rencana	nusakambangan
	gak	irian
	membenahi	jaya
	bedak	kumuh
	ilegal	makan
	sepanjang	bdn
	jalan	jalan
3	selamat	parkirnya
	pagi	diwajibkan
	saya	memberi
	ingin	karcis
	konfirmasi	parkir
	parkir	resmi
	di	kepada
	atm	kami
	bca	kalau
	cabang	tidak
	borobudur	resmi
	apakah	mohon
	resmi	ditertibkan

Tabel 4.8 Hasil tokenisasi data uji

No	Hasil Tokenisasi	
	tolong	saja
	kalau	terima
	resmi	kasih
	petugas	

Setiap token dari masing-masing dokumen akan diproses lagi untuk dilakukan pengecekan *stopword*. Dimana apabila token tersebut termasuk *stopword* maka akan dihapus. Sehingga hasil proses penghapusan stopword yang dilakukan pada sampel data latih dan data uji secara berturut-turut ditunjukkan pada Tabel 4.9 dan 4.10.

Tabel 4.9 Hasil *stopword removal* data latih

No	Hasil Stopword Removal	
1	mohon	penyeberangan
	traffic	diperbaiki
	light	berfungsi
2	pasar	jalan
	gadang	berlubang
3	jembatan	jalan
	lori	berlubang
	rw	tiang
	kel	penyangga
	bumiayu	kropos
	memperihatinkan	
4	jukir	parkir
	karcis	kota
	sukarela	malang
	titik	
5	yth	tetep
	walikota	mbayar
	bpk	puskesmas
	hm	milik
	anton	pemerinta
	jln	masak
	sulfat	parkir
	tepatnya	mbayar

Tabel 4.9 Hasil *stopword removal* data latih

No	Hasil Stopword Removal	
	areal	berharap
	puskesmas	puskesmas
	kendalkerep	kelurahan
	membayar	kecamatan
	parkir	gratiskan
	berobatnya	masuk
	gratis	areal
	parkir	parkir

Tabel 4.10 Hasil *stopword removal* data uji

No	Hasil Stopword Removal	
1	assalamu	alun
	alaikum	pecinan
	wr	alun
	wr	alun
	yth	bunder
	walikota	kahuripan
	malang	semeru
	keamanan	ijen
	kenyamanan	jalan
	pejalan	kawi
	kaki	alun
	wisatawan	alun
	pemkota	fasilitas
	malang	pejalan
	rencana	kaki
	program	parkir
	perbaikan	mobil
	fasilitas	sepeda
	pejalan	motor
	kaki	terima
	kayutungan	kasih
	alun	wassalam

Tabel 4.10 Hasil *stopword removal* data uji

No	Hasil Stopword Removal	
2	rencana	nusakambangan
	gak	irian
	membenahi	jaya
	bedak	kumuh
	ilegal	makan
	jalan	bdn
	halmahera	jalan
3	selamat	petugas
	pagi	parkirnya
	konfirmasi	diwajibkan
	parkir	karcis
	atm	parkir
	bca	resmi
	cabang	resmi
	borobudur	mohon
	resmi	ditertibkan
	tolong	terima
	resmi	kasih

Hasil dari proses penghapusan *stopword* akan dilakukan proses *stemming* untuk mendapatkan kata dasar dari setiap token. Proses *stemming* dari hasil penghapusan *stopword* data latih dan data uji secara berturut-turut ditunjukkan pada Tabel 4.11 dan 4.12. Pada tabel-tabel tersebut hasil proses *stemming* didapatkan berdasarkan algoritme nazief adriani.

Tabel 4.11 Hasil *stemming* data latih

No	Hasil Stemming	
1	mohon	seberang
	traffic	baik
	light	fungsi
2	pasar	jalan
	gadang	lubang
3	jembatan	jalan
	lori	lubang
	rw	tiang
	kel	sangga

Tabel 4.11 Hasil *stemming* data latih

No	Hasil Stemming	
	bumiayu	kropos
	memperihatinkan	
4	jukir	parkir
	karcis	kota
	sukarela	malang
	titik	
5	yth	tetep
	walikota	mbayar
	bpk	puskesmas
	hm	milik
	anton	pemerinta
	jln	masak
	sulfat	parkir
	tepat	mbayar
	areal	harap
	puskesmas	puskesmas
	kendalkerep	lurah
	bayar	camat
	parkir	gratis
	obat	masuk
	gratis	areal
	parkir	parkir

Tabel 4.12 Hasil *stemming* data uji

No	Hasil Stemming	
1	assala	alun
	alaikum	pecinan
	wr	alun
	wb	alun
	yth	bunder
	walikota	kahuripan
	malang	meru
	aman	ijen

Tabel 4.12 Hasil *stemming* data uji

No	Hasil Stemming	
	nyaman	jalan
	pejal	kawi
	kaki	alun
	wisatawan	alun
	pemkota	fasilitas
	malang	kal
	rencana	kaki
	program	parkir
	baik	mobil
	fasilitas	sepeda
	pejal	motor
	kaki	terima
	kayutangan	kasih
	alun	wassalam
2	rencana	nusakambangan
	gak	iri
	benah	jaya
	bedak	kumuh
	ilegal	makan
	jalan	bdn
	halmahera	jalan
3	selamat	tugas
	pagi	parkir
	konfirmasi	wajib
	parkir	karcis
	atm	parkir
	bca	resmi
	cabang	resmi
	borobudur	mohon
	resmi	tertib
	tolong	kian
	resmi	terima

Hasil akhir dari *pre-processing* dari data latih ini nantinya akan dikumpulkan menjadi satu untuk dijadikan sebagai fitur dengan menghapus duplikat kata.

Fitur ini nantinya akan menjadi karakteristik setiap dokumen yang akan diproses. Fitur yang telah diambil dari keseluruhan dokumen kemudian akan diseleksi dengan menggunakan teknik *information gain*. Fitur-fitur hasil dari *pre-processing* data latih ditabelkan pada Tabel 4.13.

Tabel 4.13 Daftar Fitur

No	Fitur
1	mohon
2	traffic
3	light
4	seberang
5	baik
6	fungsi
7	pasar
8	gadang
9	jalan
10	lubang
11	jembatan
12	lori
13	rw
14	kel
15	bumiayu
16	memperihatinkan
17	tiang
18	sangga
19	kropos
20	jukir
21	karcis
22	sukarela
23	titik
24	parkir
25	kota
26	malang
27	yth
28	walikota
29	bpk
30	hm
31	anton
32	jln
33	sulfat
34	tepat
35	areal
36	puskesmas

Tabel 4.13 Daftar Fitur

No	Fitur
37	kendalkerep
38	bayar
39	obat
40	gratis
41	tetep
42	mbayar
43	milik
44	pemerinta
45	masak
46	harap
47	lurah
48	camat
49	masuk

4.2.2 Information Gain

Berdasarkan pada diagram alir Gambar 4.1, *information gain* merupakan proses yang dijalankan setelah proses *pre-processing* selesai dijalankan. Langkah awal dalam proses *information gain* ini adalah menghitung ada atau tidaknya fitur pada setiap dokumen dari data latih. Pada penelitian ini ada atau tidaknya fitur pada dokumen dapat dilihat dengan menghitung frekuensi kemunculan fitur pada setiap dokumen. Frekuensi kemunculan fitur dari sampel data latih ini ditunjukkan pada Tabel 4.14.

Tabel 4.14 *Term frequency* pada data latih

Fitur	Term Frekuensi				
	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5
mohon	1	0	0	0	0
traffic	1	0	0	0	0
light	1	0	0	0	0
seberang	1	0	0	0	0
baik	1	0	0	0	0
fungsi	1	0	0	0	0
pasar	0	1	0	0	0
gadang	0	1	0	0	0
jalan	0	1	1	0	0
lubang	0	1	1	0	0
jembatan	0	0	1	0	0
lori	0	0	1	0	0
rw	0	0	1	0	0
kel	0	0	1	0	0
bumiayu	0	0	1	0	0

Tabel 4.14 *Term frequency* pada data latih

Fitur	Term Frekuensi				
	Dok 1	Dok 2	Dok 3	Dok 4	Dok 5
memperihatinkan	0	0	1	0	0
tiang	0	0	1	0	0
sangga	0	0	1	0	0
kropos	0	0	1	0	0
jukir	0	0	0	1	0
karcis	0	0	0	1	0
sukarela	0	0	0	1	0
titik	0	0	0	1	0
parkir	0	0	0	1	4
kota	0	0	0	1	0
malang	0	0	0	1	0
yth	0	0	0	0	1
walikota	0	0	0	0	1
bpk	0	0	0	0	1
hm	0	0	0	0	1
anton	0	0	0	0	1
jln	0	0	0	0	1
sulfat	0	0	0	0	1
tepat	0	0	0	0	1
areal	0	0	0	0	2
puskesmas	0	0	0	0	3
kendalkerep	0	0	0	0	1
bayar	0	0	0	0	1
obat	0	0	0	0	1
gratis	0	0	0	0	2
tetep	0	0	0	0	1
mbayar	0	0	0	0	2
milik	0	0	0	0	1
pemerinta	0	0	0	0	1
masak	0	0	0	0	1
harap	0	0	0	0	1
lurah	0	0	0	0	1
camat	0	0	0	0	1
masuk	0	0	0	0	1

Pada Tabel 4.14 dapat dilihat bahwa kata “mohon” yang menjadi salah satu fitur hasil dari *pre-processing* data latih. Kata “mohon” tersebut hanya muncul pada Dok 1 yang berarti dokumen 1 dari data latih. Hal ini ditunjukkan dengan nilai 1 pada Dok 1 yang berarti pada Dok 1 (dokumen 1) dari data latih kata “mohon” hanya muncul sekali. Sedangkan pada Dok 2, 3, 4, dan 5 bernilai 0 yang berarti bahwa kata “mohon” tidak muncul pada dokumen 2, 3, 4, dan 5,

begitupun dengan fitur lain, bahwa angka yang terdapat pada tabel tersebut melambangkan jumlah kemunculan fitur terhadap dokumen dari data latih. Sehingga dari hasil perhitungan frekuensi fitur yang terdapat pada Tabel 4.16, dapat dihitung berapa jumlah dokumen dari setiap kelas yang mengandung fitur dan tidak. Jumlah dokumen yang terdapat fitur maupun tidak dapat dihitung dengan memperhatikan setiap angka pada tabel frekuensi yaitu Tabel 4.14, dimana apabila angka kemunculan lebih dari 0 maka fitur tersebut dihitung ada pada dokumen tersebut, sedangkan apabila sama dengan 0 maka fitur tersebut dihitung tidak ada pada dokumen. Jumlah dokumen dihitung di setiap kelas, dimana pada kelas DKP hanya terdapat 1 dokumen yaitu Dok 1, pada kelas DPUPPB terdapat 2 kelas yaitu Dok 2 dan Dok 3, sedangkan pada kelas Dishub terdapat 2 kelas juga yaitu Dok 4 dan Dok 5. Hasil perhitungan jumlah dokumen pada setiap kelas yang memilikimaupun yang tidak memiliki fitur ditunjukkan pada Tabel 4.15.

Tabel 4.15 Jumlah dokumen yang mengandung fitur pada setiap kelas

Fitur	C1		C2		C3	
	Ada	Tidak	Ada	Tidak	Ada	Tidak
mohon	1	0	0	2	0	2
traffic	1	0	0	2	0	2
light	1	0	0	2	0	2
seberang	1	0	0	2	0	2
baik	1	0	0	2	0	2
fungsi	1	0	0	2	0	2
pasar	0	1	1	1	0	2
gadang	0	1	1	1	0	2
jalan	0	1	2	0	0	2
lubang	0	1	2	0	0	2
jembatan	0	1	1	1	0	2
lori	0	1	1	1	0	2
rw	0	1	1	1	0	2
kel	0	1	1	1	0	2
bumiayu	0	1	1	1	0	2
memperihatinkan	0	1	1	1	0	2
tiang	0	1	1	1	0	2
sangga	0	1	1	1	0	2
kropos	0	1	1	1	0	2
jukir	0	1	0	2	1	1
karcis	0	1	0	2	1	1
sukarela	0	1	0	2	1	1
titik	0	1	0	2	1	1
parkir	0	1	0	2	2	0
kota	0	1	0	2	1	1
malang	0	1	0	2	1	1

Tabel 4.15 Jumlah dokumen yang mengandung fitur pada setiap kelas

Fitur	C1		C2		C3	
	Ada	Tidak	Ada	Tidak	Ada	Tidak
yth	0	1	0	2	1	1
walikota	0	1	0	2	1	1
bpk	0	1	0	2	1	1
hm	0	1	0	2	1	1
anton	0	1	0	2	1	1
jln	0	1	0	2	1	1
sulfat	0	1	0	2	1	1
tepat	0	1	0	2	1	1
areal	0	1	0	2	1	1
puskesmas	0	1	0	2	1	1
kendalkerep	0	1	0	2	1	1
bayar	0	1	0	2	1	1
obat	0	1	0	2	1	1
gratis	0	1	0	2	1	1
tetep	0	1	0	2	1	1
mbayar	0	1	0	2	1	1
milik	0	1	0	2	1	1
pemerinta	0	1	0	2	1	1
masak	0	1	0	2	1	1
harap	0	1	0	2	1	1
lurah	0	1	0	2	1	1
camat	0	1	0	2	1	1
masuk	0	1	0	2	1	1

Pada Tabel 4.15, terdapat C1, C2, dan C3, dimana C1 merupakan kelas DKP, C2 merupakan kelas DPUPPB, dan C3 merupakan kelas Dishub. Pada tabel tersebut fitur kata "mohon" pada kelas DKP (C1) yaitu pada dokumen 1, jumlah dokumen yang memiliki kata "mohon" hanya 1 dokumen, hal ini ditunjukkan pada kolom ada pada fitur "mohon" tertera angka 1. Angka 1 pada kolom tersebut, berdasarkan hasil perhitungan frekuensi fitur pada tabel 4.14, pada kolom Dok 1 (dokumen 1) pada fitur "mohon" tertulis angka 1, dimana 1 lebih dari 0 sehingga fitur tersebut terhitung ada pada dokumen 1 (Dok 1) yang merupakan dokumen pada kelas DKP (C1). Sedangkan jumlah dokumen yang tidak memiliki kata mohon pada kelas DKP (C1) yaitu pada dokumen 1, jumlah dokumen yang tidak memiliki kata "mohon" berjumlah 0, sesuai yang tertera pada kolom tidak pada fitur "mohon". Hal ini dikarenakan pada Tabel 4.14, pada kolom dokumen 1 (Dok 1) pada fitur "mohon" tertulis angka 1, akan tetapi angka 1 tidak sama dengan 0, maka dokumen tersebut tidak terhitung sebagai dokumen yang tidak memiliki kata "mohon". Pertimbangan tersebut juga dilakukan sama terhadap kelas lain, C2 dan C3, akan tetapi pada C2 yang menjadi

acuan berjumlah 2 dokumen (Dok 2 dan 3), dan C3 juga berjumlah 2 dokumen (Dok 4 dan 5). Hal ini dilakukan terhadap seluruh fitur yang ada.

Setelah dilakukan perhitungan jumlah dokumen yang terdapat dan tidak terdapat fitur didalamnya, maka selanjutnya adalah menghitung total dokumen yang terdapat fitur maupun tidak, tanpa memandang kelas. Dengan melihat Tabel 4.15 dapat dihasilkan total dokumen yang terdapat fitur maupun tidak terdapat pada Tabel 4.16.

Tabel 4.16 Jumlah dokumen fitur pada data latih

Fitur	Total	
	Ada	Tidak
mohon	1	4
traffic	1	4
light	1	4
seberang	1	4
baik	1	4
fungsi	1	4
pasar	1	4
gadang	1	4
jalan	2	3
lubang	2	3
jembatan	1	4
lori	1	4
rw	1	4
kel	1	4
bumiayu	1	4
memperihatinkan	1	4
tiang	1	4
sangga	1	4
kropos	1	4
jukir	1	4
karcis	1	4
sukarela	1	4
titik	1	4
parkir	2	3
kota	1	4
malang	1	4
yth	1	4
walikota	1	4
bpk	1	4
hm	1	4
anton	1	4
jln	1	4

Tabel 4.16 Jumlah dokumen fitur pada data latih

Fitur	Total	
	Ada	Tidak
sulfat	1	4
tepat	1	4
areal	1	4
puskesmas	1	4
kendalkerep	1	4
bayar	1	4
obat	1	4
gratis	1	4
tetep	1	4
mbayar	1	4
milik	1	4
pemerinta	1	4
masak	1	4
harap	1	4
lurah	1	4
camat	1	4
masuk	1	4

Pada Tabel 4.16, kata "mohon" pada kolom total ada berjumlah 1, dan total tidak berjumlah 0, hal ini didapatkan dari penjumlahan pada nilai kolom ada pada C1, C2, dan C3 yang terdapat pada Tabel 4.15, dimana pada kata "mohon" nilai ada pada C1 berjumlah 1, pada C2 dan C3 berjumlah 0, sehingga apabila dijumlahkan akan bernilai 1 sesuai dengan total ada pada kata "mohon" yang terdapat pada Tabel 4.16. Sedangkan pada kata "mohon", nilai tidak pada C1, C2, dan C3 bernilai 0, sehingga apabila dijumlahkan akan bernilai 0 juga sesuai dengan kolom total tidak pada fitur "mohon" yang terdapat pada Tabel 4.16. Perhitungan ini juga berlaku untuk seluruh fitur yang ada.

Setelah mendapatkan jumlah dokumen yang mengandung dan tidak mengandung fitur baik pada setiap kelas maupun pada keseluruhan data latih, maka kemudian akan dicari entropy. Entropy yang dihitung adalah entropy global dan entropy kemunculan fitur. Pada entopy global dihitung menggunakan persamaan 2.6 dengan melihat jumlah data pada setiap kelas dan jumlah data keseluruhan. Pada manualisasi ini entropy global dari sampel data latih ini adalah:

$$Entropy\ Global = -\left(\left(\frac{1}{5} \times \log_2 \frac{1}{5}\right) + \left(\frac{2}{5} \times \log_2 \frac{2}{5}\right) + \left(\frac{2}{5} \times \log_2 \frac{2}{5}\right)\right)$$

$$= 1{,}521928095$$

Dari perhitungan *entropy global* tersebut, terdapat pecahan dimana yang menjadi pembilang adalah jumlah dokumen pada setiap kategori, dan yang menjadi penyebut adalah jumlah semua sampel data latih yang digunakan yaitu

pada hal ini adalah 5. Pada proses perhitungan *entropy* kemunculan pada data latih, akan dihitung dua macam kemunculan yaitu muncul dan tidak muncul. Perhitungan *entropy* kemunculan ini menggunakan persamaan 2.6 juga, akan tetapi yang menjadi pertimbangan adalah jumlah dokumen yang terdapat dan tidak terdapat fitur didalamnya pada setiap kelas dan pada data latih secara keseluruhan. Sehingga pada manualisasi ini, *entropy* ada dan tidak ada kata "mohon" secara berturut-turut adalah:

$$Entropy\ Ada = \left(\frac{1}{1} \times \log_2 \frac{1}{1}\right) + \left(\frac{0}{1} \times \log_2 \frac{0}{1}\right) + \left(\frac{0}{1} \times \log_2 \frac{0}{1}\right) = 0$$

$$Entropy\ Tidak = \left(\frac{0}{4} \times \log_2 \frac{0}{4}\right) + \left(\frac{2}{4} \times \log_2 \frac{2}{4}\right) + \left(\frac{2}{4} \times \log_2 \frac{2}{4}\right) = -1$$

Dari perhitungan diatas maka *entropy* ada pada kata "mohon" adalah 0 dan entropy tidak pada kata "mohon" adalah -1. Perhitungan seperti itu juga diterapkan terhadap seluruh fitur yang ada dengan memperhatikan Tabel 4.15 dan Tabel 4.16. Hasil akhir dari perhitungan *entropy* munculnya seluruh fitur ditunjukkan pada Tabel 4.17.

Tabel 4.17 *Entropy* kemunculan fitur pada data latih

Fitur	Entropy	
	Ada	Tidak
Mohon	0	-1
Traffic	0	-1
Light	0	-1
Seberang	0	-1
Baik	0	-1
Fungsi	0	-1
Pasar	0	-1,5
Gadang	0	-1,5
Jalan	0	-0,9183
Lubang	0	-0,9183
Jembatan	0	-1,5
Lori	0	-1,5
Rw	0	-1,5
Kel	0	-1,5
Bumiayu	0	-1,5
memperihatinkan	0	-1,5
Tiang	0	-1,5
Sangga	0	-1,5
Kropos	0	-1,5
Jukir	0	-1,5
Karcis	0	-1,5
Sukarela	0	-1,5
Titik	0	-1,5
Parkir	0	-0,9183

Tabel 4.17 *Entropy* kemunculan fitur pada data latih

Fitur	Entropy	
	Ada	Tidak
Kota	0	-1,5
Malang	0	-1,5
Yth	0	-1,5
Walikota	0	-1,5
Bpk	0	-1,5
Hm	0	-1,5
Anton	0	-1,5
Jln	0	-1,5
Sulfat	0	-1,5
Tepat	0	-1,5
Areal	0	-1,5
Puskesmas	0	-1,5
Kendalkerep	0	-1,5
Bayar	0	-1,5
Obat	0	-1,5
Gratis	0	-1,5
Tetep	0	-1,5
Mbayar	0	-1,5
Milik	0	-1,5
Pemerinta	0	-1,5
Masak	0	-1,5
Harap	0	-1,5
Lurah	0	-1,5
Camat	0	-1,5
Masuk	0	-1,5

Setelah mendapatkan *entropy global* dan *entropy* kemunculan fitur maka dapat dihitung *information gain* dari setiap fitur. Akan tetapi dalam perhitungan ini juga diperlukan jumlah dokumen yang memiliki fitur dan tidak secara keseluruhan dan jumlah sampel data latih. Perhitungan information gain ini menggunakan persamaan 2.7. Pada kata "mohon" *information gain*-nya dapat dihitung seperti berikut:

$$Information\ gain = 1{,}521928095 + \left(\frac{1}{5} \times 0\right) + \left(\frac{4}{5} \times (-1)\right)$$

$$= 0{,}721928095$$

Perhitungan *information gain* tersebut diterapkan terhadap seluruh fitur, sehingga hasil information gain dari keseluruhan fitur ditunjukkan pada Tabel 4.18. Hasil *information gain* pada Tabel 4.18 kemudian dilakukan pengurutan dari terbesar ke terkecil, dimana hasil pengurutan ditunjukkan pada Tabel 4.19.

Tabel 4.18 *Information gain* seluruh fitur

Fitur	Information Gain
Mohon	0,721928095
Traffic	0,721928095
Light	0,721928095
Seberang	0,721928095
baik	0,721928095
fungsi	0,721928095
pasar	0,321928095
gadang	0,321928095
jalan	0,970950594
lubang	0,970950594
jembatan	0,321928095
lori	0,321928095
rw	0,321928095
kel	0,321928095
bumiayu	0,321928095
memperihatinkan	0,321928095
tiang	0,321928095
sangga	0,321928095
kropos	0,321928095
jukir	0,321928095
karcis	0,321928095
sukarela	0,321928095
titik	0,321928095
parkir	0,970950594
kota	0,321928095
malang	0,321928095
yth	0,321928095
walikota	0,321928095
bpk	0,321928095
hm	0,321928095
anton	0,321928095
jln	0,321928095
sulfat	0,321928095
tepat	0,321928095
areal	0,321928095
puskesmas	0,321928095
kendalkerep	0,321928095
bayar	0,321928095
obat	0,321928095
gratis	0,321928095

Tabel 4.18 *Information gain* seluruh fitur

Fitur	Information Gain
tetep	0,321928095
mbayar	0,321928095
milik	0,321928095
pemerinta	0,321928095
masak	0,321928095
harap	0,321928095
lurah	0,321928095
camat	0,321928095
masuk	0,321928095

Tabel 4.19 Hasil pengurutan fitur berdasarkan nilai *information gain*

Fitur	Information Gain
jalan	0,970950594
lubang	0,970950594
parkir	0,970950594
mohon	0,721928095
traffic	0,721928095
light	0,721928095
seberang	0,721928095
baik	0,721928095
fungsi	0,721928095
pasar	0,321928095
gadang	0,321928095
jembatan	0,321928095
lori	0,321928095
rw	0,321928095
kel	0,321928095
bumiayu	0,321928095
memperihatinkan	0,321928095
tiang	0,321928095
sangga	0,321928095
kropos	0,321928095
jukir	0,321928095
karcis	0,321928095
sukarela	0,321928095
titik	0,321928095
kota	0,321928095
malang	0,321928095
yth	0,321928095

walikota	0,321928095
bpk	0,321928095
hm	0,321928095
anton	0,321928095
jln	0,321928095
sulfat	0,321928095
tepat	0,321928095
areal	0,321928095
puskesmas	0,321928095
kendalkerep	0,321928095
bayar	0,321928095
obat	0,321928095
gratis	0,321928095
tetep	0,321928095
mbayar	0,321928095
milik	0,321928095
pemerinta	0,321928095
masak	0,321928095
harap	0,321928095
lurah	0,321928095
camat	0,321928095
masuk	0,321928095

Dari hasil pengurutan pada tabel 4.19, kemudian diambil sejumlah fitur tertinggi sebagai hasil seleksi. Pada manualisasi ini, fitur diambil sebesar 10% dari jumlah seluruh fitur. Jumlah seluruh fitur adalah 49 sehingga jumlah fitur hasil seleksi adalah:

$$Jumlah\ fitur\ hasil\ seleksi = 10\% \times 49 = 4{,}9 \approx 5$$

Hasil perhitungan 10% dari jumlah fitur seluruhnya didapatkan hasil 5, sehingga hasil seleksi *information gain* ini adalah 5 fitur yang memiliki nilai *information gain* tertinggi. Berdasarkan pada Tabel 4.19, 5 fitur yang memiliki nilai *information gain* tertinggi ditunjukkan pada Tabel 4.20. Fitur hasil *information gain* tersebut kemudian akan dilakukan proses kombinasi dengan menggunakan *genetic algorithm* untuk mendapatkan akurasi yang baik.

Tabel 4.20 Fitur hasil seleksi *information gain*

No	Fitur
1	jalan
2	lubang
3	parkir
4	mohon
5	traffic

4.2.3 Genetic Algorithm

Dari seluruh fitur hasil seleksi menggunakan teknik *information gain*, akan dilakukan kombinasi dengan menggunakan representasi biner. Pada manualisasi ini, jumlah *popsize* yang digunakan sebesar 3, nilai *crossover rate* (cr) dan *mutation rate* (mr) yang digunakan adalah 0.5 dengan maksimum generasi sebanyak 2 . Langkah awal pada genetic algorithm ini adalah melakukan inisialisasi kromosom sebagai solusi awal yang akan diproses. Inisialisasi kromosom dilakukan secarak acak. Hasil inisialisasi kromosom pada manualisasi ini ditunjukkan pada Tabel 4.21.

Tabel 4.21 Inisialisasi kromosom

	X_1	X_2	X_3	X_4	X_5
P_1	1	0	1	1	0
P_2	0	0	1	0	1
P_3	0	1	0	1	0

Pada Tabel 4.21, terdapat simbol P yang merupakan *parent* (induk) yang berjumlah 3 yaitu sejumlah *popsize* yang telah ditentukan diawal. Sedangkan X merupakan fitur hasil seleksi dari information gain. Dikarenakan jumlah fitur hasi information gain berjumlah 5, maka nilai X yang ada pada proses *genetic algorithm* ini juga berjumlah 5. X_1 pada Tabel 4.21 adalah untuk fitur "jalan", X_2 untuk fitur "lubang", X_3 untuk fitur "parkir", X_4 untuk fitur "mohon", dan X_5 untuk fitur "traffic". Setiap gen pada setiap fitur pada setiap induk memiliki nilai 0 atau 1 dikarenakan menggunakan representasi biner.

Setelah dilakukan proses inisialisasi, maka langkah selanjutnya adalah melakukan proses reproduksi yaitu proses *crossover*. Jumlah *offspring* yang harus dihasilkan pada proses reproduksi dengan *crossover* dihitung dari perkalian nilai cr dan *popsize*. Pada manualisasi ini, jumlah offspring yang harus dihasilkan pada proses crossover adalah:

$$Jumlah\ offspring\ hasil\ crossover = cr \times popsize$$
$$= 0{,}5 \times 3 = 1{,}5 \approx 2$$

Dari hasil perhitungan tersebut maka jumlah *offspring* yang harus dihasilkan sebanyak 2 *offspring*. Sebelum *crossover* dilakukan, diharuskan memilih 2 induk secara acak dari populasi untuk dilakukan tukar silang. Pada crossover ini, induk yang terpilih adalah P_1 dan P_3. Dari kedua induk tersebut, ditentukan titik potong yang sama. Dikarenakan teknik crossover yang digunakan adalah *two cut point crossover* maka jumlah titik potong yang harus ditentukan sebanyak 2. Pada crossover ini, 2 titik potong yaitu setelah indeks 0, dan sebelum indeks 4 apabila perhitungan indeks dimulai dari 0. Apabila diilustrasikan dalam bentuk gambar ditunjukkan pada Gambar 4.33.

P_1	1	0	1	1	0
P_3	0	1	0	1	0

Gambar 4.43 Ilustrasi proses *crossover*

Garis vertikal pada Gambar 4.43 merupakan titik potong dari 2 induk yang telah dipilih secara acak. Dari gambar ilustrasi tersebut *offspring* yang dihasilkan didapatkan dari pertukaran gen kedua induk tersebut yang terletak diantara dua titik potong. Sehingga *offspring* hasil dari proses *crossover* tersebut ditunjukkan pada Tabel 4.22 dimana C merupakan simbol dari *child*.

Tabel 4.22 *Offspring* hasil *crossover*

	X_1	**X_2**	**X_3**	**X_4**	**X_5**
C_1	1	1	0	1	0
C_2	0	0	1	1	0

Setelah dilakukan reproduksi dengan *crossover*, maka reproduksi dilakukan kembali dengan cara yang berbeda yaitu dengan mutasi. Jumlah offspring yang harus dihasilkan dari reproduksi dengan mutasi ini adalah sejumlah hasil perkalian mr dengan popsize. Pada manualisasi ini, jumlah *offspring* yang harus dihasilkan pada reproduksi dengan mutasi adalah:

$$Jumlah\ offspring\ hasil\ mutasi = mr \times popsize = 0{,}5 \times 3 = 1{,}5 \approx 2$$

Sebelum proses mutasi dilakukan maka terlebih dahulu memilih satu induk untuk dilakukan proses mutasi. Dikarenakan sekali mutasi hanya menghasilkan satu *offspring* maka perlu dilakukan proses mutasi sebanyak dua kali. Pada mutasi pertama induk yang terpilih adalah P_2. Dari induk tersebut dipilih satu buah gen secara acak dan tepilih gen pada indeks 1 apabila indeks gen dihitung mulai 0. Dari gen tepilih tersebut, offspring hasil mutasi dapat didapakan dengan melakukan invers pada nilai dari gen terpilih. Ilustrasi proses mutasi pertama ditunjukkan pada Tabel 4.23.

Tabel 4.23 Ilustrasi proses mutasi pertama

P_2	0	<u>0</u>	1	0	1
C	0	<u>1</u>	1	0	1

Pada Tabel 4.23, terdapat nilai gen yang digaris bawahi. Gen tersebut adalah gen terpilih darihasil acak indeks gen. Dikarenakan pada gen tersebut nilai gen adalah 0, maka hasil offspring pada gen tersebut bernilai 1, sedangkan gen lain yang tidak terpilih bernilai sama dengan induk. Pada mutasi kedua, untuk mendapatkan *offspring* satu lagi, dilakukan dengan cara yang sama dengan mutasi pertama, akan tetapi yang menjadi induk pada mutasi kedua adalah P_1 dengan gen terpilih adalah gen pada indeks ke 2 apabila indeks dimulai dari 0. Ilistrasi proses mutasi kedua ditunjukkan pada Tabel 4.24.

Tabel 4.24 Ilustrasi proses mutasi kedua

P_1	1	0	1	1	0
C	1	0	0	1	0

Pada Tabel 4.24, nilai gen pada indeks kedua bernilai 1, sehingga gen pada indeks tersebut akan bernilai 0 untuk *offspring* yang dihasilkan. Sehingga dari kedua mutasi yang telah dijalankan didapatkan jumlah *offspring* total sebanyak 2 *offspring*. *Offspring* hasil mutasi tersebut ditunjukkan pada Tabel 4.25. Pada Tabel 4.25, C_3 adalah *offspring* hasil mutasi pertama dan C_4 adalah *offspring* hasil mutasi kedua.

Tabel 4.25 *Offspring* hasil mutasi

	X_1	X_2	X_3	X_4	X_5
C_3	0	1	1	0	1
C_4	1	0	0	1	0

Setelah dilakukan proses reproduksi baik dengan *crossover* maupun mutasi, selanjutnya adalah mengumpulkan seluruh individu baik *parent* maupun *offspring* kedalam satu kelompok. Proses ini digunakan untuk persiapan evaluasi setiap individu. Hasil pengumpulan seluruh individu kedalam satu kelompok ditunjukkan pada Tabel 4.26. Pada Tabel 4.26 P_1, P_2, dan P_3 merupakan *parent* dan C_1, C_2, C_3, dan C_4 merupakan *offspring* hasil reproduksi dimana C_1 dan C_2 hasil reproduksi dengan cara *crossover* dan C_3 dan C_4 adalah *offspring* hasil mutasi.

Tabel 4.26 Hasil pengumpulan seluruh individu dalam satu kelompok

	X_1	X_2	X_3	X_4	X_5
P_1	1	0	1	1	0
P_2	0	0	1	0	1
P_3	0	1	0	1	0
C_1	1	1	0	1	0
C_2	0	0	1	1	0
C_3	0	1	1	0	1
C_4	1	0	0	1	0

Setiap individu pada Tabel 4.26 akan dilakukan proses evaluasi untuk mendapatkan nilai *fitness*. Proses perhitungan *fitness* diawali dengan menghitung *term weighting*. Proses *term weighting* ini melibatkan data latih dan data uji serta fitur yang bernilai 1 pada individu. Pada individu pertama (P_1) berdasarkan pada Tabel 4.26, nilai 1 terletak pada X_1, X_3, dan X_4. X_1, X_3, dan X_4 merupakan fitur yang akan digunakan untuk menghitung nilai *fitness* pada individu pertama (P_1), dimana X_1 adalah jalan, X_2 adalah parkir, dan X_3 adalah mohon. Sehingga fitur yang akan digunakan untuk menghitung fitness individu pertama (P_1) adalah "jalan", "pakir", dan "mohon". Fitur-fitur tersebut dilakukan proses *term weighting* terhadap setiap data uji. Proses *term weighting* diawali dengan menghitung *term frequency*. Sehingga pada data uji pertama dan data latih hasil term frekuensinya ditunjukkan pada tabel 4.27.

Tabel 4.27 *Term frequency* data uji pertama dan data latih

	Q1	Doc1	Doc2	Doc3	Doc4	Doc5
jalan	1	0	1	1	0	0
parkir	1	0	0	0	1	4
mohon	0	1	0	0	0	0

Pada Tabel 4.27, Q merupakan kependekan dari *Query*. Yang dimaksud dengan *query* disini adalah data uji yang akan dilakukan proses klasifikasi. Pada data uji pertama mengandung kata "jalan" sebanyak 1, kata "parkir" sebanyak 1 dan pada data uji tersebut tidak mengandung kata mohon. Sedangkan pada data latih pertama (Doc1) hanya mengandung kata "mohon" dengan frekuensi kemunculan sebanyak 1, begitupun dengan data latih yang lain dan fitur yang latin terhadap data uji.

Setelah dilakukan perhitungan *term frequency* terhadap data uji dan data latih, selanjutnya adalah perhitungan bobot *term* terhadap dokumen. Pada data uji pertama dan seluruh data latih dengan fitur "jalan", "parkir" dan "mohon" didapatkan bobot *term* terhadap dokumen yang ditunjukkan pada Tabel 4.28.

Tabel 4.28 Hasil pembobotan *term* terhadap data uji pertama dan data latih

	Q1	Doc1	Doc2	Doc3	Doc4	Doc5
jalan	1,30103	0	1,30103	1,30103	0	0
parkir	1,30103	0	0	0	1,30103	5,20412
mohon	0	1,778151	0	0	0	0

Pada Tabel 4.28, perhitungan *bobot* term terhadap dokumen dihitung menggunakan persamaan 2.3. Pada Q1 dan fitur "jalan", nilai 1,30103 didapatkan dari:

$$W_{(jalan,Q1)} = 1 \times \left(1 + \log\frac{6}{3}\right) = 1{,}30103$$

Pada perhitungan diatas, niali 1 sebelum simbol × merupakan *term frequency* dari kata "jalan" pada data uji pertama (Q1), dengan kata lain bahwa Q1 hanya menganding 1 kata "jalan" didalamnya sesuai pada Tabel 4.27. Sedangkan angka 6 yang menjadi pembilang merupakan jumlah dokumen yang terlibat, dimana yang terlibat adalah 5 buah dokumen dari data latih dan satu buah dokumen dari data uji, sehingga apabila dijumlahkan menghasilkan nilai 6. Angka 3 yang menjadi penyebut adalah jumlah dokumen yang mengandung *term* dimana pada hal ini adalah kata "jalan". Berdasarkan pada Tabel 4.27, yang mengandung kata "jalan" berjumlah 3 dokumen yaitu Q1, Doc2, dan Doc3 yang masing-masing bernilai lebih dari 0 yaitu 1. Pembobotan dilakukan terhadap seluruh term seluruh dokumen.

Setelah dilakukan perhitungan bobot *term* terhadap dokumen, maka hasil pembobotan tersebut dilakukan proses normalisai. Normalisasi pembobotan *term* terhadap dokumen dihitung menggunakan persamaan 2.4. pada persamaan tersebut terdapat perhitungan penjumlahan dari perkalian kuadrat dari nilai *term frequency* dengan kuadrat dari 1+log(D/DT) yang biasa disebut dengan IDF

dari seluruh fitur setiap dokumen. Apabila perhitungan tersebut dihitung terlebih dahulu maka didapatkan hasil sebagaimana yang terdapat pada Tabel 4.29.

Tabel 4.29 Hasil perkalian kuadrat *term frequency* dengan kuadrat IDF beserta totalnya pada data uji pertama dan data latih

	Q1	Doc1	Doc2	Doc3	Doc4	Doc5
jalan	1,692679	0	1,692679	1,692679	0	0
parkir	1,692679	0	0	0	1,692679	27,08286
mohon	0	3,161822	0	0	0	0
∑ (Sigma)	3,385358	3,161822	1,692679	1,692679	1,692679	27,08286

Pada Tabel 4.29, nilai pada kata "jalan" terhadap Q1 adalah 1,692679. Nilai tersebut didapatkan dari perkalian *term frequency* dari fitur "jalan" yang terdapat pada data uji pertama yang telah dikuadratkan dengan hasil logaritma dari pembagian jumlah dokumen yang terlibat dibagi dengan jumlah dokumen yang mengandung kata "jalan" yang kemudian ditambah satu dan hasilnya dikuadratkan. Apabila dituliskan maka:

$$KuadratTFIDF_{(jalan,Q1)} = 1^2 \times \left(1 + \log\frac{6}{3}\right)^2 = 1{,}692679$$

Pada perhitungan tersebut angka 1 adalah jumlah munculnya kata "jalan" pada dokumen uji pertama dan angka 6 adalah jumlah dokumen yang terlibat serta 3 merupakan jumlah dokumen yang terlibat dan mengandung kata "jalan". Perhitngan tersebut dilakukan terhadap seluruh fitur pada setiap dokumen. Kemudian pada baris ∑ merupakan hasil penjumlahan dari seluruh hasil perkalian kuadrat TF yang ditambah dengan kuadrat dari IDF pada dokumen yang sama. Pada Tabel 4.29, nilai ∑ pada dokumen uji pertama adalah sebesar 3,385358. nilai tersebut diperoleh dari perhitungan berikut :

$$Sigma_{(Q1)} = 1{,}692679 + 1{,}692679 + 0 = 3{,}385358$$

Pada perhitungan sigma pada kolom Q1 diatas, nilai $1{,}692679$ pertama merupakan hasil kuadrat TF yang dikalikan dengan kuadrat IDF pada fitur "jalan" terhadap dokumen Q1, sedangkan nilai $1{,}692679$ yang kedua merupakan hasil kuadrat TF yang dikalikan dengan kuadrat IDF pada fitur "parkir" terhadap dokumen Q1, dan nilai 0 dari perhitungan tersebut adalah hasil kuadrat TF yang dikalikan dengan kuadrat IDF pada fitur "traffic" terhadap dokumen Q1. Dari hasil perhitungan tersebut maka normalisasi WTD dapat dihitung dengan membagi nilai bobot term terhadap dokumen yang terdapat pada Tabel 4.28 dengan hasil akar kuadrat dari ∑ yang terdapat pada Tabel 4.29. Apabila perhititungan tersebut diterapkan pada manualisasi ini, maka didapatkan hasil normalisasi pembobotan term terhadap dokumen yang ditunjukkan pada Tabel 4.30.

Tabel 4.30 Hasil normalisasi bobot term terhadap dokumen pada data ujji pertama dan data latih

	Q1	Doc1	Doc2	Doc3	Doc4	Doc5
jalan	0,707107	0	1	1	0	0

parkir	0,707107	0	0	0	1	1
mohon	0	1	0	0	0	0

Pada Tabel 4.29, nilai kata "jalan" terhadap data uji pertama (Q1) adalah 0,707107. Apabila ditelusuri, nilai tersebut didapatkan dari:

$$Normalisasi\ W_{(jalan,Q1)} = \frac{1{,}30103}{\sqrt{3{,}385358}} = 0{,}707107$$

Pada perhitungan normalisasi W tersebut, nilai 1,30103 merupakan bobot kata "jalan" pada dokumen uji pertama (Q1) berdasarkan pada Tabel 4.28. sedangkang 3,385358 merupakan hasil nilai penjumlahan (sigma) pada dokumen uji pertama (Q1) berdasarkan pada Tabel 4.29. Perhitungan diterapkan terhadap seluruh fitur pada setiap dokumen.

Setelah mendapatkan hasil normalisasi bobot term terhadap dokumen, maka dapat dihitung kemiripan dokumen antara dokumen uji dengan dokumen latih. Kemiripan dokumen dilihat berdasarkan nilai dari *cosine similarity* dimana *cosine similarity* tersebut dapat diperoleh menggunakan persamaan 2.5. Perhitungan *cosine similarity* dilakukan terhadap seluruh dokumen uji. Hasil perhitungan *cosine similarity* pada data uji pertama dengan fitur "jalan", "parkir", dan "mohon" ditunjukkan pada Tabel 4.31.

Tabel 4.31 Hasil *cosine similarity* data uji pertama

	Doc1	**Doc2**	**Doc3**	**Doc4**	**Doc5**
Q1	0	0,707107	0,707107	0,707107	0,707107

Pada Tabel 4.31, kemiripan dokumen uji pertama dengan dokumen pertama adalah 0, nilai tersebut didapatkan dari:

$$Sim_{cosine}(Q_1, Doc_1) = (0{,}707107 \times 0) + (0{,}707107 \times 0) + (0 \times 1) = 0$$

Pada perhitungan tersebut, pada perkalian pertama, nilai 0,707107 merupakan nilai hasil normalisasi pembobotan kata "jalan" pada data uji pertama (Q1), dan 0 merupakan hasil normalisasi pembobotan kata "jalan" pada data latih pertama (Doc1). Pada perkalian kedua, 0,707107 merupakan nilai hasil normalisasi pembobotan kata "jalan" pada data uji pertama (Q1), dan 0 merupakan hasil normalisasi pembobotan kata "parkir" pada data latih kedua (Dok2). Sedangkan pada perkalian ketiga, 0 merupakan merupakan nilai hasil normalisasi pembobotan kata "jalan" pada data uji pertama (Q1), dan 1 merupakan hasil normalisasi pembobotan kata "mohon" pada data latih pertama (Doc3). Nilai-nilai yang terlibat dalam perkalian pada perhitungan *cosine similarity* adalah nilai hasil normalisasi bobot term terhadap dokumen yang terdapat pada Tabel 4.30. Proses *term weighting* hingga ditemukannya kemiripan dokumen (*cosine similarity*) dilakukan terhadap seluruh data uji yang ada.

Setelah ditemukannya cosine similarity, maka proses klasifikasi dapat dilakukan dengan menggunakan NW-KNN. Langkah awal dalam proses klasifikasi adalah menentukan jumlah tetangga (k) dengan nilai *cosine similarity* tertinggi. Pada manualisasi ini, nilai k yang digunakan adalah 3. Sebagai contoh adalah pengklasifikasian data uji pertama (Q1). Apabila nilai *cosine similarity* data uji

pertama (Q1) dengan data latih yang telah diperoleh dan telah ditunjukkan pada Tabel 4.31 maka akan dilakukan pengurutan secara *dercending* (dari besar ke kecil) dan diambil sejumlah k nilai tertinggi seperti yang telah ditabelkan pada Tabel 4.32.

Tabel 4.32 K *cosine similarity* tertinggi

	Doc2	Doc3	Doc4
Q1	0,707107	0,707107	0,707107
Kelas	2	2	3

Tabel 4.32, merupakan tabel hasil pengurutan kemiripandokumen uji pertama, dari tabel tersebut dapat diketahui bahwa yang memiliki nilai *cosine similarity* adalah kemiripan dari dokumen Q1 dengan Doc2, Q1 dengan Doc3, dan Q1 dengan Doc4, dimana Doc2 dan Doc3 adalah data latih dari dari kelas 2, dan Doc4 adalah data latih dari kelas 3.

Dari Tabel 4.32, kemudian dapat dilakukan perhitungan bobot dari masing-masing kategori dengan menggunakan persamaan 2.6. Sebagai contoh adalah pada dokumen uji pertama, telah diketahui nilai *cosine similarity* tertinggi dari dokumen uji pertama yaitupada Tabel 4.32. Dari nilai-nilai pada tabel tersebut dapat dihitung bobot dari setiap kelas baik DKP, DPUPPB, maupun Dishub. Bobot dari DKP apabila dihitung secara manual adalah:

$$\boldsymbol{Weight_1 = \frac{1}{\left(\frac{1}{Min\ (2,\ 1)}\right)^{\frac{1}{exp}}} = \frac{1}{\left(\frac{1}{1}\right)^{\frac{1}{exp}}} = 1}$$

Pada perhitungan tersebut, pada penyebut, yang menjadi pembilang pada pecahan penyebut adalah jumlah data latih yang merupakankelas DKP (1) yaitu sebanyak 1 dokumen, sedangkan yang menjadi penyebut, nilai 2 merupakan jumlah dokumen kelas 2 yang terdapat pada k tertinggi, dan nilai 1adalah jumlah dokumen kelas 3 yang terdapat pada k tertinggi. Dari nilai-nilai penyebut tersebut terdapat operator min, yang mencari nilai terkecil dari2dan 1, sehingga diperoleh 1, dan apabila dihitung akan mendapatkan nilai 1 yang menjai bobot dari kelas 1 (DKP). Sehingga pada data uji pertama dengan menggunakan fitur "jalan", "parkir" dan "mohon" didapatkan hasil bobot setiap kelas seperti pada Tabel 4.33. Pada Tabel 4.33, kelas DKP memiliki bobot tebesar yaitu 1, sedangkan bobot kelas DPUPPB dan Dishub adalah 0,774921.

Tabel 4.33 Bobot masing-masing kelas pada data uji pertama

Kelas	Weight
DKP(1)	1
DPUPPB(2)	0,774921
Dishub(3)	0,774921

Setelah ditemukan bobot tiap kelas, maka kemudian menghitung *score* data uji terhadap setiap kelas sebagai penentu hasil klasifikasinya. *Score* tersebut dihitung menggunakan persamaan 2.7. Apabila persamaan tersebut diterapkan pada perhitungan *score* dari data uji pertama hasil penggunaanfitur "jalan", "parkir" dan mohon, maka perhitungan tersebut adalah sebagai berikut:

$$\boldsymbol{Score\ (Q1, DKP) = 1 \times ((0,707107 \times 0) + (0,707107 \times 0) + (0,707107 \times 0)\ ) = 0}$$

Dari perhitungan tersebut, pengali pertama adalah 1 yang merupakan bobot dari kelas DKP (1) sedangkan nilai 0,707107merupakan nilai *cosine similarity* Q1 dengan Doc2, Doc3, dan Doc4 secara berturut turut. Dan nilai 0 adalah nilai kelas Doc2, Doc3, dan Doc4 secara berturut-turut yang bukan dari kelas DKP, apabila dokumen tersebut merupakan kelas DKP maka cosine similarity dari dokumen tersebut akan dikalikan dengan 1. Sehingga apabila ditabelkan, *score* dari data uji terhadap kategori ditunjukkan pada Tabel 4.34.

Tabel 4.34 *Score* data uji pertama

Kelas	Score
DKP(1)	0
DPUPPB(2)	1,095903
Dishub(3)	0,547952

Pada Tabel 4.34, hasil score tertinggi terdapat pada DPUPPB yaitu sebesar 1,095903. Sehingga data uji pertama (Q1) terkategorikan sebagai aduan yang ditunjukkan pada SKPD Dinas Kebersihan dan Pertamanan (DKP). Perhitungan tersebut diterapkan terhadap seluruh data uji yang ada, sehingga pada manualisasi ini didapatkan hasil perhitungan score dan hasil klasifikasi seperti pada Tabel 4.35.

Tabel 4.35 *Score* seluruh data uji

Data Uji	Score			Hasil Klasifikasi
	DKP(1)	DPUPPB(2)	Dishub(3)	
Q1	0	1,095903	0,547952	2
Q2	0	1,549841	0	2
Q3	0,35395	0	1,449511	3

Untuk melakukan proses evaluasi, maka diperlukan pencocokan kelas aktual dengan kelas hasil klasifikasi. Kelas aktual dan kelas hasil klasifikasi dari seluruh data uji tersebut ditunjukkan pada Tabel 4.36.

Tabel 4.36 Kelas aktual dan hasil klasifikasi data uji

Data Uji	Kelas	
	Aktual	Hasil Klasifikasi
Q1	1	2
Q2	2	2
Q3	3	3

Dari Tabel 4.36 tersebut dapat di hitung jumlah *true positive, false positive,* dan *false negative* untuk mendapatkan nilai rata-rata *f-measure* sebagai nilai fitness dari individu pertama yaitu (P_1). Jumlah *true positive, false positive,* dan *false negative* ditunjukkan pada Tabel 4.37.

Tabel 4.37 Jumlah *true positive*, *false positive*, dan *false negative*

	DKP (1)	DPUPPB (2)	Dishub (3)
TP	0	1	1
FP	0	1	0
FN	1	0	0

Pada Tabel 4.37, jumlah dokumen uji dimana kelas aktual sama dengan hasil klasifikasi pada kelas DKP ada 0, jumlah dokumen uji dimana hasil klasifikasi daru data uji tersebut adalah DKP sedangkan kelas yang sebenarnya (aktual) bukan DKP berjumlah 0, dan jumlah dokumen uji dimana kelas sebenarnya (aktual) merupakan kelas DKP (1) sedangkan pada kelas hasil klasifikasi menunjukkan data uji tersebut bukan kelas DKP (1) berjumlah 1 dokumen.

Setelah didapatkan nilai *true positive, false positive,* dan *false negative*, maka nilai *f-measure* dapat dihitung dengan menghitung nilai *precision* dan *recall* setiap kelas. Perhitungan *precision* dan *recall* pada kelas 1 (DKP) secara berturut-turut dihitung dengan menggunakan persamaan 2.11 dan 2.12 sebagaimana berikut:

$$Precision_1 = \frac{0}{0+0} = \frac{0}{0} = \infty \approx 0$$

$$Recall_1 = \frac{0}{0+1} = \frac{0}{1} = 0$$

Pada perhitungan *precision* yang menjadipembilang adalah nilai *true positive* (TP) pada kelas DKP, sedangkan yang menjadi penyebut adalah penjumlahan nilai *true posoitive* (TP) dengan *false positive* (FP) dari kelas DKP, dikarenakan 0 dibagi 0 hasilnya adalah tak terdefinisi, maka nilai *precision* dianggap nol. Sedangkan pada perhitungan *recall* untuk kelas DKP, sama dengan pada perhitungan *precision* yang menjadi pembagi (pembilang) adalah nilai *true positive* (TP) pada kelas DKP, sedangkan yang menjadi penyebut adalah penjumlahan nilai *true posoitive* (TP) dengan *false negative* (FN) dari kelas DKP. Dikarenakan 0 dibagi dengan 1 nilainya nol, maka nilai *recall* pada kelas DKP adalah 0. Dari nilai *precision* dan *recall* dari kelas DKP yang telah diperoleh, maka nilai *f-measure* pada kelas DKP dapat diperoleh dengan menggunakan persamaan 2.10 seperti berikut:

$$f - measure_1 = 2 \times \frac{0 \times 0}{0+0} = \frac{0}{0} = \infty \approx 0$$

Dari perhitungan tersebut, nilai 0 adalah nilai *precision* dan *recall* dari data kelas DKP. Perhitungan *precision* dan *recall* tersebut diterapkan pada seluruh kelas, dimana hasil perhitungannya ditunjukkan pada Tabel 4.38.

Tabel 4.38 *Precision, recall* dan *f-measure* tiap kategori

	DKP	DPUPPB	Dishub
Precision	0	0,5	1
Recall	0	1	1
F-measure	0	0,666667	1

Dari Tabel 4.38, nilai *fitness* dari individu pertama (P_1) pada algoritme genetika ini, dihitung dengan menggunakan persamaan 2.15 sebagaimana berikut :

$$Rata - rata\ f - measure = \frac{(1\times0)+(1\times0{,}666667)+(1\times1)}{3} = \frac{1{,}666667}{3} = 0{,}555556$$

Pada perhitungan rata-rata *f-measure* nilai 1 pada pembilang secara berturut-turut adalah jumlah dokumen uji yang merupakan kelas DKP, DPUPPB, dan Dishub secara berturut-turut pada kelas sebenarnya yang dikalikan dengan nilai *f-measure* pada kelas DKP, DPUPPB, dan Dishub secara berturut-turut, yang kemudian dibagi dengan jumlah data uji yaitu berjulah 3 data uji, sehingga dihasilkan 0,555556 sebagai nilai *fitness* dari individu pertama yaitu (P_1).

Langkah-langkah tersebut dilakukan terhadap seluruh individu, sehingga didapatkan nilai *fitness* masing-masing individu sebagai hasil evaluasi. Hasil evaluasi seluruh individu ditunjukkan pada Tabel 4.39.

Tabel 4.39 Hasil evaluasi seluruh individu

Individu	*Fitness*
P_1	0,555556
P_2	0,222222
P_3	0
C_1	0,166667
C_2	0,222222
C_3	0,333333
C_4	0,222222

Hasil dari evaluasi tersebut kemudian dilakukan pengurutan secara *descending* sebagai proses seleksi pada algoritme genetika ini, sehingga didapatkan hasil pengurutan seperti pada Tabel 4.40.

Tabel 4.40 Seleksi seluruh individu

Individu	*Fitness*
P_1	0,555556
C_3	0,333333
P_2	0,222222
C_2	0,222222
C_4	0,222222
C_1	0,166667
P_3	0

Dari hasil seleksi yang tertera pada Tabel 4.40,kemudian diambil individu yang memiliki nilai *fitness* tertinggi sejumalah *popsize* yaitu 3 individu, sehingga individu yang akan digunakan pada proses generasi selanjutnya adalah P_1, P_2, dan P_3 sebagaimana yang ditunjukkan pada Tabel 4.41.

Tabel 4.41 Individu hasil proses seleksi

	X_1	X_2	X_3	X_4	X_5
P_1	1	0	1	1	0
C_3	0	1	1	0	1
P_2	0	0	1	0	1

Pada generasi selanjutnya, *parent* (induk) yang digunakan bukan lagi hasil acak seperti pada saat inisialisai, melainkan individu-individu yang terdapat pada Tabel 4.41. Dari individu-individu tersebut akan dilakukan proses reproduksi dengan *crossover* dan mutasi seperti pada generasi sebelumnya. Seluruh individu baik *parent* maupun *offspring* juga akan dikumpulkan kedalam satu kelompok sebagai persiapan proses evaluasi. Hasil pengumpulan seluruh individu kedalam satu kelompok ditunjukkan pada Tabel 4.42.

Tabel 4.42 Hasil pengumpulan seluruh individu pada generasi kedua dalam satu kelompok

	X_1	X_2	X_3	X_4	X_5
P_1	1	0	1	1	0
P_2	0	1	1	0	1
P_3	0	0	1	0	1
C_1	1	1	1	0	0
C_2	0	0	1	1	1
C_3	1	0	1	0	0
C_4	1	0	1	0	1

Pada Tabel 4.42, P_1 hingga P_3 adalah individu hasildari generasi sebelumnya yaitu generasi 1, C_1 dan C_2 merupakan *offspring* darihasil *crossover* dan C_3 dan C_4 adalah *offspring* hasil mutasi. Seperti halnya pada generasi pertama, individu-individu tersebut dilakukan evaluasi untuk mendapatkan nilai *fitness*. Nilai fitness dari individu pada generasi ke-2 ditunjukkan pada Tabel 4.43.

Tabel 4.43 Hasil evaluasi seluruh individu pada generasi kedua

Individu	*Fitness*
P_1	0,555556
P_2	0,222222
P_3	0
C_1	0,555556
C_2	0,222222
C_3	0,166667
C_4	0,555556

Dari hasil evaluasi yang terdapat pada Tabel 4.43, akan dilakukan proses seleksi seperti yang telah dilakukan pada generasi pertama,sehingga didapatkan indivdu-individu hasil seleksi yaitu ditunjukkan pada Tabel 4.44.

Tabel 4.44 Hasil seleksi seluruh individu pada generasi kedua

	X_1	X_2	X_3	X_4	X_5
P_1	1	0	1	1	0
C_1	1	1	1	0	0
C_4	1	0	1	0	1

Pada Tabel 4.45, individu P_1 yang merupakan individu pertama dari proses generasi kedua memiliki nilai *fitness* tertinggi, sehingga individu (kombinasi fitur) tersebut dianggap sebagi hasil yang paling optimal dan didapatkan fitur hasil seleksi algoritme genetika yang terdapat pada Tabel 4.45. Dari hasil seleksi fitur menggunakan algoritme genetika yang terdapat pada Tabel 4.45, maka fitur-fitur tersebut yang akan digunakan untuk mengklasifikasikan data baru yang berupa aduan masyarakat Kota Malang.

Tabel 4.45 Fitur hasil algoritme genetika

No	Fitur
1	jalan
2	parkir
3	mohon

BAB 5 IMPLEMENTASI

5.1 Pre-processing

Berdasarkan pada Gambar 4.2 proses dari pre-processing diimplementasikan ke dalam bentuk program seperti pada Tabel 5.1. Pada tabel tersebut proses yang terjadi pada pre-processing ditunjukkan pada baris 4 yaitu pemanggilan method caseFolding(). Setelah itu dilakukan proses tokenisasi setiap dokumen dari data yang terjadi pada baris 5-6. Setelah seluruh dokumen selesai dilakukan proses tokenisasi, kemudian dilakukan penghapusan *stopword* dengan memanggil method removeStopword() pada baris 9. Baris 10-15 akan dijalankan apabila baris 9 selesai dijalankan. Pada baris ini, dilakukan proses stemming dari setiap token yang tersisa hasil dari penghapusan *stopword*. Proses stemming ini lebih tepatnya terjadi pada baris 12, dimana token/kata yang dilakukan *stemming* aakan diubah menjadi output dari proses *stemming* yang dilakukan dengan memanggil method stemming().

Tabel 5.1 Implementasi pre-processing

```
1     public static ArrayList<ArrayList<String>>
      preProcessing(ArrayList<String> dt) {
2         ArrayList<ArrayList<String>> kata = new
      ArrayList<>();
3         ArrayList<String> temp = new ArrayList<>();
4         caseFolding(dt);
5         for (int i = 0; i < dt.size(); i++) {
6             temp = tokenisasi(dt.get(i));
7             kata.add(temp);
8         }
9         removeStopword(kata);
10        for (int i = 0; i < kata.size(); i++) {
11            for (int j = 0; j < kata.get(i).size(); j++) {
12                kata.get(i).set(j,
      stemming(kata.get(i).get(j)));
13            }
14        }
15        return kata;
16    }
```

Berdasarkan pada Tabel 5.1 terdapat pemanggilan method caseFolding(). Di dalam bab 4 perancangan alur dari proses case folding ditunjukkan pada Gambar 4.3, dimana apabila diimplementasikan kedalam kode-kode program ditunjukkan pada Tabel 5.2. Pada tabel tersebut proses *case folding* ditunukkan pada baris 3 dimana pada baris tersebut dokumen ke-i akan diubah menjadi bentuk lowercase dengan menggunakan fungsi toLowercase().

Tabel 5.2 Implementasi case folding

```
1     public static void caseFolding(ArrayList<String>
      list) {
2         for (int i = 0; i < list.size(); i++) {
3             list.set(i, list.get(i).toLowerCase());
4         }
5     }
```

Kembali pada Tabel 5.1 terdapat pemanggilan method tokenisasi() yang terjadi pada baris 6. Proses yang terjadi method tokenisasi sesuai dengan Gambar 4.4 pada bab perancangan, dimana alur proses tokenisasi tersebut telah diimplementasikan kedalam bentuk kode-kode program yang ditunjukkan pada Tabel 5.3. Pada tabel tersebut terdapat proses penghapusan angka dan tanda baca yang terjadi pada baris 5 yaitu pemanggilan hapusAngkadanTandaBaca() dimana hasil dari pemanggilan tersebut disimpan kedalam variabel baru berdana hasilKalimat bertipe String. Sedangkan proses tokenisasi sendiri terjadi pada baris 6 yaitu penggunakan class StringTokenizer untuk memecah nilai yang tersimpan pada variabel hasilKalimat menjadi token-token. Token-token tersebut disimpan kembali kedalam bentuk array list dengan nama listKata yang terjadi pada baris 7-10.

Tabel 5.3 Implementasi tokenisasi

```
1   public static ArrayList<String> tokenisasi(String kalimat) {
2       ArrayList<String> listKata = new ArrayList<>();
3       StringTokenizer token;
4       String words;
5       String hasilKalimat = hapusAngkadanTandaBaca(kalimat);
6       token = new StringTokenizer(hasilKalimat);
7       while (token.hasMoreTokens()) {
8           words = token.nextToken();
9           listKata.add(words);
10      }
11      return listKata;
12  }
```

Pada Tabel 5.3 terdapat pemanggilan method hapusAngkadanTandaBaca() yang terjadi pada baris 5 dimana method tersebut berisi proses dengan alur seperti pada Gambar 4.5 pada bab 4 yaitu perancangan. Alur proses pada gambar tersebut, apabila diimplementasikan kedalam bentuk kode-kode program, maka didapatkan hasil implementasi seperti pada Tabel 5.4. Pada tabel tersebut inti proses penghapusan angka dan tanda baca terletak pada baris 2. Pada baris tersebut setiap karakter yang *matches* (sesuai) dengan *punctuation* (tanda baca) maupun *digit* (angka) akan diubah menjadi karakter spasi. Proses pengubahan tersebut menggunakan fungsi replaceAll() sesuai kode program hasil implementasi pada Tabel 5.4 baris 2.

Tabel 5.4 Implementasi hapus angka dan tanda baca

```
1   private static String hapusAngkadanTandaBaca(String kalimat) {
2       kalimat = kalimat.replaceAll("\\p{Punct}|\\d", " ");
3       return kalimat;
4   }
```

Kembali pada implementasi proses pre-processing yang terdapat pada Tabel 5.1 dimana pada tabel tersebut terdapat pemanggilan method removeStopword() yang terdapat pada baris 9. Berdasarkan pada bab 4 perancangan, alur proses *remove stopword* yang digambarkan pada Gambar 4.6 apabila diimplementasikan kedalam bentuk kode program maka hasil implementasi yang dihasilkan seperti pada Tabel 5.5. Pada tabel tersebut, proses

removeStopword() diawali dengan meload daftar stopword yang ditunjukkan pada baris 2 yaitu dengan memanggil method readStopwordList(). Kemudian dilakukan pengecekan setiap token pada setiap dokumen, apakah token tersebut terdapat pada daftar *stopword*, apabila iya maka token akan dihapus sengan menggunakan fungsi remove, sedangkan apabila tidak maka akan berganti pada token selanjutnya untuk dilakukan pengecekan. Proses pengecekan yang terjadi pada implementasi *stopword removal* ditunjukkan pada baris 5. Sedangkan penghapusan token yang merupakan stopword ditunjukkan pada baris 6.

Tabel 5.5 Implementasi remove stopword

```
1   public static void removeStopword(ArrayList<ArrayList<String>> token) {
2       readStopwordList();
3       for (int i = 0; i < token.size(); i++) {
4           for (int j = 0; j < token.get(i).size(); j++) {
5               if (cekStopword(token.get(i).get(j))) {
6                   token.get(i).remove(j);
7                   j = j - 1;
8               }
9           }
10      }
11  }
```

Proses load daftar *stopword* dimana pada Tabel 5.5 ditunjukkan pada baris 2 yaitu dengan pemanggilan method readStopwordList(). Proses yang terdapat pada method tersebut ditunjukkan pada Tabel 5.6. Pada tabel tersebut file yang digunakan berekstensi .txt yang tersimpan oada direktori `E:\\SKRIPSWEET` sesuai pada baris 4. Kemudian setiap baris pada file tersebut dibaca dengan menggunakan fungsi readline(), dan setiap baris akan disimpan pada array lis listStopword sesuai pada baris 9-11.

Tabel 5.6 Implementasi load daftar stopword

```
1   private static ArrayList<String> listStopword;
2   private static void readStopwordList() {
3           listStopword = new ArrayList<String>();
4           File file = new File("E:\\SKRIPSWEET\\stopword.txt");
5           BufferedReader reader = null;
6           try {
7               reader = new BufferedReader(new FileReader(file));
8               String text = null;
9               while ((text = reader.readLine()) != null) {
10                  listStopword.add(text);
11              }
12          } catch (FileNotFoundException e) {
13              e.printStackTrace();
14          } catch (IOException e) {
15              e.printStackTrace();
16          } finally {
17              try {
18                  if (reader != null) {
19                      reader.close();
20                  }
21              } catch (IOException e) {
22              }
23          }
```

```
    }
```

Kembali pada implementasi pre-processing pada Tabel 5.1, dimana pada baris 12 terdapat pemanggilan method stemming(). Berdasarkan pada bab 4 perancangan alur proses stemming yang telah digambarkan pada Gambar 4.7, dimana apabila alur proses tersebut diimplementasikan kedalam bentuk kode-kode program maka didapatkan hasil implementasi program seperti pada Tabel 5.7. Pada tabel tersebut diawali dengan mengecek apakah kata terdapat pada kamus, jika iya maka kata akan menjadi output dan proses dihentikan. Hal ini ditunjukkan pada baris 5-7 dari hasil implementasi proses stemming. Apabila kata tidak terdapat pada kamus maka akan dilkakukan penghapusan *inflection suffix* dengan memanggil method hapusInflectionSuffix() yang ditunjukkan pada baris 8, kemudian dilakukan penghapusan *derivation suffix* dengan memanggil method hapusDerivationSuffix() yang ditunjukkan pada baris 9, dan yang terakhir adalah memanggil method hapusDerivationPreffix() untuk menghapus derivation preffix yang ditunjukkan pada baris 10.

Tabel 5.7 Implementasi stemming

```
1  public static String stemming(String kata) {
2          String tempKata;
3          String cek = "";
4          ArrayList<String> awal = new ArrayList<>();
5          if (cekKamus(kata)) {
6              return kata;
7          } else {
8              kata = hapusInfectionSuffix(kata);
9              kata = hapusDerivationSuffix(kata);
10             kata = hapusDerivationPrefix(kata);
11         }
12         return kata;
13     }
```

Pada baris 5 dalam hasil implementasi stemming pada Tabel 5.7 terdapat pemanggilan method cekKamus(). Method ini pada bab 4 perancangan memiliki alur proses yang digambarkan pada Gambar 4.8. Alur proses tersebut apabila diimplementasikan kedalam bentuk kode program maka akan dihasilkan kode program seperti pada Tabel 5.8. Pada tabel tersebut apabila pada listKamus terdapat kata yang sedang diidentifikasi maka akan mengembalikan nilai *true*, dan apabila sebaliknya maka akan mengembalikan nilai *false*. Hal ini ditunjukkan pada baris 2-6 dari implementasi cek kamus Tabel 5.8.

Tabel 5.8 Implementasi cek kamus

```
1      private static boolean cekKamus(String kata) {
2          if (listKamus.contains(kata)) {
3              return true;
4          } else {
5              return false;
6          }
7      }
```

Kembali pada Tabel 5.7 dimana pada baris 8 terdapat pemanggilan method hapusInflectionSuffix(). Method tersebut berisi proses yang telah digambarkan alur diagramnya pada bab 4 perancangan yaitu pada Gambar 4.9. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka

didapatkan hasil kode program seperti pada Tabel 5.9. Pada tabel tersbut penghapusan akhiran –lah, -kah, -pun, -nya ditunjukkan pada baris 5-18, dimana apabila akhiran tersebut telah dihapus maka akan dilakukan penghapusan akhiran –ku, -mu, dan –nya apabila akhiran sebelumnya adalah –lah, -kah, -pun. Hal ini ditunjukkan pada baris 7-16. Apabila akhiran bukan –lah, -kah, -pun, dan –nya maka akan dihapus akhiran –ku dan –mu. Hal ini ditunjukkan pada bari 18-21.

Tabel 5.9 Implementasi hapus inflection suffix

```
1     private static String hapusInfectionSuffix(String kata) {
2         String kataAsal = kata;
3         String tempKata;
4         String tempKata2;
5         if (kata.endsWith("lah") || kata.endsWith("kah") ||
      kata.endsWith("pun") || kata.endsWith("nya")) {
6             tempKata = kata.substring(0, kata.length() - 3);
7             if (kata.endsWith("lah") || kata.endsWith("kah")
      || kata.endsWith("tah") || kata.endsWith("pun")) {
8                 if (tempKata.endsWith("nya")) {
9                     tempKata2 = tempKata.substring(0,
      tempKata.length() - 3);
10                    return tempKata2;
12                } else if (tempKata.endsWith("ku") ||
      tempKata.endsWith("mu")) {
13                    tempKata2 = tempKata.substring(0,
      tempKata.length() - 2);
14                    return tempKata2;
15                }
16            }
17            return tempKata;
18        } else if (kata.endsWith("ku") ||
      kata.endsWith("mu")) {
19            tempKata = kata.substring(0, kata.length() - 2);
20            return tempKata;
21        }
22        return kataAsal;
23    }
```

Kembali pada Tabel 5.7, dimana pada baris 9 terdapat pemanggilan method hapusDerivationSuffix(). Method ini memiliki proses dengan alur seperti pada Gambar 4.10 pada bab 4 perancangan. Alur proses tersebut apabila diimplementasikan kedalam kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.10. Pada tabel tersebut terdapat penghapusan akhiran –i yang terdapat pada baris 5-10. Penghapusan akhiran –an juga terjadi pada method ini yang ditunjukkan pada baris 10-14, dan penghapusan akhiran -kan pada baris 14-20. Apabila akhiran kata adalah –kan makan akan dilakukan proses pengecekan larangan pasangan awalan dan akhiran yaitu dengan memanggil method cekDisallowedPreffixSuffix() yang ditunjukkan pada baris 21. Method tersebut mengembalikan nilai *true* atau *false*.

Tabel 5.10 Implementasi hapus derivation prefix

```
1     private static String hapusDerivationSuffix(String kata)
      {
2         String kataAsal = kata;
```

```
3           String tempKata;
4           String tempKata2;
5           if (kata.endsWith("i")) {
6               tempKata = kata.substring(0, kata.length() - 1);
7               if (cekKamus(tempKata)) {
8                   return tempKata;
9               }
10          } else if (kata.endsWith("an")) {
11              tempKata = kata.substring(0, kata.length() - 2);
12              if (cekKamus(tempKata)) {
13                  return tempKata;
14              } else {
15                  if (tempKata.endsWith("k")) {
16                      tempKata2     =     tempKata.substring(0,
    tempKata.length() - 1);
17                      if (cekKamus(tempKata2)) {
18                          return tempKata2;
19                      }
20                  }
21                  if (cekDisallowedPreffixSuffix(kata)) {
22                      return kataAsal;
23                  }
24              }
25          }
26          return kataAsal;
27      }
```

Berdasarkan pada Tabel 5.10 baris 21 yang melakukan pemanggilan method cekDisallowedPreffixSuffix(), dimana method tersbut memiliki alur proses seperti pada Gambar 4.11 pada Bab 4 Perancangan. Berdasarkan gambar tersebut, alur proses yang terjadi pada method tersebut apabila diimplementasikan kedalam bentuk kode program maka, hasil implementasi yang dihasilkan adalah seperti pada Tabel 5.11. Pada tabel tersebut kondisi pasangan awalan dan akhiran be- dan –i ditun jukkan pada baris 3-5, pasangan di- dan –an ditunjukkan pada baris 5-7, pasangan ke- dan –i atau –kan dintunjukkan pada baris 7-9, pasangan me- dan –an ditunjukkan pada baris 9-11, dan yang terakhir adalah pasangan se- dan –i atau –kan yang ditunjukkan pada baris 11-13. Apabila salah satu kondisi tersebut terpenuhi maka akan mengembalikan nilai *true*, dan apabila sebaliknya maka akan mengembalikan nilai *false*.

Tabel 5.11 Implementasi Disallowed Pasangan Preffix dan Suffix

```
1       private static boolean cekDisallowedPreffixSuffix(String
    kata) {
2           boolean hasilCek = false;
3           if (kata.startsWith("be") && kata.endsWith("i")) {
4               hasilCek = true;
5           }     else     if     (kata.startsWith("di")     &&
    kata.endsWith("an")) {
6               hasilCek = true;
7           }     else     if     (kata.startsWith("ke")     &&
    (kata.endsWith("i") || kata.endsWith("kan"))) {
8               hasilCek = true;
9           }     else     if     (kata.startsWith("me")     &&
    kata.endsWith("an")) {
10              hasilCek = true;
11          }     else     if     (kata.startsWith("se")     &&
```

```
    (kata.endsWith("i") || kata.endsWith("kan"))) {
12              hasilCek = true;
13          }
14          return hasilCek;
15      }
```

Kembali pada Tabel 5.7 dimana pada baris 10 terdapat pemanggilan method haousDerivationPrefix(). Berdasarkan pada Bab 4 Perancangan, method tersebut memiliki alur proses seperti pada Gambar 4.12. Alur proses tersebut apabila diimplementasikan kedalam bentuk kode program maka hasil implementasinya adalah seperti pada Tabel 5.12. Pada Tabel tersebut, pada baris 5-32 merupakan penghapusan awalan di-, ke-, atau se-, dimana pada baris 14-30 adalah penghapusan awalan diper-. Penghapusan awalan te-, me-, be-, dan pe- terdapat pada baris 33-327 dimana pada baris 35-96 itu sendiri adalah penghapusan awalan ter-. Pada baris 97-176 adalah penghapusan untuk awalan me-, baik meng-, meny-, mem-, meupun men-. Pada baris 177-215 adalah penghapusan awalan be- yaitu ber-. Pada baris 216-304 merupakan penghapusan awalan pe- baik peng-, peny-, pem-, pen-, maupun per-. Sedangkan pada baris 306-323 merupakan penghapusan awalan memper-.

Tabel 5.12 Implementasi hapus derivation prefix

```
1       private static String hapusDerivationPrefix(String kata)
{
2           String kataAsal = kata;
3           String tempKata;
4           String tempKata2;
5           if (kata.startsWith("di") || kata.startsWith("ke") ||
kata.startsWith("se")) {
6               tempKata = kata.substring(2);
7               if (cekKamus(tempKata)) {
8                   return tempKata;
9               }
10              tempKata2 = hapusDerivationSuffix(tempKata);
11              if (cekKamus(tempKata2)) {
12                  return tempKata2;
13              }
14              if (kata.startsWith("diper")) {
15                  tempKata = kata.substring(5);
16                  if (cekKamus(tempKata)) {
17                      return tempKata;
18                  }
19                  tempKata2 = hapusDerivationSuffix(tempKata);
20                  if (cekKamus(tempKata2)) {
21                      return tempKata2;
22                  }
23                  tempKata = "r" + kata.substring(5);
24                  if (cekKamus(tempKata)) {
25                      return tempKata;
26                  }
27                  tempKata2 = hapusDerivationSuffix(tempKata);
28                  if (cekKamus(tempKata2)) {
29                      return tempKata2;
30                  }
31              }
32          }
33          if (kata.startsWith("te") || kata.startsWith("me") ||
```

```
    kata.startsWith("be") || kata.startsWith("pe")) {
34              if (kata.length() > 3) {
35                  if (kata.startsWith("te")) {
36                      if     (kata.charAt(2)     ==     'r'     &&
    kata.charAt(3) == 'r') {
37                          return kataAsal;
38                      }
39                      if     ((kata.charAt(2)     ==     'r')     &&
    vowel(kata.charAt(3))) {
40                          tempKata = kata.substring(3);
41                          if (cekKamus(tempKata)) {
42                              return tempKata;
43                          }
44                          tempKata2                                 =
    hapusDerivationSuffix(tempKata);
45                          if (cekKamus(tempKata2)) {
46                              return tempKata2;
47                          }
48                      }
49                  }
50                  if (kata.length() > 6) {
51                      if     ((kata.charAt(2)     ==     'r')     &&
    !(vowel(kata.charAt(3))    ||    kata.charAt(3)    ==    'r')    &&
    (kata.subSequence(4,              6).equals("er"))              &&
    vowel(kata.charAt(6))) {
52                          tempKata = kata.substring(3);
53                          if (cekKamus(tempKata)) {
54                              return tempKata;
55                          }
56                          tempKata2                                 =
    hapusDerivationSuffix(tempKata);
57                          if (cekKamus(tempKata2)) {
                                return tempKata2;
58                          }
59                      }
60                      if     ((kata.charAt(2)     ==     'r')     &&
    !(vowel(kata.charAt(3))    ||    kata.charAt(3)    ==    'r')    &&
    (kata.subSequence(4,              6).equals("er"))              &&
    !(vowel(kata.charAt(6)))) {
61                          tempKata = kata.substring(3);
62                          if (cekKamus(tempKata)) {
63                              return tempKata;
64                          }
65                          tempKata2                                 =
    hapusDerivationSuffix(tempKata);
66                          if (cekKamus(tempKata2)) {
67                              return tempKata2;
68                          }
69                      }
70                      if     ((kata.charAt(2)     ==     'r')     &&
    !(vowel(kata.charAt(3))    ||    kata.charAt(3)    ==    'r')    &&
    !(kata.subSequence(4, 6).equals("er"))) {
71                          tempKata = kata.substring(3);
72                          if (cekKamus(tempKata)) {
73                              return tempKata;
74                          }
75                          tempKata2                                 =
    hapusDerivationSuffix(tempKata);
76                          if (cekKamus(tempKata2)) {
```

```
77                              return tempKata2;
78                          }
79                      }
80                  }
81                  if (kata.length() > 5) {
82                      if    (!((kata.charAt(2)    ==    'r')    ||
    vowel(kata.charAt(2)))         &&         (kata.subSequence(3,
    5).equals("er")) && vowel(kata.charAt(5))) {
83                          return kataAsal;
84                      }
85                      if    (!((kata.charAt(2)    ==    'r')    ||
    vowel(kata.charAt(2)))         &&         (kata.subSequence(3,
    5).equals("er")) && !(vowel(kata.charAt(5)))) {
86                          tempKata = kata.substring(2);
87                          if (cekKamus(tempKata)) {
88                              return tempKata;
89                          }
90                          tempKata2                                =
    hapusDerivationSuffix(tempKata);
91                          if (cekKamus(tempKata2)) {
92                              return tempKata2;
93                          }
94                      }
95                  }
96              }
97              if (kata.startsWith("me")) {
98                  if (kata.length() > 4) {
99                      if ((kata.subSequence(2, 4).equals("ng"))
    &&   (vowel(kata.charAt(4))   ||   kata.charAt(4)   ==   'k'   ||
    kata.charAt(4)   ==   'g'   ||   kata.charAt(4)   ==   'h'   ||
    kata.charAt(4) == 'q')) {
100                         tempKata = kata.substring(4);
101                         if (cekKamus(tempKata)) {
102                             return tempKata;
103                         }
104                         tempKata2                                =
    hapusDerivationSuffix(tempKata);
105                         if (cekKamus(tempKata2)) {
106                             return tempKata2;
107                         }
108                         tempKata = "k" + kata.substring(4);
109                         if (cekKamus(tempKata)) {
110                             return tempKata;
111                         }
112                         tempKata2                                =
    hapusDerivationSuffix(tempKata);
113                         if (cekKamus(tempKata2)) {
114                             return tempKata2;
115                         }
116                     }
117                     if ((kata.subSequence(2, 4).equals("ny"))
    && vowel(kata.charAt(4))) {
118                         tempKata = "s" + kata.substring(4);
119                         if (cekKamus(tempKata)) {
120                             return tempKata;
121                         }
122                         tempKata2                                =
    hapusDerivationSuffix(tempKata);
123                         if (cekKamus(tempKata2)) {
```

```
124                             return tempKata2;
125                         }
126                     }
127                 }
128                 if (kata.length() > 3) {
129                     if    ((kata.charAt(2)    ==    'm')    &&
(kata.charAt(3)   ==   'b'   ||   kata.charAt(3)   ==   'f'   ||
kata.charAt(3) == 'p' || kata.charAt(3) == 'v')) {
130                         tempKata = kata.substring(3);
131                         if (cekKamus(tempKata)) {
132                             return tempKata;
133                         }
134                         tempKata2                                =
hapusDerivationSuffix(tempKata);
135                         if (cekKamus(tempKata2)) {
136                             return tempKata2;
137                         }
138                         tempKata = "p" + kata.substring(3);
139                         if (cekKamus(tempKata)) {
140                             return tempKata;
141                         }
142                         tempKata2                                =
hapusDerivationSuffix(tempKata);
143                         if (cekKamus(tempKata2)) {
144                             return tempKata2;
145                         }
146                     }
147                     if    ((kata.charAt(2)    ==    'n')    &&
(kata.charAt(3)   ==   'c'   ||   kata.charAt(3)   ==   'd'   ||
kata.charAt(3)   ==   'j'   ||   kata.charAt(3)   ==   's'   ||
kata.charAt(3) == 'z')) {
148                         tempKata = kata.substring(3);
149                         if (cekKamus(tempKata)) {
150                             return tempKata;
151                         }
152                         tempKata2                                =
hapusDerivationSuffix(tempKata);
153                         if (cekKamus(tempKata2)) {
154                             return tempKata2;
155                         }
156                     }
157                     tempKata = kata.substring(2);
158                     if (cekKamus(tempKata)) {
159                         return tempKata;
160                     }
161                     tempKata2                                    =
hapusDerivationSuffix(tempKata);
162                     if (cekKamus(tempKata2)) {
163                         return tempKata2;
164                     }
165                     if (kata.charAt(2) == 'n') {
166                         tempKata = "t" + kata.substring(3);
167                         if (cekKamus(tempKata)) {
168                             return tempKata;
169                         }
170                         tempKata2                                =
hapusDerivationSuffix(tempKata);
171                         if (cekKamus(tempKata2)) {
172                             return tempKata2;
```

```
173                         }
174                     }
175                 }
176             }
177             if (kata.startsWith("be")) {
178                 if (kata.length() > 3) {
179                     if    (kata.charAt(2)    ==    'r'    &&
    vowel(kata.charAt(3))) {
180                         tempKata = kata.substring(3);
181                         if (cekKamus(tempKata)) {
182                             return tempKata;
183                         }
184                         tempKata = "r" + kata.substring(3);
185                         if (cekKamus(tempKata)) {
186                             return tempKata;
187                         }
188                         tempKata2                              =
    hapusDerivationSuffix(tempKata);
189                         if (cekKamus(tempKata2)) {
190                             return tempKata2;
191                         }
192                     }
193                     if    (kata.charAt(2)    ==    'r'    &&
194 !(vowel(kata.charAt(3)))) {
195                         tempKata = kata.substring(3);
196                         if (cekKamus(tempKata)) {
197                             return tempKata;
198                         }
199                         tempKata2                              =
    hapusDerivationSuffix(tempKata);
200                         if (cekKamus(tempKata2)) {
201                             return tempKata2;
202                         }
203                     }
204                     if (kata.charAt(2) == 'k') {
205                         tempKata = kata.substring(2);
206                         if (cekKamus(tempKata)) {
207                             return tempKata;
208                         }
209                         tempKata2                              =
    hapusDerivationSuffix(tempKata);
210                         if (cekKamus(tempKata2)) {
211                             return tempKata2;
212                         }
213                     }
214                 }
215             }
216             if (kata.startsWith("pe")) {
217                 if (kata.length() > 4) {
218                     if  (kata.subSequence(2,  4).equals("ng")
    &&  (vowel(kata.charAt(4))  ||  kata.charAt(4)  ==  'k'  ||
    kata.charAt(4)  ==  'g'  ||  kata.charAt(4)  ==  'h'  ||
    kata.charAt(4) == 'q')) {
219                         tempKata = kata.substring(4);
220                         if (cekKamus(tempKata)) {
221                             return tempKata;
222                         }
223                         tempKata2                              =
    hapusDerivationSuffix(tempKata);
```

```
224                         if (cekKamus(tempKata2)) {
225                             return tempKata2;
226                         }
227                         tempKata = "k" + kata.substring(4);
228                         if (cekKamus(tempKata)) {
229                             return tempKata;
230                         }
231                         tempKata2                                   =
    hapusDerivationSuffix(tempKata);
232                         if (cekKamus(tempKata2)) {
233                             return tempKata2;
234                         }
235                     }
236                     if ((kata.subSequence(2, 4).equals("ny"))
    && vowel(kata.charAt(4))) {
237                         tempKata = "s" + kata.substring(4);
238                         if (cekKamus(tempKata)) {
239                             return tempKata;
240                         }
241                         tempKata2                                   =
    hapusDerivationSuffix(tempKata);
242                         if (cekKamus(tempKata2)) {
243                             return tempKata2;
244                         }
245                     }
246                 }
247                 if (kata.length() > 3) {
248                     if ((kata.subSequence(2, 3).equals("m"))
    && (kata.charAt(3) == 'b' || kata.charAt(3) == 'f' ||
    kata.charAt(3) == 'p' || kata.charAt(3) == 'v')) {
249                         tempKata = kata.substring(3);
250                         if (cekKamus(tempKata)) {
251                             return tempKata;
252                         }
253                         tempKata2                                   =
    hapusDerivationSuffix(tempKata);
254                         if (cekKamus(tempKata2)) {
255                             return tempKata2;
256                         }
257                         tempKata = "p" + kata.substring(3);
258                         if (cekKamus(tempKata)) {
259                             return tempKata;
260                         }
261                         tempKata2                                   =
262 hapusDerivationSuffix(tempKata);
263                         if (cekKamus(tempKata2)) {
264                             return tempKata2;
265                         }
266                     }
267                     if ((kata.subSequence(2, 3).equals("n"))
    && (kata.charAt(3) == 'c' || kata.charAt(3) == 'd' ||
    kata.charAt(3) == 'j' || kata.charAt(3) == 's' ||
    kata.charAt(3) == 'z')) {
268                         tempKata = kata.substring(3);
269                         if (cekKamus(tempKata)) {
270                             return tempKata;
271                         }
272                         tempKata2                                   =
    hapusDerivationSuffix(tempKata);
```

```
273                         if (cekKamus(tempKata2)) {
274                             return tempKata2;
275                         }
276                     }
277                     if (kata.subSequence(2, 3).equals("r")) {
278                         tempKata = kata.substring(3);
279                         if (cekKamus(tempKata)) {
280                             return tempKata;
281                         }
282                         tempKata2                                   =
    hapusDerivationSuffix(tempKata);
283                         if (cekKamus(tempKata2)) {
284                             return tempKata2;
285                         }
286                         tempKata = "r" + kata.substring(3);
287                         if (cekKamus(tempKata)) {
288                             return tempKata;
289                         }
290                         tempKata2                                   =
    hapusDerivationSuffix(tempKata);
291                         if (cekKamus(tempKata2)) {
292                             return tempKata2;
293                         }
294                     }
295                     tempKata = kata.substring(2);
296                     if (cekKamus(tempKata)) {
297                         return tempKata;
298                     }
299                     tempKata2                                       =
    hapusDerivationSuffix(tempKata);
300                     if (cekKamus(tempKata2)) {
301                         return tempKata2;
302                     }
303                 }
304             }
305             if (kata.length() > 6) {
306                 if (kata.startsWith("memper")) {
307                     tempKata = kata.substring(6);
308                     if (cekKamus(tempKata)) {
309                         return tempKata;
310                     }
311                     tempKata2                                       =
    hapusDerivationSuffix(tempKata);
312                     if (cekKamus(tempKata2)) {
313                         return tempKata2;
314                     }
315                     tempKata = "r" + kata.substring(6);
316                     if (cekKamus(tempKata)) {
317                         return tempKata;
318                     }
319                     tempKata2                                       =
    hapusDerivationSuffix(tempKata);
320                     if (cekKamus(tempKata2)) {
321                         return tempKata2;
322                     }
323                 }
324             } else {
325                 return kataAsal;
326             }
```

```
327             }
328             return kataAsal;
329         }
```

Pada Tabel 5.12 terdapat beberapa baris proses yang melakukan pemanggilan method vowel() untuk melakukan pengecekan apakah karakter tersebut merupakan huruf vokal atau konsonan. Berdasarkan pada Bab 4 Perancangan, alur proses dalam pengecekan huruf vokal yang telah ditunjukkan pada Gambar 4.12 dimana apabila alur proses tersebut diimplementasikan kedalam kode program maka didapatkan hasil implementasi seperti yang ditunjukkan pada Tabel 5.13. Pada tabel tersebut pengecekan huruf *vowel* (vokal) ditunjukkan pada baris 3-5 dimana apabila huruf yang menjadi argumen dari method ini merupakan huruf 'a', 'i', 'u', 'e', atau 'o' maka hasil pengecekan bernilai *true*.

Tabel 5.13 Implementasi pengecekan vowel

```
1     private static boolean vowel(char huruf) {
2         boolean cekVowel = false;
3         if (huruf == 'a' || huruf == 'i' || huruf == 'u' ||
  huruf == 'e' || huruf == 'o') {
4             cekVowel = true;
5         }
6         return cekVowel;
7     }
```

5.2 Information Gain

Berdasarkan pada Bab 4 Perancangan tepatnya pada Gambar 4.14 telah digambarkan alur proses dari *information gain*. Apabila laur tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabe 5.14. Pada tabel tersebut langkah pertama yang harus dilakukan dalam proses *information gain* adalah menghitung frekuensi dari token setiap dokumen dari data latih dengan memanggil method frekuensi(). Hal ini ditunjukkan pada baris 14-16. Dari seluruh token yang telah dihitung frekuensinya kemudian diambil fitur dari keseluruhan dokumen dari data latih dengan memanggil method getSemuaFitur(). Proses ini terdapat pada baris 17. Kemudian dilakukan perhitungan frekuensi kelas dari data latih yaitu sama dengan menghitung frekuensi token dari data latih yaitu dengan memanggil method frekuensi yang ditunjukkan pada baris 18. Baris 19 merupakan proses untuk menghitung jumlah fitur yang akan diambil sebagai hasil seleksi information gain. Proses utama dari information gain terjadi pada baris 20-35, dimana pada baris 21 adalah proses menghitung *entropy global* yang dilakukan dengan memanggil method hitungEntropyGlobal(). Pada baris 23 adalah menghitung jumlah dokumen yang mengandung fitur pada setiap kelas yaitu dengan memanggil method hitungAdanyaFiturPdDokumenTiapKelas(). Sedangkan perhitungan jumlah dokumen yang tidak mengandung fitur pada setiap kelas ditunjukkan pada baris 25 yaitu dengan memanggil method hitungTdkAdanyaFiturPdDokumenTiapKelas(). Proses menghitung total dokumen yang mengandung fitur pada keseluruh dokumen dilakukan dengan memanggil method hitungTotalAda() yang terdapat pada baris 26. Sedangkan proses

menghitung total dokumen yang tidak mengandung fitur pada keseluruhan kelas dilakukan dengan memanggil method totalTidakAda() yang terjadi pada baris 27. Setelah itu, dilakukan proses pada baris 28-31 yaitu menghitung entropy kemunculan fitur baik ada maupun tidak ada yaitu dengan memanggil method hiturngEntropyMunculnyaFitur(). Kemudian dihitung nilai *information gain* dari setiap fitur dengan menjalankan proses pada baris 33 yaitu memanggil method hitung *information gain*. Seluruh fitur akan diurutkan berdasarkan *information gain*-nya yang dilakukan pada baris 35. Kemudian diambil sejumlah persentase fitur tertinggi yang akan menjadi keluaran dari proses ini.

Tabel 5.14 Implementasi information gain

```
1     private static ArrayList<String> dataLatih;
2     private static ArrayList<String> kelasDtLatih = new
ArrayList<>();
3     private static ArrayList<HashMap<String, Integer>>
frekuensiDtLatih = new ArrayList<>();
4     private static HashMap<String, Integer>
frekuensiKelasDtLatih = new HashMap<>();
5     private static ArrayList<String> semuaFitur = new
ArrayList<>();
6     private static double persentaseFiturGain = 5;
7     public static ArrayList<String>
informationGain(ArrayList<ArrayList<String>> kata) {
8         ArrayList<HashMap<String, Integer>> ada, tdkAda;
9         ArrayList<Integer> totalAda, totalTdkAda;
10        ArrayList<Double> entropyAda, entropyTdkAda;
11        HashMap<String, Double> infoGain;
12        ArrayList<String> fiturGain;
13        double entropyGlobal;
14        for (int i = 0; i < kata.size(); i++) {
15            frekuensiDtLatih.add(frekuensi(kata.get(i)));
16        }
17        getSemuaFitur(frekuensiDtLatih);
18        frekuensiKelasDtLatih = frekuensi(kelasDtLatih);
19        int jumlahFitur = (int)
Math.ceil((persentaseFiturGain / 100) * semuaFitur.size());
20        //Menghitung Entropy Global
21        entropyGlobal =
hitungEntropyGlobal(frekuensiKelasDtLatih, dataLatih);
22        //Menghitung Ada dan Tidak Ada per Kategori
23        ada =
hitungAdanyaFiturPdDokumenTiapKelas(frekuensiDtLatih,
semuaFitur, kelasDtLatih);
24        tdkAda = hitungTdkAdanyaFiturPdDokumenTiapKelas(ada,
frekuensiKelasDtLatih);
25        //Menghitung Total Ada dan Tidak Ada
26        totalAda = hitungTotalAda(semuaFitur, ada);
27        totalTdkAda = hitungTotalTdkAda(semuaFitur, totalAda,
dataLatih);
28        //Menghitung Entropy Ada
29        entropyAda = hitungEntropyMunculnyaFitur(totalAda,
frekuensiKelasDtLatih, ada, semuaFitur);
30        //Menghitung Entropy Tidak Ada
31        entropyTdkAda =
hitungEntropyMunculnyaFitur(totalTdkAda,
frekuensiKelasDtLatih, tdkAda, semuaFitur);
```

```
32          //menghitung information gain
33          infoGain    =    hitungInformationGain(entropyGlobal,
    totalAda, totalTdkAda, entropyAda, entropyTdkAda, dataLatih,
    semuaFitur);
34          //selected fitur
35          fiturGain      =      sortingInformationGain(infoGain,
    jumlahFitur);
36          return fiturGain;
37      }
```

Berdasarkan pada penjelasan dari Tabel 5.14, ditunjukkan bahwa terdapat pemanggilan method frekuensi yang terjadi pada baris ke 15 dan 18. Method tersebut memiliki alur proses seperti pada Gambar 4.15 yang terdapat pada Bab 4 Perancangan. Apabila akur tersebut diimplementasikan kedalam kode program maka akan didapatkan hasil program seperti pada Tabel 5.15. Pada tabel tersebut proses perhitungan frekuensi terdapat pada baris 3-11, dimana apabila pada data terdapat kata lebih dari satu maka frekuensi katanya adalah jumlah kata ditambah dengan 1, apabila tidak maka sama dengan 1.

Tabel 5.15 Implementasi perhitungan frekuensi

```
1   public static HashMap frekuensi(ArrayList<String> data) {
2           HashMap<String, Integer> map = new HashMap<String,
    Integer>();
3           for (int i = 0; i < data.size(); i++) {
4               String kata = data.get(i);
5               Integer jumlah = map.get(kata);
6               if (map.containsKey(kata)) {
7                   map.put(kata, jumlah + 1);
8               } else {
9                   map.put(kata, 1);
10              }
11          }
12          return map;
13      }
```

Pada Tabel 5.14, terdapat pemanggilan method getSemuaFitur yang terdapat pada baris 17. Method tersebut memiliki alur proses seperti pada Gambar 4.16, dimana apabila diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.16. Pada Tabel tersebut pengambilan fitur terjadi pada baris 2-10, dimana pada baris 4 terdapat kondisi apabila pada array list semua fitur belum terdapat kata yang menjadi key, maka akan ditambahkan kedalam array list.

Tabel 5.16 Implementasi pengambilan fitur

```
1       private   static   ArrayList<String>   semuaFitur   =   new
    ArrayList<>();
2       public                    static                    void
    getSemuaFitur(ArrayList<HashMap<String, Integer>> dtLatih) {
3           for (int i = 0; i < dtLatih.size(); i++) {
4               for (Map.Entry map : dtLatih.get(i).entrySet()) {
5                   if               (semuaFitur.contains((String)
    map.getKey())) {
6                       continue;
7                   } else {
8                       semuaFitur.add((String) map.getKey());
9                   }
```

```
10              }
11          }
12      }
```

Pada baris 21 pada Tabel 5.14, terdapat pemanggilan method hitungEntropyGlobal() yang memiliki alur seperti pada Gambar 4.17. Apabila alur tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.17. Pada tabel tersebut perhitungan entropy ditunjukkan pada baris ke 5, sesuai dengan persamaan 2.8 dimana nilai negatif terdapat pada baris 7 dari Tabel 5.17.

Tabel 5.17 Implementasi hitung entropy global

```
1   public static double hitungEntropyGlobal(HashMap<String, Integer> frekuensiKelasDtLatih, ArrayList<String> dtLatih) {
2       double entropyGlobal = 0;
3       for (Map.Entry map : frekuensiKelasDtLatih.entrySet()) {
4           int temporary = (int) map.getValue();
5           entropyGlobal += (((double) temporary / (double) dtLatih.size()) * (Math.log((double) temporary / (double) dtLatih.size()) / Math.log(2)));
6       }
7       entropyGlobal = -entropyGlobal;
8       return entropyGlobal;
9   }
```

Pada baris 23 pada Tabel 5.14 terdapat pemanggilan method hitungAdanyaFiturPdDokumenTiapKelas() yang memiliki alur proses seperti pada Gambar 4.18. Apabila alur proses tersebut diimplementasikan kedalam kode program maka hasil dari implementasi tersebut seperti pada Tabel 5.18. Pada tabel tersebut proses perhitungan dokumen terdapat pada baris 6-24. Pada baris tersebut terdapat pengecekan apakah pada dokumen terdapat kata fitur, apabila iya maka akn dilakukan penambahan jumlah sesuai pada kategori dari dokumen tersebut.

Tabel 5.18 Implementasi hitung adanya fitur pada dokumen tiap kelas

```
1   public static ArrayList<HashMap<String, Integer>> hitungAdanyaFiturPdDokumenTiapKelas(ArrayList<HashMap<String, Integer>> frekuensiDtLatih, ArrayList<String> semuaFitur, ArrayList<String> kelasLatih) {
2       ArrayList<HashMap<String, Integer>> ada = new ArrayList<>();
3       HashMap<String, Integer> AdaPdDkp = new HashMap<>();
4       HashMap<String, Integer> AdaPdDpuppb = new HashMap<>();
5       HashMap<String, Integer> AdaPdDishub = new HashMap<>();
6       for (int i = 0; i < semuaFitur.size(); i++) {
7           int adaPdDkp = 0;
8           int adaPdDpuppb = 0;
9           int adaPdDishub = 0;
10          for (int j = 0; j < frekuensiDtLatih.size(); j++) {
11              if (frekuensiDtLatih.get(j).containsKey(semuaFitur.get(i))) {
12                  if (kelasLatih.get(j).equalsIgnoreCase("dkp")) {
```

```
13                          adaPdDkp++;
14                      }                   else                    if
(kelasLatih.get(j).equalsIgnoreCase("dpuppb")) {
15                          adaPdDpuppb++;
16                      } else {
17                          adaPdDishub++;
18                      }
19                  }
20              }
21              AdaPdDkp.put(semuaFitur.get(i), adaPdDkp);
22              AdaPdDpuppb.put(semuaFitur.get(i), adaPdDpuppb);
23              AdaPdDishub.put(semuaFitur.get(i), adaPdDishub);
24          }
25          ada.add(AdaPdDkp);
26          ada.add(AdaPdDpuppb);
27          ada.add(AdaPdDishub);
28          return ada;
29      }
```

Pada baris 25 pada Tabel 5.14 terdapat pemanggilan method hitungTdkAdanyaFiturPdDokumenTiapKelas() yang memiliki alur proses seperti pada Gambar 4.19. Apabila alur proses tersebut diimplementasikan kedalam kode program maka hasil dari implementasi tersebut seperti pada Tabel 5.19. Pada tabel tersebut proses perhitungan dokumen terdapat pada baris 6-20. Pada baris tersebut terjadi pengurangan jumlah dokumen setiap kategori dengan jumlah dokumen yang mengandung fitur.

Tabel 5.19 Implementasi hitung tidak adanya fitur pada dokumen tiap kelas

```
1   public static ArrayList<HashMap<String, Integer>>
hitungTdkAdanyaFiturPdDokumenTiapKelas(ArrayList<HashMap<Stri
ng, Integer>> ada, HashMap<String, Integer>
frekuensiKelasDtLatih) {
2       ArrayList<HashMap<String, Integer>> tdkAda = new
ArrayList<>();
3       HashMap<String, Integer> TdkAdaPdDkp = new
HashMap<>();
4       HashMap<String, Integer> TdkAdaPdDpuppb = new
HashMap<>();
5       HashMap<String, Integer> TdkAdaPdDishub = new
HashMap<>();
6       for (int i = 0; i < ada.size(); i++) {
7           for (Map.Entry map : ada.get(i).entrySet()) {
8               switch (i) {
9                   case 0:
10                      TdkAdaPdDkp.put((String)
map.getKey(), (Integer) (frekuensiKelasDtLatih.get("dkp") -
(int) map.getValue()));
11                      break;
12                  case 1:
13                      TdkAdaPdDpuppb.put((String)
map.getKey(), (Integer) (frekuensiKelasDtLatih.get("dpuppb")
- (int) map.getValue()));
14                      break;
15                  default:
16                      TdkAdaPdDishub.put((String)
map.getKey(), (Integer) (frekuensiKelasDtLatih.get("dishub")
- (int) map.getValue()));
17                      break;
```

```
18                  }
19              }
20          }
21          tdkAda.add(TdkAdaPdDkp);
22          tdkAda.add(TdkAdaPdDpuppb);
23          tdkAda.add(TdkAdaPdDishub);
24          return tdkAda;
25      }
```

Pada baris 27 pada Tabel 5.14 terdapat pemanggilan method hitungTotalAda() yang memliki alur proses seperti pada Gambar 4.20. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.20. Pada tabel tersebut perhitungan total ada ditunjukkan pada baris 9 yaitu dengan menambahkan jumlah dokumen yang mengandung fitur pada setiap kelas.

Tabel 5.20 Implementasi hitung total ada

```
1   public static ArrayList<Integer>
    hitungTotalAda(ArrayList<String> semuaFitur,
2   ArrayList<HashMap<String, Integer>> ada) {
3           ArrayList<Integer> totalAda = new ArrayList<>();
4           for (int i = 0; i < semuaFitur.size(); i++) {
5               int jumlahAda = 0;
6               for (int j = 0; j < ada.size(); j++) {
7                   for (Map.Entry map : ada.get(j).entrySet()) {
8                       if (((String)
    map.getKey()).equalsIgnoreCase(semuaFitur.get(i))) {
9                           jumlahAda += (int) map.getValue();
10                          break;
11                      }
12                  }
13              }
14              totalAda.add(jumlahAda);
15          }
16          return totalAda;
17      }
```

Pada Tabel 5.14 pada baris 28 terdapat pemanggilan method hitungTotalTidakAda() yang memiliki alur proses seperti pada Gambar 4.21. Apabila alur tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.21. Pada tabel tersebut perhitungan total dokumen yang tidak mengandung fitur terdapat pada baris 5. Pada baris tersebut terjadi pengurangan jumlah keseluruhan dokumen dikurangi dengan total dokumen yang mengandung fitur.

Tabel 5.21 Impelementasi hitung total ada

```
1   public static ArrayList<Integer>
    hitungTotalTdkAda(ArrayList<String> semuaFitur,
2   ArrayList<Integer> totalAda, ArrayList<String> dataLatih) {
3           ArrayList<Integer> totalTdkAda = new ArrayList<>();
4           for (int i = 0; i < semuaFitur.size(); i++) {
5               totalTdkAda.add(dataLatih.size() -
    totalAda.get(i));
6           }
7           return totalTdkAda;
8       }
```

Pada Tabel 5.14, tepatnya pada baris 30 dan 31 terdapat pemanggilan method hitungEntropyMunculnyaFitur(). Method tersebut memiliki alur proses seperti pada Gambar 4.22. Apabila alur proses tersebut diimplementasikan kedalam kode program maka akan didapatkan kode program seperti pada Tabel 5.22. Pada tabel tersebut perhitungan entropy terdapat pada baris 15.

Tabel 5.22 Implementasi hitung munculnya fitur

```
1   public static ArrayList<Double> hitungEntropyMunculnyaFitur(ArrayList<Integer> totalAdaTdkFitur, HashMap<String, Integer> frekuensiKelasDtLatih, ArrayList<HashMap<String, Integer>> adaTdkFitur, ArrayList<String> semuaFitur) {
2       ArrayList<Double> EntropyFitur = new ArrayList<>();
3       double hasilLogaritma = 0;
4       for (int i = 0; i < totalAdaTdkFitur.size(); i++) {
5           double entropyFitur = 0;
6           for (int j = 0; j < frekuensiKelasDtLatih.size(); j++) {
7               for (Map.Entry map : adaTdkFitur.get(j).entrySet()) {
8                   if (((String) map.getKey()).equalsIgnoreCase(semuaFitur.get(i))) {
9                       int tampung = (int) map.getValue();
10                      if (tampung > 0) {
11                          hasilLogaritma = Math.log((double) tampung / (double) totalAdaTdkFitur.get(i)) / Math.log(2);
12                      } else {
13                          hasilLogaritma = 0;
14                      }
15                      entropyFitur += ((double) tampung / (double) totalAdaTdkFitur.get(i)) * hasilLogaritma;
16                  }
17              }
18          }
19          EntropyFitur.add(entropyFitur);
20      }
21      return EntropyFitur;
22  }
```

Pada Tabel 5.14 tepatnya pada baris 33 terdapat pemanggilan method hitungInformationGain(). Method tersebut memiliki alur proses seperti pada Gambar 4.23. Apabila alur tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada tabel 5.23. Pada tabel tersebut perhitungan *information gain* terdapat pada baris 5, dimana perhitungan tersebut sesuai dengan persamaan 2.9 pada Bab 2 Tinjauan Pustaka.

Tabel 5.23 Implementasi hitung information gain

```
1   public static HashMap<String, Double> hitungInformationGain(double entropyGlobal, ArrayList<Integer> totalAda, ArrayList<Integer> totalTdkAda, ArrayList<Double> entropyAda, ArrayList<Double> entropyTdkAda, ArrayList<String> dataLatih, ArrayList<String> semuaFitur) {
2       double informationGain = 0;
```

```
3         HashMap<String, Double> mapInformationGain = new
HashMap<>();
4         for (int i = 0; i < semuaFitur.size(); i++) {
5             informationGain = entropyGlobal + (((double)
totalAda.get(i) / (double) dataLatih.size()) *
entropyAda.get(i)) + (((double) totalTdkAda.get(i) / (double)
dataLatih.size()) * entropyTdkAda.get(i));
6             mapInformationGain.put(semuaFitur.get(i),
informationGain);
7         }
8         return mapInformationGain;
9     }
```

Pada Tabel 5.14 pada baris 35 terdapat pemanggilan method sortingInformationGain(). Method tersebut memiliki alur proses seperti pada Gambar 4.24. Apabila alur proses tersebut diimplementasikan keladam bentuk kode program maka akan didapatkan kode program seperti pada Tabel 5.24. Pada tabel tersebut proses sorting terdapat pada baris 12. Dimana hasil pengurutan tersebut diambil sejumlah persentase fitur pada baris 17-24.

Tabel 5.24 Implementasi sorting information gain

```
1     public static ArrayList<String>
sortingInformationGain(HashMap<String, Double> infoGain, int
jumlahFitur) {
2         ArrayList<String> fiturGain = new ArrayList<>();
3         Comparator<Entry<String, Double>> compare = new
Comparator<Entry<String, Double>>() {
4             @Override
5             public int compare(Entry<String, Double> o1,
Entry<String, Double> o2) {
6                 int hasilCompare =
o2.getValue().compareTo(o1.getValue());
7                 return hasilCompare;
8             }
9         };
10        Set<Entry<String, Double>> entries =
infoGain.entrySet();
11        List<Entry<String, Double>> listEntries = new
ArrayList<>(entries);
12        Collections.sort(listEntries, compare);
13        HashMap<String, Double> sorted = new
LinkedHashMap<>(listEntries.size());
14        for (Map.Entry map : listEntries) {
15            sorted.put((String) map.getKey(), (Double)
map.getValue());
16        }
17        for (Entry<String, Double> mapping :
sorted.entrySet()) {
18            if (fiturGain.size() != jumlahFitur) {
19                fiturGain.add(mapping.getKey());
20            } else {
21                break;
22            }
23            System.out.println(mapping.getKey() + " => " +
mapping.getValue());
24        }
25        return fiturGain;
26    }
```

5.3 Genetic Algorithm

Berdasarkan pada Bab 4 Perancangan, alur proses algoritme genetika adalah sebagaimana pada Gambar 4.25. Alur proses tersebut apabila diimplementasikan kedalam kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.25. Sesuai penjelasan pada Bab 2 Tinjauan Pustaka, proses algoritme genetika diawali dengan proses inisialisasi. Proses inisialisasi pada algoritme genetika diimplementasikan pada baris 33-37. Setelah itu dilakukan proses *crossover* yang dilakukan pada baris 53-72, dimana proses pertukaran silang tersebut dilakukan dengan memanggil method crossover() pada baris 64. Setelah proses *crossover* selesai dilakukan, selanjutnya dilakukan proses mutasi yang terdapat pada baris 86-93, dimana proses mutasi sendiri pada dasarnya dilakukan dengan memanggil method mutasi() yang terjadi pada baris 88. Setelah itu hasil *crossover* dan mutasi dijadikan dalam satu kelompok dengan *parent* (induk) dengan memanggil method generateIndividu() yang terdapat pada baris 103. Seluruh individu yang telah dikumpulkan dalam satu kelompok dilakukan proses evaluasi dengan mengklasifikasikan setiap data uji berdasarkan fitur yang bernilai 1 pada setiap individu. Klasifikasi data uji dilakukan dengan memanggil method nwknn() yang terdapat pada baris 123. Setelah seluruh data uji telah diklasifikasikan, maka proses evaluasi dilakukan dengan memanggil method evaluasi() yang terjadi pada baris 125. Dari hasil evaluasi kemudian dilakukan proses seleksi dengan memanggil method seleksi yang terjadi pada baris 133.

Tabel 5.25 Implementasi genetic algorithm

```
1     private static ArrayList<ArrayList<String>> kata = new
      ArrayList<>();
2     private static ArrayList<HashMap<String, Integer>>
      frekuensiDtUji = new ArrayList<>();
3     private static ArrayList<String> kelasDtUji = new
      ArrayList<>();
4     private static ArrayList<String> hasilKlasifikasi = new
      ArrayList<>();
5     private static int k = 15;
6     private static int popsize = 30;
7     private static double cr = 0.5;
8     private static double mr = 0.5;
9     private static int maxGenerasi = 100;
10    private static int offspringCr, offspringMr,
      totalIndividu;
11    private static int parent[][];
12    private static int offCr[][];
13    private static int offMr[][];
14    private static int individu[][];
15    private static double fitness[];
16    private static Random rand = new Random();
17    public static void geneticAlgorithm(ArrayList<String>
      fiturGain) {
18          offspringCr = (int) (cr * popsize);
19          offspringMr = (int) (mr * popsize);
20          totalIndividu = popsize + offspringCr + offspringMr;
21          int pjgGen = fiturGain.size();
```

```
22          parent = new int[popsize][pjgGen];
23          offCr = new int[offspringCr][pjgGen];
24          offMr = new int[offspringMr][pjgGen];
25          individu = new int[totalIndividu][pjgGen];
26          fitness = new double[individu.length];
27          ArrayList<String> fiturGA = new ArrayList<>();
28          for (int i = 0; i < kataUji.size(); i++) {
29              frekuensiDtUji.add(frekuensi(kataUji.get(i)));
30          }
31          frekuensiKelasDtUji = frekuensi(kelasDtUji);
32          int generasi = 0;
33          for (int i = 0; i < parent.length; i++) {
34              for (int j = 0; j < parent[0].length; j++) {
35                  parent[i][j] = rand.nextInt(2);
36              }
37          }
38          System.out.println("");
39          System.out.println("INISIALISASI POPULASI AWAL");
40          for (int i = 0; i < parent.length; i++) {
41              System.out.print("P" + i + ":\t");
42              for (int j = 0; j < parent[0].length; j++) {
43                  System.out.print(parent[i][j]);
44              }
45              System.out.println("");
46          }
47          while (generasi < maxGenerasi) {
48              System.out.println("");
49              System.out.println("=====GENERASI KE " + generasi
    + "=====");
50              //crossover
51              int ind[] = new int[2];
52              int jum = 0;
53              while (jum < offspringCr) {
54                  int count = 0;
55                  while (count < 2) { //ngambil indeks parent
56                      ind[count] = rand.nextInt(popsize);
57                      if (ind[0] != ind[1]) {
58                          count++;
59                      }
60                  }
61                  int       induk[][]       =       new       int
    [2][parent[0].length];
62                  induk[0] = parent[ind[0]];
63                  induk[1] = parent[ind[1]];
64                  int[][]    hasilCr    =    crossover(induk[0],
    induk[1]);
65                  for (int i = 0; i < 2; i++) {
66                      offCr[jum] = hasilCr[i];
67                      jum++;
68                      if (jum == offspringCr) {
69                          break;
70                      }
71                  }
72              }
73              System.out.println("");
74              System.out.println("OFFSPRING HASIL CROSSOVER");
75              for (int i = 0; i < offCr.length; i++) {
76                  System.out.print("C" + i + ":\t");
77                  for (int j = 0; j < offCr[0].length; j++) {
```

```
78                      System.out.print(offCr[i][j]);
79                  }
80                  System.out.println("");
81              }
82              //mutasi
83              int jml = 0;
84              int in;
85              int[]indukMutasi = new int [parent[0].length];
86              while (jml < offspringMr) {
87                  in = rand.nextInt(popsize);
                    indukMutasi = parent[in];
88                  offMr[jml] = mutasi(indukMutasi);
89                  jml++;
90                  if (jml == offspringMr) {
91                      break;
92                  }
93              }
94              System.out.println("");
95              System.out.println("OFFSPRING HASIL MUTASI");
96              for (int i = 0; i < offMr.length; i++) {
97                  System.out.print("C" + i + ":\t");
98                  for (int j = 0; j < offMr[0].length; j++) {
99                      System.out.print(offMr[i][j]);
100                 }
101                 System.out.println("");
102             }
103             individu = generateIndividu(popsize, offCr,
    offMr, parent, totalIndividu, pjgGen);
104             System.out.println("");
105             System.out.println("SEMUA INDIVIDU");
106             for (int i = 0; i < individu.length; i++) {
107                 System.out.print("I" + i + ":\t");
108                 for (int j = 0; j < individu[0].length; j++)
109 {
110                     System.out.print(individu[i][j]);
111                 }
112                 System.out.println("");
113             }
114             for (int i = 0; i < individu.length; i++) {
115                 if (!fiturGA.isEmpty()) {
                        fiturGA.clear();
116                 }
117                 for (int j = 0; j < individu[i].length; j++)
    {
118                     if (individu[i][j] == 1) {
119                         fiturGA.add(fiturGain.get(j));
120                     }
121                 }
122                 for (int j = 0; j < dataUji.size(); j++) {
123                     nwKnn(frekuensiDtUji.get(j), k, fiturGA);
124                 }
125                 fitness[i] = evaluasi(hasilKlasifikasi,
    kelasDtUji).get("fMeasure");
126             }
127             System.out.println("");
128             System.out.println("FITNESS SEMUA INDIVIDU");
129             for (int i = 0; i < fitness.length; i++) {
130                 System.out.println("I" + i + ": " +
    fitness[i]);
```

```
131             }
132             //seleksi
133             seleksi(popsize, fitness, individu);
134             System.out.println("");
135             System.out.println("HASIL   SELEKSI   :   GENERASI
    BARU");
136             for (int i = 0; i < parent.length; i++) {
137                 System.out.print("P" + i + ":\t");
138                 for (int j = 0; j < parent[0].length; j++) {
139                     System.out.print(parent[i][j]);
140                 }
141                 System.out.println("");
142             }
143             System.out.println("FITNESS    TERTINGGI    "    +
    fitness[0]);
144             generasi++;
145         }
146     }
```

Berdasarkan pada Tabel 5.25, pada baris 64 terdapat pemanggilan method corssover(). Method tersebut memiliki alur proses seperti pada Gambar 4.26 pada Bab 4 Perancangan. Alur proses tersebut apabila diimplementasikan kedalam bentuk kode program maka didapatkan hasil kode program seperti pada Tabel 5.26. Pada tabel tersebut proses *crossover* terjadi pada baris 18-21 atau 31-34 tergantung titik potong yakni variabel acak1 dan acak2.

Tabel 5.26 Implementasi crossover

```
1     private static Random rand = new Random();
2     public static int[][] crossover(int[] p1, int[] p2) {
3         int offspring[][] = new int[2][p1.length];
4         int acak1 = rand.nextInt(p1.length);
5         int acak2;
6         do {
7             acak2 = rand.nextInt(p1.length);
8         } while (acak1 == acak2);
9         if (acak1 < acak2) {
10            for (int i = 0; i < (acak1+1); i++) {
11                offspring[0][i] = p1[i];
12                offspring[1][i] = p2[i];
13            }
14            for (int i = acak1 + 1; i < acak2; i++) {
15                offspring[0][i] = p2[i];
16                offspring[1][i] = p1[i];
17            }
18            for (int i = acak2; i < offspring[0].length; i++)
    {
19                offspring[0][i]=p1[i];
20                offspring[1][i]=p2[i];
21            }
22        } else {
23            for (int i = 0; i < (acak2+1); i++) {
24                offspring[0][i] = p1[i];
25                offspring[1][i] = p2[i];
26            }
27            for (int i = acak2 + 1; i < acak1; i++) {
28                offspring[0][i] = p2[i];
29                offspring[1][i] = p1[i];
30            }
```

```
31              for (int i = acak1; i < offspring[0].length; i++)
    {
32                  offspring[0][i]=p1[i];
33                  offspring[1][i]=p2[i];
34              }
35          }
36          return offspring;
37      }
```

Pada Tabel 5.25, pada baris 88 terdapat pemanggilan method mutasi(). Method tersebut memiliki alur proses seperti pada Gambar 4.27 pada Bab 4 Perancangan. Alur proses tersebut apabila diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.27. Pada tabel tersebut inti dari proses mutasi terdapat pada baris 7-11.

Tabel 5.27 Implementasi mutasi

```
1       private static Random rand = new Random();
2       public static int[] mutasi(int[] p1) {
3           int offspring[] =new int[p1.length];
4           int acak = rand.nextInt(p1.length);
5           for (int i = 0; i < p1.length; i++) {
6               if (i == acak){
7                   if (p1[i]==0){
8                       offspring[i]=1;
9                   } else {
10                      offspring[i]=0;
11                  }
12              } else {
13                  offspring[i]=p1[i];
14              }
15          }
16          return offspring;
17      }
```

Pada Tabel 5.25, pada baris 103 terdapat pemanggilan method generateIndividu(). Method tersebut memiliki alur proses seperti pada Gambar 4.28 pada Bab 4 Perancangan. Alur proses tersebut apabila diimplementasikan kedalam bentuk kode program maka hasil implementasi seperti pada Tabel 5.28. Pada tabel tersebut proses pengumpulan seluruh individu kedalam satu kelompok terjadi pada baris 3-11. Pada baris 3-5 itu sendiri adalah untuk *parent* (induk). Pada baris 6-8 adalah untuk *offspring* hasil *crossover*. Sedangkan baris 9-11 adalah untuk *offspring* hasil mutasi.

Tabel 5.28 Implementasi generate individu

```
1       public static int[][] generateIndividu(int popsize,
    int[][] offCr, int[][] offMr, int[][] parent, int
    totalIndividu, int pjgGen) {
2           int[][] individu = new int[totalIndividu][pjgGen];
3           for (int i = 0; i < popsize; i++) {
4               individu[i] = parent[i];
5           }
6           for (int i = popsize; i < (popsize + offCr.length);
    i++) {
7               individu[i] = offCr[i - popsize];
8           }
9           for (int i = (popsize + offCr.length); i < (popsize +
    offCr.length + offMr.length); i++) {
```

```
10              individu[i] = offMr[i - (popsize + offCr.length)];
11          }
12          return individu;
13      }
```

Pada Tabel 5.25, pada baris 123 terdapat pemanggilan method nwknn(). Method tersebut memiliki alur proses seperti pada Gambar 4.29 pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka hasil implementasi yang dihasilkan adalah seperti pada Tabael 5.29. Pada tabel tersebutproses nwknn diadali dengan proses *term weighting* yang dilakukan dengan memanggil method termWeighting pada baris 3, kemudian menghitung *cosine similarity* dengan memanggil method cosSim() pada baris 4. Dari hasil nilai *cosine similarity* akan diambil sejumlah k tertinggi dengan memanggil method rank() pada baris 5. Setelah itu memanggil method weight() untuk menghitung pembobotan kelas pada baris 6. Perhitungan *score* sendiri terjadi pada baris 7 yaitu pemanggilan method score(). Score tertinggi dari kelas akan digunakan sebagai hasil klasifikasi.

Tabel 5.29 Implementasi klasifikasi dengan NWKNN

```
1       public static String nwKnn(HashMap<String, Integer> dtUji, int k, ArrayList<String> fitur) {
2           String hasil = "";
3           double wtd[][] = termWeighting(dtUji, fitur);
4           double[] cosSim = cosSim(wtd);
5           rank(k, cosSim);
6           double w[] = weight(frekuensiKelasDtLatih, kKelasLatih);
7           double score[] = score(w, kCosim);
8           double max = score[0];
9           int index = 0;
10          for (int i = 1; i < score.length; i++) {
11              if (score[i] > max) {
12                  max = score[i];
13                  index = i;
14              }
15          }
16          if (index == 0) {
17              hasil = "dkp";
18          } else if (index == 1) {
19              hasil = "dpuppb";
20          } else {
21              hasil = "dishub";
22          }
23          hasilKlasifikasi.add(hasil);
24          return hasil;
25      }
```

Pada Tabel 5.29, pada baris 3 terjadi pemanggilan method termWeighting(). Method ini memiliki alur proses seperti pada Gambar 4.30 pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.30. Pada tabel tersebut proses *term weighting* diawali dengan menghitung term frekuensi yang terdapat pada baris 2 yaitu dengan memanggil method termFrekuensi(). Apabila perhitungan term frekuensi selesai maka akan dihitung

pembobotan term terhadap dokumen dengan menjalankan method Wtd() pada baris 3. Hasil dari proses method Wtd() akan dilakukan normalisasi dengan menjalankan method WtdNormal() pada baris 4.

Tabel 5.30 Implementasin term weighting

```
1     public static double[][] termWeighting(HashMap<String,
      Integer> dtUji, ArrayList<String> fitur) {
2         TF = termFrekuensi(dtUji, fitur);
3         double wtd[][] = Wtd(TF);
4         wtd = WtdNormal(wtd, fitur, TF);
5         return wtd;
6     }
```

Pada Tabel 5.30 terdapat pemanggilan method termFrekuensi() pada baris 2. Method tersebut memiliki alur proses seperti yang telah digambarkan pada Bab 4 Perancangan yaitu pada Gambar 4.31. Apabila alur proses yang terjadi pada gambar tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.31. Pada tabel tersebut perhitungan terf frekuensi dari data uji terdapat pada baris 7-14. Sedangkan perhitungan term frekuensi pada data latih terdapat pada baris 14-21.

Tabel 5.31 Implementasi term frekuensi

```
1     private static ArrayList<String> dataLatih;
2     private static ArrayList<HashMap<String, Integer>>
      frekuensiDtLatih = new ArrayList<>();
3     public static int[][] termFrekuensi(HashMap<String,
      Integer> dtUji, ArrayList<String> fitur) {
4         int tf[][] = new int[fitur.size()][dataLatih.size() +
      1];
5         for (int i = 0; i < fitur.size(); i++) {
6             for (int j = 0; j < (dataLatih.size() + 1); j++)
      {
7                 if (j == 0) {
8                     for (Map.Entry map : dtUji.entrySet()) {
9                         if (((String)
      map.getKey()).equalsIgnoreCase(fitur.get(i))) {
10                            tf[i][j] = (int) map.getValue();
11                            break;
12                        }
13                    }
14                } else {
15                    for (Map.Entry map :
      frekuensiDtLatih.get(j - 1).entrySet()) {
16                        if (((String)
      map.getKey()).equalsIgnoreCase(fitur.get(i))) {
17                            tf[i][j] = (int) map.getValue();
18                            break;
19                        }
20                    }
21                }
22            }
23        }
24        return tf;
25    }
```

Pada Tabel 5.30 terdapat pemanggilan method Wtd() pada baris 3. Methdod tersebut memiliki alur proses seperti pada Gambar 4.32. Apabila alur tersebut

diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.32. Pada tabel tersebut perhitungan bobot term terdahap dokumen terjadi pada baris 10-12 sesuai dengan persamaan 2.2 pada Bab 2 Tinjauan Pustaka.

Tabel 5.32 Implementasi perhitungan bobot term terhadap dokumen

```
1     public static double[][] Wtd(int[][] TF) {
2         double[][] Wtd = new double[TF.length][TF[0].length];
3         for (int i = 0; i < TF.length; i++) {
4             int dokumenTerm = 0;
5             for (int j = 0; j < TF[0].length; j++) {
6                 if (TF[i][j] > 0) {
7                     dokumenTerm++;
8                 }
9             }
10            for (int j = 0; j < TF[0].length; j++) {
11                Wtd[i][j] = TF[i][j] * (1 + (Math.log((double) (dataLatih.size() + 1) / (double) dokumenTerm)));
12            }
13        }
14        return Wtd;
15    }
```

Pada Tabel 5.30 terdapat pemanggilan method WtdNormal() pada baris 4. Method tersebut memiliki alur proses seperti pada Gambar 4.33 pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam kode program maka hasil implementasi dari alur proses tersebut seperti pada Tabel 5.33. Pada tabel tersebut perhitungan normalisasi bobot term terhadap dokumen terdapat pada baris 18. Baris tersebut menerapkan persamaan 2.4 pada Bab 2 Tinjauan Pustaka.

Tabel 5.33 Implementasi normalisasi bobot term terhadap dokumen

```
1     public static double[][] WtdNormal(double[][] Wtd, ArrayList<String> fitur, int[][] TF) {
2         double[][] WtdNormal = new double[Wtd.length][Wtd[0].length];
3         for (int i = 0; i < Wtd[0].length; i++) {
4             double sum = 0;
5             for (int j = 0; j < Wtd.length; j++) {
6                 int dokumenTerm = 0;
7                 for (int l = 0; l < TF[j].length; l++) {
8                     if (TF[j][l] > 0) {
9                         dokumenTerm++;
10                    }
11                }
12                sum += Math.pow(TF[j][i], 2) * Math.pow(1 + (Math.log((double) (dataLatih.size() + 1) / (double) dokumenTerm)), 2);
13            }
14            for (int j = 0; j < Wtd.length; j++) {
15                if (sum == 0) {
16                    WtdNormal[j][i] = 0;
17                } else {
18                    WtdNormal[j][i] = Wtd[j][i] / Math.sqrt(sum);
19                }
```

```
20              }
21          }
22          return WtdNormal;
23      }
```

Kembali pada Tabel 5.29, pada baris 4, terdapat pemanggilan method cosSim(). Method tersebut memiliki alur proses seperti pada Gambar 4.34 pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.35. Pada tabel tersebut perhitungan *cosine similarity* terdapat pada baris 6 dengan menerapkan persamaan 2.5 pada Bab 2 Tinjauan Pustaka.

Tabel 5.34 Implementasi cosine similarity

```
1     private static ArrayList<String> dataLatih;
2     public static double[] cosSim(double[][] WtdNormal) {
3         double[] cosSim = new double[dataLatih.size()];
4         for (int i = 1; i < WtdNormal[0].length; i++) {
5             for (int j = 0; j < WtdNormal.length; j++) {
6                 cosSim[i - 1] += WtdNormal[j][0] *
  WtdNormal[j][i];
7             }
8         }
9         return cosSim;
10    }
```

Pada Tabel 5.29, pada baris 5, terdapat pemanggilan method rank() yang digunakan untuk mengambil dokumen yang memiliki nilai *cosine similarity* tertinggi sejumlah k. Method tersebut memiliki alur proses seperti pada Gambar 4.35 pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.35. Pada tabel tersebut pengambilan dokumen yang memiliki nilai *cosine similarity* tertinggi diawali dengan mengurutkan dokumen berdasarkan nilai *cosine similarity*-nya yang terjadi pada baris 11-22. Sedangkan pengambilan *cosine similarity* sejumlah k tertinggi terdapat pada baris 23-26.

Tabel 5.35 Implementasi pengurutan cosine similarity

```
1     private static ArrayList<String> dataLatih;
2     private static ArrayList<String> kelasDtLatih = new
  ArrayList<>();
3     private static double[] kCosim;
4     private static ArrayList<String> kKelasLatih = new
  ArrayList<>();
5     public static void rank(int k, double[] cosSimilarity) {
6         kCosim = new double[k];
7         double cosSim[] = cosSimilarity;
8         ArrayList<String> kelasDataLatih = kelasDtLatih;
9         String temp;
10        double temp2;
11        for (int i = 1; i < dataLatih.size(); i++) {
12            for (int j = 0; j < dataLatih.size() - i; j++) {
13                if (cosSim[j] < cosSim[j + 1]) {
14                    temp = kelasDataLatih.get(j);
15                    temp2 = cosSim[j];
16                    kelasDataLatih.set(j,
  kelasDataLatih.get(j + 1));
17                    cosSim[j] = cosSim[j + 1];
```

```
18                      kelasDataLatih.set(j + 1, temp);
19                      cosSim[j + 1] = temp2;
20                  }
21              }
22          }
23          for (int i = 0; i < k; i++) {
24              kCosim[i] = cosSim[i];
25              kKelasLatih.add(kelasDataLatih.get(i));
26          }
27      }
```

Pada Tabel 5.29, tepatnya pada baris 6 terdapat pemanggilan method weight(). Method tersebut memiliki alur proses seperti pada Gambar 4.36 pada Bab 4 Perancangan. Apabila alur tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Gambar 5.36. Pada tabel tersebut perhitungan bobot kelas terdapat pada baris 12 untuk bobot kelas DKP, baris 13 untuk kelas DPUPPB dan baris 14 untuk kelas Dishub. Perhitungan bobot tersebut disesuaikan dengan persamaan 2.6 yang terdapat pada Bab 2 Tinjauan Pustaka.

Tabel 5.36 Implementasi pembobotan kelas

```
1   public static double[] weight(HashMap<String, Integer>
    frekuensiKelasDtLatih, ArrayList<String> kKelasLatih) {
2       double[] weight = new
    double[frekuensiKelasDtLatih.size()];
3       HashMap<String, Integer> mapKategori = new
    HashMap<>();
4       mapKategori = frekuensi(kKelasLatih);
5       double tempMin = 99999999;
6       for (Map.Entry map : mapKategori.entrySet()) {
7           int value = (int) map.getValue();
8           if (value < tempMin) {
9               tempMin = (double) value;
10          }
11      }
12      weight[0] = 1 / (Math.pow(((double)
    frekuensiKelasDtLatih.get("dkp") / tempMin), (1 / Math.E)));
13      weight[1] = 1 / (Math.pow(((double)
    frekuensiKelasDtLatih.get("dpuppb") / tempMin), (1 /
    Math.E)));
14      weight[2] = 1 / (Math.pow(((double)
    frekuensiKelasDtLatih.get("dishub") / tempMin), (1 /
    Math.E)));
15      return weight;
16  }
```

Pada Tabel 5.29, tepatnya pada baris 7 terdapat pemanggilan method score(). Method tersebut memiliki alur proses seperti pada Bab 4 Perancangan yaitu pada Gambar 4.37. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.37. Pada tabel tersebut implementasi proses perhitungan *score* terdapat pada baris 10. Baris tersebut mengimplementasikan persamaan 2.7 yang terdapat pada Bab 2 Tinjauan Pustaka.

Tabel 5.37 Implementasi perhitungan score

```
1   private static HashMap<String, Integer>
```

```
    frekuensiKelasDtLatih = new HashMap<>();
2       private static ArrayList<String> kKelasLatih = new
    ArrayList<>();
3       public static double[] score(double[] weight, double[]
    kCosim) {
4           double[] score = new
    double[frekuensiKelasDtLatih.size()];
5           for (int i = 0; i < frekuensiKelasDtLatih.size();
    i++) {
6               double sum = 0;
7               for (int j = 0; j < k; j++) {
8                   sum += (kCosim[j] * cekKategori(i, j,
    kKelasLatih));
9               }
10              score[i] = weight[i] * sum;
11          }
12          return score;
13      }
```

Pada Tabel 5.37 terdapat pemanggilan method cekKategori() pada baris 8. Pemanggilan method tersebut digunakan untuk menilai kesesuaian kelas antara kelas yang sedang dihitung *score*-nya dengan kelas dari setiap dokumen tetangga. Method cekKategori() ini sendiri memiliki alur proses seperti pada Gambar 4.38 yang terdapat pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program, maka hasil implementasi yang didapatkan adalah seperti pada Tabel 5.38. Pada tabel tersebut peroses pengecekan kategori terdapat pada baris 10-14, dimana apabila kelasnya sesuai akan mengembalikan nilai 1, sedangkan apabila tidak sesuai akan mengembalikan nilai 0.

Tabel 5.38 Implementasi pengecekan kategori

```
1       public static int cekKategori(int indexScore, int
    indexKelasKDtLatih, ArrayList<String> kKelasLatih) {
2           String kelas;
3           if (indexScore == 0) {
4               kelas = "dkp";
5           } else if (indexScore == 1) {
6               kelas = "dpuppb";
7           } else {
8               kelas = "dishub";
9           }
10          if
    (kKelasLatih.get(indexKelasKDtLatih).equalsIgnoreCase(kelas))
    {
11              return 1;
12          } else {
13              return 0;
14          }
15      }
```

Kembali pada Tabel 5.25, dimana pada tabel tersebut terdapat pemanggilan method evaluasi() yang terjadi pada baris 125. Method tersebut memiliki alur proses seperti pada Gambar 4.39 pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program, maka akan didapatkan hasil implementasi seperti pada Tabel 5.39. Pada tabel tersebut inti dari proses evaluasi terdapat pada baris 90 karena nilai *fitness* yang digunakan

adalah nilia rata-rata *f-measure*. Perhitungan rata-rata *f-measure* yang telah diimplementasikan pada Tabel 5.39, baris 90 adalah perhitungan yang telah disesuaikan dengan persamaan 2.15 yang terdapat pada Bab 2 Tinjauan Pustaka.

Tabel 5.39 Implementasi evaluasi

```
1   private static HashMap<String, Integer> frekuensiKelasDtLatih = new HashMap<>();
2   private static HashMap<String, Integer> frekuensiKelasDtUji = new HashMap<>();
3   private static ArrayList<String> dataUji;
4   public static HashMap<String, Double> evaluasi(ArrayList<String> hasilKlasifikasi, ArrayList<String> kelasDtUji) {
5       HashMap<String, Double> eval = new HashMap<>();
6       double fMeasure[] = new double[frekuensiKelasDtLatih.size()];
7       double precision[] = new double[frekuensiKelasDtLatih.size()];
8       double recall[] = new double[frekuensiKelasDtLatih.size()];
9       double rtPrecision = 0;
10      double rtRecall = 0;
11      double rtFmeasure = 0;
12      for (int i = 0; i < frekuensiKelasDtUji.size(); i++) {
13          double tp = 0;
14          double fp = 0;
15          double fn = 0;
16          for (int j = 0; j < hasilKlasifikasi.size(); j++) {
17              switch (i) {
18                  case 0:
19                      if (hasilKlasifikasi.get(j).equalsIgnoreCase("dkp") && kelasDtUji.get(j).equalsIgnoreCase("dkp")) {
20                          tp = tp + 1;
21                      }
22                      if (hasilKlasifikasi.get(j).equalsIgnoreCase("dkp") && !kelasDtUji.get(j).equalsIgnoreCase("dkp")) {
23                          fp = fp + 1;
24                      }
25                      if (!hasilKlasifikasi.get(j).equalsIgnoreCase("dkp") && kelasDtUji.get(j).equalsIgnoreCase("dkp")) {
26                          fn = fn + 1;
27                      }
28                      break;
29                  case 1:
30                      if (hasilKlasifikasi.get(j).equalsIgnoreCase("dpuppb") && kelasDtUji.get(j).equalsIgnoreCase("dpuppb")) {
31                          tp = tp + 1;
32                      }
33                      if (hasilKlasifikasi.get(j).equalsIgnoreCase("dpuppb") && !kelasDtUji.get(j).equalsIgnoreCase("dpuppb")) {
34                          fp = fp + 1;
```

```
35                          }
36                          if
(!hasilKlasifikasi.get(j).equalsIgnoreCase("dpuppb")        &&
kelasDtUji.get(j).equalsIgnoreCase("dpuppb")) {
37                              fn = fn + 1;
38                          }
39                          break;
40                      default:
41                          if
(hasilKlasifikasi.get(j).equalsIgnoreCase("dishub")         &&
kelasDtUji.get(j).equalsIgnoreCase("dishub")) {
42                              tp = tp + 1;
43                          }
44                          if
(hasilKlasifikasi.get(j).equalsIgnoreCase("dishub")         &&
!kelasDtUji.get(j).equalsIgnoreCase("dishub")) {
45                              fp = fp + 1;
46                          }
47                          if
(!hasilKlasifikasi.get(j).equalsIgnoreCase("dishub")        &&
kelasDtUji.get(j).equalsIgnoreCase("dishub")) {
48                              fn = fn + 1;
49                          }
50                          break;
51                  }
52              }
53              if (tp == 0 && fp == 0) {
54                  precision[i] = 0;
55              } else {
56                  precision[i] = tp / (tp + fp);
57              }
58              if (tp == 0 && fn == 0) {
59                  recall[i] = 0;
60              } else {
61                  recall[i] = tp / (tp + fn);
62              }
63              if (precision[i] == 0 && recall[i] == 0) {
64                  fMeasure[i] = 0;
65              } else {
66                  fMeasure[i] = (2 * precision[i] * recall[i])
/ (precision[i] + recall[i]);
67              }
68          }
69          for (int i = 0; i < frekuensiKelasDtUji.size(); i++)
{
70              switch (i) {
71                  case 0:
72                      rtPrecision                             +=
frekuensiKelasDtUji.get("dkp") * precision[i];
73                      rtRecall                                +=
frekuensiKelasDtUji.get("dkp") * recall[i];
74                      rtFmeasure                              +=
frekuensiKelasDtUji.get("dkp") * fMeasure[i];
75                      break;
76                  case 1:
77                      rtPrecision                             +=
frekuensiKelasDtUji.get("dpuppb") * precision[i];
78                      rtRecall                                +=
frekuensiKelasDtUji.get("dpuppb") * recall[i];
```

```
79                      rtFmeasure                                      +=
    frekuensiKelasDtUji.get("dpuppb") * fMeasure[i];
80                      break;
81                  default:
82                      rtPrecision                                     +=
    frekuensiKelasDtUji.get("dishub") * precision[i];
83                      rtRecall                                        +=
    frekuensiKelasDtUji.get("dishub") * recall[i];
84                      rtFmeasure                                      +=
    frekuensiKelasDtUji.get("dishub") * fMeasure[i];
85                      break;
86              }
87          }
88          rtPrecision = rtPrecision / dataUji.size();
89          rtRecall = rtRecall / dataUji.size();
90          rtFmeasure = rtFmeasure / dataUji.size();
91          eval.put("precision", rtPrecision);
92          eval.put("recall", rtRecall);
93          eval.put("fMeasure", rtFmeasure);
94          hasilKlasifikasi.clear();
95          return eval;
96      }
```

Pada Tabel 5.25, pada baris 133, terdapat pemanggilan method seleksi(). Method tersebut memiliki alur proses seperti pada Gambar 4.40 pada Bab 4 Perancangan. Apabila alur proses tersebut diimplementasikan kedalam bentuk kode program maka akan didapatkan hasil implementasi seperti pada Tabel 5.40. Pada tabel tersebut proses seleksi diawali dengan mengurutkan individu berdasarkan nilai *fitness* secara *descending*. Proses pengurutan tersebut terjadi pada baris 5-16. Sedangkan pada baris 17-19 merupakan proses pengambilan individu sejumlah popsize yang nantinya akan digunakan pada generasi selanjutnya.

Tabel 5.40 Implementasi seleksi

```
1       private static int parent[][];
2       public static void seleksi(int popsize, double[] fitness,
    int[][] individu) {
3           int temp[];
4           double temp2;
5           for (int i = 1; i < individu.length; i++) {
6               for (int j = 0; j < individu.length - i; j++) {
7                   if (fitness[j] < fitness[j + 1]) {
8                       temp = individu[j];
9                       temp2 = fitness[j];
10                      individu[j] = individu[j + 1];
11                      fitness[j] = fitness[j + 1];
12                      individu[j + 1] = temp;
13                      fitness[j + 1] = temp2;
14                  }
15              }
16          }
17          for (int i = 0; i < popsize; i++) {
18             parent[i] = individu[i];
19          }
20      }
```

BAB 6 PENGUJIAN DAN ANALISIS

Berdasarkan pada Bab 3 Metodologi, rencana pengujian yang akan dilakukan adalah pengujian pengaruh persentase jumlah fitur, pengujian pengaruh metode NW-KNN, pengujian pengaruh algoritme genetika meliputi pengujian ukuran populaso, pengujian nilai CR dan MR, dan pengujian banyaknya generasi. Pengujian dilakukan sebanyak dua kali, yaitu terhadap data yang seimbang dan data yang tidak seimbang. Hal ini dilakukan untuk dapat dibandingkan hasil *f-measure* yang dihasilkan dari data tidak seimbang dan data seimbang.

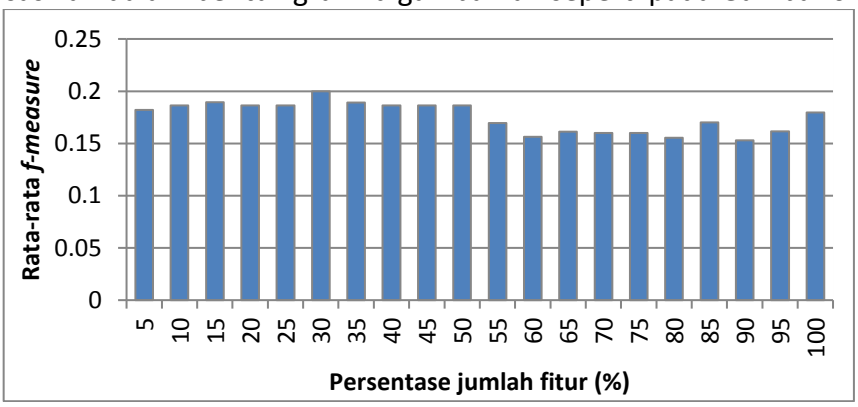
6.1 Pengujian data tidak seimbang

Pengujian yang dilakukan pertama adalah pengujian pada data yang tidak seimbang. Berdasarkan pada Bab 3 Metodologi, jumlah data yang digunakan pada pengujian ini sebesar 300 data yang terdiri dari 38 aduan untuk SKPD DKP, 111 aduan untuk SKPD DPUPPB, serta 151 aduan untuk SKPD Dishub. Dari data yang digunanakn dibagi menjadi dua dimana 70% diantaranya merupakan data latih dan 30% diantaranya merupakan data uji. Pada pengujian ini, parameter yang diuji pertama kali adalah persentase jumlah fitur untuk mendapatkan persentase terbaik dalam mereduksi fitur pada proses *information* gain. Pada pengujian persentase jumlah fitur, digunakan 3 tetangga terdekat dengan persentase jumlah fitur yang diuji 5% hingga 100% dengan kelipatan 5. Hasil pengujian persentase jumlah fitur pada data yang tidak seimbang tersebut ditunjukkan pada Tabel 6.1.

Tabel 6.1 Hasil pengujian persentase jumlah fitur pada data tidak seimbang

Persentase Jumlah Fitur (%)	Banyak Fitur	Rata-rata		
		Precision	*Recall*	*F-Measure*
5	93	0,12384	0,34444	0,182185
10	186	0,12642	0,35556	0,186521
15	279	0,13667	0,32222	0,189418
20	372	0,12642	0,35555	0,186521
25	465	0,12642	0,35556	0,186521
30	558	0,14739	0,31111	0,2
35	651	0,14048	0,28889	0,188984
40	744	0,12642	0,35556	0,186521
45	837	0,12642	0,35556	0,186521
50	929	0,12642	0,35556	0,186521
55	1022	0,12434	0,26667	0,169575
60	1115	0,11503	0,24444	0,156444
65	1208	0,11786	0,25556	0,161262
70	1301	0,11660	0,25555	0,160036
75	1394	0,11660	0,25555	0,160036
80	1487	0,11403	0,24444	0,155494
85	1580	0,12501	0,26667	0,170173
90	1673	0,11371	0,23333	0,152870
95	1766	0,12081	0,24444	0,161520
100	1858	0,13630	0,26667	0,179569

Hasil pengujian persentase jumlah fitur dengan 3 tetangga terdekat apabila divisualisasikan dalam bentuk grafik digambarkan seperti pada Gambar 6.1.



Gambar 6.1 Grafik hasil pengujian persentase jumlah fitur pada data tidak seimbang

Berdasarkan pada Gambar 6.1, rata-rata *f-measure* tertinggi terjadi ketika menggunakan 30% jumlah fitur yaitu sebesar 0,2. Hal ini menunjukkan bahwa persentase jumlah fitur mempengaruhi rata-rata *f-measure* yang dihasilkan. Pada data tidak seimbang ini, hanya menggunakan 30% jumlah fitur dapat menghasilkan jumlah fitur yang lebih tinggi dibandingkan dengan menggunakan seluruh fitur dalam proses klasifikasi. Sehingga persentase jumlah fitur 30% dianggap sebagai jumlah fitur yang optimal dalam proses klasifikasi menggunakan *information gain*.

Persentase jumlah fitur 30% kemudian digunakan dalam pengujian pengaruh metode NW-KNN. Berdasarkan pada Bab 3 Metodologi, pengujian ini dilakukan dengan menguji besarnya nilai K tertangga terdekat dengan menggunakan 2 metode klasifikasi secara bergantian yaitu NW-KNN dan KNN standar untuk dapat diketahui pengaruh NW-KNN terhadap hasil klasifikasi. Nilai K yang akan dilakukan pengujian adalah 3, 6, 9, 12, 15, 18, dan 21. Pengujian tersebut dilakukan dengan menggunakan teknik *information gain*. Hasil pengujian nilai K menggunakan NW-KNN serta menggunakan NW-KNN secara berturut-turut ditunjukkan pada Tabel 6.2 dan 6.3.

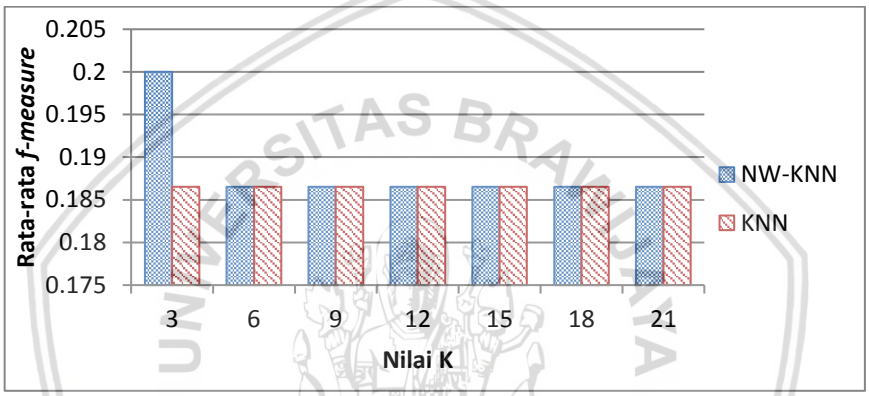
Tabel 6.2 Hasil pengujian nilai K menggunakan NW-KNN pada data tidak seimbang

Nilai K	Rata-rata		
	Precision	***Recall***	***F-Measure***
3	0,14739	0,31111	0,2
6	0,12642	0,35556	0,186521
9	0,12642	0,35556	0,186521
12	0,12642	0,35556	0,186521
15	0,12642	0,35556	0,186521
18	0,12642	0,35556	0,186521
21	0,12642	0,35556	0,186521

Tabel 6.3 Hasil pengujian nilai K menggunakan KNN pada data tidak seimbang

Nilai K	Rata-rata		
	Precision	***Recall***	***F-Measure***
3	0,12642	0,35556	0,186521
6	0,12642	0,35556	0,186521
9	0,12642	0,35556	0,186521
12	0,12642	0,35556	0,186521
15	0,12642	0,35556	0,186521
18	0,12642	0,35556	0,186521
21	0,12642	0,35556	0,186521

Hasil pengujian pada Tabel 6.2 dan Tabel 6.3 apabila divisualisasikan kedalam bentuk grafik, maka akan didapatkan hasil grafik secara berturut-turut seperti pada Gambar 6.2.



Gambar 6.2 Grafik hasil pengujian nilai K menggunakan NW-KNN dan KNN pada data tidak seimbang

Berdasarkan pada Gambar 6.2 rata-rata *f-measure* tertinggi yang terjadi ketika menggunakan metode NW-KNN terdapat pada K=3 yaitu sebesar 0,2. Hal ini menunjukkan bahwa jumlah tetangga terdekat yang sedikit lebih baik daripada menggunakan jumlah tetangga yang banyak. Pada hasil rata-rata *f-meaure* yang dihasilkan ketika menggunakan metode KNN pada setiap nilai K yang digunakan sama besar yaitu sebesar 0,186. Hal ini menunjukkan bahwa besarnya nilai K tidak mempengaruhi hasil klasifikasi. Dari kedua hasil pengujian baik menggunakan NW-KNN maupun KNN dapat dilihat bahwa rata-rata *f-measure* tertinggi terjadi ketika menggunakan NW-KNN. Hal ini membuktikan bahwa NW-KNN dapat mengatasi permasalahan data tidak seimbang, dimana hasil klasifikasi yang dihasilkan lebih baik daripada hasil klasifikasi yang dihasilkan oleh KNN standar. Hal ini disebabkan, metode NW-KNN dalam proses pengklasifikasiannya tidak semata-mata mempertimbangkan banyaknya dokumen yang masuk pada k tetangga terdakat, tetapi menggunakan pembobotan dan nilai kemiripan antara data uji dengan data latih yang terdapat pada k tetangga terdekat. NW-KNN memberikan bobot yang besar pada dokumen yang berjumlah sedikit (minoritas) dan memberikan bobot yang kecil pada dokumen yang berjumlah dominan (mayoritas). Hal tersebut membuktikan penelitian yang dilakukan oleh Ridok & Latifah (2015). Melihat rata-rata *f-measure* tertinggi terdapat pada K=3, maka

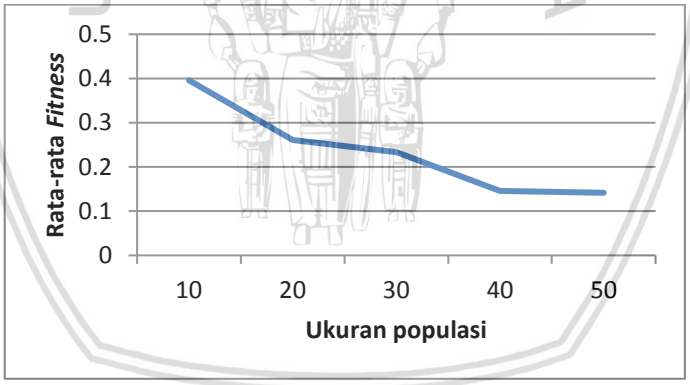
nilai tersebut dianggap sebagai jumlah tetangga terdekat yang optimal untuk digunakan.

Hasil dari pengujian persentase jumlah fitur dan nilai K kemudian digunakan pada pengujian parameter algoritme genetika. Parameter algoritme genetika yang akan diuji pertama adalah ukuran populasi yang akan digunakan. Pada pengujian ukuran populasi digunakan persentase jumlah fitur sebesar 30% dengan nilai K=3. Pengujian ini dilakukan sebanyak 3 kali percobaan dengan ukuran populasi yang akan diuji 10, 20, 30, 40 dan 50 dengan maksimum generasi 100 serta nilai CR 0.5 dan MR 0.5. Hasil dari seluruh percobaan yang telah dilakukan akan dilakukan perhitungan rata-rata *fitness*. Hasil pengujian ukuran populasi pada data tidak seimbang ditunjukkan pada Tabel 6.4.

Tabel 6.4 Hasil pengujian ukuran populasi pada data tidak seimbang

Popsize	Percobaan ke-			Rata-rata *fitness*
	1	2	3	
10	0,53452	0,40981	0,24378	0,396037
20	0,29782	0,24173	0,24323	0,260927
30	0,26693	0,22546	0,20843	0,233608
40	0,21895	0,19284	0,02662	0,146139
50	0,22017	0,02662	0,17778	0,141523

Hasil pengujian ukuran populasi pada Tabel 6.4 apabila digrafikkan maka akan didapatkan hasil seperti pada Gambar 6.3.



Gambar 6.3 Grafik hasil pengujian ukuran populasi pada data tidak seimbang

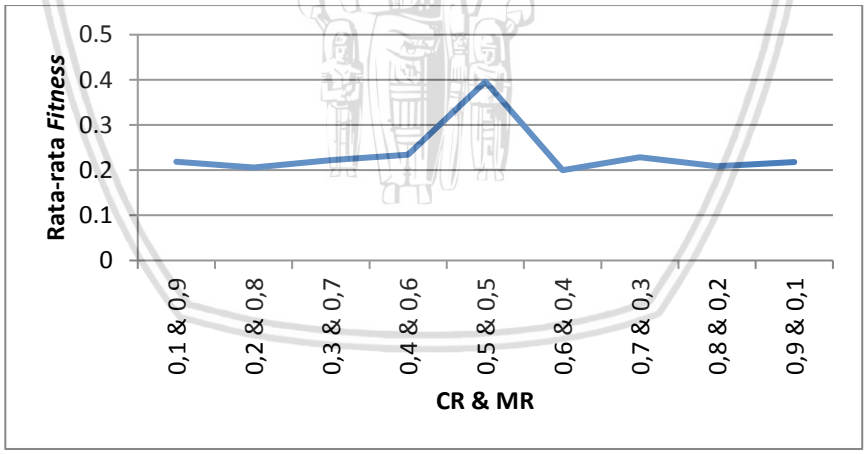
Berdasarkan pada Gambar 6.3 rata-rata *fitness* tertinggi terapat ketika ukuran populasi sebesar 10 digunakan. Rata-rata *fitness* pada ukuran populasi 10 yaitu sebesar 0,39. Pada gambar tersebut, semakin besar ukuran populasi yang digunakan, tidak memberikan hasil optimal, dimana rata-rata *fitness* yang dihasilkan semakin rendah. Hal ini disebabkan karena terjadinya penurunan kemampuan eksplorasi algoritme genetika dalam mencari solusi terbaik seiring meningkatnya ukuran populasi, dimana algoritme genetika itu sendiri bersifat *stochastic* dalam memperoleh solusi, sehingga hasil yang didapatkan tidak selalu hasil yang terbaik. Berdasarkan hasil pengujian tersebut, ukuran populasi 10, dianggap sebagai ukuran populasi terbaik yang akan digunakan pada pengujian nilai CR dan MR serta pengujian banyaknya generasi.

Pengujian selanjutnya adalah pengujian CR dan MR. Pada pengujian CR dan MR digunakan persentase jumlah fitur sebesar 30% dengan nilai k=3. Pengujian tersebut dilakukan sebanyak 3 kali percobaan dengan nilai CR dan MR yang akan diuji 0.1 dan 0.9, 0.2 dan 0,8, 0.3 dan 0.7, 0.4 dan 0.6, 0.5 dan 0.5, 0.6 dan 0.4, 0.7 dan 0.3, 0.8 dan 0.2, 0.9 dan 0.1 dengan ukuran populasi 10 serta maksimum generasi 100. Dari hasil seluruh percobaan setiap nilai CR dan MR akan dilakukan perhitungan rata-rata. Hasil pengujian nilai CR dan MR pada data tidak seimbang ditunjukkan pada Tabel 6.5.

Tabel 6.5 Hasil pengujian CR dan MR pada data tidak seimbang

CR	MR	Percobaan ke-			Rata-rata *fitness*
		1	2	3	
0,1	0,9	0,22166	0,19449	0,23932	0,218491
0,2	0,8	0,22544	0,18841	0,20369	0,205847
0,3	0,7	0,21248	0,25172	0,20311	0,222435
0,4	0,6	0,21750	0,25280	0,23135	0,233881
0,5	0,5	0,53452	0,40981	0,24378	0,396037
0,6	0,4	0,19787	0,20914	0,19284	0,199953
0,7	0,3	0,26429	0,20843	0,21360	0,228773
0,8	0,2	0,19617	0,21941	0,20977	0,208449
0,9	0,1	0,21055	0,20977	0,23281	0,217708

Rata-rata *fitness* dari hasil pengujian CR dan MR pada data tidak seimbang yang terdapat pada Tabel 6.5 apabila digrafikkan akan menghasilkan grafik seperti pada Gambar 6.4.



Gambar 6.4 Grafik hasil pengujian CR dan MR pada data tidak seimbang

Berdasarkan pada Gambar 6.4, rata-rata *fitness* tertinggi terdapat pada CR = 0,5 dan MR 0,5 digunakan. Rata-rata *fitness* tertinggi tersebut sebesar 0,39. Semakin besar, dan semakin kecil kombinasi nilai CR dan MR yang digunakan tidak menghasilkan hasil yang optimal jika dibandingkan dengan menggunakan CR = 0,5 dan MR = 0,5. Rata-rata *f-measure* tersebut dipengaruhi oleh kombinasi CR dan MR yang digunakan, dimana nilai CR dan MR yang optimal digunakan pada setiap kasus dapat berbeda karena tidak adanya ketentuan baku kombinasi nilai CR dan MR yang dapat memberikan hasil optimal. Kombinasi CR dan MR yang sama menunjukkan bahwa jumlah terbentuknya individu baru dengan tingkat

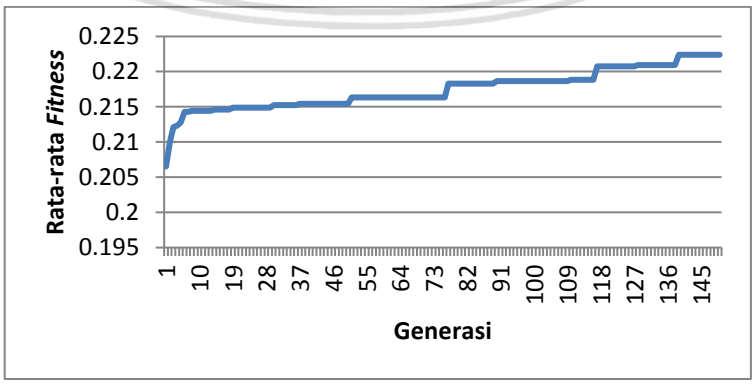
keberagaman populasi yang seimbang memberikan solusi yang optimal. Hal ini disebabkan banyaknya individu baru yang dihasilkan pada kasus ini berkaitan dengan keragaman populasi yang dihasilkan untuk mencegah terjadinya konvergensi dini dalam mencari solusi, dimana besarnya tingkat terbentuknya individu baru dan keragaman populasi berbanding lurus. Berdasarkan hasil pengujian tersebut, nilai CR=0,5 dan MR = 0,5 dianggap sebagai nilai yang terbaik untu digunakan pada pengujian banyaknya generasi.

Pengujian generasi dilakukan untuk menguji konvergensi yang terjadi pada proses algoritme genetika. Pada pengujian generasi ini digunakan persentase jumlah fitur sebesar 30% dengan nilai k = 3. Pengujian tersebut dilakukan sebanyak 3 kali percobaan dengan ukuran populasi 10, nilai CR = 0.5, MR = 0.5 dan banyak generasi yang akan diuji 1 sampai 150. Hasil pengujian generasi tersebut akan dihitung rata-rata *fitness* dari seluruh percobaan yang dilakukan. Hasil pengujian tersebut ditunjukkan pada Tabel 6.6 yang dilengkapi pada Lampiran A.

Tabel 6.6 Hasil pengujian banyaknya generasi pada data tidak seimbang

Banyaknya generasi	Percobaan ke-			Rata-rata *Fitness*
	1	2	3	
1	0,19216	0,21897	0,20843	0,206520
2	0,20251	0,21897	0,20843	0,209970
3	0,20251	0,22534	0,20843	0,212094
4	0,20251	0,22534	0,20914	0,212329
⋮	⋮	⋮	⋮	⋮
146	0,22082	0,23025	0,21612	0,222397
147	0,22082	0,23025	0,21612	0,222397
148	0,22082	0,23025	0,21612	0,222397
149	0,22082	0,23025	0,21612	0,222397
150	0,22082	0,23025	0,21612	0,222397

Hasil pengujian yang terdapat pada Tabel 6.6, apabila dilakukan visualisasi kedalam bentuk grafik, maka akan didapatkan gambar grafik seperti pada Gambar 6.5.



Gambar 6.5 Hasil pengujian banyak generasi pada data tidak seimbang

Berdasarkan pada Gambar 6.5 dari 3 percobaan yang telah dilakukan pada pengujian generasi didapatkan rata-rata *fitness* tertinggi terjadi ketika generasi

mencapai 134. Rata-rata *fitness* tertinggi yang dihasilkan pada generasi 134 yaitu sebesar 0,22. Menggunakan jumlah generasi yang lebih besar maka rata-rata *fitness* yang dihasilkan cenderung memiliki nilai yang sama. Selain itu, grafik tersebut menunjukkan bahwa semakin besar maksimum generasi yang digunakan rata-rata *f-measure* yang dihasilkan cenderung lebih besar, karena semakin besar jumlah generasi yang digunakan semakin besar pula kesempatan dalam memperoleh hasil terbaik. Berdasarkan hasil pengujian tersebut, 134 menjadi jumlah generasi maksimum yang dianggap sebagai maksimum generasi yang optimal untuk digunakan. Hasil klasifikasi pada data uji data tidak seimbang dengan menggunakan nilai parameter hasil pengujian ditunjukkan pada Tabel 6.7 yang dilengkapi pada Lampiran B.

Tabel 6.7 Hasil klasifikasi program pada data uji data tidak simbang

No	Aduan	Hasil Program
1	Selamat pagi bapak/ ibu, mohon perhatian untuk lampu penerangan jalan di sepanjang jalan M.T. Haryono yang tidak berfungsi (mati). Khususnya dari arah M. Panjaitan sepanjang Universitas Brawijaya, dan di depan Polsek Lowokwaru hingga daerah depan Mall Dinoyo. Saat malam hari, penerangan hanya berasal dari penerangan toko di sepanjang jalan. Apalagi akhir-akhir ini musim hujan sangat membahayakan para pengendara kendaraan dan pengguna jalan lainnya karena kurangnya penerangan jalan. Terima kasih atas perhatiannya.	DPUPPB
2	Pak Walikota Yth, Assalamu'alaikum Wr Wb, Saya, Kuntadi Haribowo, SH adalah Putra Malang Asli yang lahir dan besar di Malang lebih dari 47 tahun lalu. Masa kecil saya dulu adalah di jalan Guntur No 25, Kakek saya adalah Mantan Kepala Dinas Kesehatan Kota Malang, yaitu DR. R. Soepangkat, Arts. Sebagai Putra Malang Asli saya merasa ikut memiliki Kota Malang ini, dan saya ingin menyampaikan beberapa saran dan uneg2 saya kepada Bp. Walikota Malang, sebagai berikut : 1. Saya merasa sangat kecewa dengan penataan Taman Malabar yang saat ini ditutup seng sehingga berkesan kumuh sekali, padahal lokasinya tepat di pusat kota. Sangat berbeda dengan jaman tahun 70 dan 80 an yang masih terlihat asri dan indah dipandang. Tolong segera di kembalikan seperti dahulu lagi. 2. Saya melihat marka jalan di banyak jalan utama di kota Malang yang tidak ada atau tidak sesuai sebagaimana mestinya (tidak ditengah). Saya minta tolong segera dibenahi. 3. Saya melihat banyak sekali jalan yang lampu penerangan jalannya masih sangat kurang, sehingga jalan menjadi gelap karena kurang cahaya diwaktu malam hari. 4. Saat ini banyak sekali berseliweran ditengah kota BECAK MOTOR yang tentunya tidak sesuai dengan ijinnya, MOHON DITERTIBKAN. 5. Banyak sekali pengendara sepeda motor yang ugal2an dalam berkendara, tidak menggunakan helm dan saykin juga banyak yang tidak punya SIM. Khususnya daerah - daerah kampus, Dinoyo, Telogomas, Galunggung, Soekarno Hatta dll, Mohon dikoordinasikan dengan Kapolresta untuk diperbanyak operasi / razia lalu lintas. 6. Mohon ditugaskan seluruh jajaran Satpol PP untuk sesering mungkin mengadakan operasi razia MIRAS dan prostitusi. 7. Mohon agar ditertibkan parkir liar yang ada di seluruh wilayah Malang. Demikian saran dan usulan ini saya sampaikan, mohon untuk segera bisa ditindaklanjuti dengan baik. Saya sebagai Putra Malang Asli ingin melihat kota Malang yang lebih baik, lebih tertib di masa yang akan	DPUPPB

Tabel 6.7 Hasil klasifikasi program pada data uji data tidak simbang

No	Aduan	Hasil Program
	datang, terima kasih. Wassalamu'alaikum, Hormat saya, Kuntadi Haribowo, SH.	
3	Assalamu'alaikum wr.wb. Yth. Pak Walikota Malang, , untuk keamanan dan kenyamanan pejalan kaki / wisatawan , apakah pemkota malang mempunyai rencana program perbaikan fasilitas untuk pejalan kaki :1) Kayutangan - Alun-alun - Pecinan; 2). Alun-alun bunder - Kahuripan - Semeru - Ijen; 3). jalan. Kawi - Alun-alun. ? Fasilitas untuk pekalan kaki jangan/tidak boleh untuk parkir mobil dan sepeda motor ! . Terima kasih, wassalam	DPUPPB
⋮	⋮	⋮
31	Traffic Light Pertigaan Janti Mati sudah beberapa hari ini, harap segera di prioritaskan perbaikannya	DPUPPB
32	MOHON SEGERA DIPERBAIKI TRAFFIC LIGHT PERTIGAAN SUKUN-JANTI MATI SUDAH 1 MINGGU YG LALU, JANGAN TUNGGU KORBAN!	DPUPPB
33	sebelumnya Terimakasih atas respon cepat permasalahan TL Janti, menyampaikan pula kondisi di Jalan S.Supriadi tepatnya di depan Pabrik Es Ngaglik, kalau malam di pinggir jalan banyak dijadikan parkir liar mobil-mobil dan hal ini menjadikan sempit jalan, mohon Dishub dapat bertindak tegas, kedua kondisi di Jalan Jakarta banyak yg berlubang dan sangat membahayakan pengguna jalan, terimakasih	DPUPPB

6.2 Pengujian data seimbang

Pengujian kedua yang dilakukan yaitu pengujian terhadap data yang seimbang. Berdasarkan pada Bab 3 Metodologi, jumlah data yang digunakan pada pengujian data seimbang adalah 114 aduan dimana terdapat 38 aduan pada masing-masing SKPD. 70% dari seluruh aduan yang digunakan merupakan data latih, dan 30% dari seluruh aduan yang digunakan merupakan data uji. Pengujian yang pertama dilakukan adalah pengujian persentase jumlah fitur pada teknik *information gain*. Pengujian tersebut menggunakan 3 tetangga terdekat dengan persentase jumlah fitur yang diuji 5% hingga 100% dengan kelipatan 5. Hasil pengujian persentase jumlah fitur pada data seimbang ditunjukkan pada Tabel 6.8.

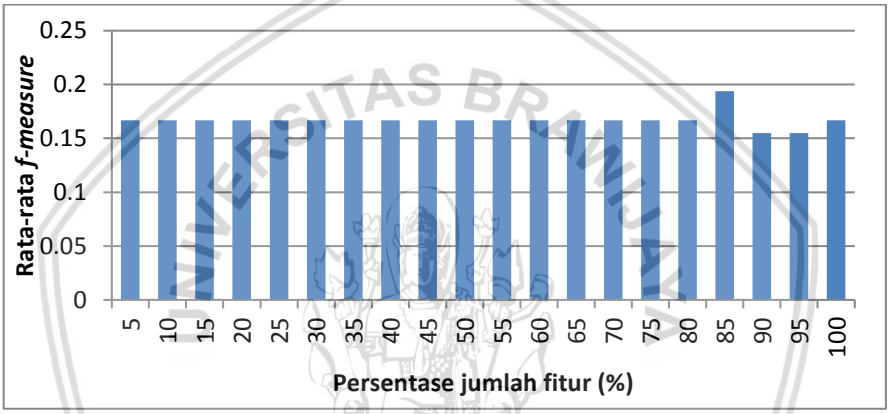
Tabel 6.8 Hasil pengujian persentase jumlah fitur pada data seimbang

Persentase Jumlah Fitur (%)	Banyak Fitur	Rata-rata		
		Precision	*Recall*	*F-Measure*
5	48	0,11111	0,33333	0,16667
10	95	0,11111	0,33333	0,16667
15	143	0,11111	0,33333	0,16667
20	190	0,11111	0,33333	0,16667
25	237	0,11111	0,33333	0,16667
30	285	0,11111	0,33333	0,16667
35	332	0,11111	0,33333	0,16667
40	380	0,11111	0,33333	0,16667
45	427	0,11111	0,33333	0,16667
50	474	0,11111	0,33333	0,16667
55	522	0,11111	0,33333	0,16667

Tabel 6.8 Hasil pengujian persentase jumlah fitur pada data seimbang

Persentase Jumlah Fitur (%)	Banyak Fitur	Rata-rata		
		Precision	*Recall*	*F-Measure*
60	569	0,11111	0,33333	0,16667
65	617	0,11111	0,33333	0,16667
70	664	0,11111	0,33333	0,16667
75	711	0,11111	0,33333	0,16667
80	759	0,11111	0,33333	0,16667
85	806	0,21111	0,30303	0,19396
90	854	0,11642	0,30303	0,15503
95	901	0,11642	0,30303	0,15503
100	948	0,11111	0,33333	0,16667

Hasil pengujian pada Tabel 6.8 apabila divisualisasikan kedalam bentuk grafik, maka akan didapatkan hasil grafik seperti pada Gambar 6.6.



Gambar 6.6 Grafik hasil pengujian persentase jumlah fitur pada data seimbang

Berdasarkan pada Gambar 6.6, banyaknya jumlah fitur yang digunakan mempengaruhi rata-rata *f-measure* yang dihasilkan. Pada gambar tersebut, rata-rata *f-measure* tertinggi terjadi ketika 85% jumah fitur hasil pengurutan *information gain* digunakan yaitu sebesar 0,21. Rata-rata *f-measure* tersebut lebih tinggi daripada ketika seluruh fitur digunakan. Hal ini membuktikan bahwa seleksi fitur dengan *information gain* dapat meningkatkan rata-rata *f-measure* yang dihasilkan.

Pengujian yang selanjutnya adalah pengujian pengaruh penggunaan metode NW-KNN teradap hasil klasifikasi pada data yang seimbang. Berdasarkan pada Bab 3 Metodologi, pengujian ini dilakukan dengan menguji nilai K dimana dalam pengujiannya menggunakan 85% dari seluruh jumlah fitur dengan menggunakan seleksi fitur *information gain*. Untuk dapat diketahui pengaruh metode NW-KNN maka perlu dilakukan dua kali uji dengan metode yang berbeda yaitu NW-KNN dan KNN standar. dengan nilai K yang diuji yaitu 3, 6, 9, 12, 15, 18, dan 21. Hasil pengujian nilai K dengan menggunakan NW-KNN dan KNN pada data seimbang secara berturut-turut ditunjukkan pada Tabel 6.9 dan 6.10.

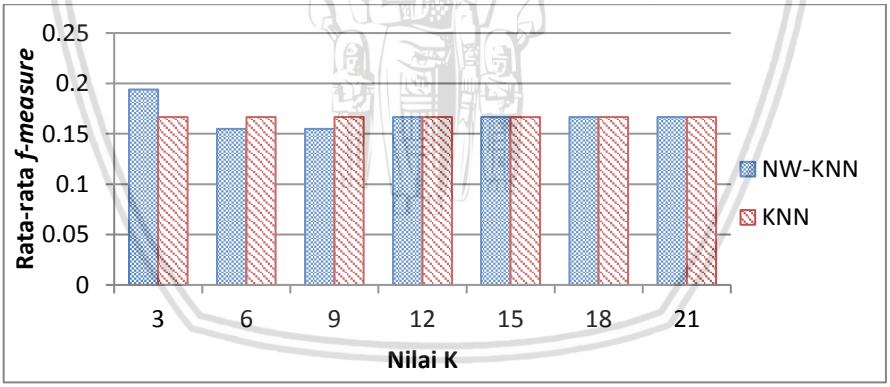
Tabel 6.9 Hasil pengujian nilai K menggunakan NW-KNN pada data seimbang

Nilai K	Rata-rata		
	Precision	*Recall*	*F-Measure*
3	0,21111	0,3030	0,19396
6	0,10417	0,30303	0,15504
9	0,10417	0,30303	0,15504
12	0,11111	0,33333	0,16667
15	0,11111	0,33333	0,16667
18	0,11111	0,33333	0,16667
21	0,11111	0,33333	0,16667

Tabel 6.10 Hasil pengujian nilai K menggunakan KNN pada data seimbang

Nilai K	Rata-rata		
	Precision	*Recall*	*F-Measure*
3	0,11111	0,33333	0,16667
6	0,11111	0,33333	0,16667
9	0,11111	0,33333	0,16667
12	0,11111	0,33333	0,16667
15	0,11111	0,33333	0,16667
18	0,11111	0,33333	0,16667
21	0,11111	0,33333	0,16667

Hasil pengujian pada Tabel 6.9 dan 610 apabila divisualisasikan kedalam bentuk grafik, maka akan didapatkan hasil grafik secara berturut turut seperti pada Gambar 6.7.



Gambar 6.7 Grafik hasil pengujian nilai K menggunakan NW-KNN pada data seimbang

Berdasarkan pada Gambar 6.7, rata-rata *f-measure* tertinggi yang terjadi ketika menggunakan metode NW-KNN terdapat pada nilai K = 3. Rata-rata *f-measure* yang didapatkan menggunakan 3 tetangga terdekat sebesar 0,19. Gambar tersebut menunjukkan bahwa menggunakan jumlah tetangga terdekat yang besar tidak memberikan hasil yang lebih baik. Ketika menggunakan metode KNN dengan menggunakan jumlah nilai K yang berbeda, besarnya nilai K tersebut tidak mempengaruhi rata-rata *f-measure* yang dihasilkan. Rata-rata *f-measure* yang dihasilkan menggunakan metode KNN memiliki besar yang sama yaitu sebesar 0,16. Berasarkan dua gambar tersebut, rata-rata *f-measure* tertinggi terdapat pada hasil klasifikasi menggunakan NW-KNN. Hal ini membuktikan

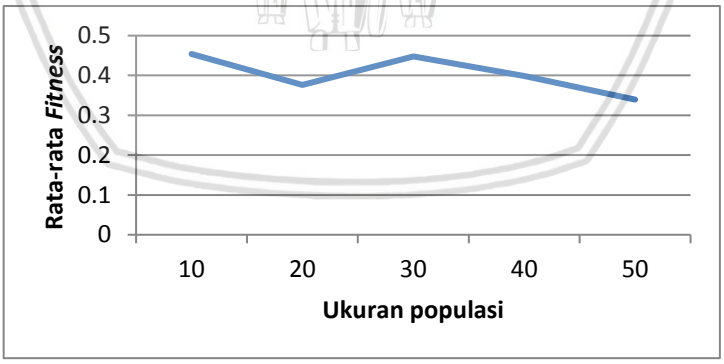
bahwa metode NW-KNN menghasilkan hasil klasifikasi yang lebih baik daripada KNN dimana rata-rata *f-measure* yang dihasilkan dengan NW-KNN lebih besar daripada yang dihasilkan dengan KNN pada K=3. Dikarenakan NW-KNN menghasilkan rata-rata *f-measure* yang lebih besar dibandingkan dengan menggunakan metode KNN maka, NW-KNN terbukti dapat mengatasi data yang seimbang. Hal ini sesuai dengan hasil penelitian yang telah dilakukan oleh Ridok & Latifah (2015). Berdasarkan hasil pengujian tersebut K=3 dianggap sebagai nilai K yang paling baik dan akan digunakan pada pengujian parameter algoritme genetika.

Pengujian selanjutnya adalah pengujian parameter algoritme genetika. Parameter pertama yang akan diuji adalah ukuran populasi. Pengujian ini menggunakan 85% jumlah fitur dengan 3 tetangga terdekat. Ukuran populasi yang akan diuji yaitu 10 hingga 50 dengan kelipatan 10. Nilai parameter algoritme lain yang digunakan dalam pengujian ini yaitu CR 0.5, MR 0.5 dan maksimum generasi 10. Hasil pengujian ukuran populasi pada data seimbang ditunjukkan pada Tabel 6.11.

Tabel 6.11 Hasil pengujian ukuran populasi pada data seimbang

Popsize	Percobaan ke-			Rata-rata *fitness*
	1	2	3	
10	0,45454	0,45005	0,45714	0,453912
20	0,43080	0,22778	0,46965	0,376077
30	0,48895	0,48179	0,37152	0,447419
40	0,51434	0,32172	0,36111	0,399056
50	0,27717	0,27717	0,46398	0,339438

Hasil pengujian pada Tabel 6.11 apabila divisualisasikan kedalam bentuk grafik, maka akan didapatkan hasil grafik seperti pada Gambar 6.8.



Gambar 6.8 Grafik hasil pengujian ukuran populasi pada data seimbang

Berdasarkan pada Gambar 6.8, rata-rata *fitness* tertinggi terdapat pada ukuran populasi 10 yaitu sebesar 0,45. Gambar tersebut menunjukkan bahwa semakin besar ukuran populasi yang digunakan, rata-rata *fitness* yang dihasilkan cenderung lebih rendah. Sehingga menggunakan ukuran populasi yang besar tidak memberikan hasil klasifikasi yang lebih baik daripada menggunakan ukuran populasi yang kecil. Hal ini disebabkan karena terjadinya penurunan kemampuan eksplorasi algoritme genetika dalam mencari solusi terbaik seiring meningkatnya

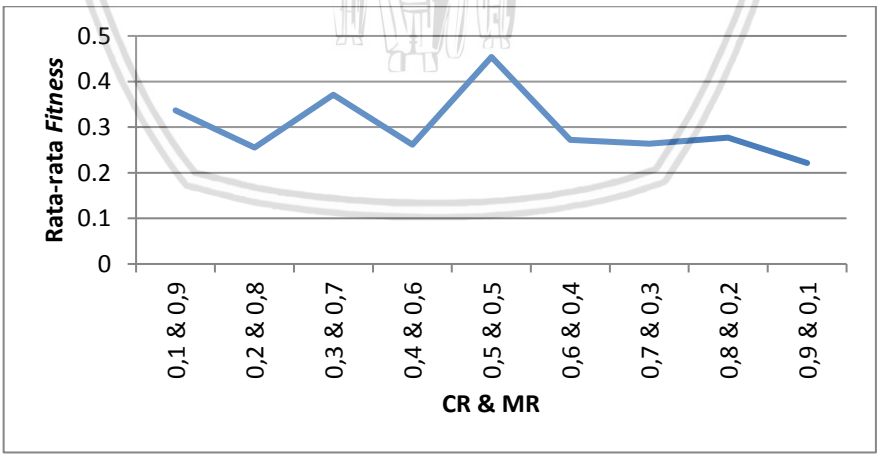
ukuran populasi, dimana algoritme genetika itu sendiri bersifat stochastic dalam memperoleh solusi, sehingga hasil yang didapatkan tidak selalu hasil yang terbaik. Pengujian ukuran populasi pada data seimbnag menghasilkan ukuran populasi 10 sebagai ukuran populasi yang akan digunakan pada pengujian kombinasi nilai CR dan MR dan pengujian banyaknya generasi.

Pengujian selanjutnya adalah pengujian CR dan MR. Pada pengujian ini, digunakan 85% jumlah fitur dengan 3 tetangga terdekat. Sedangkan pada parameter algoritme genetika yang digunakan adalah ukuran populasi 10, dan maksimum generasi 100. Nilai CR dan MR yang akan diuji adalah 0,1 dan 0,9, 0,2 dan 0,8, 0,3 dan 0,7, 0,4 dan 0,6, 0,5 dan 0,5, 0,7 dan 0,3, 0,8 dan 0, 2, serta 0,9 dan 0,1. Hasil pengujian nilai CR dan MR pada data seimbang ditunjukkan pada Tabel 6.12.

Tabel 6.12 Hasil pengujian CR dan MR pada data seimbang

CR	MR	Percobaan ke-			Rata-rata *fitness*
		1	2	3	
0,1	0,9	0,17460	0,43080	0,40404	0,33648
0,2	0,8	0,32172	0,27717	0,16667	0,25518
0,3	0,7	0,27717	0,43044	0,40476	0,37079
0,4	0,6	0,29594	0,32172	0,16667	0,26144
0,5	0,5	0,45454	0,45005	0,45714	0,45391
0,6	0,4	0,16667	0,42381	0,22648	0,27232
0,7	0,3	0,34734	0,27717	0,16667	0,26372
0,8	0,2	0,43044	0,22610	0,17460	0,27705
0,9	0,1	0,16667	0,27141	0,22610	0,22139

Hasil pengujian pada Tabel 6.12 apabila divisualisasikan kedalam bentuk grafik, maka akan didapatkan hasil grafik seperti pada Gambar 6.9.



Gambar 6.9 Grafik hasil pengujian CR dna MR pada data seimbang

Berdasarkan pada Gambar 6.9, rata-rata *fitness* tertinggi terjadi ketika menggunakan nilai CR dan MR 0,5. Rata-rata *fitness* tersebut sebesar 0,45. Menggunakan kombinasi nilai CR dan MR yang lebih kecil maupun besar tidak memberikan hasil yang lebih baik. Rata-rata *f-measure* tersebut dipengaruhi oleh kombinasi CR dan MR yang digunakan, dimana nilai CR dan MR yang optimal

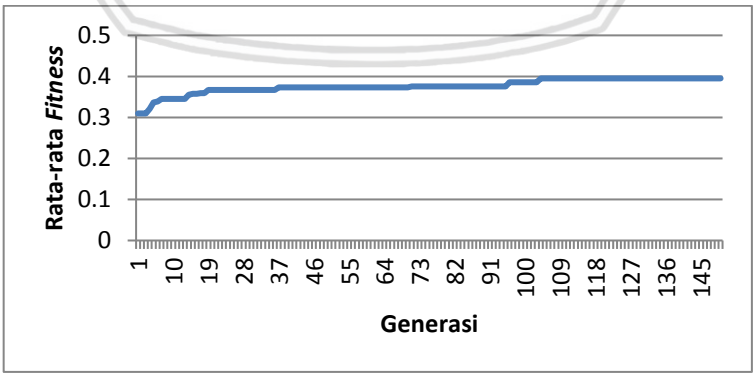
digunakan pada setiap kasus dapat berbeda karena tidak adanya ketentuan baku kombinasi nilai CR dna MR yang dapat memberikan hasil optimal. Kombinasi CR dan MR yang sama menunjukkan bahwa jumlah terbentuknya individu baru dengan tingkat keberagaman populasi yang seimbang memberikan solusi yang optimal. Hal ini disebabkan banyaknya individu baru yang dihasilkan pada kasus ini berkaitan dengan keragaman populasi yang dihasilkan untuk mencegah terjadinya konvergensi dini dalam mencari solusi, dimana besarnya tingkat terbentuknya individu baru dan keragaman populasi berbanding lurus. Sehingga, nilai CR dan MR 0,5 dianggap sebagai nilai CR dan MR yang paling baik yang akan digunakan pada pengujian banyaknya generasi.

Pengujian selanjutnya adalah pengujian banyaknya generasi. Pada pengujian ini, digunakan 85% jumlah fitur dengan 3 tetangga terdekat. Parameter algoritme genetika yang digunakan pada pengujian ini adalah ukuran populasi 10 dan nilai CR dan MR 0,5. Banyaknya generasi yang akan diuji yaitu 1 sampai 150. Hasil pengujian ini ditunjukkan pada Tabel 6.13 yang dilengkapi pada Lampiran C.

Tabel 6.13 Hasil pengujian banyaknya generasi pada data seimbang

Banyaknya generasi	Percobaan ke-			Rata-rata *Fitness*
	1	2	3	
1	0,37143	0,25784	0,3	0,309756
2	0,37143	0,25784	0,3	0,309756
3	0,37143	0,25784	0,3	0,309756
4	0,40476	0,25784	0,3	0,320867
⋮	⋮	⋮	⋮	⋮
146	0,47735	0,27717	0,43080	0,395107
147	0,47735	0,27717	0,43080	0,395107
148	0,47735	0,27717	0,43080	0,395107
149	0,47735	0,27717	0,43080	0,395107
150	0,47735	0,27717	0,43080	0,395107

Hasil pengujian pada Tabel 6.13 apabila divisualisasikan kedalam bentuk grafik, maka akan didapatkan hasil grafik seperti pada Gambar 6.10.



Gambar 6.10 Grafik hasil pengujian banyaknya generasi

Berdasarkan pada Gambar 6.12 rata-rata *fitness* tertinggi terjadi ketika mencapai generasi ke 102. Rata-rata *fitness* pada generasi ke 102 sebesar 0,39. Menggunakan generasi yang lebih banyak tidak memberikan hasil yang lebih

baik, dimana rata-rata *fitness* yang dihasilkan sama besar. Selain itu, grafik tersebut menunjukkan bahwa semakin besar maksimum generasi yang digunakan rata-rata *f-measure* yang dihasilkan cenderung lebih besar, karena semakin besar jumlah generasi yang digunakan semakin besar pula kesempatan dalam memperoleh hasil terbaik. Hasil tersebut menjadikan 102 menjadi maksimum generasi yang baik untuk digunakan pada proses *genetic algorithm*. Hasil klasifikasi program menggunakan nilai parameter hasil pngujian ditunjukkan pada Tabel 6.14 yang dilengkapi pada lampiran D.

Tabel 6.14 Hasil klasifikasi program pada data uji data seimbang

No	Aduan	Hasil Program
1	Selamat pagi bapak/ ibu, mohon perhatian untuk lampu penerangan jalan di sepanjang jalan M.T. Haryono yang tidak berfungsi (mati). Khususnya dari arah M. Panjaitan sepanjang Universitas Brawijaya, dan di depan Polsek Lowokwaru hingga daerah depan Mall Dinoyo. Saat malam hari, penerangan hanya berasal dari penerangan toko di sepanjang jalan. Apalagi akhir-akhir ini musim hujan sangat membahayakan para pengendara kendaraan dan pengguna jalan lainnya karena kurangnya penerangan jalan. Terima kasih atas perhatiannya.	DKP
2	Pak Walikota Yth, Assalamu'alaikum Wr Wb, Saya, Kuntadi Haribowo, SH adalah Putra Malang Asli yang lahir dan besar di Malang lebih dari 47 tahun lalu. Masa kecil saya dulu adalah di jalan Guntur No 25, Kakek saya adalah Mantan Kepala Dinas Kesehatan Kota Malang, yaitu DR. R. Soepangkat, Arts. Sebagai Putra Malang Asli saya merasa ikut memiliki Kota Malang ini, dan saya ingin menyampaikan beberapa saran dan uneg2 saya kepada Bp. Walikota Malang, sebagai berikut : 1. Saya merasa sangat kecewa dengan penataan Taman Malabar yang saat ini ditutup seng sehingga berkesan kumuh sekali, padahal lokasinya tepat di pusat kota. Sangat berbeda dengan jaman tahun 70 dan 80 an yang masih terlihat asri dan indah dipandang. Tolong segera di kembalikan seperti dahulu lagi. 2. Saya melihat marka jalan di banyak jalan utama di kota Malang yang tidak ada atau tidak sesuai sebagaimana mestinya (tidak ditengah). Saya minta tolong segera dibenahi. 3. Saya melihat banyak sekali jalan yang lampu penerangan jalannya masih sangat kurang, sehingga jalan menjadi gelap karena kurang cahaya diwaktu malam hari. 4. Saat ini banyak sekali berseliweran ditengah kota BECAK MOTOR yang tentunya tidak sesuai dengan ijinnya, MOHON DITERTIBKAN. 5. Banyak sekali pengendara sepeda motor yang ugal2an dalam berkendara, tidak menggunakan helm dan saykin juga banyak yang tidak punya SIM. Khususnya daerah - daerah kampus, Dinoyo, Telogomas, Galunggung, Soekarno Hatta dll, Mohon dikoordinasikan dengan Kapolresta untuk diperbanyak operasi / razia lalu lintas. 6. Mohon ditugaskan seluruh jajaran Satpol PP untuk sesering mungkin mengadakan operasi razia MIRAS dan prostitusi. 7. Mohon agar ditertibkan parkir liar yang ada di seluruh wilayah Malang. Demikian saran dan usulan ini saya sampaikan, mohon untuk segera bisa ditindaklanjuti dengan baik. Saya sebagai Putra Malang Asli ingin melihat kota Malang yang lebih baik, lebih tertib di masa yang akan datang, terima kasih. Wassalamu'alaikum, Hormat saya, Kuntadi Haribowo, SH.	DPUPPB
3	Assalamu'alaikum wr.wb. Yth. Pak Walikota Malang, , untuk keamanan dan kenyamanan pejalan kaki / wisatawan , apakah pemkota malang	DKP

Tabel 6.14 Hasil klasifikasi program pada data uji data seimbang

No	Aduan	Hasil Program
	mempunyai rencana program perbaikan fasilitas untuk pejalan kaki :1) Kayutangan - Alun-alun - Pecinan; 2). Alun-alun bunder - Kahuripan - Semeru - Ijen; 3). jalan. Kawi - Alun-alun. ? Fasilitas untuk pekalan kaki jangan/tidak boleh untuk parkir mobil dan sepeda motor ! . Terima kasih, wassalam	
⋮	⋮	⋮
31	Traffic Light Pertigaan Janti Mati sudah beberapa hari ini, harap segera di prioritaskan perbaikannya	DPUPPB
32	MOHON SEGERA DIPERBAIKI TRAFFIC LIGHT PERTIGAAN SUKUN-JANTI MATI SUDAH 1 MINGGU YG LALU, JANGAN TUNGGU KORBAN!	DPUPPB
33	sebelumnya Terimakasih atas respon cepat permasalahan TL Janti, menyampaikan pula kondisi di Jalan S.Supriadi tepatnya di depan Pabrik Es Ngaglik, kalau malam di pinggir jalan banyak dijadikan parkir liar mobil-mobil dan hal ini menjadikan sempit jalan, mohon Dishub dapat bertindak tegas, kedua kondisi di Jalan Jakarta banyak yg berlubang dan sangat membahayakan pengguna jalan, terimakasih	DPUPPB

Berdasarkan pengujian yang telah dilakukan baik terhadap data yang jumlahnya tidak seimbang maupun data yang jumlahnya seimbang, Gambar 6.1 dan Gambar 6.6 mengenai pengujian terhadap persentase jumlah fitur menunjukkan bahwa *information gain* lebih banyak mereduksi fitur pada data seimbang. Jumlah fitur yang digunakan pada data tidak seimbang yaitu sebesar 558 fitur, sedangkan pada data seimbang jumlah fitur terbaik yang digunakan adalah sebesar 806. Hal ini dikarenakan jumlah data latih yang digunakan pada data tidak seimbang lebih banyak, sehingga jumlah fitur yang didapatkan lebih banyak. Dikarenakan jumlah fitur yang lebih banyak, maka mempengaruhi hasil pengurutan fitur menggunakan *information gain* yang juga mempengaruhi hasil klasifikasi yang dihasilkan.

Rata-rata *f-measure* yang dihasilkan dengan menggunakan 30% jumlah fitur sebesar 0,2, sedangkan apabila menggunakan 100% jumlah fitur menghasilkan rata-rata *f-measure* sebesar 0,18 pada data tidak seimbang. Pada data seimbang, ratarata *f-measure* yang dihasilkan menggunakan 85% jumlah fitur sebesar 0,19 sedangkan menggunakan 100% jumlah fitur menghasilkan rata-rata *f-mesure* sebesar 0,17. Hasil tersebut menunjukkan bahwa dengan menggunakan seleksi *information gain* meningkatkan rata-rata *f-measure* yang dihasilkan, dimana hasil ini sesuai dengan hasil penelitian yang telah dilakukan Sari B. N. (2016). Hal ini juga terjadi pada hasil pengujian nilai K yang ditunjukkan pada Gambar 6.2 dan 6.8. Pada gambar tersebut rata-rata *f-measure* yang dihasilkan pada data tidak seimbang cenderung lebih besar dibandingkan rata-rata *f-measure* yang dihasilkan pada data seimbang. Hal ini terjadi karena terdapat penyusutan data, dimana jumlah data yang digunakan pada pengujian data seimbang lebih sedikit dibandingkan jumlah data yang digunakan pada pengujian data tidak seimbang. Hal tersebut menunjukkan bahwa jumlah data yang digunakan mempengaruhi besarnya rata-rata *f-measure* yang dihasilkan.

Pada pengujian parameter algoritme genetika konvergensi yang dicapai pada data yang jumlahnya tidak seimbang lebih lama dibandingkan pada data yang jumlahnya seimbang karena perbedaan jumlah data yang signifikan sehingga mempengaruhi lamanya proses untuk mencapai hasil yang optimal. Hasil dari proses algoritme genetika menghasilkan rata-rata fitness sebesar 0,22 pada data seimbang dan 0,39 pada data seimbang. Hasil tersebut lebih besar dibandingkan tanpa menggunakan seleksi fitur (100% jumlah fitur) yaitu mengalami peningkatan sebesar 0,04 untuk data seimbang dan 0,22 untuk data seimbang dari hasil klasifikasi tanpa menggunakan seleksi fitur. Apabila dibandingkan dengan hasil klasifikasi yang hanya menggunakan seleksi fitur *information gain* mengalami peningkatan sebesar 0,02 untuk data tidak seimbang dan 0,19 untuk data seimbang. Hal ini menunjukkan bahwa proses klasifikasi dengan menggunakan 2 tahap seleksi yaitu *information gain* dan *genetic algorithm* dapat menghasilkan hasil klasifikasi yang lebih baik dan dapat meningkatkan rata-rata *f-measure* lebih besar dibandingkan hanya dengan menggunakan seleksi fitur *information gain* saja. Hal tersebut sesuai dengan hasil penelitian yang telah dilakukan oleh Uguz (2011).

Rata-rata *f-measure* yang dihasilkan dari proses klasifikasi menggunakan NW-KNN terbilang kecil karena nilai *precision* dan nilai *recall* yang rendah. Rendahnya nilai *precision* ini sendiri dikarenakan banyaknya jumlah dokumen yang tidak relevan. Semakin banyak jumlah dokumen yang tidak relevan, maka nilai precision yang dihasilkan semakin kecil. Nilai *recall* yang kecil dikarenakan sedikitnya jumlah *true positive* atau jumlah dokumen yang sesuai dengan target klasifikasi. Sedikitnya jumlah dokumen yang relevan dikarenakan fitur yang digunakan masih belum dapat membedakan antar kelas secara sempurna sehingga dalam proses pengklasifikasiannya tidak memberikan hasil yang cukup bagus.

Hasil penelitian ini telah menunjukkan bahwa penggunaan seleksi fitur dapat meningkatkan hasil klasifikasi yaitu sebesar 0,04 pada data yang jumlahnya tidak seimbang dan 0,22 pada data yang jumlahnya seimbang dibandingkan tanpa menggunakan seleksi fitur. Dari hasil peningkatan hasil klasifikasi tersebut, *information gain* dan *genetic algorithm* dapat digunakan sebagai teknik seleksi fitur yang dilakukan sebelum dilakukannya proses klasifikasi. Hal ini dikarenakan fitur yang digunakan berpengaruh terhadap hasil klasifikasi yang dihasilkan. Dengan menggunakan seleksi fitur, fitur yang tidak relevan dengan kelas yang terlibat dapat diabaikan, sehingga hanya fitur yang dianggap relevan, yang dapat membedakan setiap kelas, yang digunakan dalam proses klasifikasi, sehingga hasil klasifikasi yang dihasilkan dapat meningkat.

BAB 7 PENUTUP

7.1 Kesimpulan

Dari hasil penelitian yang telah dilakukan mengenai pengklasifikasian aduan masyarakat Kota Malang menggunakan metode NW-KNN dan seleksi fitur *information gain* dan *genetic algorithm* dapat disimpulkan bahwa:

1. Persentase jumlah fitur 30% pada data tidak seimbang serta persentase jumlah fitur 85% pada data seimbang menghasilkan rata-rata *f-measure* yang lebih tinggi dibandingkan dengan menggunakan nilai K dan persentase jumlah fitur yang lain. Selain itu klasifikasi aduan masyarakat menggunakan metode NW-KNN dapat mengatasi permasalahan keseimbangan data, dimana rata-rata *f-measure* yang dihasilkan mengalami peningkatan. Pada nilai K = 3 NW-KNN menghasilkan rata-rata *f-measure* sebesar 0,2 pada data tidak seimbang dan 0,19 pada data seimbang, sedangkan pada KNN rata-rata *f-measure* yang dihasilkan sebesar 0,18 pada data tidak seimbang, dan 0,17 pada data seimbang. Hasil tersebut telah mengalami peningkatan hingga 0,02 baik untuk data tidak seimbang maupun data seimbang. Hal ini menunjukkan bahwa mengklasifikasikan aduan masyarakat menggunakan NW-KNN dengan seleksi fitur *information gain* dapat meningkatkan rata-rata *f-measure* yang dihasilkan dari proses klasifikasi.
2. Kombinasi fitur dari hasil seleksi menggunakan *information gain* mengahsilkan rata-rata *fitness* tertinggi yaitu sebesar 0,22 pada data tidak seimbang dan 0,39 pada data seimbang dengan parameter algoritme genetika optimal yang digunakan yaitu ukuran populasi 10, CR=0,5, MR=0,5 dan maksimum generasi 134 pada data tidak seimbang dan 102 pada data seimbang. Hasil tersebut telah mengalami peningkatan 0,02 pada data seimbang dan 0,2 pada data seimbang dari hasil klasifikasi menggunakan *information gain*. Hal ini menunjukkan bahwa mengklasifikasikan aduan masyarakat menggunakan NW-KNN dan seleksi fitur *information gain – genetic algorithm* menghasilkan rata-rata *f-measure* yang lebih tinggi dibandingkan tanpa menggunakan *genetic algorithm*.

7.2 Saran

Dari penelitian yang telah dilakukan, terdapat beberapa hal yang dapat dilakukan untuk mengembangkan penelitian ini, yaitu:

1. Diperlukannya penelitian dengan menggunakan teknik seleksi fitur yang lain seperti *chi-square* yang kemudian diseleksi lagi menggunakan algoritme optimasi baik *genetic algorithm, particle swarm optimization*, dll untuk mendapatkan hasil klasifikasi yang lebih baik.
2. Perlu adanya proses untuk mengubah kata singkatan dan kata non-baku menjadi kata baku agar proses *pre-processing* dapat dilakukan secara maksimal.

DAFTAR PUSTAKA

Abadi, D. 2013. *Perbandingan Algoritme Feature Selection Information Gain dan Symmetrical Uncertainty pada Data Ketahanan Pangan*. Departemen Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Pertanian Bogor. [Diakses 10 Mei 2017]

Anon. 2016. *SlideShare*. [*online*] Tersedia di: <https://www.slideshare.net/mobile/LalSaid/08-dimensionality-reduction1> [Diakses 4 April 2018]

Asian, J. 2007. *Effective Techniques for Indonesian Text Retrieval*. *School of Computer Science and Information Technology, RMIT University*. [Diakses 11 Mei 2017]

BariCkly, F. 2013. *SCRIBD*. [*online*] Tersedia di: <https://www.scribd.com/document/171292118/ALGORITME-GENETIKA> [Diakses 4 April 2018]

Darmawan, A. 2015. *Penerapan Model Support Vector Machine Text Mining pada Komentar Review Smartphone Android Vs Blackberry dengan Teknik Optimasi Genetic Algorithm*. Faktor Exacta 8(2), 100-115. [Diakses 30 November 2017]

Fitri, A. & Mahmudy, W. F. 2017. *Optimasi Kenggotaan Fuzzy Tsukamoto Menggunakan Algoritme Genetika pada Penentuan Priorotas Penerima Zakat*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol. 1, No.2, 125-138. [Diakses 3 April 2018]

Handoyo, R., M, R. R., & Nasution, S. M. 2014. *Perbandingan Metode Clustering Menggunakan Metode Single Linkage dan K-Means pada Pengelompokan Dokumen*. JSM STMIK Mikroskil Vol. 15, No. 2, 73-82. [Diakses 11 Mei 2017]

Hariri, F. R., Utami, E., & Amborowati, A. 2015. *Learning Vector Quantization untuk Klasifikasi Abstrak Tesis*. *Citec Journal*, Vol. 2, No. 2, 129-143. [Diakses 9 Mei 2017]

Indriati, & Ridok, A. 2016. *Sentiment Analysis For Review Mobile Applications Using Neighbor Method Weighted K-Nearest Neighbor (NW-KNN).Journal of Environmental Engineering & Sustainable Technology* Vol. 03 No. 01, 23-32. [Diakses 22 Desember 2017]

Kustiyahningsih, Y., & Syafa'ah, N. 2015. *Sistem Pendukung Keputusan untuk Menentukan Jurusan pada Siswa SMA Menggunakan Metode KNN dan Smart*. Jurnal Sistem Informasi Indonesia Vol. 1 No. 1, 19-28. [Diakses 21 Desember 2017]

Liyantanto, R. 2011. *Kata Dasar Bahasa Indonesia*. Kompasiana Beyond Blogging, [*online*] Tersedia di: <https://www.kompasiana.com/liyantanto/kata-

dasar-bahasa-indonesia_5500d0d8a333115b74511d03> [DIakses 20 Maret 2017]

Mahmudy, W. F. 2015. *Dasar-dasar Algoritme Evolusi*. Malang: Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. [Diakses 6 April 2018]

Maulida, I., Suyatno, A., & Hatta, H. R. 2016. *Seleksi Fitur pada Dokumen Abstrak Teks Bahasa Indonesia Menggunakan Metode Information Gain*. JSM STMIK Mikroskil Vol. 17, No. 2, 249-258. [Diakses 9 Mei 2017]

Mu'asyaroh, F. L & Mahmudy, W. F. 2016. *Implementasi Algoritme Genetika dalam Optimasi Model AHP dan TOPSIS untuk Penentuan Kelayakan Pengisian Bibit Ayam Broiler di Kandang Peternak*. Jurnal Teknologi Informasi dan Ilmu Komputer, Vol. 3, No. 4, 208-219. [Diakses 3 April 2018]

Muthia, D. A. 2016. *Opinion Mining pada Review Buku Menggunakan Algoritme Naive Bayes*. Jurnal Teknik Komputer AMIK BSI, Vol. II, No. 1, 1-8. [Diakses 29 November 2017]

Nazief, B. A., & Adriani, M. 1996. *Confix-Stripping: Approach to Stemming Algorithm for Bahasa Indonesia*. Laporan Penelitian, Fakultas Ilmu Komputer, Universitas Indonesia.

Patel, H., & Thakur, G. S. 2016. *Classification of Imbalanced Data Using a Modified Fuzzy-Neighbor Weighted*. *International Journal of Intelligent Engineering & Systems*, 56-64. [Diakses 22 Desember 2017]

Pramudiono, I. 2003. *Pengantar Data Mining: Menambang Permata Pengetahuan di Gunung Data*. Kuliah Umum IlmuKomputer.Com. [Diakses 9 Mei 2017]

Pramudita. 2014. *Penerapan Algoritme Stemming Nazief & Adriani dan Similarity pada Penerimaan Judul Thesis*. Jurnal Ilmiah DASI, Vol. 15, No. 04, 15-19. [Diakses 26 Maret 2018].

Prasanti, A. A., Fauzi, M. A., & Furqon, M. T. 2018. *Klasifikasi Teks Pengaduan pada Sambat Online Menggunakan Metode N-Gram dan Neighbor Weighted K-Nearest Neighbor (NW-KNN)*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer Vol. 2 No. 2, 594-601. [Diakses 6 Desember 2017]

Riany, J., Fajar, M., & Lukman, M. P. 2016. *Penerapan Deep Sentement Analysis pada Angket Penilaian Terbuka Menggunakan K-Nearest Neighbor*. Jurnal Sisfo Vol. 06, No. 01, 147-156. [Diakses 21 Desember 2017]

Ridok, A., & Latifah, R. 2015. *Klasifikasi Teks Bahasa Indonesia pada Corpus Tak Seimbang Menggunakan NWKNN*. Konferensi Nasional Sistem & Informatika 2015, STMIK STIKOM, 222-227. [Diakses 28 Desember 2017]

Sari, B. N. 2016. *Implementasi Teknik Seleksi Fitur Information Gain pada Algoritme Klasifikasi Machine Learning untuk Prediksi Performa Akademik Siswa*. Seminar Nasional Teknologi Informasi dan Multimedia 2016. [Diakses 9 Mei 2017]

Sari, N. H., Fauzi, M. A., & Adikara, P. P. 2018. *Klasifikasi Dokumen Sambat Online Menggunakan Metode K-Nearest Neighbor dan Features Selection Berbasis Categorical Proportional Difference*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer Vol. 2 No. 8, 2449-2454. [Diakses 6 Desember 2017]

Sari, Y. A., & Puspanigrum, E. Y. 2013. *Pencarian Semantic Dokumen Berita Menggunakan Essential Dimension of Latent Semantic Indexing dengan Memakai Reduksi Fitur Document Frequency dan Information Gain Thresholding*. Seminar Nasional Teknologi Informasi dan Multimedia 2013. [Diakses 7 Maret 2017]

Somantri, O., & Khambali, M. 2017. *Feature Selection Klasifikasi Kategori Cerita Pendek Menggunakan Naive Bayes dan Algoritme Genetika*. JNTETI, Vol. 6, No. 3, 301-306. [Diakses 2 Desember 2017]

Suharno, C. F., Fauzi, M. A., &Perdana, R. S. 2017. *Klasifikasi Teks Bahasa Indonesia pada Dokumen Pengaduan Sambat Online Menggunakan Metode K-Nearest Neighbor dan Chi-Square*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 1000-1007. [Diakses 6 Desember 2017]

Sulistiyorini, R. & Mahmudy, W. F. 2015. *Penerapan Algoritme Genetika untuk Permasalahan Optimasi DIstribusi Barang Dua Tahap*. DORO: *Repository* Jurnal Mahasiswa PTIIK Universitas Brawijaya Vol. 5 No. 12, 1-12. [Diakses 3 April 2018]

Tala, F. Z. 2003. *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*. *Master of Logic Project. Institute for Logic, Language and Computation*, *Universiteit van Amsterdam*, *The Netherlands*, 30. [Diakses 19 Maret 2018]

Taufik, A. 2017. *Optimasi PSO sebagai Seleksi FItur pada Analisis Sentimen Review Hotel Berbahasa Indonesia Menggunakan Naive Bayes*. Jurnal Teknik Komputer, Vol. III, No.2. [Diakses 2 Desember 2017]

Uguz, H. 2011. *A Two-Stage Feature Selection Method For Text Categorization By Using Information Gain, Principal Component Analysis And Genetic Algorithm*. *Knowledge-Based Systems 24*, 1024-1032. [Diakses 1 Desember 2017]

Wati, R. 2016. *Penerapan Algoritme Genetika untuk Seleksi Fitur pada Analisis Sentimen Review Jasa Maskapai Penerbangan Menggunakan Naive Bayes*. Jurnal Evolusi Volume 4 Nomor 1, 26-32. [Diakses 29 November 2017]