IMPLEMENTASI METODE *IMPROVED K-MEANS* UNTUK MENGELOMPOKKAN DOKUMEN JURNAL PENGEMBANGAN TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Abdurasyid
NIM: 135150200111057

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

PENGESAHAN

IMPLEMENTASI METODE *IMPROVED K-MEANS* UNTUK MENGELOMPOKKAN DOKUMEN JURNAL PENGEMBANGAN TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Abdurasyid
NIM: 135150200111057

Skripsi ini telah diuji dan dinyatakan lulus pada
11 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Indriati, S.T., M.Kom.	Rizal Setya Perdana, S.Kom., M.Kom.
NIP: 19831013 201504 2 002	NIK: 201603 910118 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

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

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

Malang, 11 Januari 2018

Muhammad Abdurasyid

NIM: 135150200111057

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Implementasi Metode *Improved K-Means* untuk Mengelompokkan Dokumen Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer" sebagai salah satu syarat untuk mendapatkan gelar Sarjana Komputer pada Fakultas Ilmu Komputer Universitas Brawijaya.

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

1. Ibu Indriati, S.T., M.Kom. dan Bapak Rizal Setya Perdana, S.Kom., M.Kom. selaku Pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini,
2. Bapak Agus Wahyu Widodo, S.T., M.Cs. selaku Ketua Program Studi Teknik Informatika,
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika,
4. Bapak Agi Putra Kharisma, S.T., M.T. selaku dosen Penasehat Akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi,
5. Papa dan Mama, dan seluruh keluarga besar atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta yang senantiasa tiada henti-hentinya memberikan doa dan semangat demi terselesaikannya skripsi ini,
6. Seluruh civitas akademik Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Fakultas Ilmu Komputer Universitas Brawijaya dan selama penyelesaian skripsi ini.

Penulis menyadari bahwa skripsi ini jauh dari kata sempurna, untuk itu diharapkan adanya saran dan kritik yang membangun bagi penulis. Semoga skripsi ini dapat memberikan sumbangan pikiran yang berguna bagi fakultas, pengembangan ilmu dan bermanfaat bagi masyarakat.

Malang, 11 Januari 2018

Penulis

muhammadabdurasyid@gmail.com

ABSTRAK

Muhammad Abdurasyid, Implementasi Metode *Improved K-Means* untuk Mengelompokkan Dokumen Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer

Pembimbing: Indriati, S.T., M.Kom. dan Rizal Setya Perdana, S.Kom., M.Kom.

Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) merupakan jurnal keilmuan di bidang komputer yang memuat tulisan ilmiah hasil penelitian mahasiswa/i FILKOM Universitas Brawijaya yang diterbitkan secara berkala. J-PTIIK merupakan sebuah dokumen jurnal yang memiliki topik jurnal yang berada dalam bidang teknologi informasi dan ilmu komputer. Pada saat ini J-PTIIK dikelompokkan berdasarkan arsip *volume* dan nomor terbit jurnal. Untuk memudahkan identifikasi topik jurnal yang terdapat pada J-PTIIK, maka dokumen J-PTIIK dapat dikelompokkan berdasarkan kemiripan topik yang terdapat dalam J-PTIIK. Pengelompokan dokumen J-PTIIK dibuat dengan menggunakan metode *improved k-means*. Metode *improved k-means* merupakan salah satu teknik klasterisasi *unsupervised* dengan penentuan *centroid* awal klaster diperoleh dengan cara menggabungkan metode optimasi jarak dan densitas. Praproses dokumen dan pembentukan *vector space model* untuk melakukan pembobotan kata dilakukan terlebih dahulu sebelum mengelompokkan dokumen J-PTIIK dengan menggunakan metode *improved k-means*. Berdasarkan hasil pengujian, pengelompokan dokumen J-PTIIK memperoleh hasil *silhouette coefficient* optimal sebesar 0,026574 pada k = 19 dan α = 0,50. Hasil pengujian *purity* optimal diperoleh sebesar 0,738197 pada k = 23 dan α = 0,50. Hasil penelitian menunjukkan penggunaan metode *improved k-means* memiliki *silhouette coefficient* yang lebih baik dibandingkan metode *k-means,* dengan nilai rata-rata *silhouette coefficient* pada metode *improved k-means* sebesar 0,016457654 dan metode *k-means* sebesar 0,011820563.

Kata kunci: praproses teks, *vector space model*, pembobotan kata, klasterisasi, *improved k-means*

ABSTRACT

Journal of Information Technology and Computer Science Development (J-PTIIK) is a scientific journal in the field of computer that contains scientific writings of research results FILKOM Brawijaya University students that published periodically. J-PTIIK is a journal document that has journal topics that are in the field of information technology and computer science. At this time J-PTIIK is clustered by volume archive and published journal number. To facilitate the identification of journal topics contained in J-PTIIK, J-PTIIK documents can be clustered based on similarity of topics contained in J-PTIIK. J-PTIIK documents clustering is made using improved k-means method. The improved k-means method is one of the unsupervised clustering techniques with the initial centroid determination obtained by combining the optimization method of distance and density. Document pre-processing and formation of vector space model to perform term weighting is done first before clustering the J-PTIIK documents using improved k-means method. Based on the evaluation results, J-PTIIK documents clustering obtained an optimal silhouette coefficient by 0.026574 at k = 19 and α = 0.50. Optimal purity test results obtained by 0.738197 at k = 23 and α = 0.50. The research result shows that the use of improved k-means method has better silhouette coefficient than k-means method, with average value of silhouette coefficient at improved k-means method by 0.016457654 and k-means method by 0.011820563.

Keywords: text pre-processing, vector space model, term weighting, clustering, improved k-means

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCE CODE

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Menurut Hakim (2012) jurnal ilmiah adalah "majalah publikasi yang memuat KTI (Karya Tulis Ilmiah) yang secara nyata mengandung data dan informasi yang mengajukan iptek dan ditulis sesuai dengan kaidah-kaidah penulisan ilmiah serta diterbitkan secara berkala". Di Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya terdapat publikasi jurnal ilmiah yang dikenal dengan nama Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer atau yang biasa disingkat J-PTIIK. J-PTIIK merupakan jurnal keilmuan di bidang komputer yang memuat tulisan ilmiah hasil penelitian mahasiswa/i FILKOM Universitas Brawijaya, dimana hasil penelitian berasal dari keminatan program studi FILKOM Universitas Brawijaya.

J-PTIIK dapat diakses secara *online* dengan menggunakan *web browser*. J-PTIIK pada saat ini dikelompokkan berdasarkan arsip *volume* dan nomor terbit jurnal. J-PTIIK yang merupakan sebuah dokumen jurnal memiliki topik jurnal yang berada dalam bidang teknologi informasi dan ilmu komputer. Untuk memudahkan identifikasi topik jurnal yang terdapat pada J-PTIIK, maka dokumen J-PTIIK dapat dikelompokkan berdasarkan kemiripan topik yang terdapat dalam J-PTIIK. Pengelompokan dokumen J-PTIIK dapat dilakukan dengan menggunakan teknik klasterisasi agar dokumen J-PTIIK dikelompokkan berdasarkan kemiripan objek topik antar dokumen J-PTIIK di dalam suatu klaster.

Klasterisasi merupakan salah satu teknik yang efisien digunakan pada *data mining* dalam melakukan pengelompokan objek ke dalam kelas yang sama berdasarkan tingkat kemiripan antar objek dalam kelas. Teknik klasterisasi terdiri dari dua model, antara lain model hierarki dan model partisi. Salah satu algoritme klasterisasi yang banyak digunakan pada model partisi adalah metode *k-means* (Reddy & Jana, 2012). Metode *k-means* merupakan algoritme yang cukup sederhana dan termasuk ke dalam teknik klasterisasi *unsupervised* (Karimov & Ozbayoglu, 2015).

Metode *k-means* membandingkan nilai jarak terhadap *centroid* pada masing-masing klaster yaitu nilai rata-rata *centroid* klaster (Chayangkoon & Srivihok, 2016). Metode *k-means* memiliki beberapa batasan masalah dalam prosesnya. Salah satu batasan masalah yang terdapat dalam metode *k-means* adalah pemilihan *centroid* awal yang dilakukan secara acak. Pemilihan *centroid* awal secara acak ini dapat menghasilkan kualitas klaster yang tidak baik (Rahman, Islam, & Bossomaier, 2015). Pemilihan *centroid* awal secara acak ini juga menyebabkan hasil klaster yang selalu berbeda pada tiap proses pengelompokan (Karimov & Ozbayoglu, 2015).

Oleh karena batasan masalah tersebut, metode *improved k-means* digunakan sebagai solusi untuk memilih *centroid* awal yang akan digunakan dalam proses *k-means*. Dalam prosesnya metode *improved k-means* terlebih dahulu mencari *centroid* awal klaster, hal ini berbeda dengan metode *k-means* yang menentukan

centroid awal secara acak. Hasil evaluasi dari metode *improved k-means* memiliki kualitas klaster yang lebih tinggi berdasarkan pengukuran jarak *intra* klaster dan *inter* klaster bila dibandingkan dengan *k-means* (Poomagal & Hamsapriya, 2011). Hasil evaluasi metode *improved k-means* juga memiliki akurasi yang lebih baik bila dibandingkan dengan menggunakan metode *k-means* (Sutariya & Amin, 2013). Hasil evaluasi metode *improved k-means* juga memiliki nilai presisi dan *recall* yang lebih tinggi bila dibandingkan dengan metode *k-means* (Xiong, et al., 2016).

Dalam penelitian sebelumnya yang dilakukan oleh Poomagal dan Hamsapriya (2011), pemilihan *centroid* awal pada *improved k-means* dilakukan dengan menghitung nilai *midpoint* menggunakan metode *scale factor*. Hasil evaluasi dalam penelitian menunjukkan bahwa metode *improved k-means* menghasilkan kualitas klaster yang tinggi. Sementara pada penelitian yang dilakukan oleh Shen dan Meng (2012), pemilihan *centroid* awal *k-means* dilakukan berdasarkan *small world network.* Hasil evaluasi menunjukkan nilai total kohesi sebesar 162,56, *purity* dengan nilai 85,14 %, dan *recall* dengan nilai 70,5 %. Dalam penelitian lain yang dilakukan oleh Shoolihah, Furqon, dan Widodo (2017), pemilihan *centroid* awal *improved k-means* dilakukan dengan cara memilih jarak terbesar antar tiap data sebagai *centroid* awal klaster. Hasil evaluasi menunjukkan bahwa nilai *silhouette coefficient* memiliki nilai tertinggi dengan jumlah klaster 2.

Berdasarkan uraian yang telah dijelaskan di atas, maka penulis mengajukan penelitian yang memiliki luaran berupa sebuah sistem pengelompokan dokumen J-PTIIK dengan menggunakan metode *improved k-means*.

1.2 Rumusan Masalah

Berdasarkan latar belakang permasalahan yang telah diuraikan di atas, maka diperoleh rumusan masalah penelitian sebagai berikut:

1. Bagaimana mengimplementasikan metode *improved k-means* untuk mengelompokkan dokumen J-PTIIK?
2. Bagaimana hasil evaluasi dengan menggunakan *silhouette coefficient* dan *purity* terhadap hasil pengelompokan dokumen J-PTIIK yang menggunakan metode *improved k-means*?

1.3 Tujuan

Sesuai dengan latar belakang dan rumusan masalah yang telah diuraikan di atas, maka penelitian ini memiliki tujuan sebagai berikut:

1. Mengimplementasikan metode *improved k-means* untuk mengelompokkan dokumen J-PTIIK.
2. Mengetahui hasil evaluasi dengan menggunakan *silhouette coefficient* dan *purity* terhadap hasil pengelompokan dokumen J-PTIIK yang menggunakan metode *improved k-means*.

1.4 Manfaat

Dengan tercapainya tujuan dalam penelitian ini, diharapkan penelitian ini dapat bermanfaat bagi pembaca dalam menambah wawasan pengetahuan teknik klasterisasi khususnya penerapan implementasi metode *improved k-means.* Penelitian ini juga diharapkan menjadi masukan bagi *admin web* J-PTIIK untuk menerapkan implementasi metode *improve k-means* dalam mengelompokkan dokumen J-PTIIK pada sistem yang sudah ada.

1.5 Batasan Masalah

Batasan-batasan masalah yang diterapkan dalam penelitian ini antara lain sebagai berikut:

1. Isi dokumen J-PTIIK yang diambil sebagai data hanya bagian judul dan abstrak karena judul dan abstrak dapat mewakili isi keseluruhan dokumen J-PTIIK.
2. Tahap *stemming* menggunakan algoritme Nazief dan Adriani dengan bantuan *library jsastrawi*.
3. Judul dan abstrak dokumen J-PTIIK diperoleh dari *web* J-PTIIK dimulai dari arsip J-PTIIK Vol 1 No 1 (2017) hingga Vol 1 No 12 (2017).
4. Jumlah judul dan abstrak yang digunakan sebanyak 233 objek data.

1.6 Sistematika Pembahasan

Dalam penyusunan laporan penelitian ini terdapat penulisan yang terstruktur sebagai berikut:

BAB 1 PENDAHULUAN

Bab 1 menguraikan masalah umum terkait penelitian yang tertulis secara bersistematis dan berkesinambungan. Penulisan pada Bab 1 dimulai dari latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Bab 2 berisi kajian pustaka terhadap penelitian-penelitian terkait yang berhubungan dengan penelitian penulis. Bab 2 juga membahas dasar teori pendukung dalam penyusunan penelitian.

BAB 3 METODOLOGI

Bab 3 menguraikan langkah-langkah secara sistematis dalam mengimplementasikan metode *improved k-means* untuk mengelompokkan dokumen J-PTIIK yang dimulai dari studi literatur, analisis kebutuhan, pengumpulan data, perancangan, implementasi, pengujian dan penarikan kesimpulan.

BAB 4 ANALISIS DAN PERANCANGAN

Bab 4 membahas tentang analisis dan perancangan yang akan dilakukan dalam mengimplementasikan metode *improved k-means* untuk mengelompokkan dokumen J-PTIIK. Bab 4 ini meliputi perancangan algoritme dalam menerapkan metode *improved k-means*.

BAB 5 IMPLEMENTASI

Bab 5 membahas tentang implementasi hasil dari perancangan algoritme yang dilakukan pada bab sebelumnya meliputi implementasi antarmuka pengguna (*user interface*) dan implementasi ke dalam bentuk bahasa pemrograman yang telah ditentukan pada bab sebelumnya.

BAB 6 PENGUJIAN

Bab 6 menguraikan tentang hasil evaluasi dari pengelompokan dokumen J-PTIIK menggunakan metode *improved k-means* dengan menggunakan teknik evaluasi *silhouette coefficient* dan *purity*.

BAB 7 PENUTUP

Bab 7 menguraikan kesimpulan yang diperoleh terhadap penelitian dan menyertakan saran yang dapat digunakan untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Dalam penelitian sebelumnya yang dilakukan oleh Poomagal dan Hamsapriya (2011), pemilihan *centroid* awal dilakukan dengan menghitung nilai *midpoint* menggunakan metode *scale factor*. *Improved k-means* digunakan untuk mengelompokkan hasil pencarian dokumen *web* dengan menggunakan fitur URL dan konten *tag*. Hasil pencarian dokumen ini terdiri dari 200 query yang diperoleh dari Yahoo, Google, dan Bing. Evaluasi dalam penelitian ini dilakukan dengan membandingkan nilai jarak *intra* klaster *dan inter* klaster pada *k-means* dan *improved k-means*. Evaluasi dalam penelitian ini menunjukkan bahwa metode *improved k-means* menghasilkan kualitas klaster yang tinggi.

Pada penelitian yang dilakukan oleh Shen dan Meng (2012), pemilihan *centroid* awal *k-means* dilakukan berdasarkan *small world network* untuk mengatasi permasalahan pemilihan *centroid* awal secara acak pada *k-means.* Pemilihan *centroid* awal dilakukan dengan cara memodelkan dokumen ke dalam sebuah jaringan yang memiliki phenomenon *small world.* Hasil karakteristik *small world* ini digunakan untuk membentuk *centroid* awal. *Dataset* dokumen diperoleh secara *online* dengan jumlah dokumen sebanyak 842 yang terdiri dari kategori *news, reading, education, science* dan technology. Hasil evaluasi dalam penelitian ini menunjukkan bahwa nilai total kohesi, *purity,* dan *recall* dalam penggunaan metode ini lebih baik bila dibandingkan pada *k*-means. Hasil evaluasi metode ini memiliki nilai total kohesi sebesar 162.56, *purity* dengan nilai 85.14 %, dan *recall* dengan nilai 70.5 %.

Pada penelitian lain yang dilakukan oleh Sutariya dan Amin (2013), pemilihan *centroid* awal klaster dalam metode *improved k-means* dilakukan dalam dua tahap. Tahap pertama yang dilakukan adalah menentukan *centroid* awal klaster yang kemudian diikuti oleh tahap kedua dalam melakukan pengelompokan. Pada tahap pertama penentuan *centroid* awal klaster dilakukan berdasarkan jarak yang diberikan nilai batas kemudian dicari nilai rata-ratanya. *Dataset* yang digunakan pada penelitian ini berasal dari UCI *repository machine learning* dimana data yang diambil adalah *Libras Movement, iris, wine quality, blood transfusion* dan *Teaching Assistant Evaluation.* Hasil evaluasi dalam penelitian ini menunjukkan bahwa metode *improved k-means* memiliki akurasi dan penggunaan waktu yang lebih baik bila dibandingkan dengan metode *k-means*. Nilai akurasi terbaik diperoleh pada *dataset iris* dengan akurasi sebesar 90 % dan waktu eksekusi terbaik pada *dataset Teaching Assistant Evaluation* dalam waktu 2 ms.

Pada penelitian lain yang dilakukan oleh Xiong, Hua, Lv dan Li (2016), pemilihan *centroid* awal klaster dilakukan dengan menggabungkan metode optimasi jarak dan densitas. Metode *improved k-means* dalam penelitian ini digunakan untuk mengelompokkan dokumen teks berbahasa Cina dasar. Data yang digunakan dalam penelitian ini berjumlah 880 teks yang terdiri dari 5 kategori, dimana data teks ini diambil dari korpus berbahasa Cina yang disediakan oleh Profesor Li Ronglu

dari Universitas Fudan. Hasil dalam penelitian ini menunjukkan bahwa metode *improved k-means* memiliki nilai presisi dan *recall* yang lebih tinggi bila dibandingkan dengan *k-means*. Dimana presisi dan *recall* tertinggi diperoleh pada data teks yang berada di kategori *Environment* dengan nilai presisi 82.71 % dan *recall* 83.75%.

Dalam penelitian lain yang dilakukan oleh Shoolihah, Furqon, dan Widodo (2017), metode *improved k-means* digunakan untuk mengelompokkan data titik panas bumi (*hotspot*). Data titik panas bumi (*hotspot*) yang digunakan dalam penelitian ini diperoleh dari web NASA LANCE-FIRMS MODIS yang diakses pada tanggal 24 Januari 2017. Pemilihan *centroid* awal dilakukan dengan cara memilih jarak terbesar antar tiap data sebagai *centroid* awal klaster. Hasil evaluasi dalam penelitian dengan jumlah data sebanyak 700 menunjukkan bahwa nilai *silhouette coefficient* tertinggi sebesar 0.908000874 pada nilai *k* sama dengan 2.

Berikut adalah penelitian-penelitian sebelumnya yang memiliki keterkaitan terhadap penelitian penulis yang dibahas secara detail meliputi judul, objek, metode dan hasil penelitian yang dapat dilihat pada Tabel 2.1.

Tabel 2.1 Perbandingan Terhadap Penelitian Terkait

No.	Judul	Objek	Metode	Hasil
1.	*Optimized K-Means Clustering with Intelligent Initial Centroid Selection for Web Search Using URL and Tag Contents* (Poomagal dan Hamsapriya, 2011)	URL, *tag* judul, dan *tag meta* dari kumpulan dokumen *web*	Proses: 1. Ekstraksi kata dari URL, *tag* judul, dan *tag meta* 2. Menghilangkan *stopword* dan melakukan *stemming* 3. Pembobotan tf.idf 4. Dari tabel pembobotan, hitung nilai terbesar masing-masing kata 5. Bagi nilai terbesar untuk masing-masing kata pada langkah sebelumnya dengan nilai *k* 6. Gunakan hasil pembagian ini untuk membentuk tabel *scale factor* yang terdiri dari *k*+1 baris 7. Hitung nilai *centroid* dengan mencari nilai rata-rata dari baris yang berdekatan 8. Lakukan *k-means* dengan menggunakan *centroid* pada langkah sebelumnya 9. Pembentukan label hasil klaster dengan mencari nilai frekuensi terbesar kata	Berdasarkan hasil dari 200 *query* dokumen *web* diperoleh bahwa nilai jarak *intra* klaster *dan inter* klaster metode *improved k-means* lebih baik dari metode *k-means*. Evaluasi menunjukkan bahwa metode *improved k-means* menghasilkan kualitas klaster yang tinggi.

Tabel 2.1 Perbandingan Terhadap Penelitian Terkait (lanjutan)

No.	Judul	Objek	Metode	Hasil
2.	*Optimization of Initial Centroids for K-Means Algorithm Based on Small World Network* (Shen & Meng, 2012)	*Dataset* dokumen berjumlah 842 dokumen yang diperoleh dari http://www.163.com (27 November 2011) dan terdiri dari 5 kategori yakni: 1. *News* 2. *Reading* 3. *Education* 4. *Science* 5. *Technology*	Proses: 1. Pembentukan *vector space model* dokumen 2. Tentukan nilai *threshold* λ untuk *cosine similarity*. Buat *link* antar dua dokumen yang memiliki nilai *cosine similarity* lebih besar dari λ dan bentuk jaringan 3. Bentuk koleksi sampel klasifikasi 4. Tentukan nilai *threshold* SNN v 5. Cari nilai derajat terbesar *node* dari koleksi sampel dan digabung kedalam M_1 6. Identifikasi *node* dengan kemiripan tetangga terdekat yang tidak kurang dari v berada dalam set sampel, tambahkan ke dalam koleksi M_1 7. Ulangi langkah 5 & 6 hingga data M_1 tidak mengalami perubahan	Hasil dari penelitian ini menunjukkan bahwa nilai *purity* meningkat. Pemilihan *centroid* awal dengan metode ini juga meningkatkan kompleksitas waktu hingga O (n^2). Hasil evaluasi: • *Total cohesion* = 162.56 • *Purity* = 85.14 % • *Recall* = 70.5 %

Tabel 2.1 Perbandingan Terhadap Penelitian Terkait (lanjutan)

No.	Judul	Objek	Metode	Hasil
			8. Hapus *node* yang berada dalam M_1 dari koleksi sampel 9. Ulangi langkah 5-8 hingga M_k terbentuk 10. Hitung pusat dari M1...Mk dan gunakan pusat ini sebagai *centroid* awal 11. Lakukan *k-means* seperti pada umumnya	
3.	*An Improvement in K-means Clustering Algorithm* (Sutariya & Amin, 2013)	Data diperoleh dari UCI *repository machine learning:* 1. *Libras Movement* 2. *Iris* 3. *Wine quality* 4. *Blood transfusion* 5. *Teaching Assistant Evaluation*	Proses: 1. Tentukan $m = 1$ 2. Hitung jarak antar tiap data dengan menggunakan *euclidean distance* 3. Temukan pasangan data terdekat dari *D* dan bentuk *Am* ($1 \leq m \leq k$) yang terdiri dari pasangan data ini. Hapus pasangan data dari *D* 4. Temukan data dalam *D* yang dekat dari *Am*. Tambahkan ke dalam *Am* dan hapus dari *D* 5. Ulangi langkah 4 hingga jumlah data di *Am* $0.75*(n/k)$ 6. Jika $m < k$, maka $m = m + 1$. Temukan pasangan lain dari *D*	Hasil evaluasi penelitian menunjukkan bahwa metode *improved k-means* memiliki akurasi dan waktu eksekusi lebih baik bila dibandingkan *k-means*. Nilai akurasi terbaik diperoleh pada *dataset iris* dengan akurasi sebesar 90 % dan waktu eksekusi terbaik pada *dataset Teaching Assistant Evaluation* dalam waktu 2 ms.

Tabel 2.1 Perbandingan Terhadap Penelitian Terkait (lanjutan)

No.	Judul	Objek	Metode	Hasil
			yang memiliki jarak terdekat, bentuk *Am* kembali dan hapus pasangan data ini dari *D.* Lakukan kembali langkah 4 7. Untuk setiap data yang berada dalam *Am* ($1 \leq m \leq k$), temukan rata-rata dari vektor data *Am*. Gunakan rata-rata ini sebagai *centroid* awal klaster 8. Lakukan *k-means* seperti biasa	
4.	*An Improved K-means text clustering algorithm By Optimizing initial cluster centers* (Xiong, et al., 2016)	Data teks berbahasa Cina dengan jumlah 880 teks yang terdiri dari 5 kategori yakni: 1. *Art* 2. *Economy* 3. *Environment* 4. *Political* 5. *Sports*	Proses: 1. Hitung jarak antar tiap data dengan menggunakan *euclidean distance* 2. Hitung rata-rata jarak 3. Hitung nilai parameter densitas data 4. Hitung rata-rata parameter densitas 5. Tentukan objek data yang terisolasi dan hapus data dari *D* sehingga menghasilkan *A* dengan densitas tertinggi	Hasil pengujian memiliki nilai presisi dan *recall* yang lebih tinggi dibandingkan *k-means*. Dimana presisi dan *recall* tertinggi diperoleh pada data teks yang berada di kategori *Environment* dengan nilai presisi 82.71 % dan *recall* 83.75%.

Tabel 2.1 Perbandingan Terhadap Penelitian Terkait (lanjutan)

No.	Judul	Objek	Metode	Hasil
			6. Pilih objek data yang memiliki densitas tertinggi dari *A* dan gunakan sebagai *centroid* awal klaster pertama. Masukkan ke *B* dan hapus dari *A* 7. Dari koleksi *A*, pilih objek data yang memiliki jarak terjauh dari objek data yang berada dalam *B* dan gunakan sebagai *centroid* awal klaster kedua. Masukkan ke *B* dan hapus dari *A* 8. Ulangi langkah 7 hingga *k* objek data berada di *B* 9. Berdasarkan *k centroid* awal, lakukan *k-means* seperti biasa	
5.	Implementasi Metode *Improved K-Means* untuk Mengelompokkan Titik Panas Bumi (Shoolihah, Furqon, & Widodo, 2017)	Data titik panas bumi (*hotspot*) diperoleh dari web NASA LANCE-FIRMS MODIS pada tanggal 24 Januari 2017	Proses: 1. Tentukan nilai *k* dan masukkan *dataset* titik panas bumi (*hotspot*) 2. Lakukan normalisasi data 3. Persiapkan klaster kosong sebanyak nilai *k* 4. Temukan data x_i dan x_j dengan jarak maksimal pada tiap data.	Hasil pengujian diperoleh nilai terbesar *silhouette coefficient* yakni sebesar 0.908000874 untuk nilai *k* 2 dan jumlah data 700. Klaster dengan nilai *k* 2 menghasilkan nilai *confidence* sebesar 100 pada klaster 2 dan 61.73986486 pada klaster 1.

Tabel 2.1 Perbandingan Terhadap Penelitian Terkait (lanjutan)

No.	Judul	Objek	Metode	Hasil
			Labeli sebagai klaster X dan Y, masukkan pada klaster pada langkah sebelumnya 5. Hitung jarak antara x_i dan x_j terhadap *dataset*. Jika jarak ke x_i minimal maka masuk klaster X dan sebaliknya jika jarak ke x_j minimal maka masuk klaster Y 6. Jika *dataset* sudah terbagi menjadi *k* klaster, proses berhenti dan ambil nilai titik pusat klaster. Jika belum maka ulangi langkah 4 7. Lakukan *k-means* seperti pada umumnya 8. Hitung evaluasi dengan menggunakan *silhouette coefficient*	

2.2 Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer

Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer atau yang biasa disingkat J-PTIIK merupakan jurnal keilmuan di bidang komputer yang memuat tulisan ilmiah hasil dari penelitian mahasiswa/i Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya. J-PTIIK memiliki luaran dalam mengembangkan penelitian dan memberikan kontribusi yang berarti dalam meningkatkan sumber daya penelitian di bidang teknologi informasi dan ilmu komputer. J-PTIIK diterbitkan oleh Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya, dimana jurnal ini merupakan hasil penelitian yang berasal dari program studi yang ada di Fakultas Ilmu Komputer (FILKOM) seperti magister ilmu komputer, teknik informatika, sistem komputer, sistem informasi, teknologi informasi dan pendidikan teknologi informasi.

J-PTIIK dapat dilihat dan diakses secara *online* melalui web J-PTIIK yang beralamat di *j-ptiik.ub.ac.id*. J-PTIIK dikelompokkan berdasarkan arsip volume dan nomor terbit, dimana volume menandakan tahun terbit jurnal sementara nomor terbit menandakan bulan terbit jurnal. J-PTIIK diterbitkan sebulan sekali oleh Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya. J-PTIIK pertama kali terbit di bulan Januari 2017 dengan kode arsip Vol 1 No 1 (2017).

J-PTIIK memiliki topik jurnal yang berada dalam bidang teknologi informasi dan ilmu komputer. Beberapa topik J-PTIIK yang terbit antara lain adalah seperti sistem pakar, *fuzzy*, algoritme genetika, pengembangan perangkat lunak, pengembangan jaringan dan topik-topik teknologi informasi dan ilmu komputer lainnya. Sehingga dokumen J-PTIIK dapat dilakukan pengelompokan berdasarkan kemiripan topik yang terdapat dalam J-PTIIK.

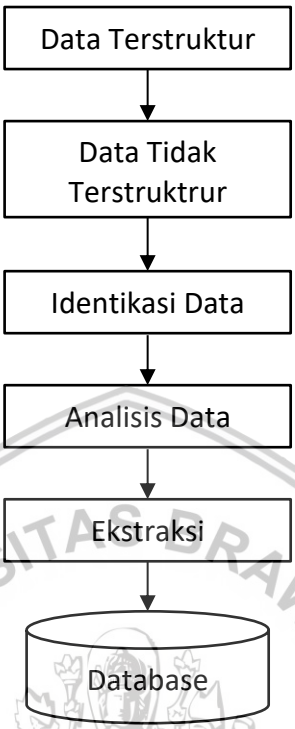
2.3 *Text Mining*

Text mining adalah suatu proses dalam menemukan dan mengekstrak informasi dari sekumpulan sumber teks yang banyak dan tidak terstruktur (KM & Reddy, 2016). Sumber teks dapat berupa *e-mail*, *chat*, SMS, artikel koran, jurnal, ulasan produk dan catatan organisasi. Teks yang tidak terstruktur ini tidak mudah diproses oleh komputer sehingga dibutuhkan beberapa teknik untuk mengekstrak beberapa informasi (Dang & Ahmad, 2014). *Text mining* biasa dilalukan untuk tujuan khusus dan informasi hasil *text mining* disimpan ke dalam sebuah *database* (Kumar & Bhatia, 2013).

Terdapat lima langkah dalam melakukan *text mining* seperti yang terdapat pada Gambar 2.1 antara lain (Dang & Ahmad, 2014):

1. Mengumpulkan informasi dari data yang tidak terstruktur
2. Mengubah informasi ke dalam data yang terstruktur
3. Mengidentifikasi pola pada data yang terstruktur
4. Menganalisis pola data

5. Mengekstrak informasi yang bernilai dan menyimpan ke dalam sebuah *database*



Gambar 2.1 Langkah-langkah *Text Mining*

Sumber: Dang & Ahmad (2014)

Penelitian dalam *text mining* merupakan pengembangan dalam beberapa teknik matematika, statistik, linguistik dan pengenalan pola yang mampu menganalisis informasi tidak terstruktur secara otomatis sehingga menghasilkan ekstraksi data yang berkualitas dan relevan. Dokumen teks terdiri dari karakter yang secara bersama membentuk suatu kata yang selanjutnya dapat membentuk frasa. *Text mining* harus mampu mengenali, mengekstrak dan menggunakan informasi ini, baik pencarian dalam bentuk kata maupun pengenalan semantik sehingga menghasilkan pencarian dalam level tertinggi (Kumar & Bhatia, 2013).

Text mining memiliki aktivitas proses secara terurut yang harus dilakukan agar perolehan informasi didapat secara efisien (Kumar & Bhatia, 2013). Proses dari *text mining* dapat dilihat pada Gambar 2.2.

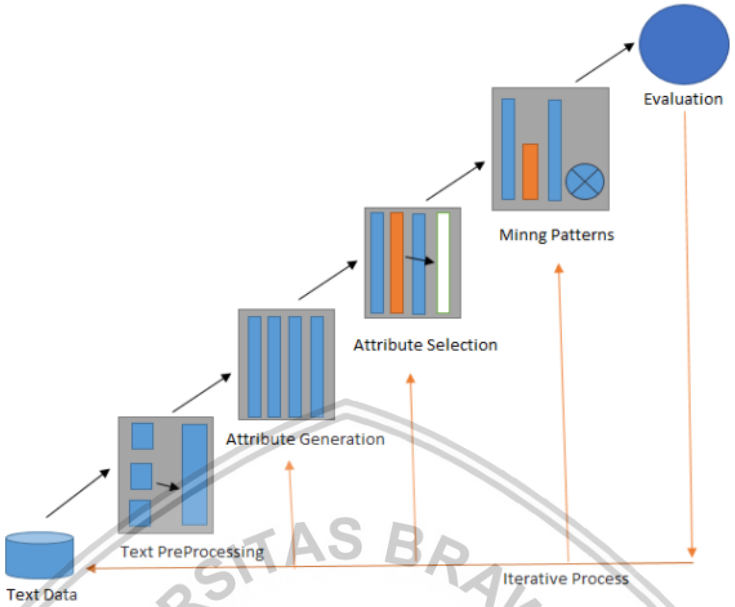
1. *Text Preprocessing*

   Text preprocessing terdiri dari proses *text cleanup* untuk membuang informasi yang tidak diinginkan, tokenisasi untuk memisahkan teks berdasarkan *white space* dan tanda baca, dan *part of speech* (POS) *tagging*.

2. *Attribute generation* (transformasi teks)

   Dalam proses transformasi teks, dokumen teks direpresentasikan dalam bentuk kata yang terdapat dalam dokumen dan jumlah kemunculannya.

Terdapat dua pendekatan untuk merepresentasikan dokumen yakni *bag of words* dan *vector space model.*



Gambar 2.2 Proses dari *Text Mining*

Sumber: Kumar & Bhatia (2013)

3. *Attribute selection* (pemilihan fitur)

   Pemilihan fitur juga dikenal dengan sebutan pemilihan variabel yang merupakan sebuah proses dalam memilih sekumpulan fitur penting yang digunakan dalam pembentukan model. Teknik pemilihan fitur merupakan sebuah bagian umum dari ekstraksi fitur.

4. *Mining patterns* (*data mining*)

   Dalam proses ini, teknik *data mining* digunakan untuk mengolah data yang telah terstruktur melalui proses sebelumnya.

5. *Evaluation*

   Proses terakhir adalah melakukan evaluasi terhadap hasil. Hasil evaluasi dapat digunakan untuk menilai kinerja dari *text mining* atau dapat digunakan sebagai *input* untuk melakukan proses *text mining* berikutnya.

Text mining mirip dengan *data mining*, bedanya adalah jika *data mining* dikhususkan untuk mengolah data terstruktur, sementara *text mining* dapat mengolah data yang tidak terstruktur dan semi terstruktur (Kumar & Bhatia, 2013). *Text mining* terdiri dari beberapa bidang seperti sistem temu kembali informasi, analisis teks, ekstraksi informasi, kategorisasi, klasterisasi, visualisasi, *data mining* dan *machine learning* (Dang & Ahmad, 2014). Berikut merupakan beberapa bidang yang terdapat dalam *text mining* antara lain:

1. Sistem temu kembali informasi

   Sistem temu kembali informasi merupakan cara dalam menemukan dokumen teks berisi informasi yang dibutuhkan dari koleksi yang cukup besar yang biasanya disimpan dalam sebuah database (A. & Aghila, 2010).

2. Ekstraksi informasi

   Ekstraksi informasi merupakan suatu proses yang mengidentifikasi kata kunci dan hubungannya yang terdapat dalam teks. Proses ini biasanya dilakukan dengan melihat urutan yang terdapat dalam teks yang biasanya disebut dengan pencocokan pola. Ekstraksi informasi mengambil kesimpulan hubungan antara tempat, orang, dan waktu yang diberikan oleh pengguna sebagai informasi berguna (Vijayarani, Ilamathi, & Nithya, 2011).

3. Kategorisasi

   Kategorisasi merupakan suatu proses yang mengidentifikasi sebuah dokumen berada di kelas mana dengan memasukkan dokumen ini ke dalam sekumpulan kelas yang telah ditentukan (Vijayarani, Ilamathi, & Nithya, 2011). Dalam pelatihan kategorisasi teks, seluruh kumpulan dokumen dilabeli berdasarkan kelasnya, sehingga kategorisasi mampu mengetahui suatu kelas dari dokumen baru (A. & Aghila, 2010).

4. Klasterisasi

   Klasterisasi dokumen atau yang biasa dikenal dengan sebutan klasterisasi teks merupakan salah satu metode penting dalam *text mining* yang dikembangkan untuk membantu pengguna dalam melakukan navigasi, peringkasan, dan mengorganisasi dokumen teks secara efektif (A. & Aghila, 2010). Klasterisasi dokumen memiliki tujuan untuk menemukan struktur intrinsik informasi dan menyusunnya ke dalam beberapa sub grup untuk penelitian dan analisis lebih lanjut. Klasterisasi dokumen termasuk sebuah proses *unsupervised* dimana objek dikelompokkan ke dalam grup yang disebut dengan klaster (Dang & Ahmad, 2014).

5. Peringkasan

   Peringkasan teks merupakan suatu proses yang secara otomatis membuat sebuah versi ringkas dari sebuah teks yang menghasilkan informasi berguna untuk pengguna. Sebuah ringkasan teks yang diperoleh dari satu atau lebih teks memiliki panjang teks yang kurang dari dokumen aslinya dan memiliki makna yang sama dengan dokumen asli. Peringkasan teks menggunakan beberapa teknik kategorisasi teks seperti jaringan syaraf, pohon keputusan, graf semantik, model regresi, logika fuzzy dan kecerdasan kelompok (Dang & Ahmad, 2014).

6. Data mining

   Data mining dapat didefinisikan sebagai mencari pola yang terdapat dalam data, biasa dilakukan untuk mengekstraksi data tersembunyi yang sebelumnya tidak diketahui dan informasi berguna dari suatu data. Secara keseluruhan

data mining memiliki tujuan untuk mengekstraksi informasi dari sekumpulan data dan mengubahnya menjadi bentuk yang terstruktur untuk kepentingan tertentu (Kumar & Bhatia, 2013).

7. Pemrosesan bahasa alami

   Pemrosesan bahasa alami merupakan sebuah area penelitian dan aplikasi yang mencari cara bagaimana komputer dapat digunakan untuk memahami dan memanipulasi teks bahasa alami. Dasar-dasar pengetahuan pemrosesan bahasa alami terletak dalam beberapa bidang yang berbeda yakni teknologi informasi dan ilmu komputer, teknik elektro dan elektronika, kecerdasan buatan dan robotika, psikologi dan sebagainya (Vijayarani, Ilamathi, & Nithya, 2011).

2.4 Praproses Teks

Praproses teks memiliki peranan yang sangat penting dalam teknik dan aplikasi *text mining*. Praproses teks merupakan langkah pertama yang dilakukan dalam proses *text mining* (Vijayarani, Ilamathi, & Nithya, 2011). Praproses teks ditujukan untuk membentuk *corpus* dan *lexicon* dari keseluruhan dokumen yang ada (Çakir & Güldamlasroglu, 2016).

Sebelum teks diterjemahkan ke dalam vektor, praproses teks dilakukan untuk membersihkan dan menyusun data. Praproses teks mengubah data teks mentah menjadi sebuah kata linguistik yang terdefinisikan dengan baik. Tahapan dari praproses teks meliputi penghapusan tanda baca dan angka, mengubah huruf kapital menjadi huruf kecil, tokenisasi, mengubah kata menjadi bentuk kata dasar (*stemming*) dan menghapus *stop word* (Çakir & Güldamlasroglu, 2016). Berikut adalah penjelasan detail dari masing-masing tahapan Praproses teks menurut Çakir & Güldamlasroglu (2016):

1. Menghapus tanda baca dan angka

   Menghapus tanda baca merupakan langkah pertama dalam tahapan praproses teks. Tanda baca tidak mengubah makna kata dan tanda baca ini dapat menimbulkan *noise* dalam data. Tanda baca juga dapat mengubah perhitungan jumlah frekuensi kata sebab kata dengan dan tanpa tanda baca dihitung berbeda. Angka seperti representasi tahun, representasi keuangan, rumus matematika dan yang lainnya juga dapat dihapus sebab angka tidak dibutuhkan untuk tahapan berikutnya.

2. *Case folding*

   Kata yang terdiri dari huruf kecil dan kata yang memiliki huruf kapital dihitung berbeda dalam kasus perhitungan frekuensi kemunculan kata yang cukup sensitif. Oleh sebab itu seluruh teks diubah ke dalam bentuk huruf kecil yang dikenal dengan istilah *case folding.*

3. Tokenisasi

   Tokenisasi dilakukan untuk memisahkan dokumen ke dalam bentuk kata-kata berdasarkan *white space* yang ada. Urutan dari token dapat dengan mudah ditemukan sehingga menghitung frekuensi kemunculan kata mudah dilakukan. Tokenisasi dilakukan untuk menghasilkan vektor dari dokumen.

4. *Stemming*

   Stemming adalah proses mengubah kata berimbuhan menjadi bentuk kata dasar. Jika bentuk dasar dari sebuah kata dan kata yang memiliki imbuhan memiliki makna yang sama tetapi berbeda bentuk, maka kata ini dapat diasumsikan sebagai sebuah kata unik. Proses *stemming* dilakukan untuk meningkatkan akurasi dan mengurasi variasi kata.

5. Menghapus *stop word*

   Stop word merupakan kata-kata yang sering muncul dimana kata-kata ini tidak memiliki makna yang cukup berarti. Sehingga *stop word* dapat dihapus dalam praproses teks. Menghapus *stop word* dapat meningkatkan akurasi dan berkontribusi dalam mengurangi ukuran vektor.

2.4.1 Algoritme *Stemming* Nazief dan Adriani

Salah satu algoritme *stemming* bahasa Indonesia yang cukup terkenal adalah algoritme Nazief dan Adriani. Algoritme ini berdasarkan pada aturan morfologi komprehensif yang mengelompokkan secara bersama dan mengenkapsulasi imbuhan yang diperbolehkan dan tidak diperbolehkan. Imbuhan ini termasuk awalan, akhiran, sisipan dan konfiks yakni kombinasi awalan dan akhiran. Algoritme ini juga mendukung proses *recoding*, yakni sebuah pendekatan untuk mengembalikan sebuah huruf awal yang telah dihilangkan dari kata dasar (Asian, Williams, & Tahaghoghi, 2005).

Secara umum kata berimbuhan dapat diklasifikasikan sebagai berikut:

[DP+[DP+[DP+]]]kata-dasar[[+DS][+PP][+P]] (2.1)

dimana DP merupakan awalan derivatif, DS merupakan akhiran derivatif, PP merupakan kata ganti kepunyaan dan P merupakan partikel (Asian, 2007).

Terdapat pengecualian dan batasan dalam membentuk kata dasar yang sesuai dengan Persamaan 2.1 antara lain (Asian, Williams, & Tahaghoghi, 2005):

1. Tidak seluruh kombinasi awalan dan akhiran (konfiks) dapat dilakukan, daftar pengecualian dapat dilihat pada Tabel 2.2.
2. Imbuhan dengan jenis yang sama tidak dapat digunakan secara berulang dalam kata.
3. Jika sebuah kata terdiri dari satu atau dua karakter, maka proses *stemming* tidak dapat dilakukan.
4. Menambah awalan memiliki kemungkinan mengubah kata dasar atau awalan sebelumnya (*recoding*).

Tabel 2.2 Kombinasi Awalan dan Akhiran yang Tidak Diperbolehkan

Awalan	Akhiran yang tidak diperbolehkan
ber-	-i
di-	-an
ke-	-i dan -kan
me-	-an
ter-	-an
per-	-an

Sumber: Asian (2007)

Algoritme Nazief dan Adriani terdiri dari tiga komponen utama yaitu antara lain pengelompokan imbuhan, urutan dalam penggunaan aturan dan pengecualian, dan sebuah kamus kata dasar (Asian, Williams, & Tahaghoghi, 2005). Berikut merupakan langkah-langkah dari algoritme Nazief dan Adriani (Asian, 2007):

1. Kata yang ingin diubah bentuknya menjadi kata dasar dicari dalam sebuah kamus kata dasar. Jika kata tersebut ditemukan dalam kamus, maka kata ini merupakan kata dasar dan proses berhenti. Jika tidak ditemukan, maka lanjutkan ke langkah 2.
2. Hapus akhiran infleksional (‘-lah’, ‘-kah’, ‘-tah’, ‘-pun’, ‘-ku’, ‘-mu’, ‘-nya’). Jika akhiran yang dihapus merupakan partikel (‘-lah’, ‘-kah’, ‘-tah’, ‘-pun’), lanjutkan dengan menghapus akhiran kata ganti kepunyaan (‘-ku’, ‘-mu’, ‘-nya’). Cari kata hasil penghapusan akhiran di dalam kamus kata dasar, jika ditemukan maka kata ini adalah kata dasar dan proses berhenti. Jika tidak, maka lanjutkan ke langkah selanjutnya.
3. Hapus akhiran derivatif (‘-i’, ‘-kan’, ‘-an’). Cari kata hasil penghapusan akhiran di dalam kamus kata dasar, jika ditemukan maka kata ini merupakan kata dasar dan proses berhenti. Jika tidak, maka lanjutkan ke langkah selanjutnya.
4. Hapus awalan derivatif (‘be-’, ‘di-’, ‘ke-’, ‘me-’, ‘pe-’, ‘se-’, ‘te-’).
    a. Proses berhenti jika:
        i. Akhiran yang telah dihapus pada langkah 3 terdapat dalam kombinasi awalan dan akhiran yang tidak diperbolehkan sesuai pada Tabel 2.2.
        ii. Awalan sekarang sama dengan awalan sebelumnya yang telah dihapus.
        iii. Terdapat tiga awalan yang telah dihapus.
    b. Tipe awalan dapat ditentukan dalam langkah berikut:
        i. Jika awalan ‘di-’, ‘ke-’, dan ‘se-‘, maka awalan dapat dihapus secara langsung.

ii. Jika awalan ‘te-’, ‘be-’, ‘me-’ dan ‘pe-’, maka gunakan aturan yang terdapat pada Tabel 2.3.

Tabel 2.3 Aturan untuk Menghapus Awalan ‘te-’, ‘be-’, ‘me-’ dan ‘pe-’

Aturan	Bentuk Kata	Kata yang Dikembalikan
1.	berV. . .	ber-V. . . | be-rV. . .
2.	berCAP. . .	ber-CAP. . . dimana C!=’r’ dan P!=’er’
3.	berCAerV. . .	ber-CAerV. . . dimana C!=’r’
4.	belajar	bel-ajar
5.	beC_1erC_2. . .	be-C_1erC_2. . . dimana C_1!=’r’|’l’
6.	terV. . .	ter-V. . . | te-rV. . .
7.	terCerV. . .	ter-CerV. . . dimana C!=’r’
8.	terCP. . .	ter-CP. . . dimana C!=’r’ dan P!=’er’
9.	teC_1erC_2. . .	te-C_1erC_2. . . dimana C_1!=’r’
10.	me{l|r|w|y}V. . .	me-{l|r|w|y}V. . .
11.	mem{b|f|v}. . .	mem-{b|f|v}. . .
12.	mempe{r|l}. . .	mem-pe. . .
13.	mem{rV|V}. . .	me-m{rV|V}. . . | me-p{rV|V}. . .
14.	men{c|d|j|z}. . .	men-{c|d|j|z}. . .
15.	menV. . .	me-nV. . . | me-tV. . .
16.	meng{g|h|q}. . .	meng-{g|h|q}. . .
17.	mengV. . .	meng-V. . . | meng-kV. . .
18.	menyV. . .	meny-sV. . .
19.	mempV. . .	mem-pV. . .where V!=‘e’
20.	pe{w|y}V. . .	pe-{w|y}V. . .
21.	perV. . .	per-V. . . | pe-rV. . .
22.	perCAP. . .	per-CAP. . . dimana C!=‘r’ and P!=‘er’
23.	perCAerV. . .	per-CAerV. . . dimana C!=‘r’
24.	pem{b|f|v}. . .	pem-{b|f|v}. . .
25.	pem{rV|V}. . .	pe-m{rV|V}. . . | pe-p{rV|V}. . .
26.	pen{c|d|j|z}. . .	pen-{c|d|j|z}. . .
27.	penV. . .	pe-nV. . . | pe-tV. . .
28.	peng{g|h|q}. . .	peng-{g|h|q}. . .
29.	pengV. . .	peng-V. . . | peng-kV. . .

Tabel 2.3 (lanjutan)

Aturan	Bentuk Kata	Kata yang Dikembalikan
30.	penyV. . .	peny-sV. . .
31.	pelV. . .	pe-lV. . . kecuali untuk "pelajar", kembalikan "ajar"
32.	peCerV. . .	per-erV. . . dimana C!={r|w|y|l|m|n}
33.	peCP. . .	pe-CP. . . dimana C!={r|w|y|l|m|n} and P!='er'

Sumber: Asian (2007)

dimana V merupakan huruf vokal, C merupakan huruf konsonan, dan A menandakan setiap huruf.

iii. Jika dua karakter pertama dari kata tidak cocok pada awalan 'di-', 'ke-', 'se-', 'te-', 'be-', 'me-' dan 'pe-', maka proses berhenti.

c. Ulangi langkah 4. Jika kata ditemukan dalam kamus kata dasar, maka proses berhenti. Jika tidak ditemukan setelah mengulangi langkah 4 dan salah satu kondisi 4.a terpenuhi, maka lanjutkan ke langkah selanjutnya.

5. Lakukan recoding.
6. Jika tidak ada langkah yang berhasil, maka kembalikan kata tersebut sebagai kata dasar.

2.5 *Vector Space Model*

Vector space model merupakan salah satu model klasik dalam sistem temu kembali informasi berdasarkan pengetahuan matematika yang mudah dikenali dan dipahami secara sederhana, efisien dan mudah diimplementasikan (A & Aghila, 2010). *Vector space model* merepresentasikan dokumen dan *query* sebagai vektor yang berada dalam ruang multi dimensi (Singh & Dwivedi, 2012). Dimana *vector space model* dibuat secara statistik dengan mengesktrak kata-kata dari dokumen dan kata-kata ini dilakukan pembobotan untuk direpresentasikan secara *vector* dari keseluruhan dokumen (Lee, Chuang, & Seamons, 1997).

Prosedur dari *vector space model* dibagi ke dalam tiga tahapan (Singh & Dwivedi, 2012). Tahapan pertama adalah pengindeksan dokumen dimana kata-kata diekstrak dari dokumen teks. Tahapan kedua adalah melakukan pembobotan kata untuk meningkatkan pencarian dokumen yang relevan terhadap pengguna. Tahapan terakhir adalah menghitung kemiripan antar dokumen teks untuk memperoleh hasil secara berperingkat. *Vector space model* memiliki performa yang baik dalam menentukan kemiripan makna antar kata, frasa, dan dokumen (Turney & Pantel, 2010). Untuk menghitung kemiripan antar dokumen teknik yang paling banyak digunakan adalah *cosine similarity*.

Menurut A & Aghila (2010) penerapan *vector space model* dalam sistem temu kembali informasi memiliki kelebihan antara lain sebagai berikut:

1. Dapat melakukan pembobotan terhadap suatu kata dalam sebuah *query*
2. Ukuran kemiripan (*cosine similarity*) dapat digunakan untuk menampilkan hasil dalam bentuk urutan yang relevan
3. Banyak penelitian yang mengatakan bahwa hasil dari *vector space model* lebih baik dari penggunaan *boolean model* dalam sistem temu kembali informasi

Di samping penerapan *vector space model* dalam sistem temu kembali informasi yang memiliki kelebihan, *vector space model* juga memiliki kekurangan antara lain sebagai berikut (A & Aghila, 2010):

1. Penggunaan kata yang berdiri sendiri
2. Dalam penggunaan *query* tidak dapat menggunakan relasi logika seperti AND, OR, dan NOT

2.5.1 Pembobotan Kata

Pembobotan kata di dalam vektor dokumen dapat ditentukan dengan menggunakan metode *tf.idf* (Singh & Dwivedi, 2012). Metode *tf.idf* merupakan salah satu metode pembobotan yang banyak digunakan dalam *vector space model*. Metode *tf.idf* menghitung bobot dari masing-masing komponen vektor (masing-masing kata yang ada) dari tiap masing-masing dokumen (Soucy & Guy, 2005).

Tahap pertama dalam melakukan pembobotan kata adalah dengan menghitung frekuensi kemunculan suatu kata atau yang dikenal dengan sebutan *tf*. Semakin sering suatu kata muncul dalam dokumen maka kata ini dapat dikatakan cukup penting dalam dokumen ini (Soucy & Guy, 2005). Terdapat modifikasi dari *tf* yang menggunakan logaritma dari frekuensi kemunculan kata untuk menghitung nilai bobot *tf* (Manning, Raghavan, & Schutze, 2009):

$$wf_{t,d} = \begin{cases} 1 + log\ tf_{t,d} & \text{jika } tf_{t,d} > 0 \\ 0 & \text{selainnya} \end{cases} \quad (2.2)$$

dimana *t* menandakan posisi kata dan *d* menandakan dokumen ke berapa.

Tahap selanjutnya adalah menghitung nilai *idf*. Nilai *idf* menghitung seberapa tidak sering suatu kata muncul dalam koleksi keseluruhan dokumen. Sehingga kata yang sering muncul dalam dokumen dianggap tidak merepresentasikan dokumen, sebaliknya kata yang jarang muncul dianggap paling relevan terhadap dokumen (Soucy & Guy, 2005). Nilai *idf* dapat diperoleh dengan cara sebagai berikut (Lee, Chuang, & Seamons, 1997):

$$idf_t = \log N / df_t \quad (2.3)$$

Karena nilai *tf* telah dilakukan modifikasi menjadi *wf* maka penyebutan *tf.idf* dapat diubah menjadi *wf.idf* (Manning, Raghavan, & Schutze, 2009). Sehingga *wf.idf* memiliki persamaan sebagai berikut:

$$wf.idf_{t,d} = wf_{t,d} \times idf_t \quad (2.4)$$

Nilai *wf.idf* memiliki variasi normalisasi nilai, salah satunya adalah sebagai berikut:

$$wf.idf_{t,d} = \frac{wf.idf_{t,d}}{\sqrt{\sum_{t=1}^{n} wf.idf_{t,d}^{\ 2}}} \tag{2.5}$$

2.5.2 *Cosine Similarity*

Cosine similarity merupakan teknik pengukuran kemiripan antar dokumen yang banyak digunakan dimana teknik pengukuran ini menentukan sudut antara vektor dokumen dan vektor *query*. Sudut antar kedua vektor ini dianggap sebagai ukuran divergensi antar vektor. Sudut *cosine* digunakan untuk menghitung nilai kemiripan, dimana ditentukan berdasarkan sudut antara vektor dokumen dan vektor *query* ketika keduanya direpresentasikan dalam ruang *Euclidean* dimensi *V* dengan *V* merupakan ukuran dimensi (Singh & Dwivedi, 2012).

Karena nilai *cosine* dari sudut 0° adalah 1 dan *cosine* dari sudut 90° adalah 0, maka dapat dikatakan nilai kemiripan berada pada nilai -1 dan 1. Vektor memiliki arah yang sama jika nilai *cosine similarity* 1, kedua vektor berada dalam sudut 90° memiliki nilai *cosine similarity* 0, dan vektor yang berlawanan arah memiliki nilai *cosine similarity* -1 (Sahu & Mohan, 2014). Fungsi *cosine similarity* antara dokumen d_j dan *query* q adalah sebagai berikut (Singh & Dwivedi, 2012):

$$CosSim(d_j, q) = \frac{\sum_{i=1}^{t} (w_{ij}.w_{iq})}{\sqrt{\sum_{i1}^{t} w_{ij}^{\ 2} . \sum_{i1}^{t} w_{iq}^{\ 2}}} \tag{2.6}$$

Karena pembobotan kata *wf.idf* pada Persamaan 2.5 menggunakan normalisasi panjang dari vektor menjadi unit vektor, maka nilai *cosine similarity* dapat diperoleh dengan cara (Manning, Raghavan, & Schutze, 2009):

$$CosSim(d_j, q) = \sum_{i=1}^{t} (w_{ij}.w_{iq}) \tag{2.7}$$

2.6 Klasterisasi

Klasterisasi dokumen atau yang biasa dikenal dengan sebutan klasterisasi teks merupakan salah satu metode penting dalam *text mining* yang dikembangkan untuk membantu pengguna dalam melakukan navigasi, peringkasan, dan mengorganisasi dokumen teks secara efektif (A. & Aghila, 2010). Klasterisasi dokumen memiliki tujuan untuk menemukan struktur intrinsik informasi dan menyusunnya ke dalam beberapa sub grup untuk penelitian dan analisis lebih lanjut. Klasterisasi dokumen termasuk sebuah proses *unsupervised* dimana objek dikelompokkan ke dalam grup yang disebut dengan klaster (Dang & Ahmad, 2014). Terdapat tiga cara dalam melakukan klasterisasi yaitu antara lain klasterisasi

berdasarkan partisi, klasterisasi hirarki, dan klasterisasi berdasarkan model (Al-Azzawy & Al-Rufaye, 2017).

2.6.1 Algoritme *K-Means*

Algoritme *k-means* merupakan salah satu algoritme yang paling cukup populer digunakan dan merupakan algoritme partisi sederhana yang termasuk dalam teknik *unsupervised* (Kaur & Kalra, 2016). *K-means* merupakan salah satu metode dalam klasterisasi yang bersifat matematika, tidak dapat dimanajemen, tidak dapat ditentukan dan merupakan proses iteratif (Chandrawanshi, Tripathi, & Khan, 2016). *K-means* merupakan sebuah metode vektor kuantisasi yang memiliki tujuan untuk membagi *n* data ke dalam *k* klaster dimana setiap data dikelompokkan berdasarkan nilai jarak rata-rata terdekat terhadap klaster (Lu, et al., 2016).

Dalam penggunaan algoritme *k-means*, algoritme ini memiliki tantangan jika diimplementasikan ke dalam sebuah sistem antara lain sebagai berikut (Kaur & Kalra, 2016):

1. Algoritme *k-means* mengasumsikan bahwa nilai *k* sudah diketahui, hal ini tidak sesuai jika diaplikasikan ke dalam data yang bersifat *real*.
2. Algoritme *k-means* cukup sensitif dalam pemilihan *centroid* awal klaster.
3. Algoritme *k-means* memiliki kemungkinan untuk bertemu dengan kondisi *local minima*.

Berikut adalah langkah-langkah dari algoritme *k-means* menurut Sahu & Mohan (2014):

1. Tentukan jumlah *k.*
2. Pilih secara acak *k centroid* awal klaster.
3. Tentukan kondisi konvergen. Menurut Chandrawanshi, Tripathi, & Khan (2016) kondisi konvergen dalam algoritme *k-means* antara lain sebagai berikut:
    a. Hentikan proses ketika telah mencapai pada batasan iterasi yang telah ditentukan
    b. Hentikan proses ketika klaster tidak mengalami perubahan pada iterasi sebelumnya
4. Hitung jarak antara tiap data dengan *centroid* klaster dengan menggunakan *cosine similarity* yang terdapat pada Persamaan 2.7. *Cosine similarity* digunakan sebab memiliki performa yang baik dan efisien jika dibandingkan dengan *k-means* yang menggunakan *Euclidean distance* (Sabthami, Thirumoorthy, & Muneeswaran, 2016).
5. Tentukan data yang terdekat terhadap *centroid* klaster.
6. Hitung kembali *centroid* baru dengan menggunakan Persamaan 2.8:

$$v_i = (\frac{1}{C_i}).\sum_{j=1}^{C_i}(b_j) \quad (2.8)$$

dimana C_i menandakan jumlah data yang berada dalam klaster *i*, v_i merupakan *centroid* klaster baru, dan b_j adalah kumpulan data.

7. Ulangi langkah 4 jika kondisi konvergen belum terpenuhi.

Terdapat beberapa teknik atau cara yang dapat dilakukan agar algoritme *k-means* memperoleh hasil yang baik. Cara ini adalah dengan memilih parameter-parameter yang ada dalam algoritme *k-means* sebagai berikut (Lu, et al., 2016):

1. Jumlah klaster: lakukan secara berulang proses *k-means* dengan menggunakan nilai *k* yang berbeda sehingga dapat ditentukan nilai *k* mana yang baik digunakan dalam algoritme *k-means*.
2. *Centroid* awal: lakukan proses *k-means* berulang kali untuk mengetahui *centroid* awal mana yang memiliki hasil terbaik. Cara lain adalah dengan menggunakan metode hirarki untuk menentukan nilai *centroid* awal. Atau dapat juga dengan menggunakan analisis *cross validation*.

2.6.2 Algoritme *Improved K-Means*

Dalam penggunaan algoritme *k-means*, pemilihan *centroid* awal yang tepat memiliki pengaruh besar terhadap hasil klaster. Dalam beberapa pendekatan untuk mengoptimasi *k-means*, pemilihan *centroid* awal dilakukan dengan mengambil nilai jarak terjauh atau yang memiliki nilai densitas terbesar antar objek data. Akan tetapi jika di dalam dataset terdapat data yang kotor atau terisolasi maka terdapat kemungkinan data ini digunakan sebagai *centroid* awal. Oleh karena itu dalam melakukan pemilihan *centroid* awal dapat dilakukan dengan menggabungkan metode optimasi jarak dan densitas untuk memperoleh *centroid* awal terbaik (Xiong, et al., 2016).

Langkah-langkah dari algoritme *improved k-means* yang merupakan kombinasi antara metode optimasi jarak dan densitas dijelaskan sebagai berikut (Xiong, et al., 2016):

Input: dataset dokumen $D = \{d_1, d_2, ..., d_n\}$ dan *k*

Output: *k* klaster

1. Hitung jarak antara setiap pasangan dua objek data yang berada dalam dataset *D*

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{im} - x_{jm})^2} \quad (2.9)$$

dimana $x_i = (x_{i1}, x_{i2}, ..., x_{im})$ dan $x_j = (x_{j1}, x_{j2}, ..., x_{im})$ merupakan dua *m* dimensi objek data

2. Hitung rata-rata jarak pada langkah 1

$$MeanDist = \frac{1}{C_2^n} \times \sum d(x_i, x_j) \quad (2.10)$$

3. Hitung nilai parameter densitas seluruh objek data yang berada dalam dataset *D*

$$Dens(x_i) = \sum_{j=1}^{n} u(MeanDist - d(x_i, x_j)) \quad (2.11)$$

dimana $u(z)$ merupakan sebuah fungsi:

$$u(z) = \begin{cases} 1, z \geq 0 \\ 0, z < 0 \end{cases}$$

4. Hitung rata-rata nilai parameter densitas dataset *D*

$$MeanDens(D) = \frac{1}{n} \sum_{i=1}^{n} Dens(x_i) \quad (2.12)$$

5. Dengan menggunakan Persamaan 2.12, tentukan objek data yang terisolasi dan hapus data ini dari *D* sehingga menghasilkan koleksi *A* yang memiliki nilai parameter densitas tertinggi

$$Dens(x_i) < \alpha \times MeanDens(D) \quad (2.13)$$

dimana $0 < \alpha < 1$ untuk menentukan titik terisolasi x_i

6. Pilih objek data yang memiliki nilai parameter densitas tertinggi dari *A* sebagai nilai *centroid* awal klaster pertama, masukkan ke dalam koleksi *B*, dan hapus dari *A*
7. Dari koleksi *A*, pilih objek data yang memiliki jarak terjauh dari objek data yang berada dalam *B* sebagai nilai *centroid* awal klaster berikutnya, masukkan ke dalam *B*, dan hapus dari *A*
8. Ulangi langkah 7 hingga jumlah objek data *k* berada dalam koleksi *B*
9. Berdasarkan *k centroid* awal klaster, lakukan *k-means* untuk melakukan pengelompokan terhadap objek data

2.7 Evaluasi

Evaluasi dalam sebuah teknik klasterisasi dilakukan untuk melakukan pengukuran yang independen dan dapat diandalkan untuk penilaian dan perbandingan dari eksperimen dan hasil klasterisasi. Secara teori sudah banyak penelitian yang menghasilkan metode yang dapat digunakan dalam melakukan evaluasi hasil klasterisasi, akan tetapi dalam prakteknya jumlah representasi data yang besar membuat evaluasi sulit dilakukan. Oleh karena itu evaluasi secara introspektif hanya dapat digunakan pada objek yang kecil, untuk penelitian yang menggunakan objek cukup besar maka dibutuhkan metode yang bersifat secara objektif. Dalam penerapannya terdapat banyak jenis-jenis dari metode evaluasi

dalam berbagai bidang seperti teori statistik, *machine vision*, dan klasterisasi halaman *web*.

2.7.1 *Silhouette Coefficient*

Silhouette coefficient merupakan suatu teknik evaluasi klasterisasi yang digunakan untuk melihat kualitas dan kekuatan klaster, dimana seberapa baik suatu objek ditempatkan dalam suatu klaster (Shoolihah, Furqon, & Widodo, 2017). Tahap pertama untuk menentukan *silhouette coefficient* adalah menghitung nilai rata-rata jarak dari data *i* dengan semua data yang lain dalam satu klaster yang sama.

$$a(i) = \frac{1}{|A|-1} \sum_{j \in A, j \neq i} d(i, j) \tag{2.14}$$

Dimana *j* adalah data lain yang berada dalam klaster A dan *d(i,j)* merupakan jarak antara data *i* dan *j*. Tahap berikutnya adalah menghitung nilai rata-rata jarak dari data *i* dengan semua data lain yang berada pada klaster berbeda yang kemudian diambil nilai terkecilnya.

$$b(i) = \min_{B \neq A} \frac{1}{|B|} \sum_{j \in B} d(i, j) \tag{2.15}$$

Sehingga *silhouette coefficient* dari data *i* yang berada pada klaster A dapat diperoleh dengan menggunakan Persamaan 2.16 seperti berikut (Zoubi & Rawi, 2008):

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.16}$$

Hasil dari *silhouette coefficient* merupakan sebuah bilangan *real* yang berada di antara -1 dan 1. Jika nilai *silhouette coefficient* mendekati -1, maka klaster kurang sesuai karena nilai rata-rata jarak ke data lain dalam klaster lebih besar bila dibandingkan dengan nilai terkecil rata-rata jarak ke data lain yang berada pada klaster berbeda. Lebih besar nilai yang diperoleh *silhouette coefficient* maka lebih baik kualitas suatu hasil klasterisasi (Chaimontree, Atkinson, & Coenen, 2010).

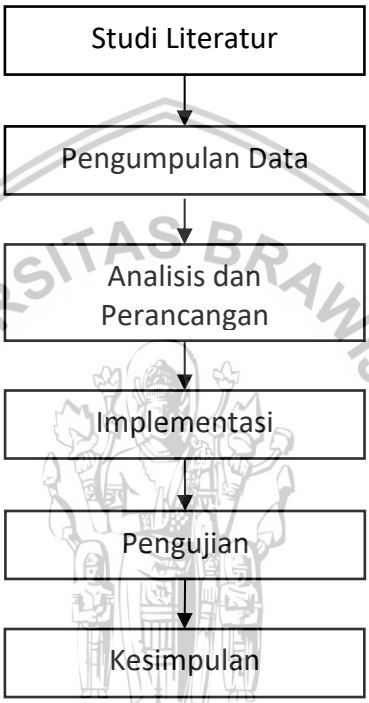
2.7.2 *Purity*

Purity merupakan salah satu teknik evaluasi klaster yang bersifat eksternal. Dalam menghitung nilai *purity*, tiap-tiap klaster dilihat label mana yang memiliki jumlah anggota terbanyak pada klaster tersebut (Deepa & Revathy, 2012). Nilai *purity* ini dapat dihitung dengan cara membagi hasil penjumlahan label yang sering muncul pada tiap klaster terhadap jumlah data. Sehingga *purity* dapat diperoleh dengan cara sebagai berikut:

$$purity(\Omega, C) = \frac{1}{N} \sum_{k} \max_{j} |\omega_k \cap c_j| \tag{2.17}$$

BAB 3 METODOLOGI

Bab ini berisi tahapan dalam menyelesaikan penelitian yang memiliki tujuan untuk mengelompokkan dokumen J-PTIIK dengan menggunakan metode *improved k-means.* Penelitian yang dilakukan merupakan termasuk jenis penelitian implementatif yang bersifat pembangunan. Penelitian ini akan menghasilkan luaran berupa sebuah perangkat lunak yang dapat mengelompokkan dokumen J-PTIIK. Berikut merupakan alur metodologi penelitian yang dilakukan ditunjukkan pada Gambar 3.3.



Gambar 3.3 Alur Metodologi Penelitian

3.1 Studi Literatur

Penelitian yang dilakukan penulis membutuhkan studi literatur dari dasar teori pendukung yang secara detail telah dibahas sebelumnya dalam bab 2. Dasar teori ini disusun berdasarkan referensi yang diperoleh melalui buku, jurnal, dan serta penelitian-penelitian lainnya yang terkait baik dalam skala nasional maupun internasional. Studi literatur merupakan pedoman dasar dalam melakukan analisis, perancangan, implementasi dan pengujian dalam penelitian yang dilakukan. Dasar teori yang dibutuhkan sebagai pedoman dan pendukung dalam penelitian yang dilakukan antara lain sebagai berikut:

1. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK)
2. *Text mining*
3. Praproses teks
4. *Vector space model* dan *cosine similarity*

5. Klasterisasi
6. Metode *improved k-means*
7. Evaluasi *silhouette coefficient* dan *purity*

3.2 Pengumpulan Data

Kebutuhan data dibutuhkan untuk mendukung penelitian yang dilakukan dengan luaran berupa pengelompokan dokumen J-PTIIK menggunakan metode *improved k-means*. Data dokumen J-PTIIK diperoleh secara langsung melalui *web* J-PTIIK yang dapat diakses melalui *j-ptiik.ub.ac.id*. Data dokumen J-PTIIK yang digunakan hanya merupakan judul dan abstrak dari masing-masing dokumen J-PTIIK yang diperoleh dari *web* J-PTIIK dimulai dari arsip J-PTIIK Vol 1 No 1 (2017) hingga Vol 1 No 12 (2017).

3.3 Analisis dan Perancangan

Analisis dan perancangan merupakan langkah selanjutnya yang dilakukan dalam penelitian setelah mengumpulkan data dokumen J-PTIIK. Penelitian ini termasuk jenis penelitian implementatif yang bersifat pembangunan sehingga dibutuhkan analisis kebutuhan sistem. Setelah analisis kebutuhan sistem dilakukan, selanjutnya adalah melakukan tahapan perancangan dalam melakukan penelitian pengelompokan dokumen J-PTIIK.

3.3.1 Analisis Kebutuhan Sistem

Dalam melaksanakan penelitian yang memiliki luaran untuk mengelompokkan dokumen J-PTIIK menggunakan metode *improved k-means* dibutuhkan sebuah sistem. Adapun sistem yang dibutuhkan dalam penelitian ini terdiri dari dua yaitu perangkat keras (*hardware*) dan perangkat lunak (*software*). Berikut merupakan spesifikasi perangkat keras (*hardware*) ditunjukkan pada Tabel 3.4 dan spesifikasi perangkat lunak (*software*) ditunjukkan pada Tabel 3.5 yang digunakan penulis dalam melaksanakan penelitian ini.

Tabel 3.4 Spesifikasi Perangkat Keras (*Hardware*)

Nama Komponen	Spesifikasi
Prosesor	Intel® Core™ i3-2330M CPU @ 2.20GHz (4 CPUs)
Memori (RAM)	TEAM Elite DDR3 4+2 GB PC12800 1600Mhz
Kartu Grafis	NVIDIA GeForce GT 520M (1GB)
Harddisk	Hitachi 500GB
Laptop	ASUS A43S

Tabel 3.5 Spesifikasi Perangkat Lunak (*Software*)

Nama Komponen	Spesifikasi
Sistem Operasi	Microsoft Windows 10 Pro 64-bit
Editor Dokumentasi	Microsoft Office 2016
Editor Pemrograman	NetBeans IDE 8.1

3.3.2 Perancangan

Perancangan dilakukan untuk mempermudah pada tahapan implementasi pengelompokan dokumen J-PTIIK menggunakan metode *improved k-means*. Bagian ini memuat perancangan proses algoritme dalam memecahkan permasalahan pengelompokan dokumen J-PTIIK dengan menggunakan model *flowchart.* Perancangan *flowchart* algoritme yang digunakan dalam penelitian ini antara lain sebagai berikut:

1. *Flowchart* praproses teks dokumen J-PTIIK.
2. *Flowchart* algoritme *stemming* Nazief Adriani dengan bantuan *library jsastrawi* yang dibutuhkan untuk mengubah kata berimbuhan menjadi kata dasar.
3. *Flowchart* pembentukan *vector space model* dari hasil praproses teks dokumen J-PTIIK.
4. *Flowchart* metode *improved k-means* yang digunakan untuk mengelompokkan dokumen J-PTIIK.

Pada bagian ini juga terdapat perancangan manualisasi penyelesaian permasalahan secara sederhana dengan menggunakan sebagian data dari dokumen J-PTIIK yang menggunakan metode *improve k-means.* Di bagian ini juga terdapat perancangan pengujian evaluasi dengan menggunakan *silhouette coefficient* dan *purity.*

3.4 Implementasi

Implementasi merupakan tahapan yang dilakukan setelah perancangan pengelompokan dokumen J-PTIIK dengan menggunakan metode *improved k-means*. Implementasi dalam penelitian dilakukan dengan mengacu pada hasil perancangan yang telah dilakukan sebelumnya. Pada penelitian ini, penulis menggunakan bahasa pemrograman Java dalam melakukan implementasi. Implementasi dilakukan dalam sebuah sistem yang memiliki spesifikasi perangkat keras (*hardware*) dan perangkat lunak (*software*) yang telah dijabarkan pada sub bab analisis kebutuhan sistem. Implementasi yang dilakukan dalam penelitian ini terdiri dari dua bagian yaitu antara lain sebagai berikut:

1. Implementasi hasil perancangan *flowchart* algoritme yang digunakan ke dalam *source code* bahasa pemrograman Java.
2. Implementasi tampilan antarmuka pengguna (*user interface*) dengan menggunakan Java JFrame.

3.5 Pengujian

Dalam penelitian pengelompokan dokumen J-PTIIK menggunakan metode *improved k-means* perlu dilakukan pengujian terhadap hasil pengelompokan. Pengujian ini dilakukan untuk mengukur kualitas klaster dari hasil pengelompokan dokumen J-PTIIK menggunakan metode *improved k-means.* Teknik pengujian dilakukan secara berulang dengan menggunakan evaluasi *silhouette coefficient* dan *purity* dengan nilai *k* = 2 hingga nilai *k* = 25. Hasil evaluasi ini ditampilkan ke dalam bentuk grafik nilai *k* - nilai *silhouette coefficient* dan nilai *k* – nilai *purity.*

3.6 Kesimpulan

Setelah tahapan penelitian yang dimulai dari studi literatur hingga pengujian selesai dilakukan, maka tahap terakhir adalah pengambilan kesimpulan penelitian. Kesimpulan dapat diperoleh melalui hasil pengelompokan dokumen J-PTIIK menggunakan metode *improved k-means* berdasarkan pengujian evaluasi *silhouette coefficient* dan *purity*. Sehingga kesimpulan yang diperoleh dapat menjawab rumusan masalah yang ada dalam penelitian ini.

Selain melakukan pengambilan kesimpulan penelitian, dilakukan pula evaluasi seperti kekurangan yang terdapat dalam penelitian. Sehingga penulis memberikan saran yang dapat diterapkan pada penelitian selanjutnya baik dari segi penggunaan metode pengelompokan (klasterisasi) ataupun objek yang digunakan dalam penelitian. Melalui saran ini diharapkan penelitian selanjutnya dapat menghasilkan penelitian yang lebih baik dari penelitian sebelumnya.

BAB 4 PERANCANGAN

4.1 Alur Kerja Sistem

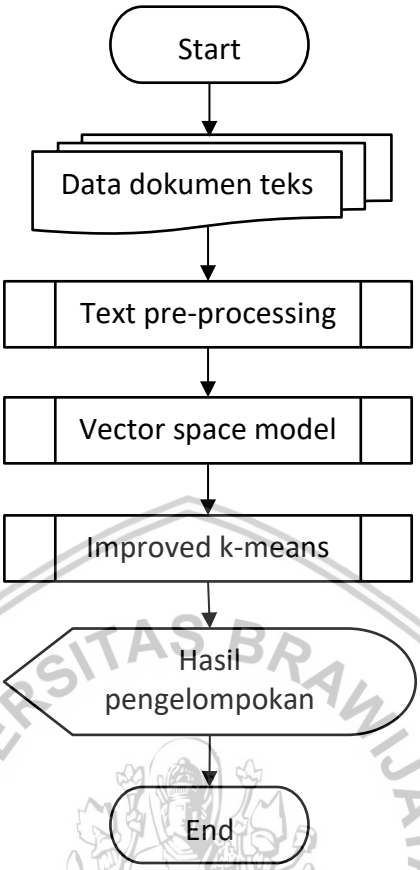
Secara umum, prinsip kerja dari sistem yang mengimplementasikan metode *improved k-means* untuk mengelompokkan dokumen J-PTIIK menghasilkan luaran hasil berupa pengelompokan dokumen J-PTIIK. Sistem membutuhkan objek data judul dan abstrak dokumen J-PTIIK yang diperoleh secara manual melalui *web* J-PTIIK yang beralamat di *j-ptiik.ub.ac.id.* Jumlah objek data dokumen J-PTIIK yang diperoleh dari arsip J-PTIIK Vol 1 No 1 (2017) hingga Vol 1 No 12 (2017) sebanyak 233. Objek data judul dan abstrak dokumen J-PTIIK disimpan ke dalam file teks untuk masing-masing dokumen J-PTIIK. Pada saat menyimpan judul dan abstrak ke dalam file teks ditambahkan informasi berupa *tag* JUDUL dan ABSTRAK untuk memudahkan identifikasi bagian judul dan abstrak.

Proses berikutnya adalah sistem membaca file teks masing-masing dokumen J-PTIIK yang kemudian akan dilakukan praproses teks. Dimana pada praproses teks, sistem menghapus *tag* JUDUL dan ABSTRAK, menghapus tanda baca dan angka, melakukan *case folding* atau mengubah semua huruf menjadi huruf kecil dan melakukan tokenisasi untuk menghasilkan token. Hasil token ini kemudian dilakukan *filtering* untuk menghapus *stopword* dari token yang ada, kemudian diikuti dengan melakukan *stemming* dengan menggunakan algoritme Nazief dan Adriani untuk menghasilkan kata dasar.

Proses berikutnya adalah menyusun kata hasil dari praproses teks ke dalam sebuah bentuk vektor kata-dokumen. Dimana proses ini dilakukan untuk menghitung jumlah frekuensi kata pada masing-masing dokumen. Sehingga pada proses ini dapat dilakukan pembobotan kata dengan menggunakan metode *wf.idf.* Pembobotan kata *wf.idf* diperoleh dengan mengkalikan nilai bobot jumlah frekuensi kata pada masing-masing dokumen dengan nilai frekuensi *inverse* pada dokumen.

Hasil pembobotan kata dengan menggunakan metode *wf.idf* selanjutnya digunakan untuk melakukan pengelompokan dengan menggunakan metode *improved k-means.* Pada proses ini yang pertama kali dilakukan adalah mencari nilai *centroid* awal dengan menggunakan metode *improved k-means* yang telah dijelaskan pada sub bab 2.5.2. Kemudian nilai *centroid* awal yang diperoleh dapat digunakan untuk melakukan proses *k-means* seperti pada umumnya untuk menghasilkan pengelompokan dokumen J-PTIIK.

Proses terakhir dari sistem adalah melakukan pengujian terhadap hasil pengelompokan dokumen J-PTIIK dengan menggunakan teknik *silhouette coefficient* dan *purity.* Proses ini dilakukan untuk mengetahui seberapa baik kualitas hasil pengelompokan dokumen J-PTIIK berdasarakan judul dan abstrak. Secara garis besar, diagram alir kerja sistem pengelompokan dokumen J-PTIIK menggunakan metode *improved k-means* dapat dilihat pada Gambar 4.4.



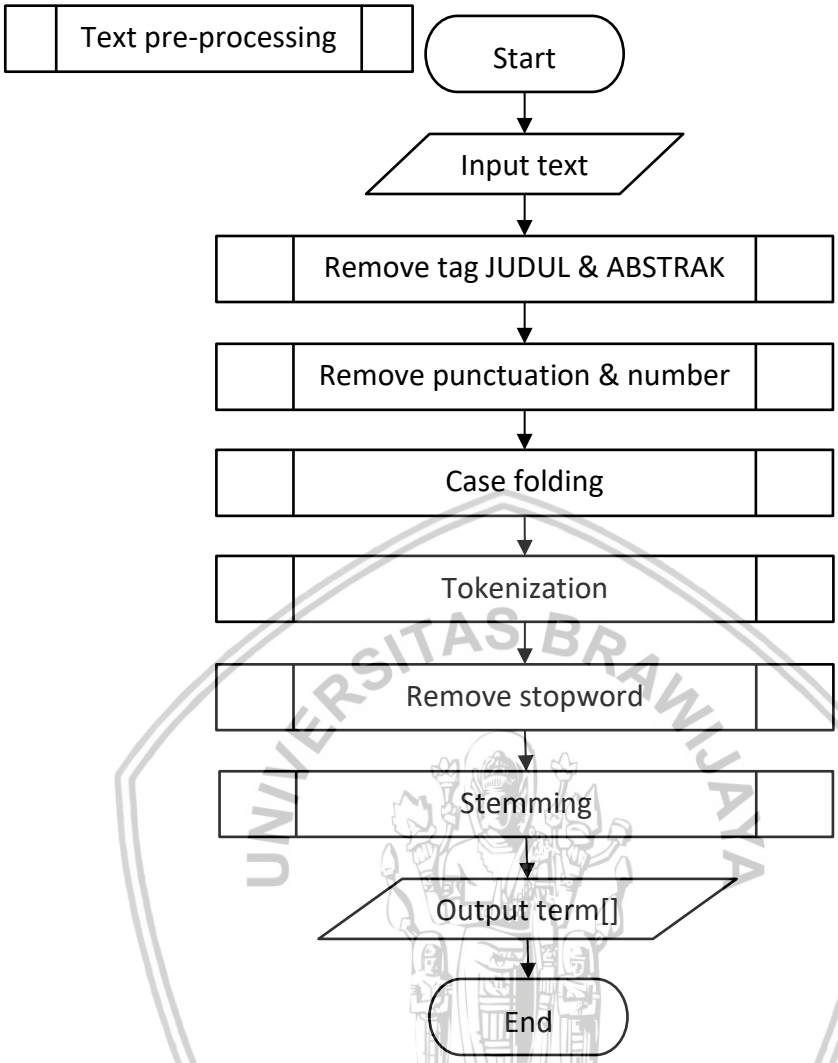
Gambar 4.4 Diagram Alir Kerja Sistem

4.2 Perancangan Sistem

Pada perancangan sistem diuraikan *flowchart* dari masing-masing proses yang berjalan pada sistem pengelompokan dokumen J-PTIIK dengan menggunakan metode *improved k-means.* Secara garis besar proses tersebut antara lain adalah praproses teks, pembentukan *vector space model* untuk memperoleh hasil pembobotan kata, dan penggunaan metode *improved k-means* untuk mengelompokkan dokumen J-PTIIK.

4.2.1 Praproses Teks

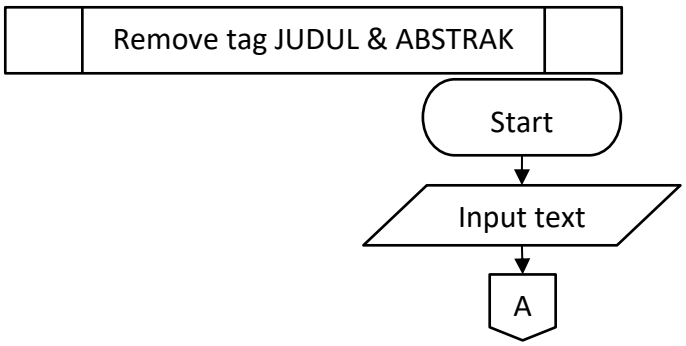
Pada praproses teks dilakukan beberapa sub proses yang bertujuan untuk memecah dokumen J-PTIIK menjadi kumpulan kata. Hasil kumpulan kata ini nantinya akan digunakan untuk proses pada tahap berikutnya. Praproses teks terdiri dari beberapa sub proses antara lain adalah menghapus *tag* JUDUL dan ABSTRAK, menghapus tanda baca dan angka, *case folding*, tokenisasi, menghapus *stopword* dan *stemming*. *Flowchart* untuk praproses teks dapat dilihat pada Gambar 4.5.



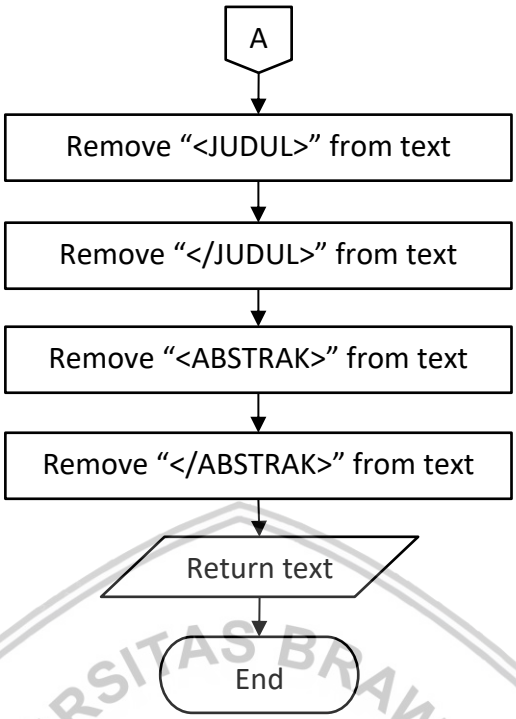
Gambar 4.5 *Flowchart* Praproses Teks

4.2.1.1. Hapus *Tag* JUDUL & ABSTRAK

Pada sub proses hapus *tag* JUDUL dan ABSTRAK dilakukan untuk menghapus *tag* JUDUL dan ABSTRAK dari teks. *Flowchart* untuk sub proses hapus *tag* JUDUL dan ABSTRAK dapat dilihat pada Gambar 4.6.



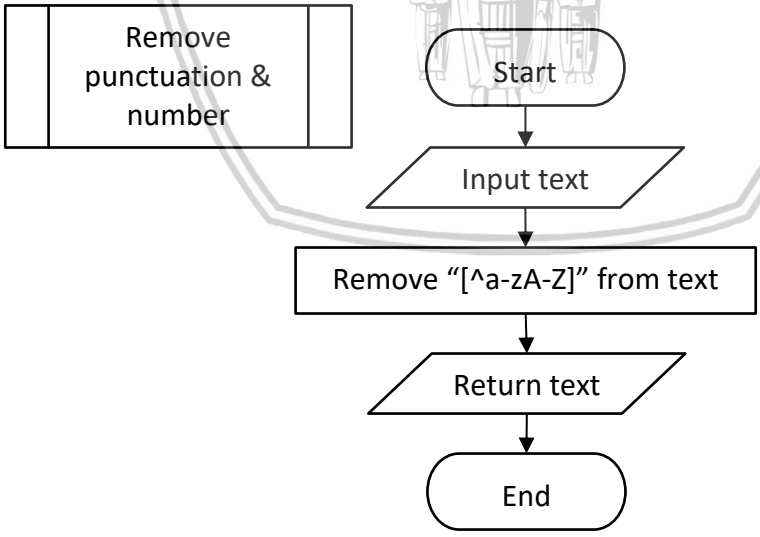
Gambar 4.6 *Flowchart* Hapus *Tag* JUDUL & ABSTRAK



Gambar 4.6 (lanjutan)

4.2.1.2. Hapus Tanda Baca Dan Angka

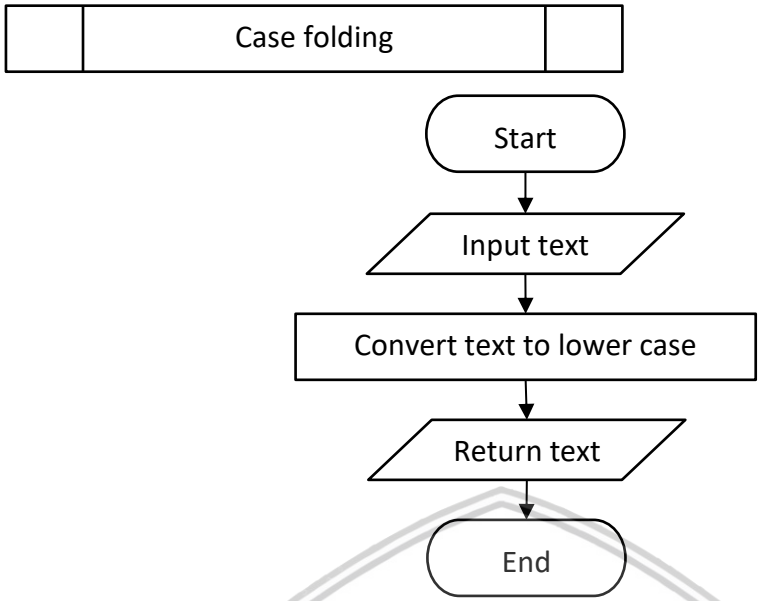
Pada sub proses hapus tanda baca dan angka dilakukan untuk menghapus tanda baca dan angka dari teks hasil sub proses hapus *tag* JUDUL dan ABSTRAK. *Flowchart* untuk sub proses hapus tanda baca dan angka dapat dilihat pada Gambar 4.7.



Gambar 4.7 *Flowchart* Hapus Tanda Baca & Angka

4.2.1.3. *Case Folding*

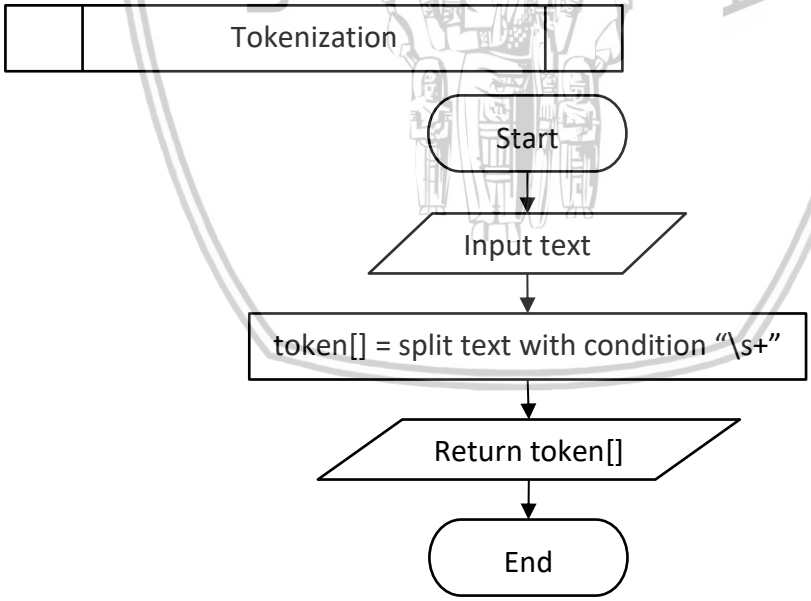
Pada sub proses *case folding* dilakukan untuk mengubah seluruh huruf pada teks hasil sub proses hapus tanda baca dan angka menjadi huruf kecil. *Flowchart* untuk sub proses *case folding* dapat dilihat pada Gambar 4.8.



Gambar 4.8 *Flowchart Case Folding*

4.2.1.4. Tokenisasi

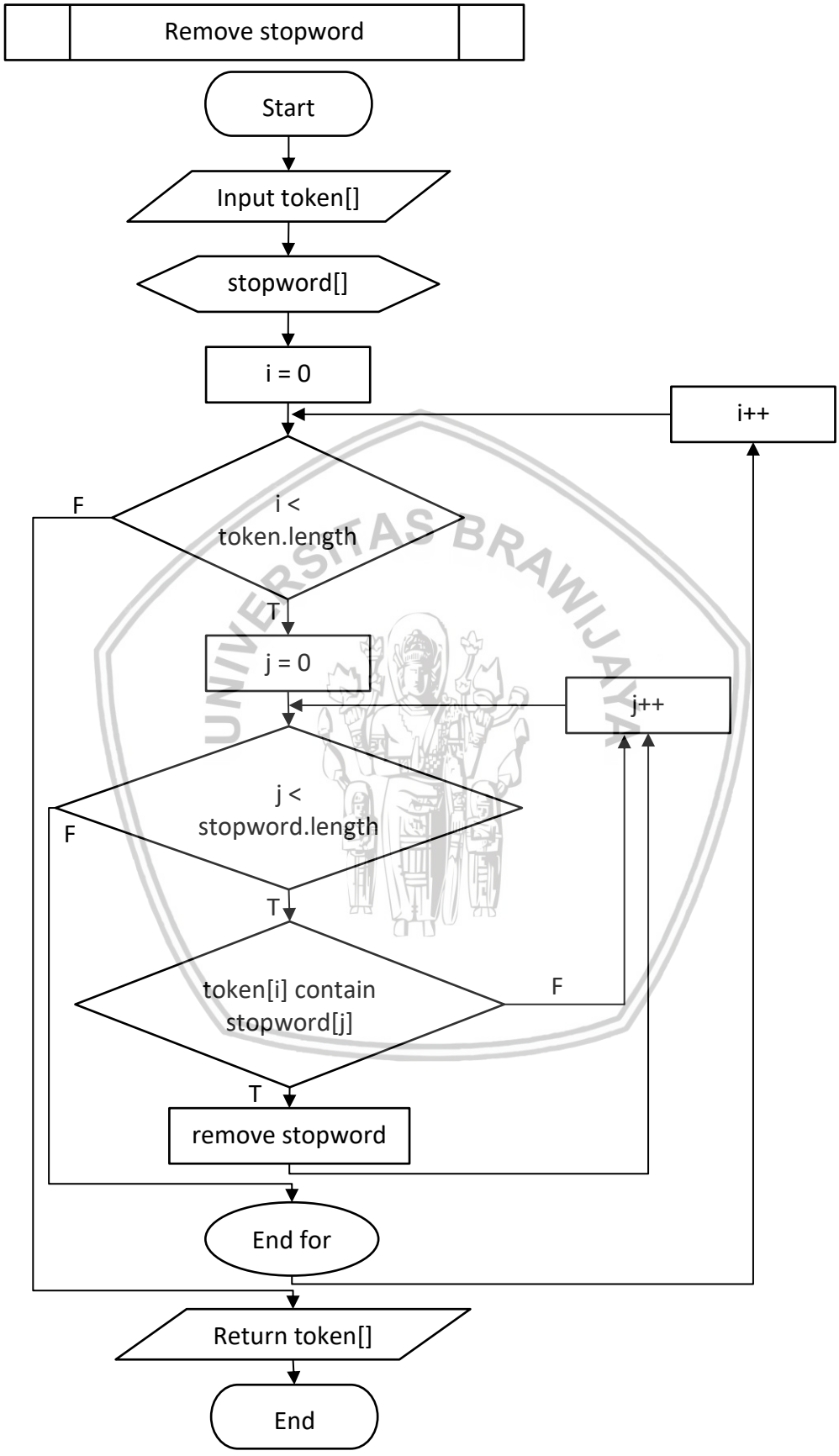
Pada sub proses tokenisasi dilakukan untuk mengubah teks hasil sub proses *case folding* menjadi bentuk kumpulan token. *Flowchart* untuk sub proses tokenisasi dapat dilihat pada Gambar 4.9.



Gambar 4.9 *Flowchart* Tokenisasi

4.2.1.5. Hapus *Stopword*

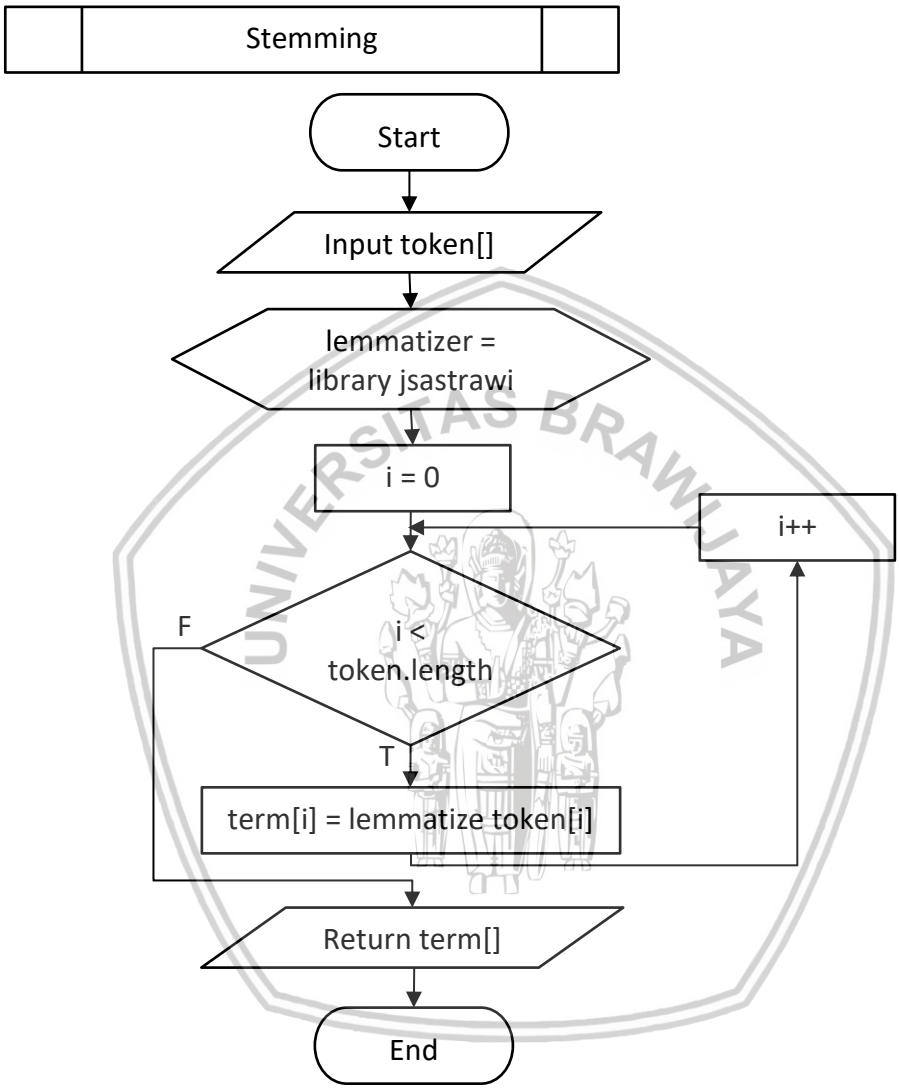
Pada sub proses hapus *stopword* dilakukan untuk menghapus *stopword* yang terdapat dalam token hasil dari sub proses tokenisasi. *Flowchart* untuk sub proses hapus *stopword* dapat dilihat pada Gambar 4.10.



Gambar 4.10 *Flowchart* Hapus *Stopword*

4.2.1.6. *Stemming*

Pada sub proses *stemming* dilakukan untuk mengubah token hasil dari sub proses hapus *stopword* menjadi bentuk kata dasar dengan menggunakan algoritme Nazief dan Adriani melalui bantuan *library jsastrawi*. *Flowchart* untuk sub proses *stemming* dapat dilihat pada Gambar 4.11.

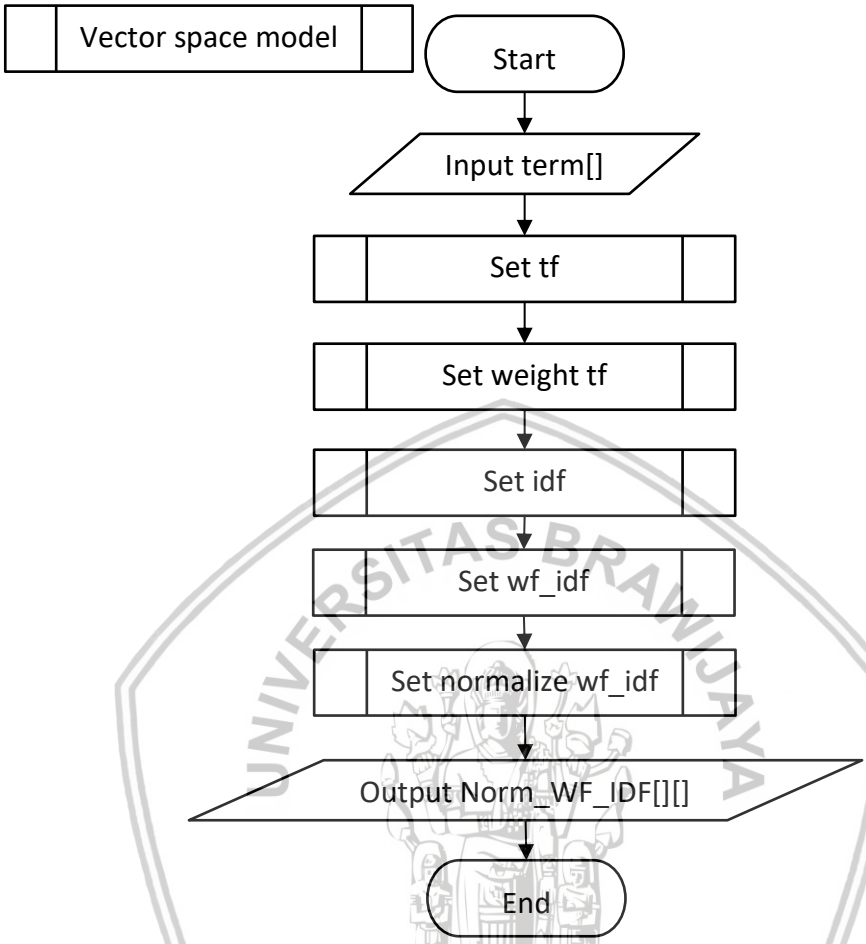


Gambar 4.11 *Flowchart Stemming*

4.2.2 *Vector Space Model*

Pada proses *vector space model* dilakukan beberapa sub proses yang bertujuan untuk memodelkan hasil kata dari praproses teks ke dalam bentuk vektor yang kemudian dilakukan pembobotan kata. Hasil pembobotan kata ini nantinya akan digunakan untuk proses pada tahap berikutnya. Proses *vector space model* terdiri dari beberapa sub proses antara lain adalah menghitung nilai *tf* atau frekuensi kata pada dokumen, menghitung nilai bobot *tf*, menghitung nilai *idf*, menghitung pembobotan kata yang merupakan perkalian bobot *tf* dan *idf* dan menghitung

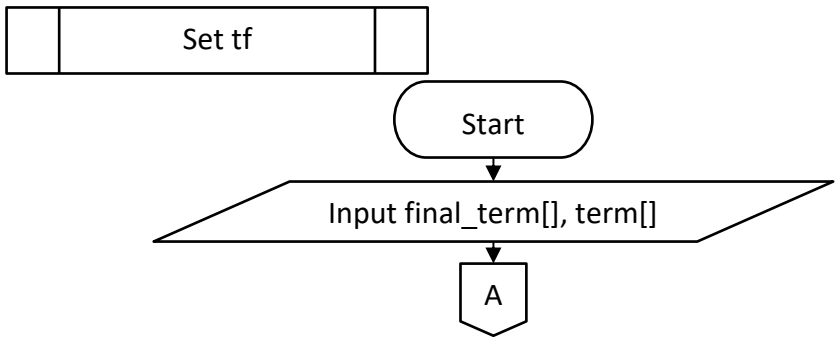
normalisasi pembobotan kata *wf.idf*. *Flowchart* untuk proses *vector space model* dapat dilihat pada Gambar 4.12.



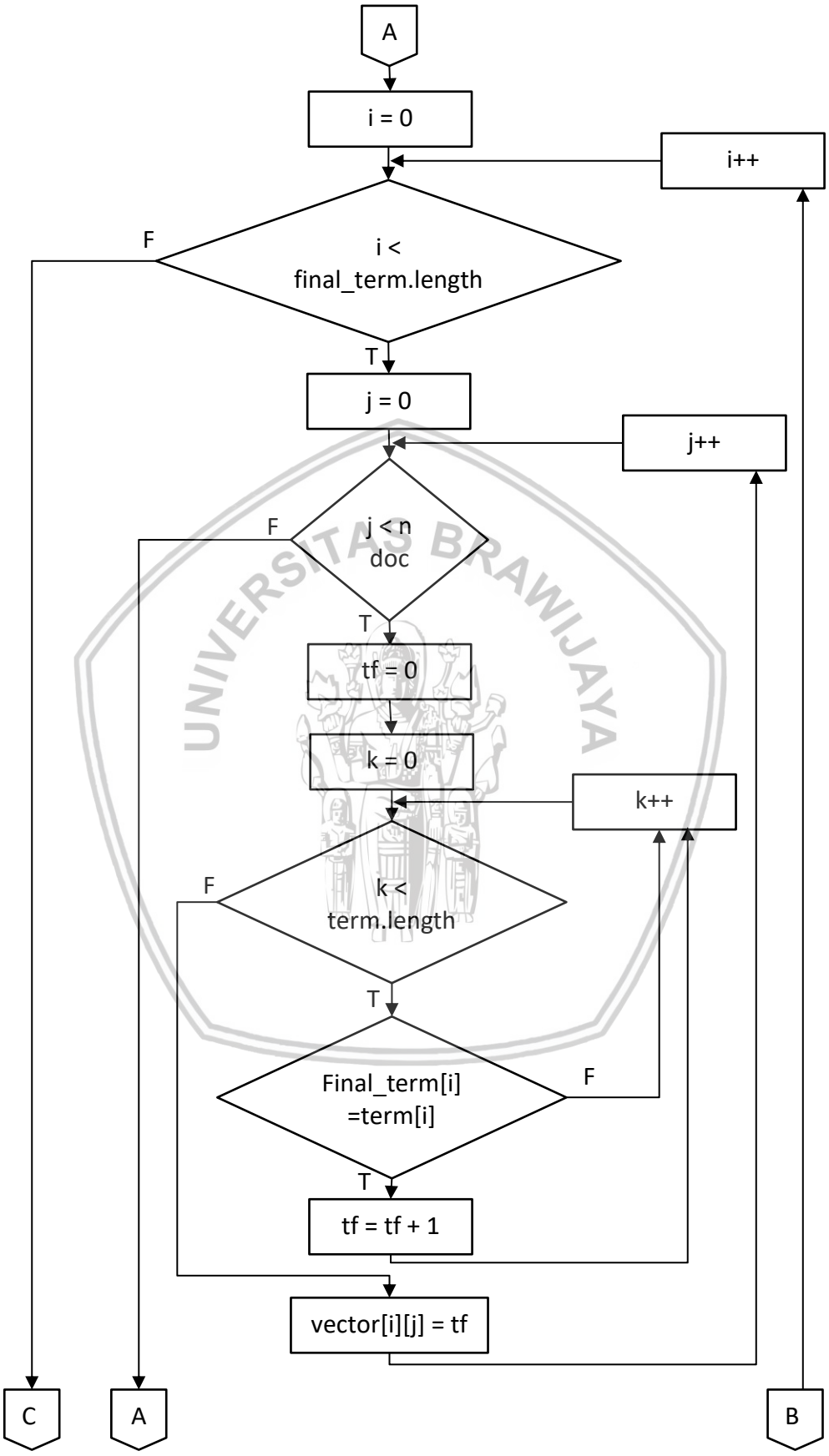
Gambar 4.12 *Flowchart Vector Space Model*

4.2.2.1. Hitung *tf*

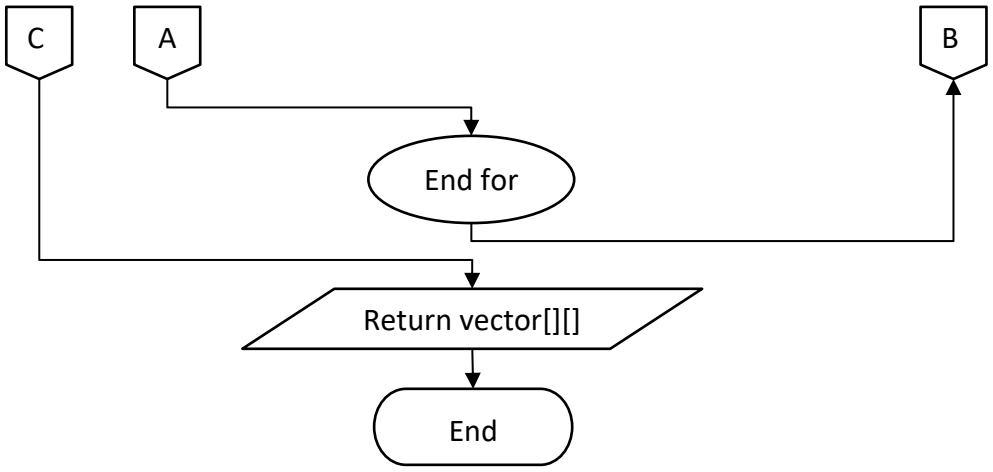
Pada sub proses hitung *tf* dilakukan untuk menghitung frekuensi kata yang muncul pada dokumen, dimana hasil perhitungan ini ditampilkan dalam bentuk model vektor kata-dokumen. *Flowchart* untuk sub proses hitung *tf* dapat dilihat pada Gambar 4.13.



Gambar 4.13 *Flowchart* Hitung *tf*



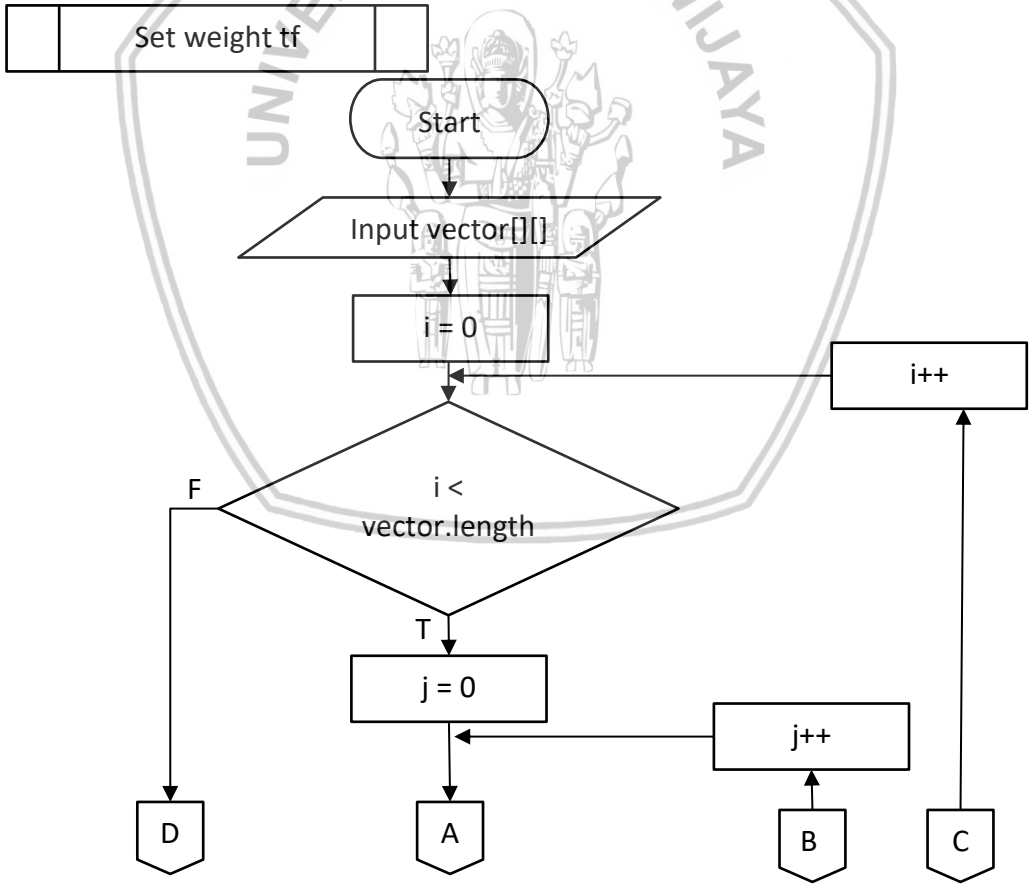
Gambar 4.13 (lanjutan)



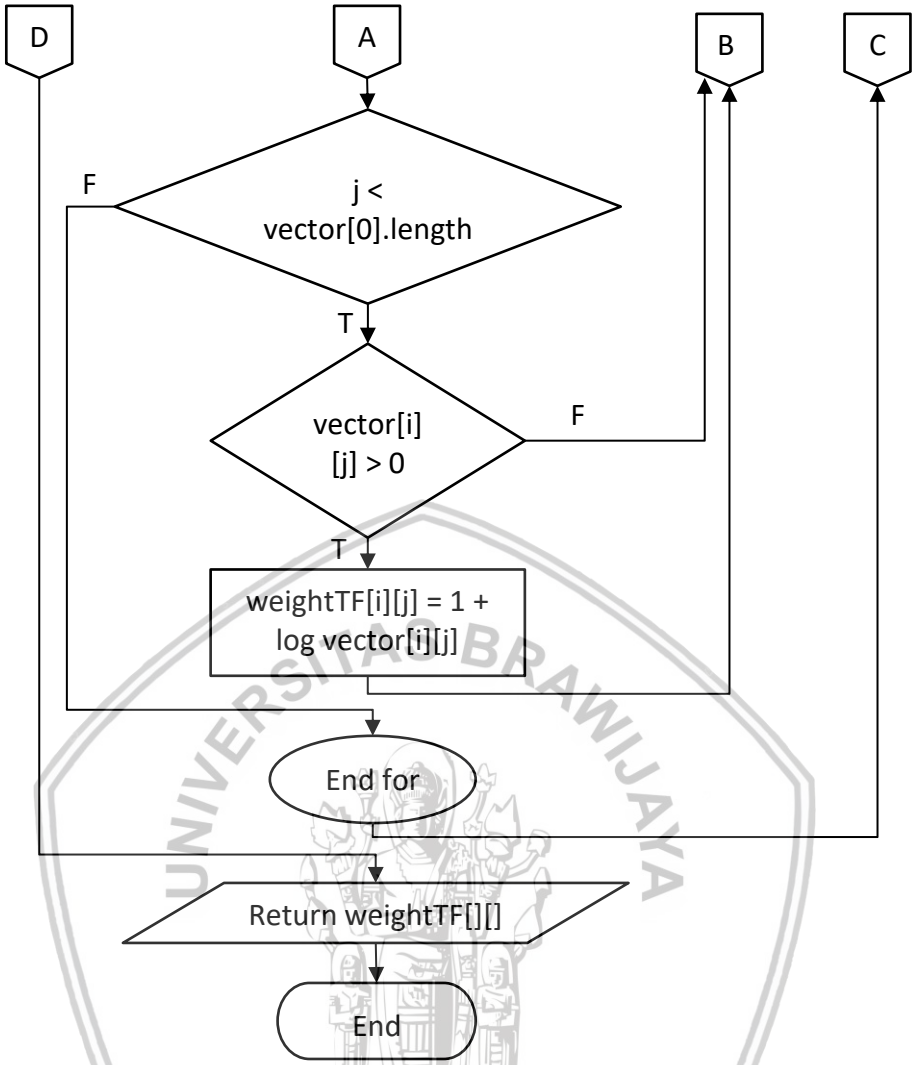
Gambar 4.13 (lanjutan)

4.2.2.2. Hitung Bobot *tf*

Pada sub proses hitung bobot *tf* dilakukan untuk menghitung nilai bobot frekuensi kata yang muncul pada dokumen. *Flowchart* untuk sub proses hitung bobot *tf* dapat dilihat pada Gambar 4.14.



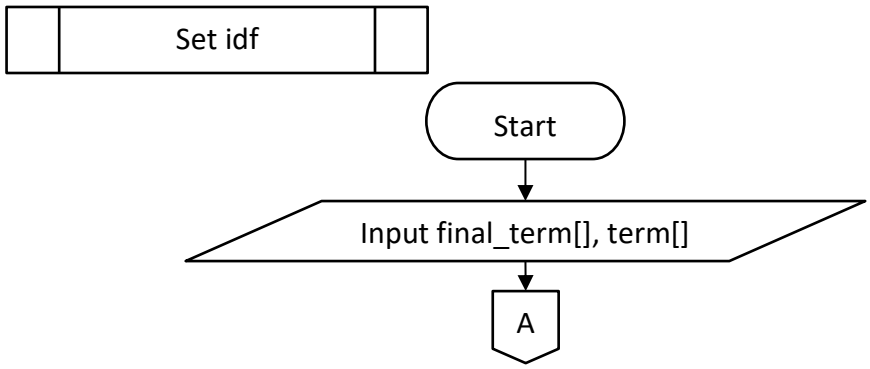
Gambar 4.14 *Flowchart* Hitung Bobot *tf*



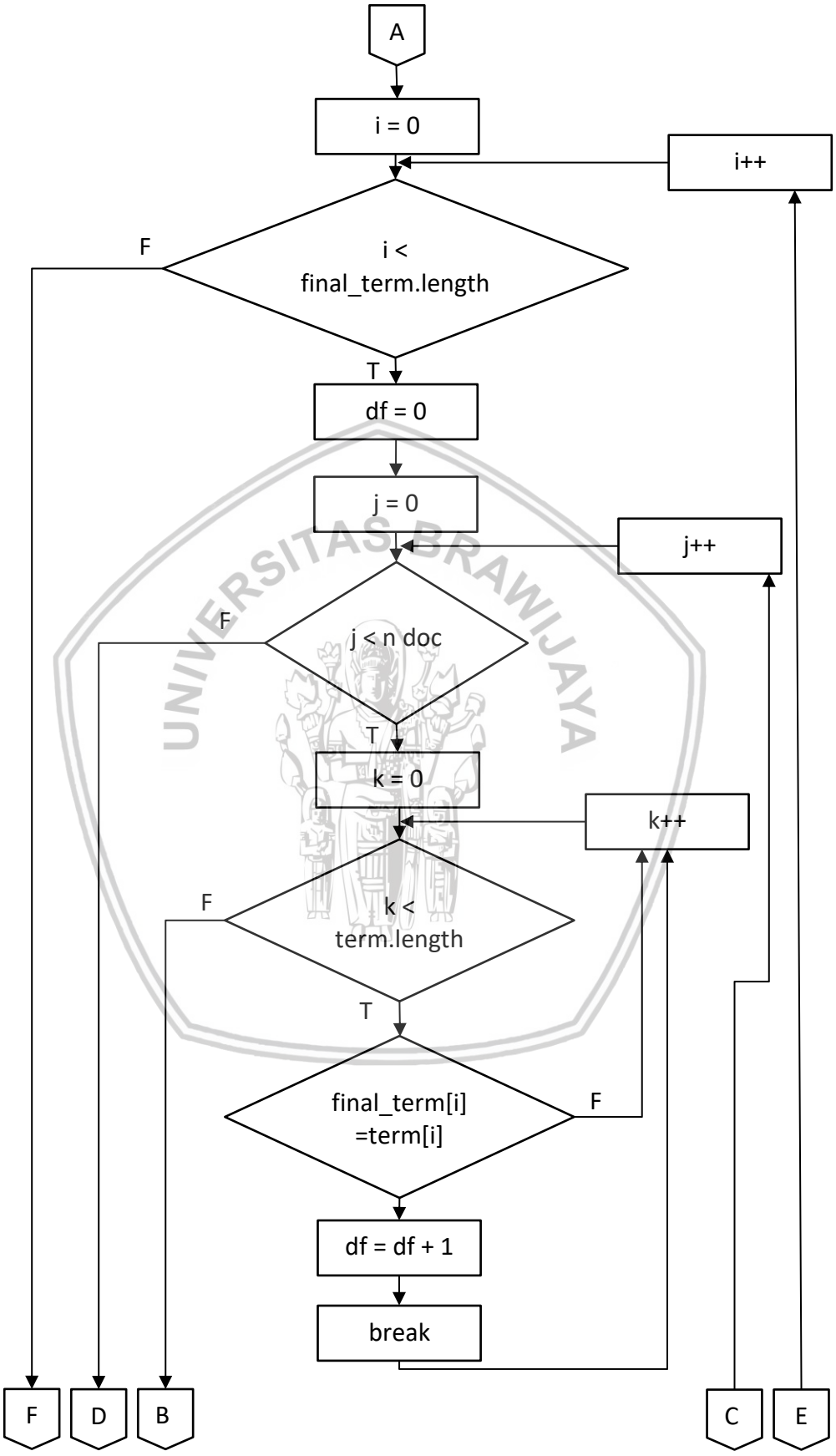
Gambar 4.14 (lanjutan)

4.2.2.3. Hitung *idf*

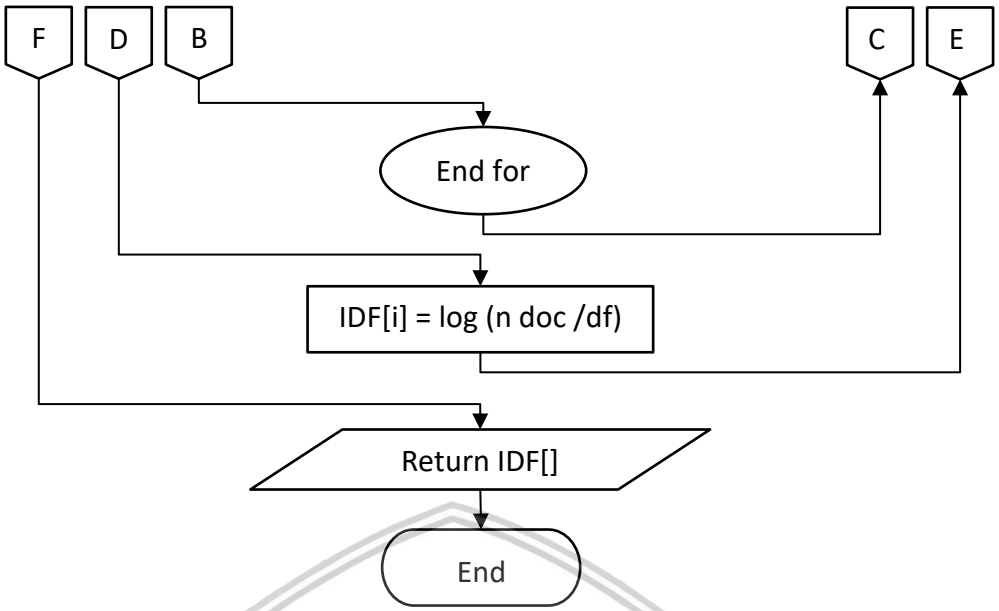
Pada sub proses hitung *idf* dilakukan untuk menghitung nilai *idf* atau seberapa tidak sering suatu kata muncul dalam koleksi keseluruhan dokumen. *Flowchart* untuk sub proses hitung *idf* dapat dilihat pada Gambar 4.15.



Gambar 4.15 *Flowchart* Hitung *idf*



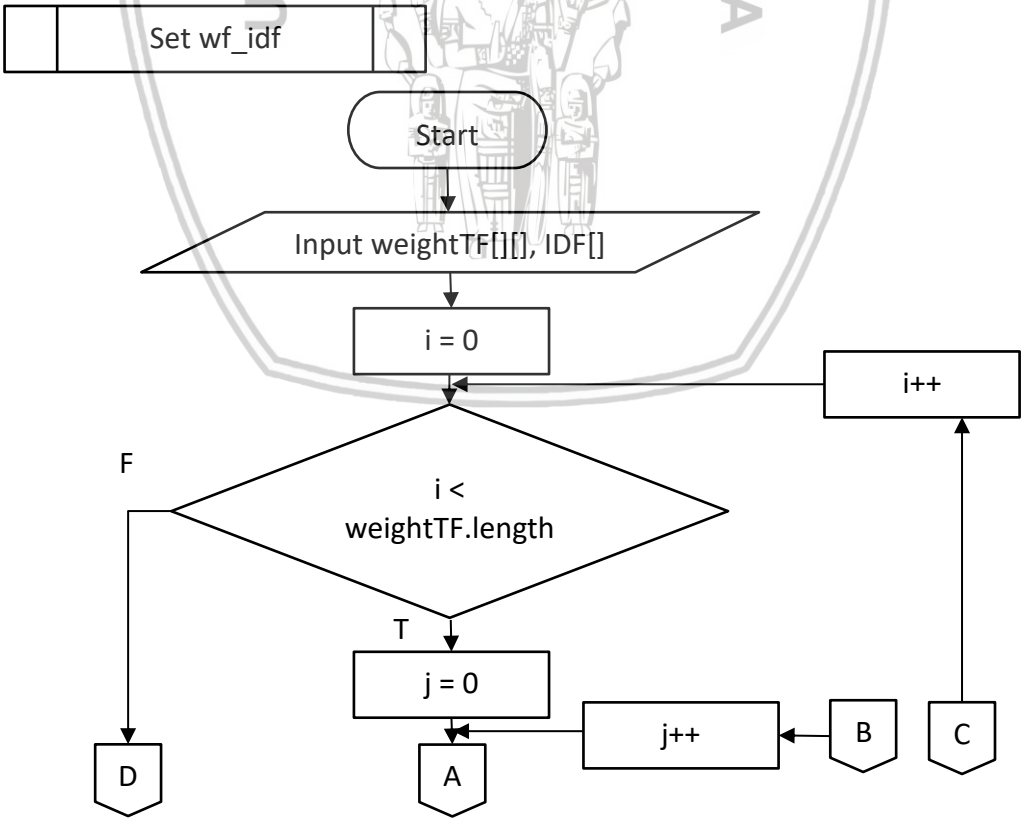
Gambar 4.15 (lanjutan)



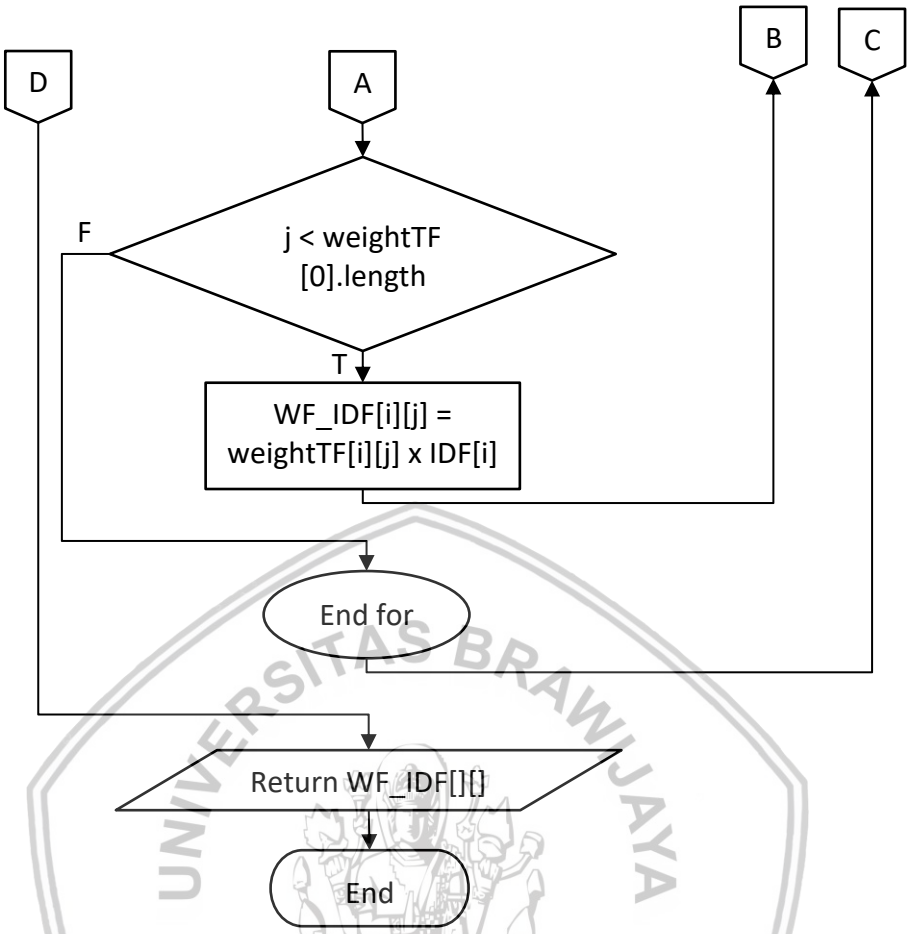
Gambar 4.15 (lanjutan)

4.2.2.4. Hitung *wf.idf*

Pada sub proses hitung *wf.idf* dilakukan untuk menghitung pembobotan kata yang merupakan perkalian antara nilai bobot *tf* dan *idf*. *Flowchart* untuk sub proses hitung *wf.idf* dapat dilihat pada Gambar 4.16.



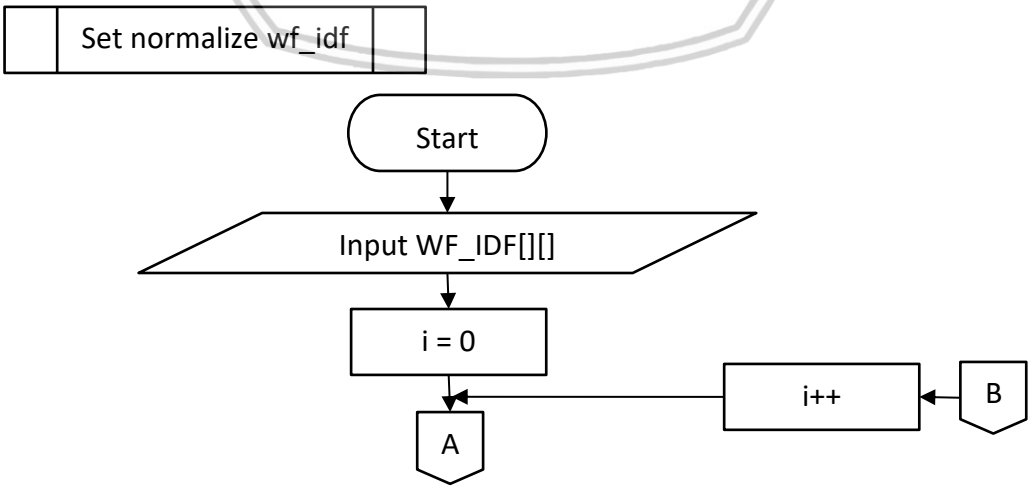
Gambar 4.16 *Flowchart* Hitung *wf.idf*



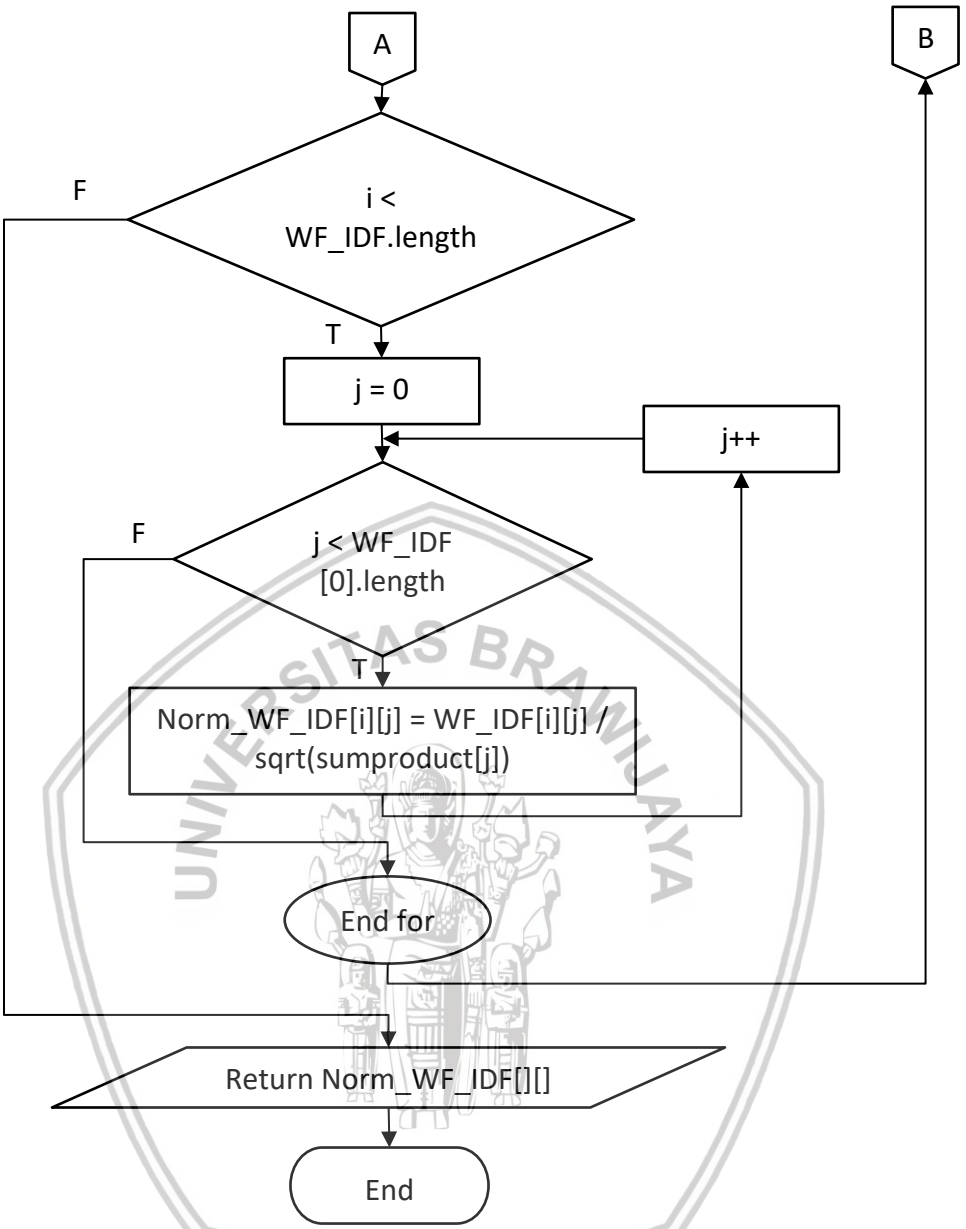
Gambar 4.16 (lanjutan)

4.2.2.5. Hitung Normalisasi *wf.idf*

Pada sub proses hitung normalisasi *wf.idf* dilakukan untuk menghitung nilai normalisasi dari pembobotan kata *wf.idf*. *Flowchart* untuk sub proses hitung normalisasi *wf.idf* dapat dilihat pada Gambar 4.17.



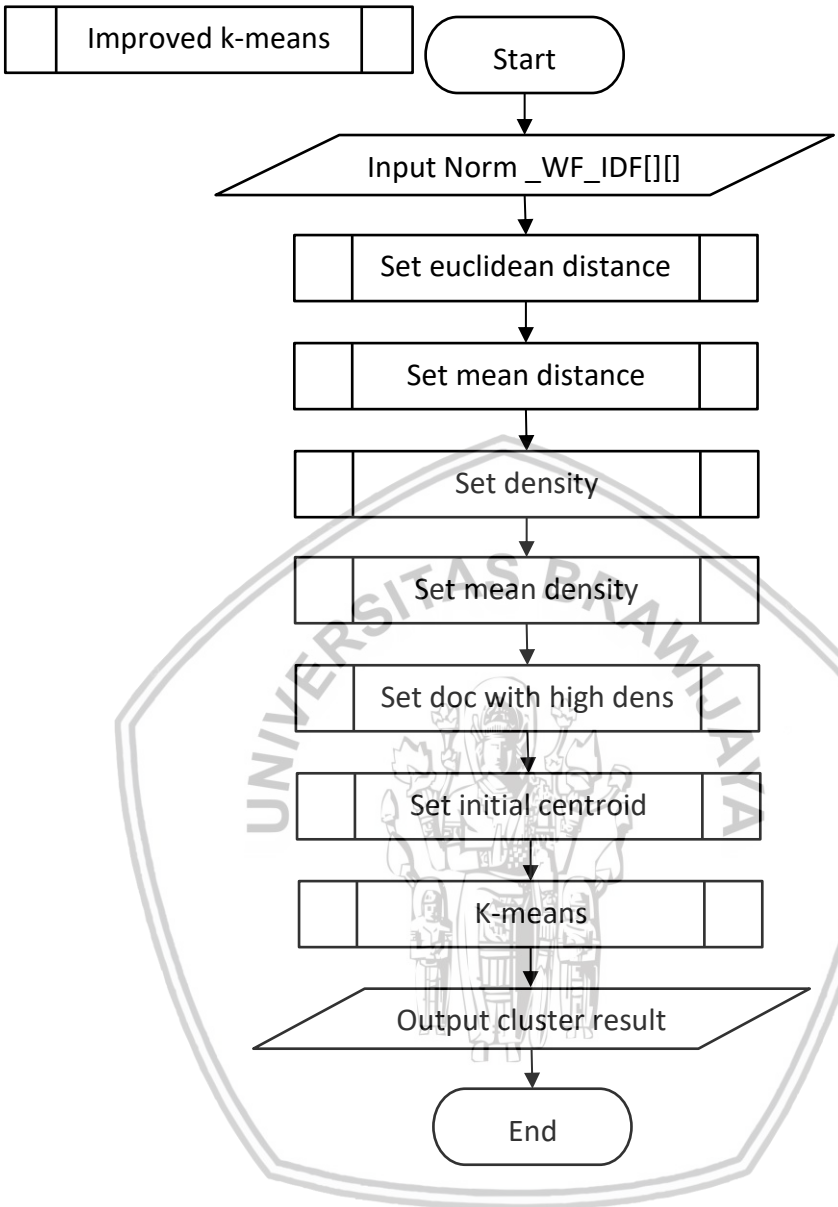
Gambar 4.17 *Flowchart* Hitung Normalisasi *wf.idf*



Gambar 4.17 (lanjutan)

4.2.3 *Improved K-Means*

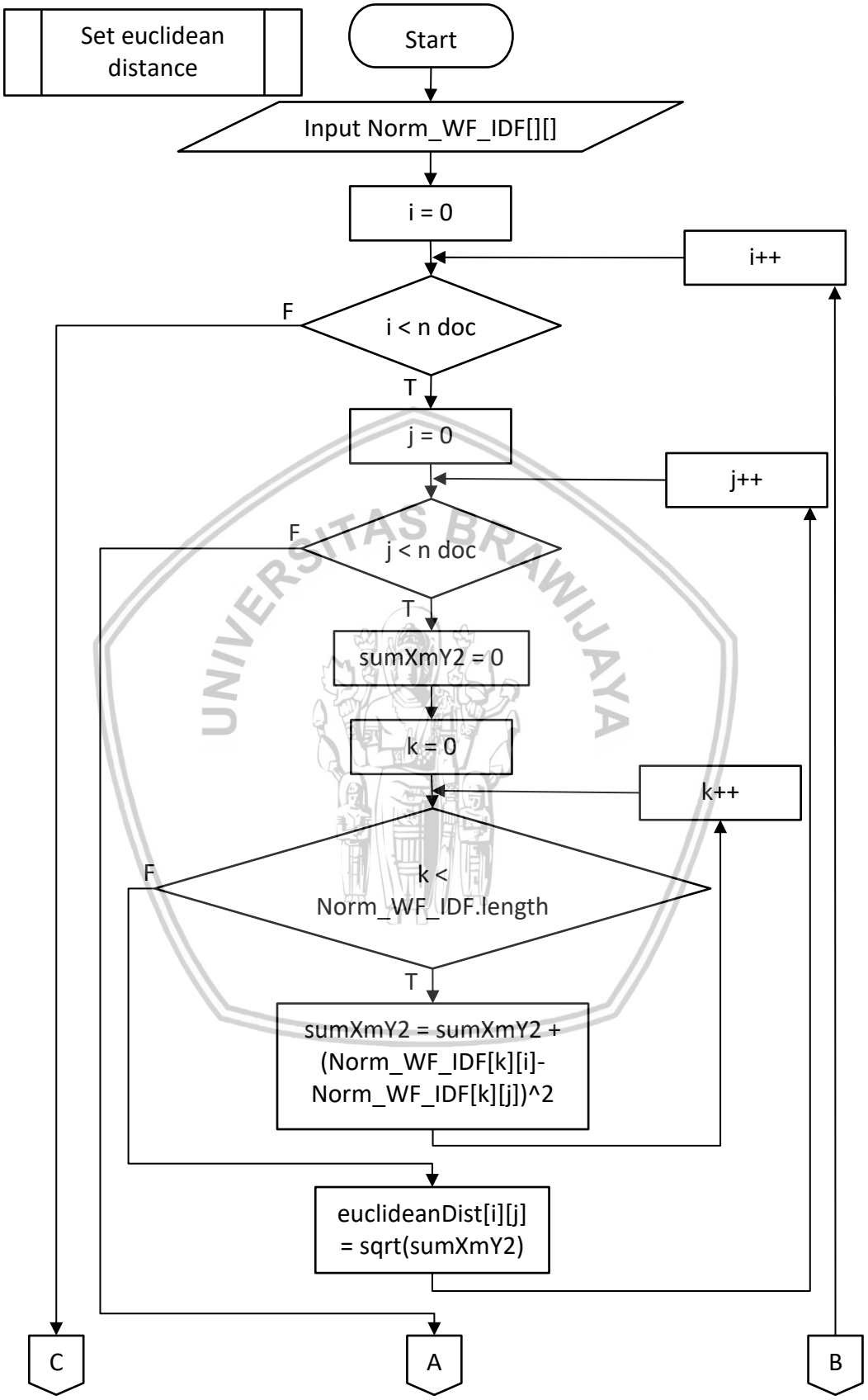
Pada proses *improved k-means* dilakukan beberapa sub proses yang bertujuan untuk melakukan pengelompokan dokumen J-PTIIK. Pengelompokan dilakukan dengan menggunakan hasil pembobotan kata *wf.idf* yang telah dilakukan pada proses *vector space model*. Proses *improved k-means* terdiri dari dua sub proses antara lain adalah menentukan *centroid* awal klaster dan penggunaan *k-means* dalam mengelompokkan dokumen J-PTIIK. *Flowchart* untuk proses *improved k-means* dapat dilihat pada Gambar 4.18.



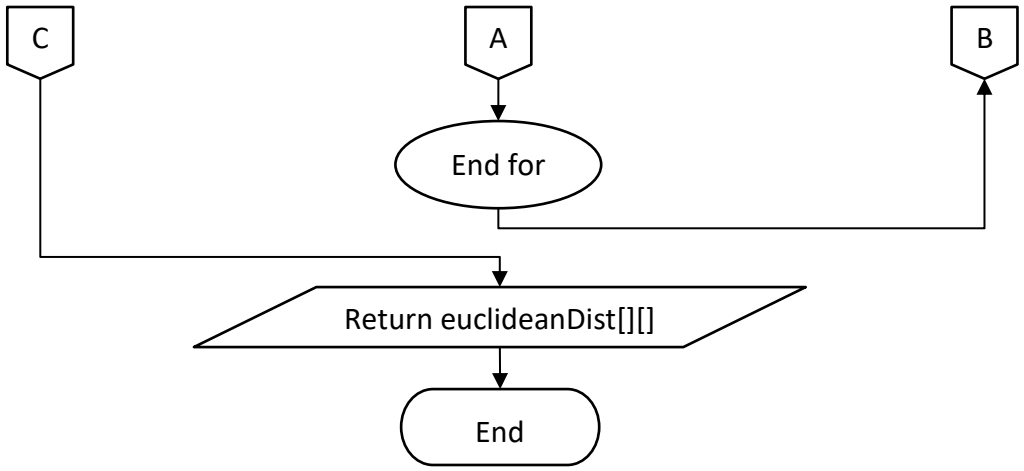
Gambar 4.18 *Flowchart Improved K-Means*

4.2.3.1. Hitung Jarak Tiap Dokumen

Pada sub proses ini dilakukan untuk menghitung jarak antar tiap dokumen dengan menggunakan persamaan *Euclidean distance*. *Flowchart* untuk sub proses hitung jarak tiap dokumen dapat dilihat pada Gambar 4.19.



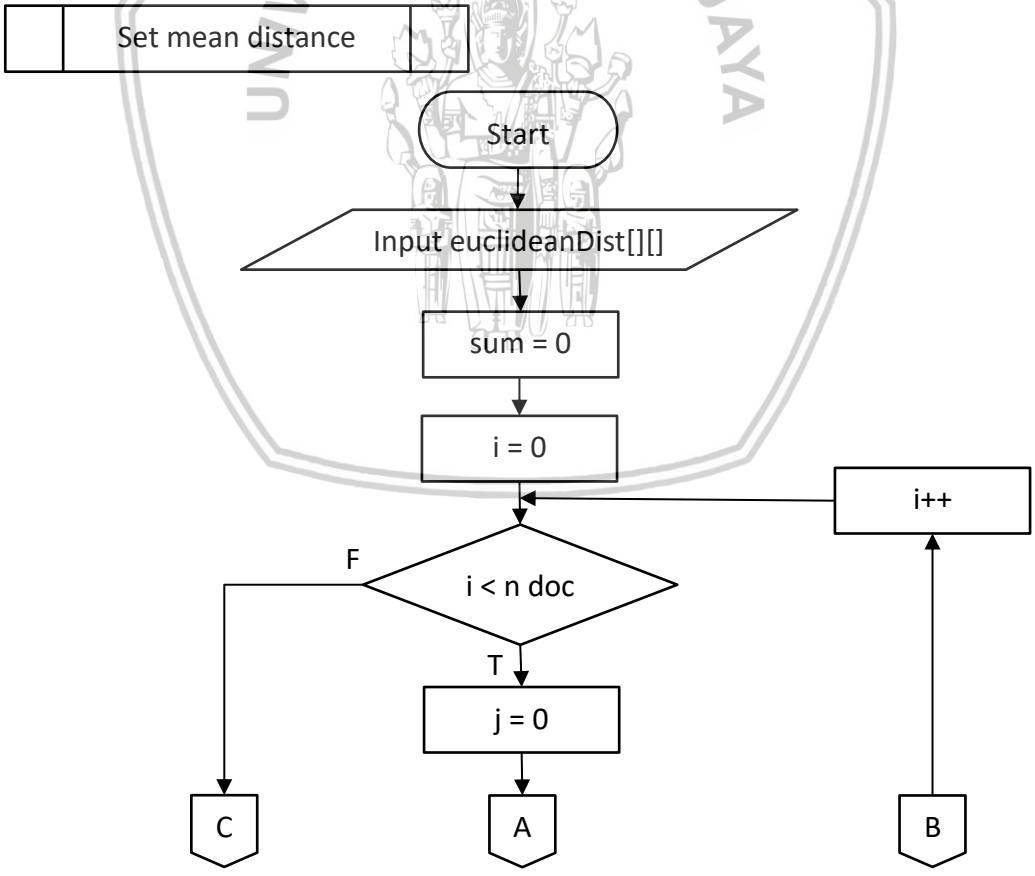
Gambar 4.19 *Flowchart* Hitung Jarak Tiap Dokumen



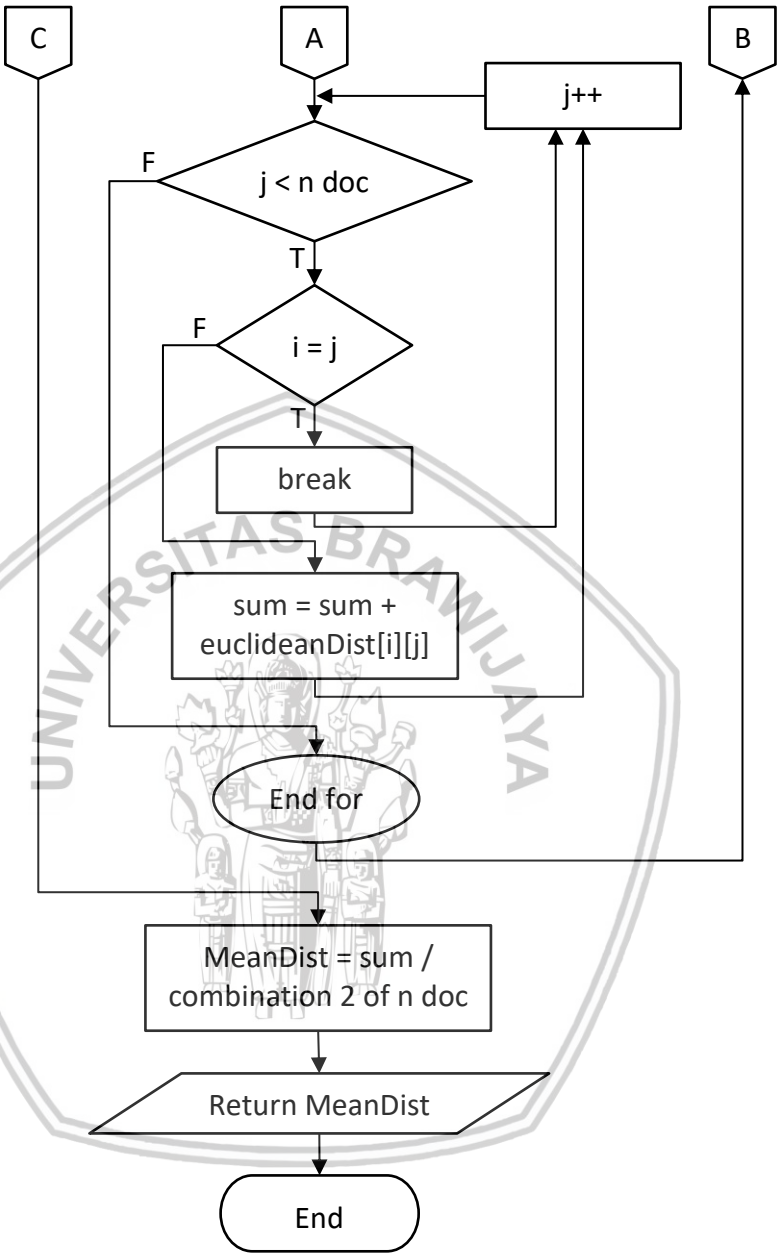
Gambar 4.19 (lanjutan)

4.2.3.2. Hitung Rata-rata Jarak

Pada sub proses ini dilakukan untuk menghitung rata-rata jarak dokumen, dimana hanya jarak antar dokumen yang berbeda saja digunakan dan penggunaan jarak antar dokumen juga hanya diperbolehkan sekali. *Flowchart* untuk sub proses hitung rata-rata jarak dokumen dapat dilihat pada Gambar 4.20.



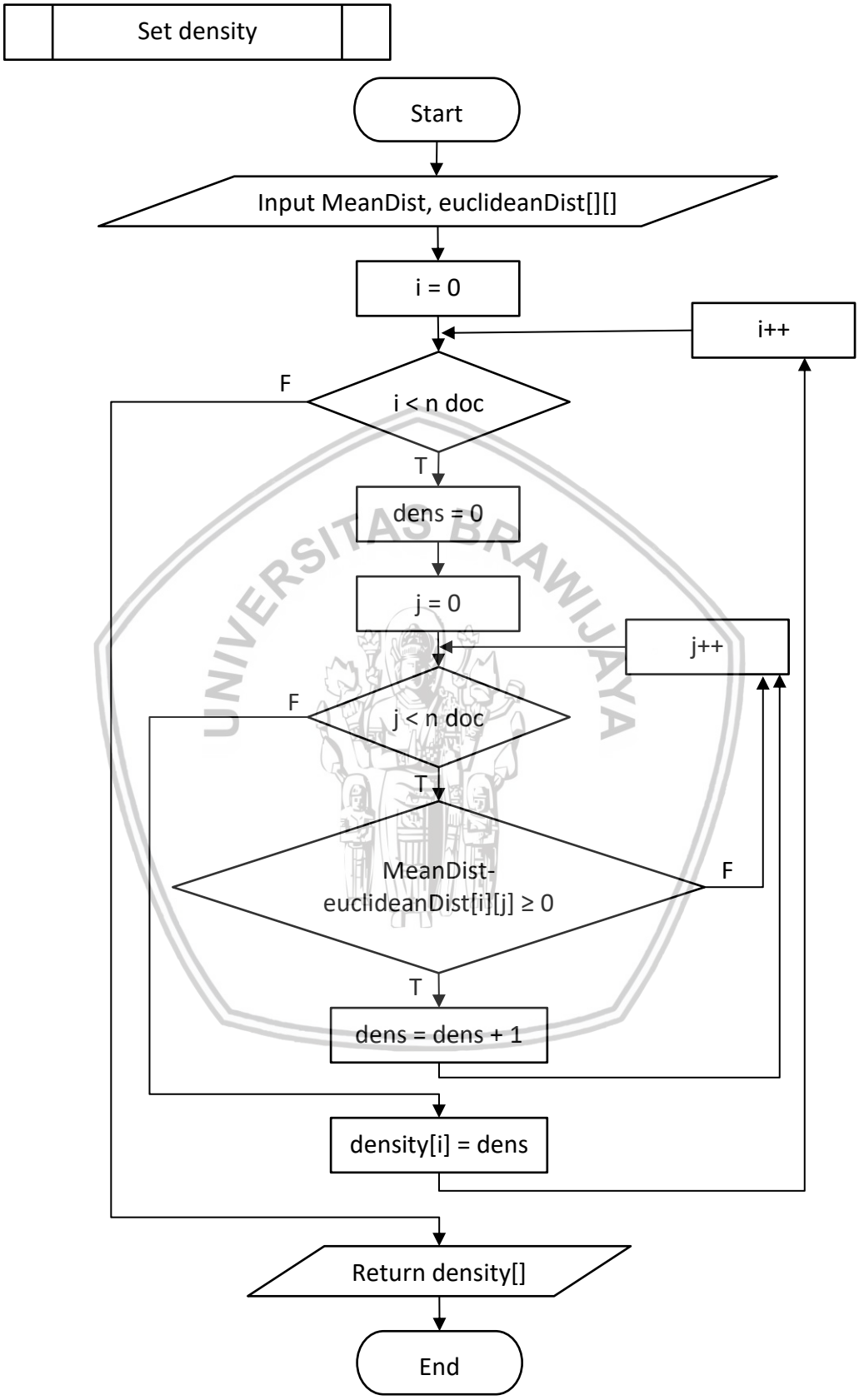
Gambar 4.20 *Flowchart* Hitung Rata-rata Jarak



Gambar 4.20 (lanjutan)

4.2.3.3. Hitung Densitas Dokumen

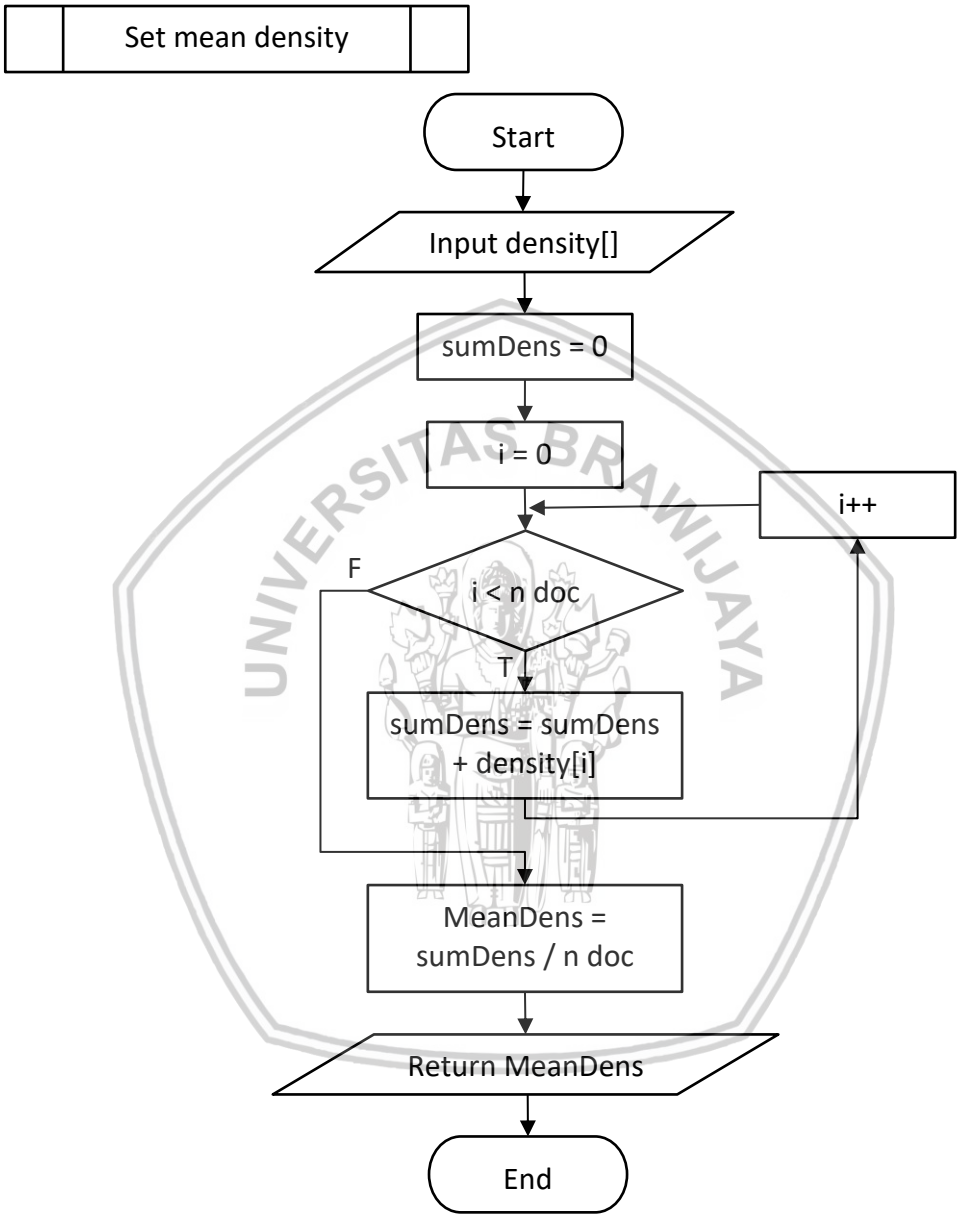
Pada sub proses ini dilakukan untuk menghitung nilai parameter densitas dari masing-masing dokumen. *Flowchart* untuk sub proses hitung densitas dokumen dapat dilihat pada Gambar 4.21.



Gambar 4.21 *Flowchart* Hitung Densitas Dokumen

4.2.3.4. Hitung Rata-rata Densitas

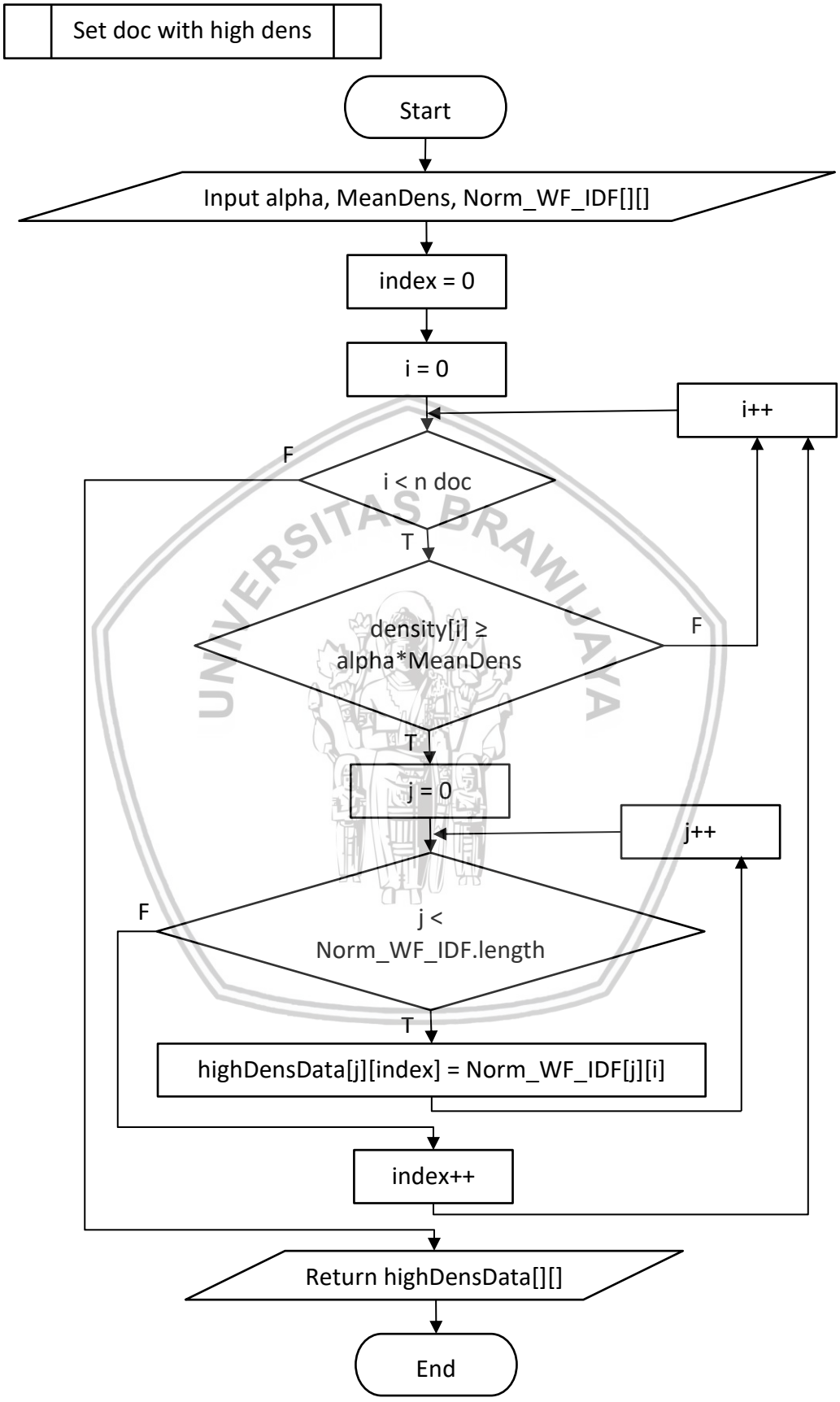
Pada sub proses ini dilakukan untuk menghitung rata-rata dari nilai parameter densitas dokumen. *Flowchart* untuk sub proses hitung rata-rata densitas dapat dilihat pada Gambar 4.22.



Gambar 4.22 *Flowchart* Hitung Rata-rata Densitas

4.2.3.5. Menentukan Dokumen dengan Densitas Tertinggi

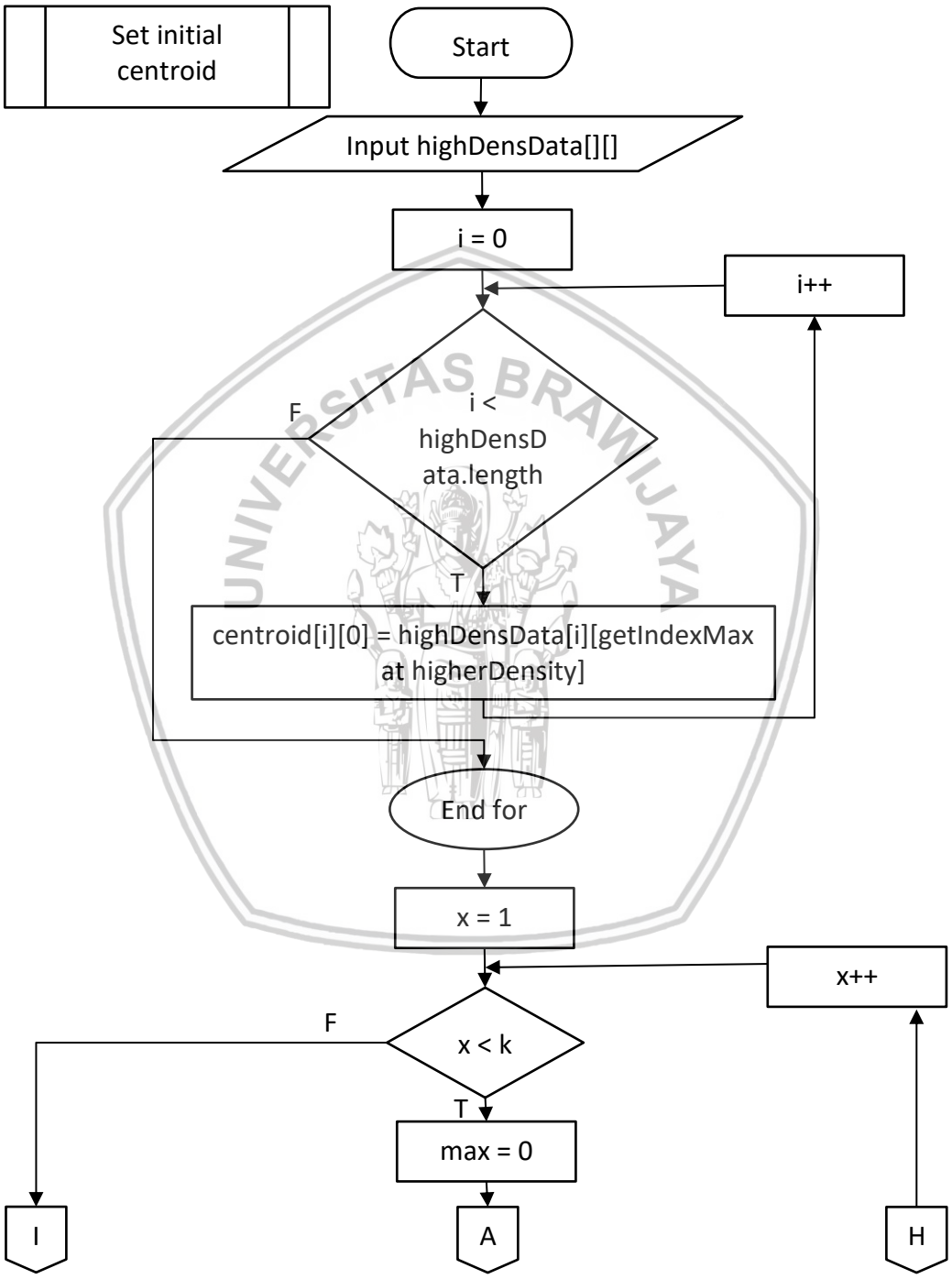
Pada sub proses ini dilakukan untuk menentukan dokumen yang memiliki nilai densitas tertinggi, dimana dokumen ini diperoleh dari dokumen yang memiliki densitas lebih besar dari nilai batas yang merupakan perkalian antara α dan rata-rata densitas. *Flowchart* untuk sub proses menentukan dokumen dengan densitas tertinggi dapat dilihat pada Gambar 4.23.



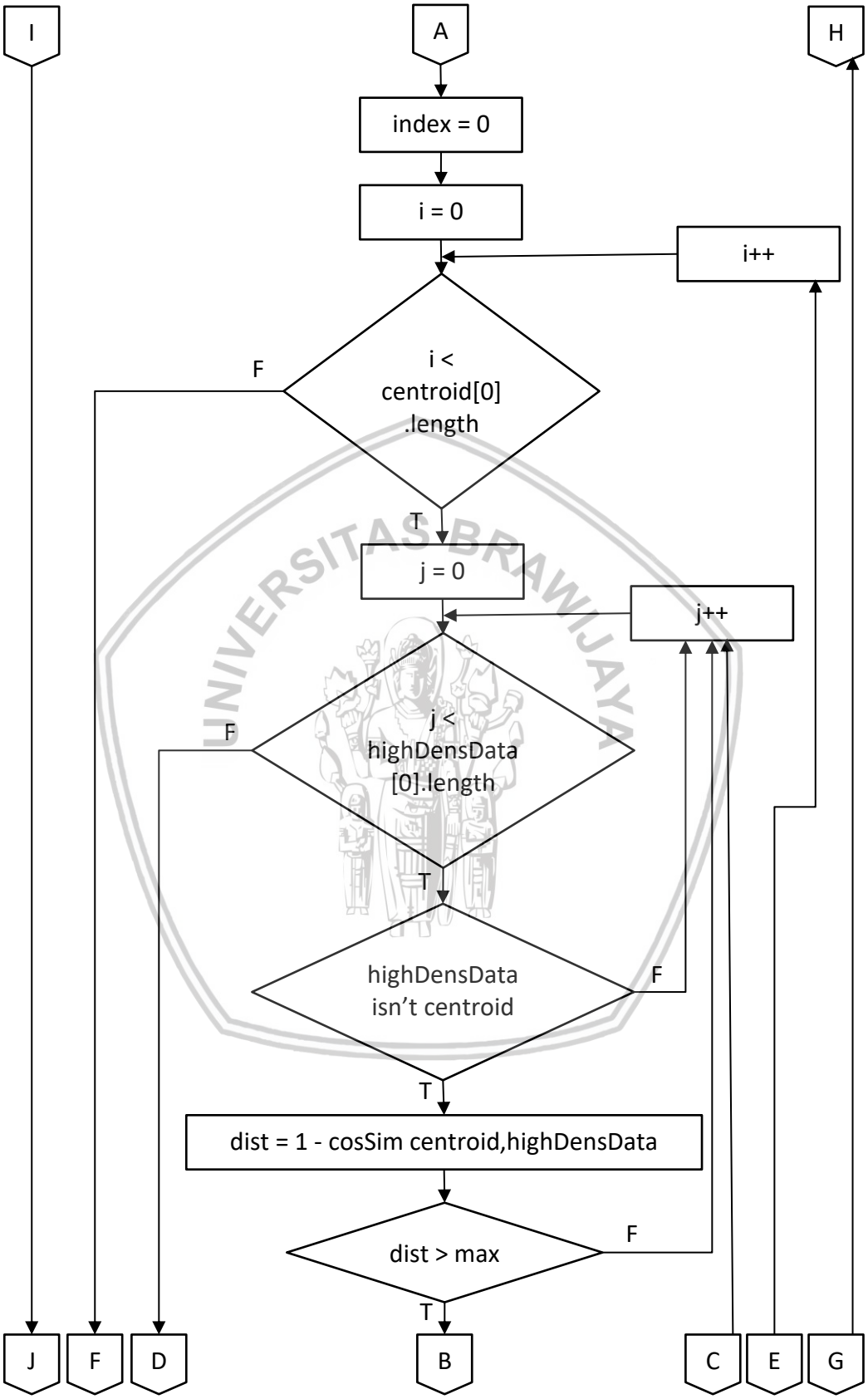
Gambar 4.23 *Flowchart* Menentukan Dokumen dengan Densitas Tertinggi

4.2.3.6. Menentukan *Centroid* Awal Klaster

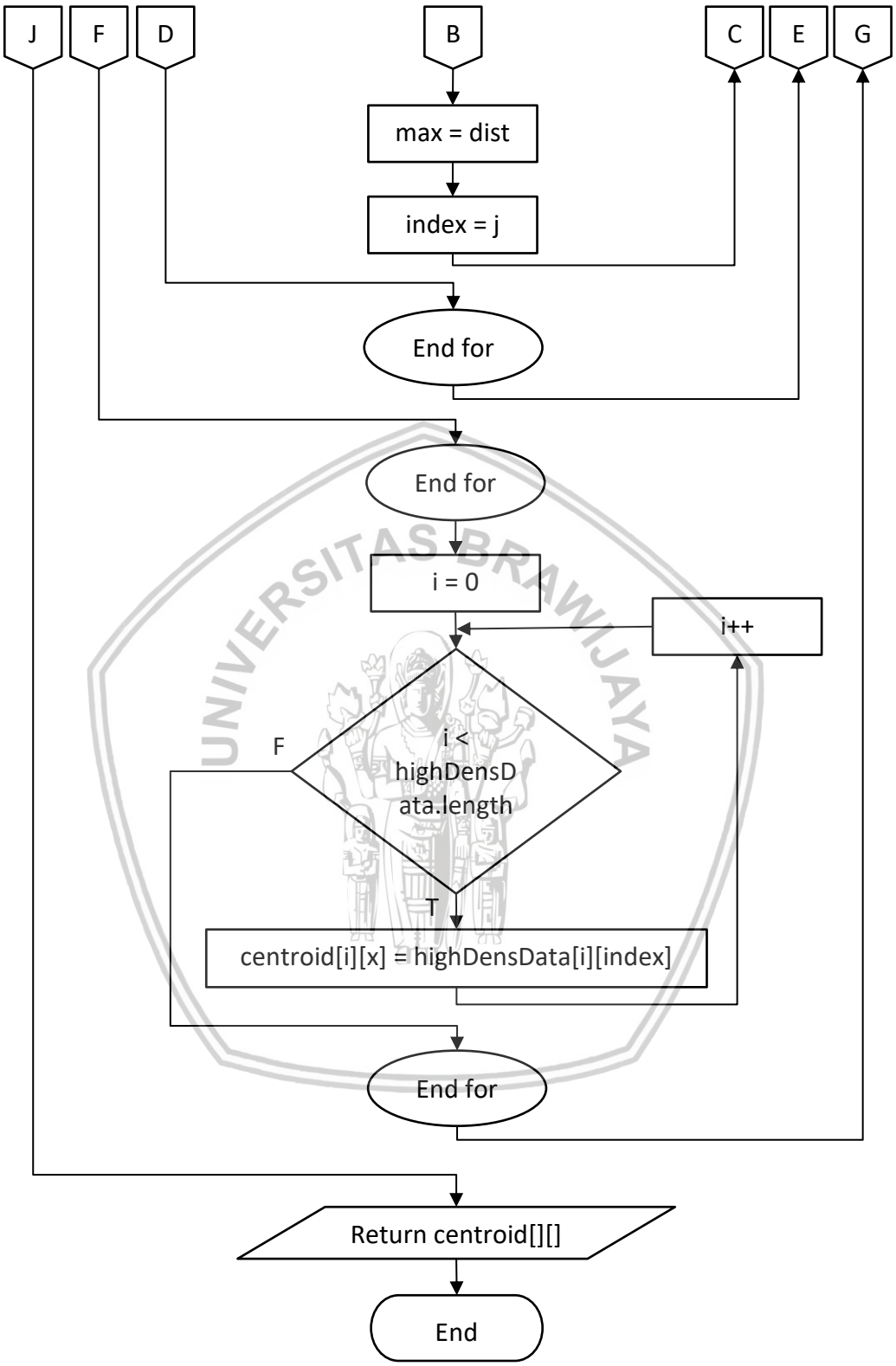
Pada sub proses ini dilakukan untuk menentukan *centroid* awal klaster yang akan digunakan pada proses pengelompokan dengan menggunakan metode *k-means*. *Flowchart* untuk sub proses menentukan *centroid* awal klaster dapat dilihat pada Gambar 4.24.



Gambar 4.24 *Flowchart* Menentukan *Centroid* Awal Klaster



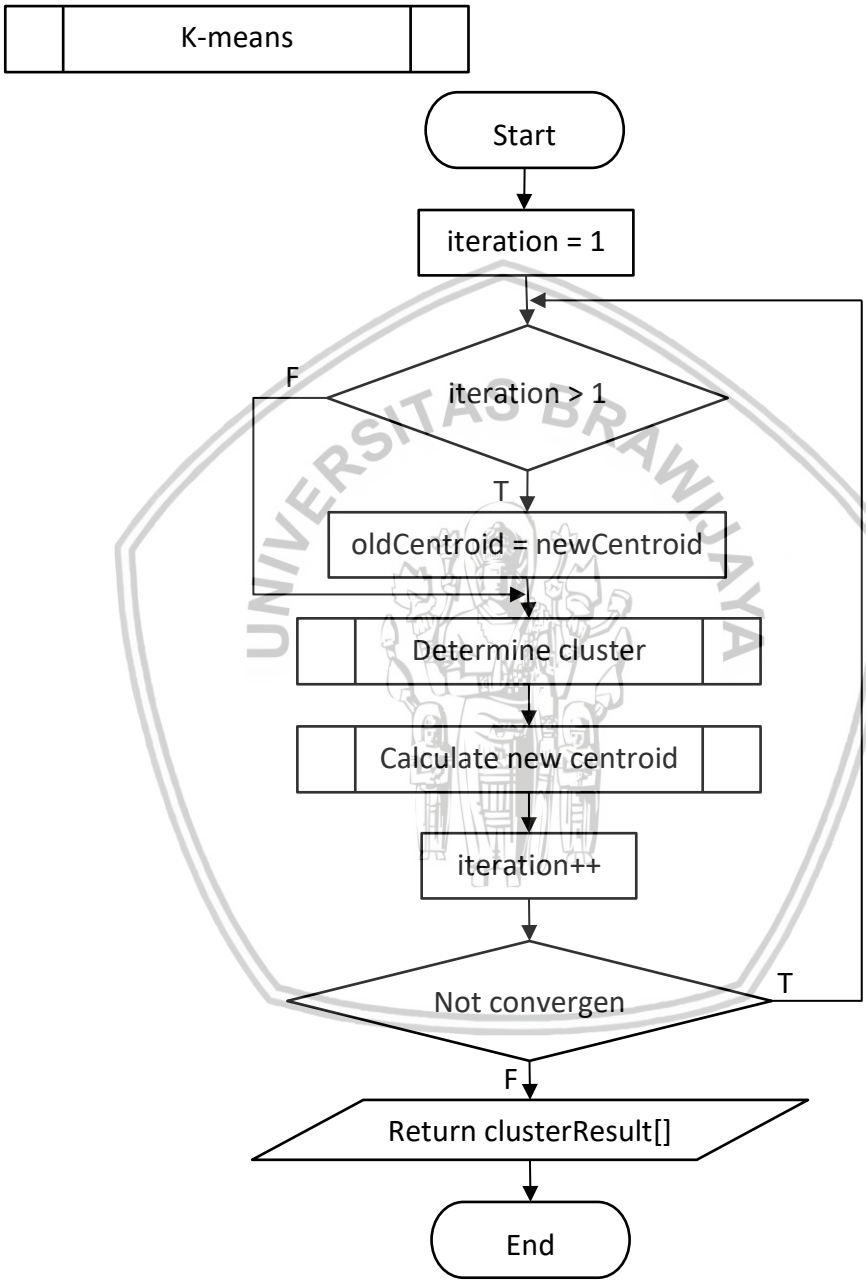
Gambar 4.24 (lanjutan)



Gambar 4.24 (lanjutan)

4.2.3.7. K-Means

Pada sub proses *k-means* dilakukan untuk melakukan pengelompokan dokumen J-PTIIK dengan menggunakan metode *k-means*. Dimana *centroid* awal yang digunakan adalah *centroid* yang telah diperoleh pada sub proses sebelumnya. *Flowchart* untuk sub proses *k-means* dapat dilihat pada Gambar 4.25.

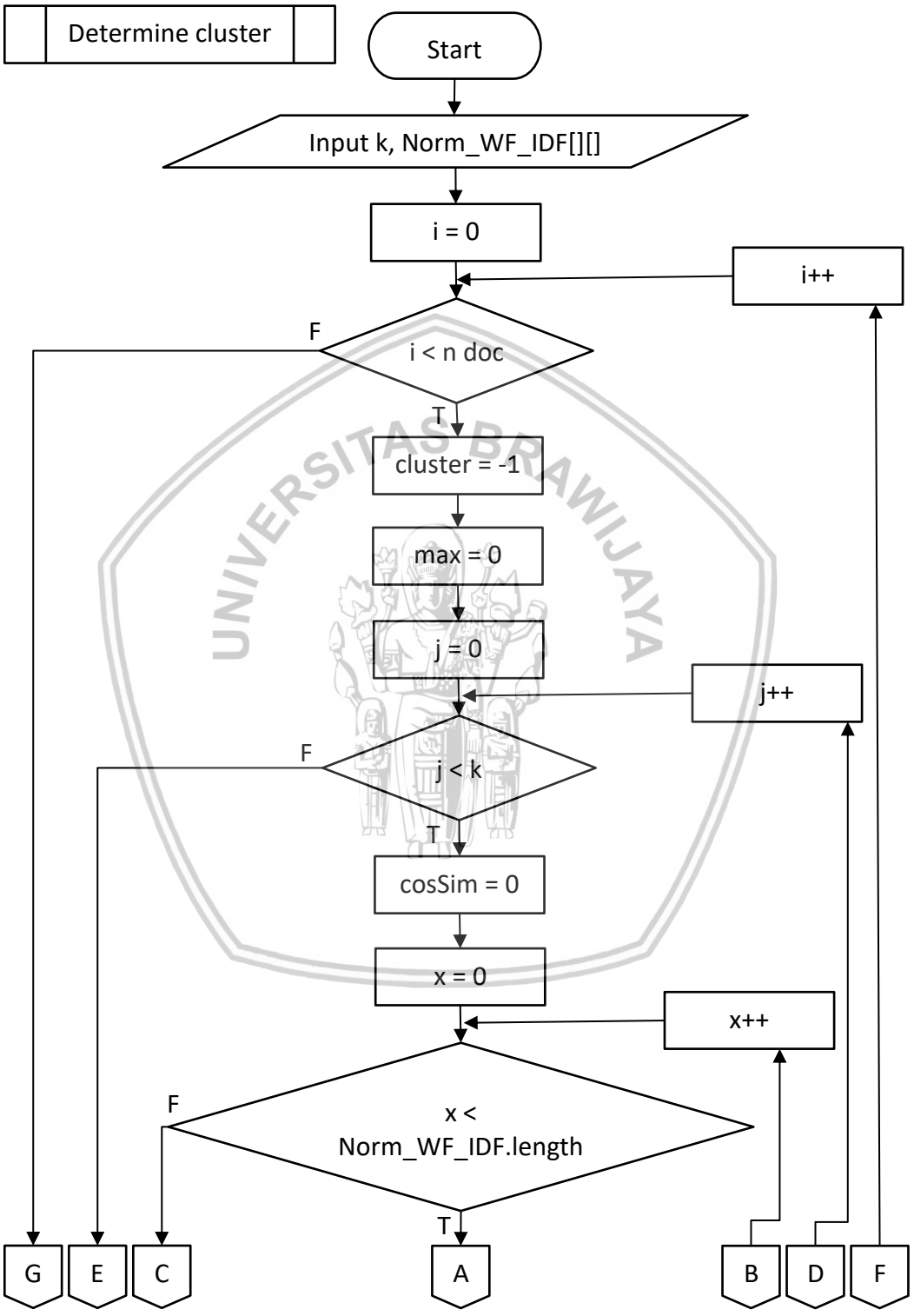


Gambar 4.25 *Flowchart K-Means*

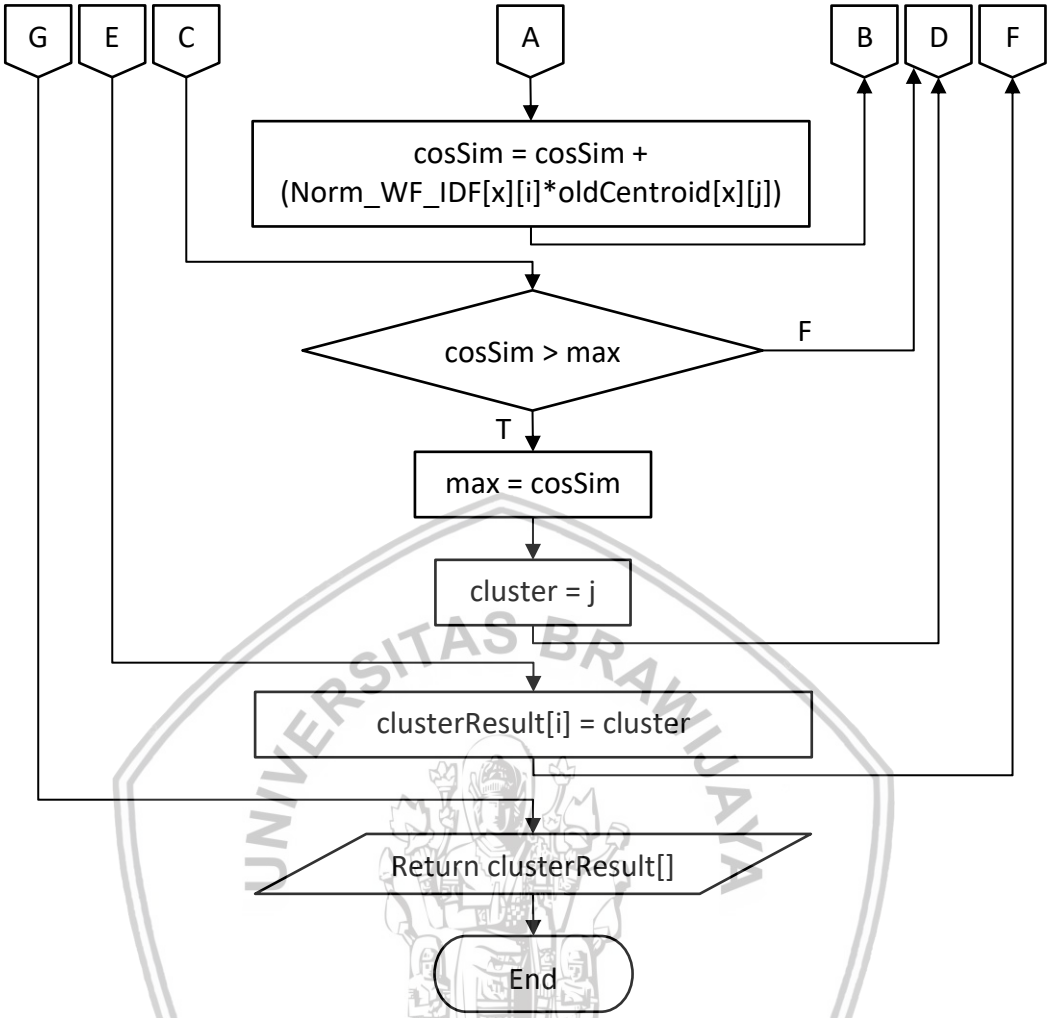
1. Menentukan Klaster Dokumen

Pada proses ini dilakukan untuk menentukan suatu dokumen berada di posisi klaster mana. Persamaan yang digunakan untuk menentukan klaster adalah nilai cosine similarity terbesar pada masing-masing dokumen terhadap *centroid* awal

yang telah diperoleh pada proses sebelumnya. *Flowchart* untuk proses menentukan klaster dokumen dalam pengelompokan dapat dilihat pada Gambar 4.26.



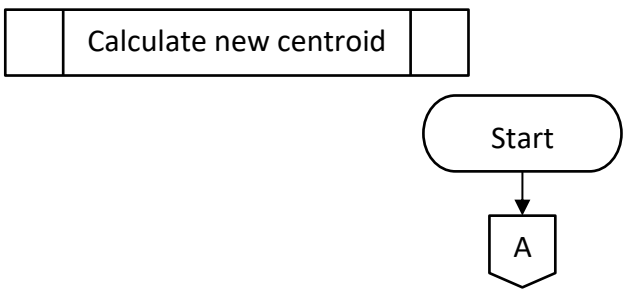
Gambar 4.26 *Flowchart* Menentukan Klaster Dokumen



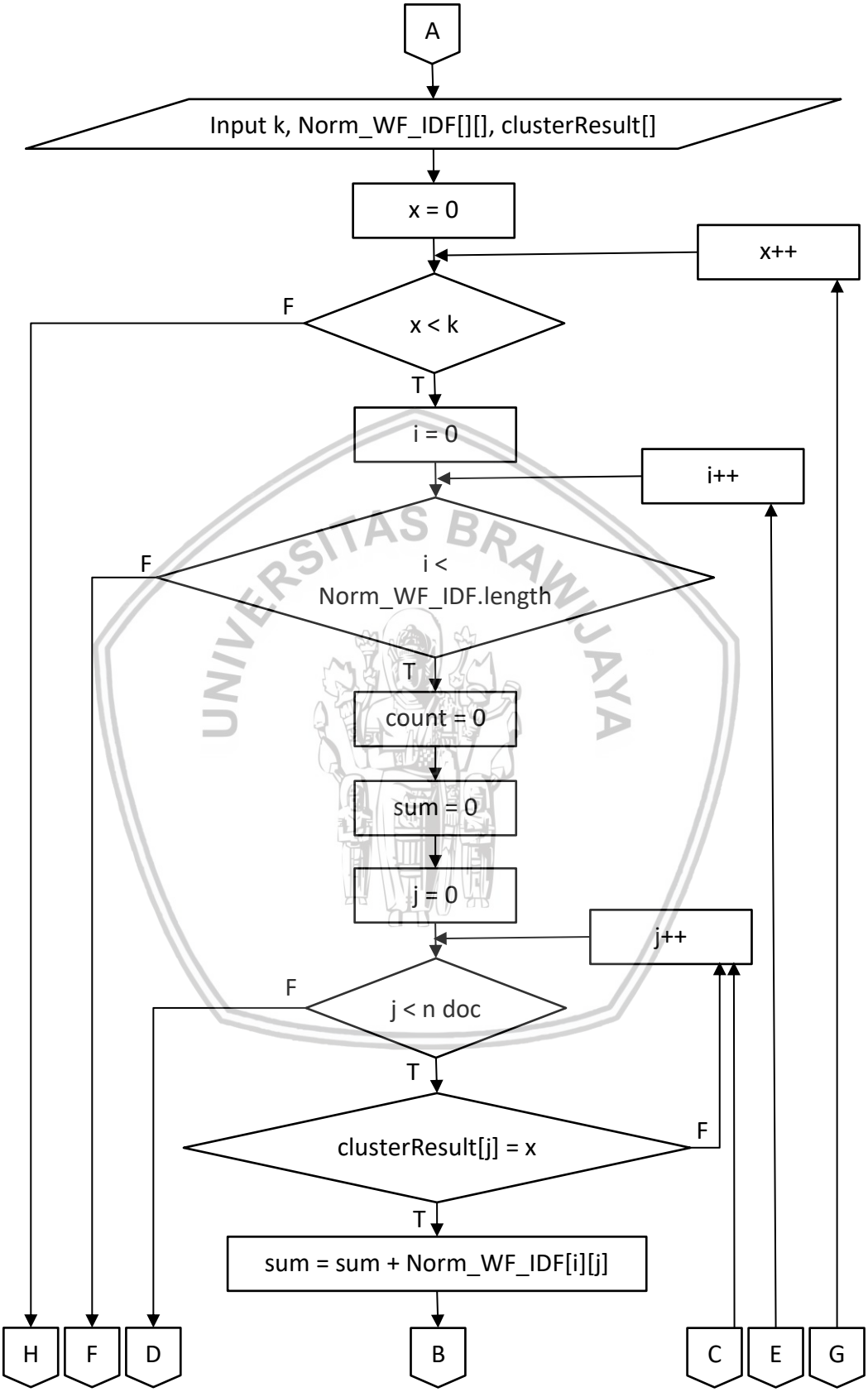
Gambar 4.26 (lanjutan)

2. Menghitung *Centroid* Baru

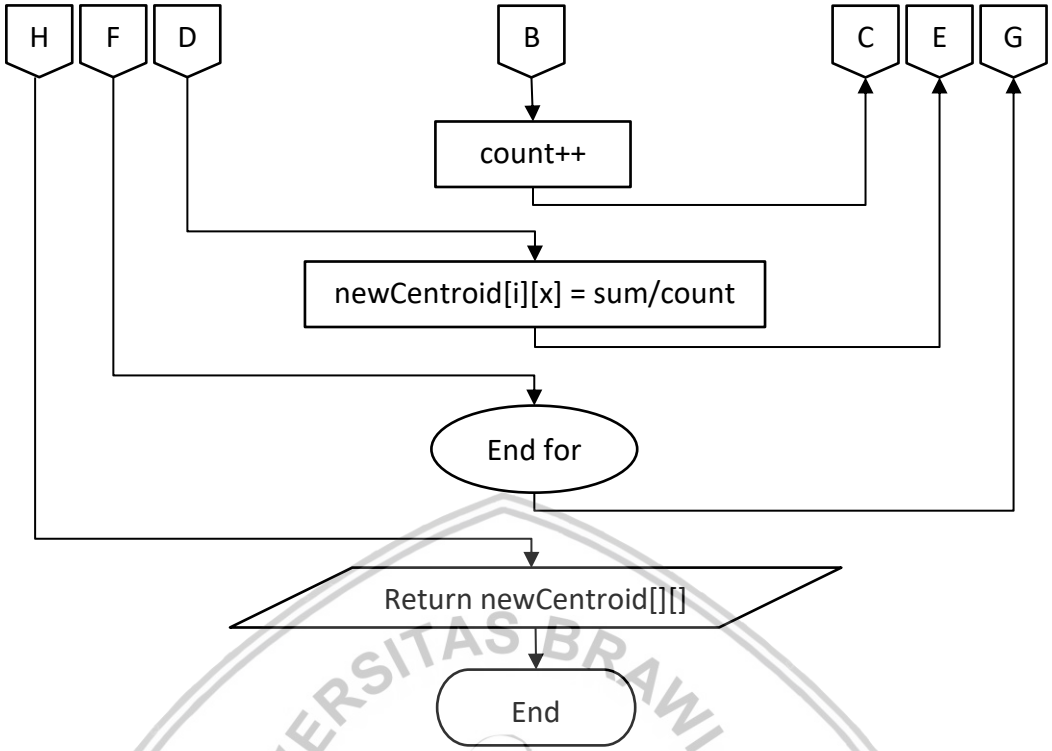
Pada proses ini dilakukan untuk menghitung kembali *centroid* baru setelah dilakukan proses menentukan klaster dokumen. *Centroid* baru dapat diperoleh dengan cara mencari nilai rata-rata vektor dokumen untuk masing-masing klaster. *Flowchart* untuk proses menghitung *centroid* baru dalam pengelompokan dapat dilihat pada Gambar 4.27.



Gambar 4.27 *Flowchart* Menghitung *Centroid* Baru



Gambar 4.27 (lanjutan)



Gambar 4.27 (lanjutan)

4.3 Manualisasi Perhitungan Data

Manualisasi perhitungan data menunjukkan proses perhitungan data pada implementasi praproses teks, *vector space model, improved k-means* dan pengujian *silhouette coefficient* yang terdapat dalam implementasi metode *improved k-means* untuk mengelompokkan dokumen J-PTIIK. Manualisasi perhitungan data dibutuhkan untuk membantu menjelaskan bagian perancangan secara semantik. Dari 233 objek data yang digunakan dalam penelitian, hanya 7 objek data yang digunakan sebagai manualisasi perhitungan data. Berikut objek data yang digunakan sebagai manualisasi perhitungan data dapat dilihat pada Tabel 4.6.

Tabel 4.6 Teks yang Digunakan pada Manualisasi Perhitungan Data

Doc	Teks
Doc1	<JUDUL>Interactive Mixed Reality System Menggunakan Pepper Ghost System dan Kendali Gerakan Tangan Berbasis Kinect</JUDUL> <ABSTRAK> Salah satu penggunaan teknologi mixed reality adalah penggunaan pepper ghost. Pepper ghost adalah salah satu teknologi yang bertujuan untuk menciptakan sebuah objek holografik yang interaktif.</ABSTRAK>
Doc2	<JUDUL>Penyelesaian Penjadwalan Flexible Job Shop Problem Menggunakan Real Coded Genetic Algorithm</JUDUL> <ABSTRAK>Model penjadwalan jobshop merupakan salah satu contoh masalah penjadwalan yang banyak ditemui dalam industri manufaktur.

Tabel 4.6 (lanjutan)

Doc	Teks
	Penyelesaiannya rumit dan solusi terbaik hanya bisa didapatkan dengan mencoba semua kemungkinan.</ABSTRAK>
Doc3	<JUDUL>Penentuan Portofolio Saham Optimal Menggunakan Algoritma Genetika</JUDUL> <ABSTRAK>Dalam melakukan investasi di pasar modal investor sering kali dihadapkan pada dua hal yaitu tingkat keuntungan dan tingkat kerugian. Maka dari itu untuk mengurangi tingkat kerugian investor melakukan diversifikasi dengan mengkombinasikan berbagai sekuritas dalam investasi atau disebut dengan portofolio saham.</ABSTRAK>
Doc4	<JUDUL>Deteksi Tepi Danau Pada Citra Satelit Menggunakan Metode Canny</JUDUL> <ABSTRAK>Danau adalah sebuah fitur darat yang berperan penting dalam kehidupan manusia. Perubahan pada danau dapat memengaruhi keadaan lingkungan sekitar serta kehidupan masyarakat yang berada di sekitarnya.</ABSTRAK>
Doc5	<JUDUL>Perancangan Sistem Pemetaan Ruangan Secara Dua Dimensi Menggunakan Sensor Ultrasonik</JUDUL> <ABSTRAK>Alat yang dirancang dan direalisasikan dalam tugas akhir ini berfungsi untuk memetakan ruangan yang hasilnya dapat dilihat pada layar komputer dalam bentuk peta ruang 2 dimensi. Alat ini terdiri dari Arduino Uno, ultrasonik HC-SR04 dan motor DC.</ABSTRAK>
Doc6	<JUDUL>Analisis Perbandingan Penetration Testing Tool Untuk Aplikasi Web</JUDUL> <ABSTRAK>Metodologi uji penetrasi mencakup tiga tahap: persiapan pengujian, tes dan analisa tes. Tahap uji coba melibatkan langkah-langkah berikut: analisa kerentanan, pengumpulan informasi, dan analisa tool.</ABSTRAK>
Doc7	<JUDUL>Analisis Penerimaan OS Windows 10 Dengan Unified Theory of Acceptance and Use of Technology 2 (UTAUT2)</JUDUL> <ABSTRAK>Responden penelitian merupakan pengguna mahasiswa di Indonesia. Hasil menunjukkan bahwa kemudahan, kondisi fasilitas dan harga tidak berpengaruh positif terhadap niat pengguna.</ABSTRAK>

4.3.1 Manualisasi Praproses Teks

Tahap pertama yang dilakukan dalam manualisasi adalah melakukan praproses teks. Proses ini meliputi beberapa sub proses yang terdiri dari menghapus *tag* JUDUL dan ABSTRAK, menghapus tanda baca dan angka, *case folding*, tokenisasi, menghapus *stopword* dan *stemming*.

4.3.1.1. Hapus *Tag* JUDUL dan ABSTRAK

Pada sub proses hapus *tag* JUDUL dan ABSTRAK yang dilakukan adalah menghapus *tag* JUDUL dan ABSTRAK dari teks. *Tag* JUDUL dan ABSTRAK ini merupakan karakter '<JUDUL>', '</JUDUL>', '<ABSTRAK>' dan '</ABSTRAK>', dimana karakter ini tidak dibutuhkan dalam proses berikutnya sehingga dapat

dihapus pada teks. Hasil penghapusan *tag* JUDUL dan ABSTRAK pada teks dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil Teks setelah Menghapus *Tag* JUDUL & ABSTRAK

Doc	Teks
Doc1	Interactive Mixed Reality System Menggunakan Pepper Ghost System dan Kendali Gerakan Tangan Berbasis Kinect Salah satu penggunaan teknologi mixed reality adalah penggunaan pepper ghost.
Doc2	Penyelesaian Penjadwalan Flexible Job Shop Problem Menggunakan Real Coded Genetic Algorithm Penyelesaiannya rumit dan solusi terbaik hanya bisa didapatkan dengan mencoba semua kemungkinan.
Doc3	Penentuan Portofolio Saham Optimal Menggunakan Algoritma Genetika Dalam melakukan investasi di pasar modal investor sering kali dihadapkan pada dua hal yaitu tingkat keuntungan dan tingkat kerugian.
Doc4	Deteksi Tepi Danau Pada Citra Satelit Menggunakan Metode Canny Danau adalah sebuah fitur darat yang berperan penting dalam kehidupan manusia.
Doc5	Perancangan Sistem Pemetaan Ruangan Secara Dua Dimensi Menggunakan Sensor Ultrasonik Alat ini terdiri dari Arduino Uno, ultrasonik HC-SR04 dan motor DC.
Doc6	Analisis Perbandingan Penetration Testing Tool Untuk Aplikasi Web Metodologi uji penetrasi mencakup tiga tahap: persiapan pengujian, tes dan analisa tes.
Doc7	Analisis Penerimaan OS Windows 10 Dengan Unified Theory of Acceptance and Use of Technology 2 (UTAUT2) Responden penelitian merupakan pengguna mahasiswa di Indonesia.

4.3.1.2. Hapus Tanda Baca Dan Angka

Pada sub proses hapus tanda baca dan angka yang dilakukan adalah menghapus tanda baca dan angka dari teks hasil sub proses hapus *tag* JUDUL dan ABSTRAK. Sebagai contoh jika pada Doc1 terdapat tanda baca berupa karakter titik, maka tanda baca ini dihapus dari teks. Secara lengkap hasil penghapusan tanda baca dan angka pada teks dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil Teks setelah Menghapus Tanda Baca & Angka

Doc	Teks
Doc1	Interactive Mixed Reality System Menggunakan Pepper Ghost System dan Kendali Gerakan Tangan Berbasis Kinect Salah satu penggunaan teknologi mixed reality adalah penggunaan pepper ghost
Doc2	Penyelesaian Penjadwalan Flexible Job Shop Problem Menggunakan Real Coded Genetic Algorithm Penyelesaiannya rumit dan solusi terbaik hanya bisa didapatkan dengan mencoba semua kemungkinan

Tabel 4.8 (lanjutan)

Doc	Teks
Doc3	Penentuan Portofolio Saham Optimal Menggunakan Algoritma Genetika Dalam melakukan investasi di pasar modal investor sering kali dihadapkan pada dua hal yaitu tingkat keuntungan dan tingkat kerugian
Doc4	Deteksi Tepi Danau Pada Citra Satelit Menggunakan Metode Canny Danau adalah sebuah fitur darat yang berperan penting dalam kehidupan manusia
Doc5	Perancangan Sistem Pemetaan Ruangan Secara Dua Dimensi Menggunakan Sensor Ultrasonik Alat ini terdiri dari Arduino Uno ultrasonik HC SR dan motor DC
Doc6	Analisis Perbandingan Penetration Testing Tool Untuk Aplikasi Web Metodologi uji penetrasi mencakup tiga tahap persiapan pengujian tes dan analisa tes
Doc7	Analisis Penerimaan OS Windows Dengan Unified Theory of Acceptance and Use of Technology UTAUT Responden penelitian merupakan pengguna mahasiswa di Indonesia

4.3.1.3. *Case Folding*

Pada sub proses *case folding* yang dilakukan adalah mengubah seluruh huruf teks hasil sub proses hapus tanda baca dan angka menjadi huruf kecil. Hasil dari *case folding* terhadap data teks dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil Teks setelah Dilakukan *Case Folding*

Doc	Teks
Doc1	interactive mixed reality system menggunakan pepper ghost system dan kendali gerakan tangan berbasis kinect salah satu penggunaan teknologi mixed reality adalah penggunaan pepper ghost
Doc2	penyelesaian penjadwalan flexible job shop problem menggunakan real coded genetic algorithm penyelesaiannya rumit dan solusi terbaik hanya bisa didapatkan dengan mencoba semua kemungkinan
Doc3	penentuan portofolio saham optimal menggunakan algoritma genetika dalam melakukan investasi di pasar modal investor sering kali dihadapkan pada dua hal yaitu tingkat keuntungan dan tingkat kerugian
Doc4	deteksi tepi danau pada citra satelit menggunakan metode canny danau adalah sebuah fitur darat yang berperan penting dalam kehidupan manusia
Doc5	perancangan sistem pemetaan ruangan secara dua dimensi menggunakan sensor ultrasonik alat ini terdiri dari arduino uno ultrasonik hc sr dan motor dc
Doc6	analisis perbandingan penetration testing tool untuk aplikasi web metodologi uji penetrasi mencakup tiga tahap persiapan pengujian tes dan analisa tes
Doc7	analisis penerimaan os windows dengan unified theory of acceptance and use of technology utaut responden penelitian merupakan pengguna mahasiswa di indonesia

4.3.1.4. Tokenisasi

Pada sub proses tokenisasi yang dilakukan adalah mengubah teks hasil sub proses *case folding* menjadi bentuk kumpulan token. Token diperoleh melalui mengubah teks menjadi bentuk kata yang dipisahkan berdasarkan karakter *white space* seperti penggunaan karakter spasi dan karakter *new line.* Hasil dari tokenisasi pada teks dapat dilihat pada Tabel 4.10.

Tabel 4.10 Hasil Tokenisasi pada Teks

Doc	Token
Doc1	interactive mixed reality system menggunakan pepper ghost system dan kendali gerakan tangan berbasis kinect salah satu penggunaan teknologi mixed reality adalah penggunaan pepper ghost
Doc2	penyelesaian penjadwalan flexible job shop problem menggunakan real coded genetic algorithm penyelesaiannya rumit dan solusi terbaik hanya bisa didapatkan dengan mencoba semua kemungkinan
Doc3	penentuan portofolio saham optimal menggunakan algoritma genetika dalam melakukan investasi di pasar modal investor sering kali dihadapkan pada dua hal yaitu tingkat keuntungan dan tingkat kerugian
Doc4	deteksi tepi danau pada citra satelit menggunakan metode canny danau adalah sebuah fitur darat yang berperan penting dalam kehidupan manusia
Doc5	perancangan sistem pemetaan ruangan secara dua dimensi menggunakan sensor ultrasonik alat ini terdiri dari arduino uno ultrasonik hc sr dan motor dc
Doc6	analisis perbandingan penetration testing tool untuk aplikasi web metodologi uji penetrasi mencakup tiga tahap persiapan pengujian tes dan analisa tes
Doc7	analisis penerimaan os windows dengan unified theory of acceptance and use of technology utaut responden penelitian merupakan pengguna mahasiswa di indonesia

4.3.1.5. Hapus *Stopword*

Pada sub proses hapus *stopword* yang dilakukan adalah menghapus *stopword* yang terdapat dalam token hasil dari sub proses tokenisasi. Sebagai contoh bentuk *stopword* dalam penggunaan bahasa Indonesia adalah kata ‘dengan’ yang terdapat pada Doc7. Sehingga token ini dapat dihapus dari Doc7. Hasil lengkap penghapusan *stopword* pada token dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil Penghapusan *Stopword* pada Token

Doc	Token
Doc1	interactive mixed reality system pepper ghost system kendali gerakan tangan berbasis kinect salah penggunaan teknologi mixed reality penggunaan pepper ghost
Doc2	penyelesaian penjadwalan flexible job shop problem real coded genetic algorithm penyelesaiannya rumit solusi terbaik didapatkan mencoba
Doc3	penentuan portofolio saham optimal algoritma genetika investasi pasar modal investor kali dihadapkan tingkat keuntungan tingkat kerugian

Tabel 4.11 (lanjutan)

Doc	Token
Doc4	deteksi tepi danau citra satelit metode canny danau fitur darat berperan kehidupan manusia
Doc5	perancangan sistem pemetaan ruangan dimensi sensor ultrasonik alat arduino uno ultrasonik hc sr motor dc
Doc6	analisis perbandingan penetration testing tool aplikasi web metodologi uji penetrasi mencakup tahap persiapan pengujian tes analisa tes
Doc7	analisis penerimaan windows unified theory of acceptance and use of technology utaut responden penelitian pengguna mahasiswa indonesia

4.3.1.6. *Stemming*

Pada sub proses *stemming* yang dilakukan adalah mengubah token hasil dari sub proses hapus *stopword* menjadi bentuk kata dasar. Sebagai contoh pada Doc7 terdapat kata 'penerimaan', sehingga jika kata ini diubah bentuknya menjadi kata dasar akan menghasilkan kata 'terima'. Secara lebih lengkap hasil *stemming* pada token dapat dilihat pada Tabel 4.12.

Tabel 4.12 Hasil *Stemming* pada Token

Doc	Kata Dasar
Doc1	interactive mixed reality system pepper ghost system kendali gerak tangan basis kinect salah guna teknologi mixed reality guna pepper ghost
Doc2	selesai jadwal flexible job shop problem real coded genetic algorithm selesai rumit solusi baik dapat coba
Doc3	tentu portofolio saham optimal algoritma genetika investasi pasar modal investor kali hadap tingkat untung tingkat rugi
Doc4	deteksi tepi danau citra satelit metode canny danau fitur darat peran hidup manusia
Doc5	rancang sistem peta ruang dimensi sensor ultrasonik alat arduino uno ultrasonik hc sr motor dc
Doc6	analisis banding penetration testing tool aplikasi web metodologi uji penetrasi cakup tahap siap uji tes analisa tes
Doc7	analisis terima windows unified theory of acceptance and use of technology utaut responden teliti guna mahasiswa indonesia

4.3.2 Manualisasi *Vector Space Model*

Tahap berikutnya yang dilakukan dalam manualisasi adalah melakukan proses pembentukan *vector space model* untuk memperoleh hasil pembobotan kata terhadap kata hasil dari tahap praproses teks. Proses ini meliputi beberapa sub proses yang terdiri dari hitung jumlah frekuensi kata pada dokumen, hitung w_{tf}, hitung *idf*, hitung *wf.idf* dan hitung normalisasi dari *wf.idf.*

4.3.2.1. Hitung *tf*

Pada sub proses hitung *tf* yang dilakukan adalah menghitung jumlah frekuensi kata hasil dari praproses teks pada masing-masing dokumen. Nilai frekuensi ini disusun ke dalam bentuk vektor. Sebagai contoh kata 'danau' dihitung jumlah

kemunculan kata ini pada masing-masing dokumen. Sehingga diperoleh hasil bahwa kata ini hanya terdapat di Doc4 dengan jumlah kemunculan sebesar 2. Secara lebih lengkap hasil dari perhitungan *tf* dapat dilihat dalam Tabel 4.13 dan Lampiran A.

Tabel 4.13 Vektor Jumlah Frekuensi *Term* pada Dokumen

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
coba	0	1	0	0	0	0	0
coded	0	1	0	0	0	0	0
danau	0	0	0	2	0	0	0
dapat	0	1	0	0	0	0	0
darat	0	0	0	1	0	0	0
dc	0	0	0	0	1	0	0
deteksi	0	0	0	1	0	0	0
dimensi	0	0	0	0	1	0	0

4.3.2.2. Hitung *wf*

Pada sub proses hitung *wf* yang dilakukan adalah menghitung nilai bobot *tf* dengan menggunakan Persamaan 2.2 yang telah dibahas pada sub bab 2.4.1. Sebagai contoh untuk menghitung nilai *wf* kata 'danau' pada Doc4, berdasarkan Tabel 4.13 nilai *tf* kata 'danau' pada Doc4 lebih besar dari 0 sehingga untuk menghitung nilai *wf* kata 'danau' pada Doc4 dapat diperoleh dengan cara sebagai berikut:

$$\begin{aligned} wf_{danau,Doc4} &= 1 + \log tf_{danau,Doc4} \\ &= 1 + \log 2 \\ &= 1 + 0.30103 \\ &= 1.30103 \end{aligned} \tag{4.18}$$

Secara lebih lengkap hasil dari perhitungan nilai *wf* untuk masing-masing kata pada dokumen dapat dilihat pada Tabel 4.14 dan Lampiran B.

Tabel 4.14 Hasil Perhitungan *wf*

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
coba	0	1	0	0	0	0	0
coded	0	1	0	0	0	0	0
danau	0	0	0	1,30103	0	0	0
dapat	0	1	0	0	0	0	0
darat	0	0	0	1	0	0	0
dc	0	0	0	0	1	0	0
deteksi	0	0	0	1	0	0	0
dimensi	0	0	0	0	1	0	0

4.3.2.3. Hitung *idf*

Pada sub proses hitung *idf* yang dilakukan adalah menghitung nilai *idf* masing-masing kata dengan menggunakan Persamaan 2.3 yang telah dibahas pada sub bab 2.4.1. Sebagai contoh untuk menghitung nilai *idf* kata 'danau', maka langkah pertama adalah menghitung nilai *df* kata 'danau'. Nilai *df* kata 'danau' dapat diperoleh dengan cara menghitung frekuensi dokumen yang mengandung kata 'danau'. Berdasarkan Tabel 4.13 diperoleh hasil bahwa kata 'danau' hanya terdapat di Doc4, sehingga nilai *df* kata 'danau' adalah 1.

Langkah berikutnya adalah menghitung nilai *idf* kata 'danau' yang dapat diperoleh dengan cara sebagai berikut:

$$\begin{aligned} idf_{danau} &= \log N / df_{danau} \\ &= \log 7/1 \\ &= \log 7 \\ &= 0.845098 \end{aligned} \tag{4.19}$$

Secara lebih lengkap hasil dari perhitungan *idf* masing-masing kata dapat dilihat pada Tabel 4.15 dan Lampiran C.

Tabel 4.15 Hasil Perhitungan *idf*

Kata	df_t	idf_t
coba	1	0,845098
coded	1	0,845098
danau	1	0,845098
dapat	1	0,845098
darat	1	0,845098
dc	1	0,845098
deteksi	1	0,845098
dimensi	1	0,845098

4.3.2.4. Hitung *wf.idf*

Pada sub proses hitung *wf.idf* yang dilakukan adalah melakukan pembobotan kata dengan cara menghitung perkalian antara *wf* dan *idf* dengan menggunakan Persamaan 2.4 yang telah dibahas pada sub bab 2.4.1. Sebagai contoh untuk menghitung nilai *wf.idf* kata 'danau' pada Doc4 yang merupakan perkalian antara Persamaan 4.17 dan 4.18 dapat diperoleh dengan cara sebagai berikut:

$$\begin{aligned} wf.idf_{danau,Doc4} &= wf_{danau,Doc4} \times idf_{danau} \\ &= 1.30103 \times 0.845098 \\ &= 1.099498 \end{aligned} \tag{4.20}$$

Secara lebih lengkap hasil dari perhitungan *wf.idf* dapat dilihat pada Tabel 4.16 dan Lampiran D.

Tabel 4.16 Hasil Perhitungan *wf.idf*

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
coba	0	0,845098	0	0	0	0	0
coded	0	0,845098	0	0	0	0	0
danau	0	0	0	1,099498	0	0	0
dapat	0	0,845098	0	0	0	0	0
darat	0	0	0	0,845098	0	0	0
dc	0	0	0	0	0,845098	0	0
deteksi	0	0	0	0,845098	0	0	0
dimensi	0	0	0	0	0,845098	0	0

4.3.2.5. Hitung Normalisasi *wf.idf*

Pada sub proses hitung normalisasi *wf.idf* yang dilakukan adalah menghitung nilai normalisasi dari pembobotan kata *wf.idf* dengan menggunakan Persamaan 2.5 yang telah dibahas pada sub bab 2.4.1. Sebagai contoh untuk menghitung nilai normalisasi dari pembobotan *wf.idf* kata 'danau' pada Doc4 dapat diperoleh dengan cara sebagai berikut:

$$wf.idf_{danau,Doc4} = \frac{wf.idf_{danau,Doc4}}{\sqrt{\sum_{t=1}^{n} wf.idf_{t,Doc4}^{2}}}$$
$$= \frac{1.099498}{\sqrt{0^2+0^2+0^2+\ldots+0^2}}$$
$$= \frac{1.099498}{9.06499}$$
$$= 0.365183 \quad (4.21)$$

Secara lebih lengkap hasil dari perhitungan nilai normalisasi pembobotan kata *wf.idf* dapat dilihat pada Tabel 4.17 dan Lampiran E.

Tabel 4.17 Hasil Perhitungan Normalisasi *wf.idf*

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
coba	0	0,252436	0	0	0	0	0
coded	0	0,252436	0	0	0	0	0
danau	0	0	0	0,365183	0	0	0
dapat	0	0,252436	0	0	0	0	0
darat	0	0	0	0,280688	0	0	0
dc	0	0	0	0	0,260885	0	0
deteksi	0	0	0	0,280688	0	0	0
dimensi	0	0	0	0	0,260885	0	0

4.3.3 Manualisasi *Improved K-Means*

Tahap berikutnya yang dilakukan dalam manualisasi adalah melakukan proses pengelompokan terhadap data dengan menggunakan pembobotan kata yang diselesaikan dalam proses *vector space model*. Proses ini meliputi beberapa sub proses yang terdiri dari penentuan *centroid* awal klaster yang terdiri dari 2 klaster dan pengelompokan menggunakan *k-means* dengan nilai *k* = 2.

4.3.3.1. Penentuan *Centroid* Awal Klaster

Pada sub proses penentuan *centroid* awal yang pertama kali dilakukan adalah menghitung jarak antar tiap data dengan menggunakan *Euclidean Distance* pada Persamaan 2.9 yang telah dibahas pada sub bab 2.5.2. Data yang digunakan untuk menghitung jarak adalah hasil normalisasi pembobotan *wf.idf* yang telah selesai dilakukan pada proses sebelumnya. Sebagai contoh untuk menghitung jarak *Euclidean Distance* antara Doc1 dan Doc7 dapat diperoleh dengan cara sebagai berikut:

$$\begin{aligned} d(Doc1, Doc7) &= \sqrt{(t_{Doc1_1} - t_{Doc7_1})^2 + (t_{Doc1_2} - t_{Doc7_2})^2 + ... + (t_{Doc1_m} - t_{Doc7_m})^2} \\ &= \sqrt{(0 - 0.253823)^2 + (0 - 0)^2 + ... + (0 - 0.253823)^2} \\ &= \sqrt{1.93393} \\ &= 1.390657 \end{aligned} \quad (4.22)$$

Secara lebih lengkap hasil dari perhitungan jarak antar tiap data dapat dilihat pada Tabel 4.18.

Tabel 4.18 Hasil Perhitungan Jarak Data dengan *Euclidean Distance*

d(i,j)	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
Doc1	0	1,414214	1,414214	1,414214	1,414214	1,414214	1,390657
Doc2	1,414214	0	1,414214	1,414214	1,414214	1,414214	1,414214
Doc3	1,414214	1,414214	0	1,414214	1,414214	1,414214	1,414214
Doc4	1,414214	1,414214	1,414214	0	1,414214	1,414214	1,414214
Doc5	1,414214	1,414214	1,414214	1,414214	0	1,414214	1,414214
Doc6	1,414214	1,414214	1,414214	1,414214	1,414214	0	1,395373
Doc7	1,390657	1,414214	1,414214	1,414214	1,414214	1,395373	0

Langkah berikutnya adalah menghitung nilai rata-rata jarak dengan menggunakan Persamaan 2.10 yang telah dibahas pada sub bab 2.5.2. Untuk menghitung nilai rata-rata jarak, hanya jarak antar dokumen yang berbeda saja digunakan, sehingga jarak Doc1 ke Doc1 tidak digunakan. Penggunaan jarak antar dokumen juga hanya diperbolehkan sekali, karena jarak Doc1 ke Doc2 sama saja dengan jarak Doc2 ke Doc1. Sehingga hasil dari menghitung nilai rata-rata jarak dapat diperoleh dengan cara sebagai berikut:

$$MeanDist = \frac{1}{C_2^7} \times \sum d(x_i, x_j)$$

$$= \frac{1}{C_2^7} \times (d(Doc2, Doc1) + d(Doc3, Doc1) + ... + d(Doc7, Doc6))$$

$$= \frac{1}{7!/2!(7-2)!} \times (1.414214 + 1.414214 + ... + 1.395373)$$

$$= \frac{1}{21} \times 29.6561$$

$$= 1.41219 \quad (4.23)$$

Setelah menghitung rata-rata jarak langkah berikutnya adalah menghitung nilai parameter densitas seluruh objek data dengan menggunakan Persamaan 2.11 yang telah dibahas pada sub bab 2.5.2. Sebagai contoh untuk menghitung nilai parameter densitas Doc1 dapat diperoleh dengan cara sebagai berikut:

$$Dens(Doc1) = \sum_{j=1}^{n} u(MeanDist - d(x_i, x_j))$$

$$= u(MeanDist - d(Doc1, Doc1)) + u(MeanDist - d(Doc1, Doc2)) + ... + u(MeanDist - d(Doc1, Doc7))$$

$$= u(1.41219 - 0) + u(1.41219 - 1.414214) + ... + u(1.41219 - 1.390657)$$

$$= u(1.41219) + u(-0.002024) + ... + u(0.021533)$$

Dimana nilai $u(z)$ merupakan sebuah fungsi yang jika nilai *z* lebih besar sama dengan 0, maka nilai $u(z)$ adalah 1. Dan sebaliknya jika nilai *z* lebih kecil dari 0, maka nilai $u(z)$ adalah 0. Sehingga nilai parameter densitas Doc1 adalah sebagai berikut:

$$Dens(Doc1) = 1 + 0 + ... + 1$$
$$= 2 \quad (4.24)$$

Secara lebih lengkap hasil dari perhitungan nilai parameter densitas seluruh objek data dapat dilihat pada Tabel 4.19.

Tabel 4.19 Hasil Perhitungan Parameter Densitas Dokumen

	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
Dens	2	1	1	1	1	2	3

Setelah menghitung nilai parameter densitas langkah berikutnya adalah menghitung nilai rata-rata parameter densitas dengan menggunakan Persamaan 2.12 yang telah dibahas pada sub bab 2.5.2. Untuk menghitung nilai rata-rata parameter densitas dapat diperoleh dengan cara sebagai berikut:

$$MeanDens(D) = \frac{1}{n}\sum_{i=1}^{n} Dens(x_i)$$
$$= \frac{1}{7} \times (Dens(Doc1) + Dens(Doc2) + ... + Dens(Doc7))$$
$$= \frac{1}{7} \times (2 + 1 + ... + 3)$$
$$= \frac{1}{7} \times 11$$
$$= 1.57143 \quad (4.25)$$

Langkah berikutnya adalah menentukan nilai α, dimana dalam manualisasi perhitungan data ini diberikan nilai α = 0,90. Kemudian hitung nilai dari α x $MeanDens(D)$ sehingga diperoleh hasil sebagai berikut:

$$\alpha \times MeanDens(D) = 0.90 \times 1.57143$$
$$= 1.41429 \quad (4.26)$$

Sehingga dengan menggunakan Persamaan 2.13 yang telah dibahas pada sub bab 2.5.2 diperoleh nilai parameter densitas tertinggi dengan batasan parameter densitas sebagai berikut:

$$Dens(x_i) < 1.41429 \quad (4.27)$$

Nilai parameter densitas yang kurang dari nilai ini akan tereliminasi sebagai calon kandidat *centroid* awal untuk proses pengelompokan, sehingga berdasarkan Tabel 4.19 maka Doc2, Doc3, Doc4 dan Doc5 tereliminasi. Hasil dari pengeliminasian nilai parameter densitas yang menghasilkan nilai parameter densitas tertinggi dapat dilihat pada Tabel 4.20.

Tabel 4.20 Nilai Parameter Densitas Tertinggi Dokumen

	Doc1	Doc6	Doc7
Dens	2	2	3

Berdasarkan nilai parameter densitas tertinggi yang terdapat pada Tabel 4.20 dapat ditentukan *centroid* awal klaster 1, dimana diperoleh bahwa Doc7 memiliki parameter densitas tertinggi dengan nilai densitas sebesar 3. Langkah selanjutnya adalah mencari *centroid* awal klaster 2 dengan cara mencari dokumen dengan jarak terjauh dari Doc7. Jarak yang digunakan dalam proses ini diperoleh dengan cara 1 – *cosine similarity*. Sehingga dokumen yang memiliki jarak terjauh dari Doc7 adalah Doc6 dengan nilai jarak sebesar 0,97353. Sehingga untuk nilai *k* = 2 diperoleh hasil bahwa *centroid* awal klaster 1 dan 2 masing-masing adalah Doc7 dan Doc6.

4.3.3.2. *K-Means*

Pada sub proses pengelompokan dilakukan dengan menggunakan *k-means. Centroid* awal klaster yang digunakan pada sub proses ini adalah *centroid* awal klaster yang telah diperoleh pada sub proses sebelumnya. Berikut adalah *centroid* awal dari masing-masing klaster 1 dan 2 yang mengambil nilai normalisasi pembobotan kata *wf.idf* dapat dilihat pada Tabel 4.21 dan Lampiran F.

Tabel 4.21 *Centroid* Awal Klaster 1 dan Klaster 2

Kata	Klaster 1 = Doc7	Klaster 2 = Doc6
acceptance	0,253823	0
alat	0	0
algorithm	0	0
algoritma	0	0
analisa	0	0,251579
analisis	0,16341	0,161965
and	0,253823	0
aplikasi	0	0,251579

Langkah berikutnya adalah menentukan kondisi konvergen dari proses pengelompokan. Untuk manualisasi perhitungan data ini yang digunakan sebagai kondisi konvergen adalah jika posisi klaster tidak ada perubahan terhadap posisi klaster pada iterasi sebelumnya.

Setelah menentukan kondisi konvergen *k-means* langkah berikutnya adalah menentukan posisi dari masing-masing dokumen berada di klaster mana dengan cara menghitung jarak antara dokumen dengan *centroid* klaster. Pendekatan jarak yang digunakan adalah *cosine similarity* sebab memiliki performa yang baik dan efisien jika digunakan pada bentuk data teks (Sabthami, Thirumoorthy, & Muneeswaran, 2016). Sebagai contoh untuk menghitung nilai *cosine similarity* Doc1 terhadap *centroid* klaster 1 dapat diperoleh dengan cara sebagai berikut:

$$
\begin{aligned}
CosSim(Doc1, K1) &= \sum_{i=1}^{t} (w_{iDoc1}.w_{iK1}) \\
&= (w_{1Doc1}.w_{1K1}) + (w_{2Doc1}.w_{2K1}) + \ldots + (w_{tDoc1}.w_{tK1}) \\
&= (0 \times 0.253823) + (0 \times 0) + \ldots + (0 \times 0.253823) \\
&= 0 + 0 + \ldots + 0 \\
&= 0.033036
\end{aligned}
\qquad (4.28)
$$

Kemudian untuk menentukan posisi dokumen berada di klaster 1 atau klaster 2 dapat diperoleh dengan cara menentukan nilai *cosine similarity* terbesar terhadap *centroid* klaster. Contoh untuk Doc1 memiliki nilai *cosine similarity* terbesar terhadap klaster 1 dengan nilai *cosine similarity* 0,033036, sehingga Doc1 ditempatkan pada klaster 1. Hasil penentuan posisi dokumen dalam klaster secara lebih lengkap dapat dilihat pada Tabel 4.22.

Tabel 4.22 Hasil Klaster Dokumen pada Iterasi Pertama

Doc	*CosSim*		Terbesar	Klaster
	Klaster 1	Klaster 2		
Doc1	0,033036	0	0,033036	Klaster 1
Doc2	0	0	0	Klaster 1
Doc3	0	0	0	Klaster 1
Doc4	0	0	0	Klaster 1
Doc5	0	0	0	Klaster 1
Doc6	0,026467	1	1	Klaster 2
Doc7	1	0,026467	1	Klaster 1

Langkah berikutnya adalah menghitung kembali *centroid* baru dari klaster 1 dan 2. Untuk memperoleh nilai dari *centroid* baru klaster dapat menggunakan Persamaan 2.8 yang telah dibahas pada sub bab 2.5.1. Dimana *centroid* baru diperoleh dengan cara mencari nilai rata-rata vektor yang terdapat pada masing-masing klaster. Sebagai contoh untuk menghitung *centroid* baru klaster 1 pada kata 'alat' dapat diperoleh dengan cara sebagai berikut:

$$
\begin{aligned}
v_{alat} &= \frac{1}{K1} \times \sum_{j=1}^{K1} b_{alat_j} \\
&= \frac{1}{6} \times \sum_{j=1}^{6} b_{alat_j} \\
&= \frac{1}{6} \times (b_{alat_1} + b_{alat_2} + ... + b_{alat_6}) \\
&= \frac{1}{6} \times (0 + 0 + ... + 0) \\
&= \frac{1}{6} \times 0.260885 \\
&= 0.043481
\end{aligned}
\tag{4.29}
$$

Secara lebih lengkap hasil perhitungan *centroid* baru dari klaster 1 dan klaster 2 pada iterasi pertama dapat dilihat pada Tabel 4.23 dan Lampiran G.

Tabel 4.23 *Centroid* Baru Klaster 1 dan Klaster 2 pada Iterasi Pertama

Kata	Klaster 1	Klaster 2
acceptance	0,042304	0
alat	0,043481	0
algorithm	0,042073	0
algoritma	0,042073	0
analisa	0	0,251579
analisis	0,027235	0,161965
and	0,042304	0
aplikasi	0	0,251579

Setelah iterasi pertama selesai dilakukan, langkah berikutnya adalah melakukan iterasi kedua untuk menentukan posisi dari masing-masing dokumen berada di klaster mana dengan cara mencari nilai *cosine similarity* terbesar antara dokumen dengan *centroid* klaster baru yang terdapat pada Lampiran G. Hitung kembali nilai *cosine similarity* Doc1 terhadap *centroid* klaster 1 dengan cara yang sama pada iterasi sebelumnya sebagai berikut:

$$\begin{aligned} CosSim(Doc1, K1) &= \sum_{i=1}^{t}(w_{iDoc1}.w_{iK1}) \\ &= (w_{1Doc1}.w_{1K1}) + (w_{2Doc1}.w_{2K1}) + \ldots + (w_{tDoc1}.w_{tK1}) \\ &= (0 \times 0.042304) + (0 \times 0.043481) + \ldots + (0 \times 0.042304) \\ &= 0 + 0 + \ldots + 0 \\ &= 0.172173 \end{aligned} \quad (4.30)$$

Kemudian tentukan kembali apakah dokumen berada di klaster 1 atau klaster 2 dengan cara melihat nilai *cosine similarity* terbesar terhadap *centroid* klaster. Untuk Doc1 memiliki nilai *cosine similarity* terbesar terhadap klaster 1, sehingga Doc1 ditempatkan pada klaster 1. Secara lebih lengkap hasil penentuan posisi dokumen dalam klaster pada iterasi kedua dapat dilihat pada Tabel 4.24.

Tabel 4.24 Hasil Klaster Dokumen pada Iterasi Kedua

Doc	*CosSim*		Terbesar	Klaster
	Klaster 1	Klaster 2		
Doc1	0,172173	0	0,172173	Klaster 1
Doc2	0,166667	0	0,166667	Klaster 1
Doc3	0,166667	0	0,166667	Klaster 1
Doc4	0,166667	0	0,166667	Klaster 1
Doc5	0,166667	0	0,166667	Klaster 1
Doc6	0,004411	1	1	Klaster 2
Doc7	0,172173	0,026467	0,172173	Klaster 1

Langkah berikutnya pada iterasi kedua ini adalah menghitung kembali *centroid* baru dari klaster 1 dan 2. Sebagai contoh untuk menghitung *centroid* baru klaster 1 pada kata 'alat' dapat diperoleh dengan cara sebagai berikut:

$$\begin{aligned} v_{alat} &= \frac{1}{K1} \times \sum_{j=1}^{K1} b_{alat_j} \\ &= \frac{1}{6} \times \sum_{j=1}^{6} b_{alat_j} \\ &= \frac{1}{6} \times (b_{alat_1} + b_{alat_2} + \ldots + b_{alat_6}) \\ &= \frac{1}{6} \times (0 + 0 + \ldots + 0) \\ &= \frac{1}{6} \times 0.260885 \\ &= 0.043481 \end{aligned} \quad (4.31)$$

Secara lebih lengkap hasil perhitungan *centroid* baru dari klaster 1 dan klaster 2 pada iterasi kedua dapat dilihat pada Tabel 4.25 dan Lampiran G.

Tabel 4.25 *Centroid* Baru Klaster 1 dan Klaster 2 pada Iterasi Kedua

Kata	Klaster 1	Klaster 2
acceptance	0,042304	0
alat	0,043481	0
algorithm	0,042073	0
algoritma	0,042073	0
analisa	0	0,251579
analisis	0,027235	0,161965
and	0,042304	0
aplikasi	0	0,251579

Berdasarkan kondisi konvergen yang telah ditentukan sebelumnya, dapat dilihat bahwa hasil klaster pada iterasi kedua yang terdapat pada Tabel 4.24 (Lampiran G) dan iterasi pertama yang terdapat pada Tabel 4.22 (Lampiran G) tidak mengalami perubahan. Sehingga dapat disimpulkan proses iterasi berhenti pada iterasi kedua. Hasil akhir dari proses pengelompokan dokumen adalah untuk klaster 1 terdiri dari Doc1, Doc2, Doc3, Doc4, Doc5 dan Doc7, sementara untuk klaster 2 terdiri dari Doc6.

4.3.4 Manualisasi Evaluasi *Silhouette Coefficient*

Tahap berikutnya yang dilakukan dalam manualisasi adalah melakukan evaluasi atau pengujian dengan menghitung nilai *silhouette coefficient* masing-masing dokumen. Pengujian dalam manualisasi ini dilakukan untuk mengetahui kualitas dari hasil pengelompokan dokumen. Langkah pertama dalam menghitung *silhouette coefficient* adalah menghitung rata-rata jarak (1 – *cosine similarity*) dari data *i* ke semua data lain yang berada dalam klaster yang sama. Sebagai contoh untuk menghitung jarak dari Doc1 ke dokumen lain yang berada dalam klaster yang sama ditunjukkan dengan cara sebagai berikut:

$$a(Doc1) = \frac{1}{|A|-1} \sum_{j \in A, j \neq i} d(i,j)$$

$$= \frac{1}{6-1} \times (d(Doc1, Doc2) + d(Doc1, Doc3) + \ldots + d(Doc1, Doc7))$$

$$= \frac{1}{5} \times ((1-0) + (1-0) + \ldots + (1-0.03304))$$

$$= \frac{1}{5} \times (1+1+\ldots+0.96696)$$

$$= \frac{1}{5} \times 4.96696$$

$$= 0.9934 \qquad (4.32)$$

Langkah berikutnya dalam menghitung *silhouette coefficient* adalah menghitung rata-rata jarak (1 – *cosine similarity*) dari data *i* ke semua data lain yang berada dalam klaster berbeda kemudian diambil nilai terkecilnya. Sebagai contoh untuk menghitung jarak dari Doc1 ke dokumen lain yang berada dalam klaster berbeda ditunjukkan dengan cara sebagai berikut:

$$\begin{aligned} b(Doc1) &= \min_{B \neq A} \frac{1}{|B|} \sum_{j \in B} d(i, j) \\ &= \min\{\frac{1}{1} \times (d(Doc1, Doc6))\} \\ &= \min\{1 \times (1-0)\} \\ &= 1 \end{aligned} \quad (4.33)$$

Sehingga nilai *silhouette coefficient* Doc1 dapat diperoleh dengan cara sebagai berikut:

$$\begin{aligned} s(Doc1) &= \frac{b(Doc1) - a(Doc1)}{\max\{a(Doc1), b(Doc1)\}} \\ &= \frac{1 - 0.9934}{\max\{0.9934, 1\}} \\ &= \frac{0.00661}{1} \\ &= 0.00661 \end{aligned} \quad (4.34)$$

Secara lebih lengkap hasil perhitungan dari nilai *silhouette coefficient* masing-masing dokumen dapat dilihat pada Tabel 4.26.

Tabel 4.26 Hasil Pengujian *Silhouette Coefficient* Dokumen

Doc	*Silhouette Coefficient*
Doc1	0,006607
Doc2	0
Doc3	0
Doc4	0
Doc5	0
Doc6	1
Doc7	-0,01999

Langkah terakhir dalam menghitung *silhouette coefficient* adalah menghitung rata-rata *silhouette coefficient* keseluruhan dokumen sebagai berikut:

$$\begin{aligned} AVG\_Sil &= \frac{s(Doc1) + s(Doc2) + \ldots + s(Doc7)}{N} \\ &= \frac{0.006607 + 0 + \ldots + (-0.01999)}{7} \\ &= \frac{0.98662}{7} \\ &= 0.14095 \end{aligned} \quad (4.35)$$

Maka dengan demikian diperoleh hasil untuk pengujian manualisasi perhitungan data memiliki nilai rata-rata *silhouette coefficient* sebesar 0,14095.

4.3.5 Manualisasi Evaluasi *Purity*

Tahap terakhir yang dilakukan dalam manualisasi adalah melakukan evaluasi atau pengujian dengan menghitung nilai *purity*. Langkah pertama dalam melakukan evaluasi dengan menggunakan *purity* adalah memberikan label pada masing-masing dokumen berdasarkan keminatan. Hasil pemberian label dilakukan secara manual dapat dilihat pada Tabel 4.27.

Tabel 4.27 Hasil *Labeling* Manual pada Dokumen

Doc	Label	Klaster
Doc1	MG	Klaster 1
Doc2	KC	Klaster 1
Doc3	KC	Klaster 1
Doc4	KC	Klaster 1
Doc5	SC	Klaster 1
Doc6	JKI	Klaster 2
Doc7	SI	Klaster 1

Setelah masing-masing dokumen diberikan label, langkah berikutnya adalah menghitung nilai *purity* yang dapat diperoleh dengan cara sebagai berikut:

$$\begin{aligned} purity(\Omega, C) &= \frac{1}{N}\sum_{k}\max_{j} | \omega_k \cap c_j | \\ &= \frac{1}{7}x(KC + JKI) \\ &= \frac{1}{7}x(3+1) \\ &= \frac{1}{7}x4 \\ &= 0.57143 \end{aligned} \quad (4.36)$$

Sehingga evaluasi dengan menggunakan *purity* menunjukkan hasil pengelompokan dokumen memiliki nilai *purity* sebesar 0,57143.

4.4 Perancangan Pengujian

Dalam perancangan pengujian menjelaskan cara melakukan evaluasi atau pengujian yang dilakukan dalam penelitian ini. Penelitian ini melakukan pengujian dengan menggunakan teknik *silhouette coefficient* yang telah dijelaskan pada sub bab 2.6.1 untuk mengukur kualitas dari hasil pengelompokan dokumen J-PTIIK. Pengujian ini dilakukan secara berulang untuk memperoleh hasil *silhouette*

coefficient dan *purity* optimal pada nilai *k* = 2 hingga nilai *k* = 25. Untuk masing-masing nilai *k* juga ditentukan batas terendah dan tertinggi dari nilai α dalam menentukan *centroid* awal klaster. Nilai α yang diberikan adalah α = 0,40 hingga α = 0,95 dengan interval 0,05. Pengujian ini juga dilakukan dengan mengukur nilai *purity* hasil dari pengelompokan dokumen J-PTIIK.

BAB 5 IMPLEMENTASI

5.1 Implementasi Program

Implementasi program dalam penelitian dilakukan dengan mengacu pada hasil perancangan *flowchart* yang telah dilakukan pada bab sebelumnya. Pada penelitian ini, penulis menggunakan bahasa pemrograman Java dalam melakukan implementasi program. Implementasi program dilakukan dalam sebuah sistem yang memiliki spesifikasi perangkat keras (*hardware*) dan perangkat lunak (*software*) yang telah dijabarkan pada sub bab analisis kebutuhan sistem. Dalam implementasi program terdiri dari tiga bagian yakni antara lain praproses teks, *vector space model*, dan *improved k-means.*

5.1.1 Implementasi Praproses Teks

Implementasi program pada bagian ini ditulis ke dalam sebuah kelas yang diberi nama *Preprocessing.* Pada kelas ini terdiri dari beberapa *method* untuk melaksanakan proses praproses teks pada data dokumen secara terprogram. *Method* yang ada dalam kelas ini memiliki fungsi seperti hapus *tag* JUDUL dan ABSTRAK, hapus tanda baca dan angka, melakukan *case folding,* melakukan tokenisasi, menghapus *stopword* dan melakukan *stemming*.

5.1.1.1. Hapus *Tag* JUDUL Dan ABSTRAK

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *String* yang diberi nama *removeTag* dengan parameter *data* bertipe *String. Method* ini akan mengembalikan data teks yang *tag* JUDUL dan ABSTRAK-nya sudah dihapus dengan menggunakan fungsi *replaceAll* yang tersedia di Java. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.1.

Source Code 5.1 Hapus *Tag* JUDUL dan ABSTRAK

```
1  public String removeTag(String data) {
2        return
3  data.replaceAll("<JUDUL>|</JUDUL>|<ABSTRAK>|</ABSTRAK>", "");
4  }
5
```

5.1.1.2. Hapus Tanda Baca Dan Angka

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *String* yang diberi nama *removeNumberPunctuation* dengan parameter *data* bertipe *String*. *Method* ini akan mengembalikan data teks yang sudah dilakukan penghapusan tanda baca dan angka dengan menggunakan fungsi *replaceAll* yang tersedia di Java. Pada baris 2 terdapat penggunaan *regex* [^a-zA-Z] yang maksudnya adalah mengganti seluruh karakter yang bukan termasuk karakter alfabet baik penggunaan huruf kecil dan besar menjadi karakter spasi (kosong). Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.2.

Source Code 5.2 Hapus Tanda Baca Dan Angka

```
1  public String removeNumberPunctuation(String data) {
2          return data.replaceAll("[^a-zA-Z]", " ");
3  }
```

5.1.1.3. *Case Folding*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *String* yang diberi nama *caseFolding* dengan parameter *data* bertipe *String*. *Method* ini akan mengembalikan data teks yang sudah diubah seluruh karakternya menjadi huruf kecil dengan menggunakan fungsi *toLowerCase* yang tersedia di Java. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.3.

Source Code 5.3 *Case Folding*

```
1  public String caseFolding(String data) {
2          return data.toLowerCase();
3  }
```

5.1.1.4. Tokenisasi

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *array String* yang diberi nama *tokenization* dengan parameter *data* bertipe *String*. Pada baris 2 terdapat proses memecah data teks menjadi bentuk token dengan menggunakan fungsi *split* yang tersedia di Java. *Method* ini akan mengembalikan kumpulan *array* token hasil dari proses tokenisasi. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.4.

Source Code 5.4 Tokenisasi

```
1  public String[] tokenization(String data) {
2          String[] term = data.split("\\s+");
3          return term;
4  }
```

5.1.1.5. Hapus *Stopword*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *ArrayList String* yang diberi nama *removeStopword* dengan parameter *data* bertipe *array String*. Pada baris 2-3 program terlebih dahulu membaca file daftar *stopword* yang terdapat dalam bahasa Indonesia, kemudian daftar *stopword* ini disimpan dalam sebuah variabel *stopwordsList.* Pada baris 6-14 program mencoba untuk menghapus *stopword* yang terdapat pada kumpulan token. Dapat dilihat pada baris 9 terdapat kondisional dimana jika token memiliki *stopword,* maka pada baris 10 program akan menghapus *stopword* tersebut dari *ArrayList.*

Method ini akan mengembalikan kumpulan *ArrayList String* token hasil dari menghapus *stopword.* Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.5.

Source Code 5.5 Hapus *Stopword*

```
1  public ArrayList<String> removeStopwords(String[] data) {
2         ReadFile read = new ReadFile();
3         String[] stopwordsList = read.getStopwordsList();
4         this.data = new ArrayList<>(Arrays.asList(data));
5
6         try {
7                 this.data.stream().forEach((_item) -> {
8                         for (String stopwords : stopwordsList) {
9                                 if (this.data.contains(stopwords)) {
10                                        this.data.remove(stopwords);
11                                }
12                        }
13                });
14        } catch (Exception ex) {}
15        return this.data;
16 }
```

5.1.1.6. *Stemming*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *ArrayList String* yang diberi nama *stemming* dengan parameter *token* bertipe *ArrayList String*. Pada baris 2-4 program terlebih dahulu membaca file daftar kata dasar yang terdapat dalam bahasa Indonesia, kemudian pada baris 3-4 terdapat proses inisialisasi variabel *lemmatizer* yang menggunakan bantuan *library jsastrawi* dalam proses *stemming* Nazief dan Adriani. Pada baris 6-8 program mencoba untuk melakukan proses *stemming* terhadap kumpulan token dengan menggunakan fungsi *lemmatize* dari *library jsastrawi*.

Method ini akan mengembalikan kumpulan *ArrayList String term* yang merupakan hasil dari proses *stemming* Nazief dan Adriani dengan bantuan *library jsastrawi.* Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.6.

Source Code 5.6 *Stemming*

```
1  public ArrayList<String> stemming(ArrayList<String> token) {
2         ReadFile read = new ReadFile();
3         Lemmatizer lemmatizer = new
4  DefaultLemmatizer(read.getRootwordList());
5
6         for (int i = 0; i < token.size(); i++) {
7                 term.add(lemmatizer.lemmatize(token.get(i)));
8         }
9         return term;
10 }
```

5.1.2 Implementasi *Vector Space Model*

Implementasi program pada bagian ini ditulis ke dalam sebuah kelas yang diberi nama *VectorSpaceModel.* Pada kelas ini terdiri dari beberapa *method* untuk melaksanakan proses *vector space model* atau pembobotan kata secara terprogram. *Method* yang ada dalam kelas ini memiliki fungsi seperti menghitung frekuensi kemunculan *term* pada dokumen (*tf*), menghitung bobot *tf* (*wf*), menghitung *idf,* menghitung perkalian antara *wf* dan *idf*, dan melakukan normalisasi terhadap hasil *wf.idf*.

5.1.2.1. Hitung *tf*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setTF* yang memiliki fungsi untuk menghitung nilai *tf* pada masing-masing dokumen. Pada baris 2 terdapat perulangan *for* untuk melakukan perulangan sebanyak jumlah *term* hasil dari praproses teks. Pada baris 5-6 terdapat perulangan *for* untuk melakukan perulangan sebanyak jumlah dokumen yang ada. Proses menghitung nilai *tf* masing-masing dokumen terjadi pada baris 9-13, dimana nilai *tf* ini disimpan pada sebuah variabel *vector* yang bertipe *array double* dua dimensi. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.7.

Source Code 5.7 Hitung *tf*

```
1  private void setTF() {
2         for (int i = 0; i < term.size(); i++) {
3                int count = 0;
4
5                for (Map.Entry<String, String> data :
6  term_doc.entrySet()) {
7                      double tf = 0;
8                      String[] token = data.getValue().split("\\s");
9                      for (String t : token) {
10                           if (term.get(i).equals(t)) {
11                                 tf = tf + 1;
12                           }
13                     }
14                     vector[i][count] = tf;
15                     count = count + 1;
16               }
17        }
18 }
```

5.1.2.2. Hitung *wf*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setWeightTF* yang memiliki fungsi untuk menghitung nilai bobot *tf* atau yang disebut dengan nilai *wf* pada masing-masing dokumen. Proses menghitung nilai bobot *tf* atau *wf* masing-masing dokumen terjadi pada baris 5-8, dimana jika nilai *tf*-nya lebih besar dari 0 maka bobot dapat dihitung dengan menggunakan persamaan 1 + log *tf*. Hasil perhitungan bobot *tf* atau *wf* ini disimpan pada sebuah variabel *weightTF* yang bertipe *array double* dua dimensi. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.8.

Source Code 5.8 Hitung *wf*

```
1  private void setWeightTF() {
2         double[][] vector = getTF();
3         for (int i = 0; i < vector.length; i++) {
4                for (int j = 0; j < vector[0].length; j++) {
5                      if (vector[i][j] > 0) {
6                            weightTF[i][j] = 1 +
7  Math.log10(vector[i][j]);
8                      }
9                }
10        }
11 }
```

5.1.2.3. Hitung *idf*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setIDF* yang memiliki fungsi untuk menghitung nilai *idf.* Pada baris 5 terdapat perulangan *for* untuk melakukan perulangan sebanyak jumlah *term* atau panjang *vector*. Pada baris 8-9 terdapat perulangan *for* untuk melakukan perulangan sebanyak jumlah dokumen yang ada. Langkah pertama dalam proses ini adalah menghitung nilai *df* terlebih dahulu yang terjadi pada baris 11-16. Dimana nilai *df* ini dihitung untuk mengetahui berapa jumlah dokumen yang memiliki suatu *term* atau kata.

Langkah berikutnya adalah menghitung nilai *idf* masing-masing *term* yang terjadi pada baris 19. Hasil dari perhitungan nilai *idf* ini disimpan pada sebuah variabel *IDF* yang bertipe *array double.* Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.9.

Source Code 5.9 Hitung *idf*

```
1  private void setIDF() {
2         double[][] vector = getTF();
3         Map<String, String> term_doc = this.term_doc;
4
5         for (int i = 0; i < vector.length; i++) {
6                 int count = 0, df = 0;
7
8                 for (Map.Entry<String, String> data :
9  term_doc.entrySet()) {
10                        String[] token = data.getValue().split("\\s");
11                        for (String term : token) {
12                              if (this.term.get(i).equals(term)) {
13                                      df = df + 1;
14                                      break;
15                              }
16                        }
17                        count = count + 1;
18                }
19                IDF[i] = Math.log10((double) term_doc.size() / df);
20        }
21 }
```

5.1.2.4. Hitung *wf.idf*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setWF_IDF* yang memiliki fungsi untuk menghitung nilai *wf.idf* yang merupakan perkalian antara nilai *wf* dan *idf.* Proses menghitung nilai *wf.idf* ini terjadi pada baris 4-8, dimana terdapat *nested loop* dengan menggunakan perulangan *for* untuk mendapatkan nilai *wf.idf.* Baris 6 merupakan persamaan yang digunakan dalam melakukan perkalian antara nilai *wf* dan *idf* untuk menghasilkan nilai *wf.idf*. Hasil perhitungan nilai *wf.idf* ini disimpan pada sebuah variabel *WF_IDF* yang bertipe *array double* dua dimensi. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.10.

Source Code 5.10 Hitung *wf.idf*

```
1  private void setWF_IDF() {
2         double[][] weightTF = getWeightTF();
3
4         for (int i = 0; i < weightTF.length; i++) {
5               for (int j = 0; j < weightTF[0].length; j++) {
6                     WF_IDF[i][j] = weightTF[i][j] * IDF[i];
7               }
8         }
9  }
```

5.1.2.5. Hitung Normalisasi *wf.idf*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setNorm_WF_IDF* yang memiliki fungsi untuk menghitung normalisasi nilai *wf.idf.* Pada baris 3 terdapat inisialisasi variabel *array sp* yang digunakan untuk menyimpan sementara nilai *sumproduct* yang merupakan jumlah dari $wf.idf^2$. Proses untuk menghitung *sumproduct* ini terjadi pada baris 5-12 dengan menggunakan perulangan *for* dalam *nested loop*. Nilai *sumproduct* yang merupakan jumlah dari $wf.idf^2$ masing-masing dokumen dihitung pada baris 8-9.

Setelah nilai *sumproduct* yang merupakan jumlah dari $wf.idf^2$ masing-masing dokumen diperoleh, maka langkah selanjutnya adalah melakukan normalisasi nilai *wf.idf* yang terjadi pada baris 13-18 dengan menggunakan perulangan *for* dalam *nested loop.* Baris 15-16 merupakan persamaan yang digunakan dalam melakukan normalisasi nilai *wf.idf.* Hasil perhitungan normalisasi *wf.idf* ini disimpan pada sebuah variabel *Norm_WF_IDF* yang bertipe *array double* dua dimensi. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.11.

Source Code 5.11 Hitung Normalisasi *wf.idf*

```
1  private void setNorm_WF_IDF() {
2         double[][] WF_IDF = getWF_IDF();
3         double[] sp = new double[WF_IDF[0].length];
4
5         for (int i = 0; i < WF_IDF[0].length; i++) {
6               double sumproduct = 0;
7               for (double[] wf_idf : WF_IDF) {
8                     sumproduct = sumproduct + Math.pow(wf_idf[i],
9  2);
10              }
11              sp[i] = sumproduct;
12        }
13        for (int i = 0; i < WF_IDF.length; i++) {
14              for (int j = 0; j < WF_IDF[0].length; j++) {
15                    Norm_WF_IDF[i][j] = WF_IDF[i][j] /
16 Math.sqrt(sp[j]);
17              }
18        }
19 }
```

5.1.3 Implementasi *Improved K-Means*

Implementasi program pada bagian ini ditulis ke dalam sebuah kelas yang diberi nama *ImprovedKMeans.* Pada kelas ini terdiri dari beberapa *method* untuk melaksanakan proses pengelompokan data dokumen dengan menggunakan

metode *improved k-means* secara terprogram. *Method* yang ada dalam kelas ini memiliki fungsi seperti menghitung jarak tiap dokumen, menghitung rata-rata jarak, menghitung densitas dokumen, menghitung rata-rata densitas, menentukan dokumen yang memiliki densitas tertinggi, menentukan *centroid* awal klaster dan melakukan *k-means* seperti pada umumnya.

5.1.3.1. Hitung Jarak Tiap Dokumen

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setEuclideanDist* yang memiliki fungsi untuk menghitung jarak tiap dokumen dengan menggunakan *Euclidean distance.* Pada baris 2-7 terdapat perulangan *for* dalam *nested loop* yang digunakan untuk menghitung nilai jarak *Euclidean distance* terhadap masing-masing dokumen. Perulangan ini dilakukan sebanyak jumlah dokumen yang ada. Nilai jarak *Euclidean distance* ini disimpan pada sebuah variabel *euclideanDist* yang bertipe *array double* dua dimensi.

Pada baris 4-5 program akan melakukan proses untuk menghitung nilai jarak *Euclidean distance* dengan memanggil *method euclideanDistance*. Dimana parameter yang digunakan dalam *method* ini adalah nilai normalisasi *wf.idf* pada masing-masing vektor dokumen. Langkah pertama untuk menghitung nilai jarak *Euclidean distance* ini adalah menghitung jumlah antara vektor dokumen *i* dikurangi vektor dokumen *j* yang kemudian dikuadratkan yang dapat dilihat pada baris 14-16. Kemudian *method euclideanDistance* mengembalikan nilai jarak *Euclidean distance* yang dapat dilihat pada baris 17. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.12.

Source Code 5.12 Hitung Jarak Tiap Dokumen

```
1  private void setEuclideanDist() {
2         for (int i = 0; i < euclideanDist.length; i++) {
3               for (int j = 0; j < euclideanDist[0].length; j++) {
4                     euclideanDist[i][j] =
5  euclideanDistance(Norm_WF_IDF, Norm_WF_IDF, i, j);
6               }
7         }
8  }
9
10 private double euclideanDistance(double[][] X, double[][] Y, int i,
11 int j) {
12        double sumXmY2 = 0;
13
14        for (int k = 0; k < Norm_WF_IDF.length; k++) {
15              sumXmY2 = sumXmY2 + Math.pow((X[k][i] - Y[k][j]), 2);
16        }
17        return Math.sqrt(sumXmY2);
18 }
```

5.1.3.2. Hitung Rata-rata Jarak

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setMeanDist* yang memiliki fungsi untuk menghitung nilai rata-rata jarak. Pada baris 2 terdapat inisialisasi sebuah variabel *comb* bertipe *integer* yang digunakan untuk menyimpan nilai kombinasi berpasangan jarak

antara dokumen *i* dan *j*. Kombinasi ini menyebabkan untuk menghitung rata-rata jarak jika dokumen *i* dan *j* merupakan dokumen yang sama maka nilai ini tidak perlu dihitung. Hal ini dapat dilihat dalam program pada baris 5-13 pada perulangan *for* dalam *nested loop* terdapat pengecekan pada program apakah kondisi ini sudah terpenuhi. Pada baris 7-9 jika kondisi terpenuhi maka program akan melakukan proses *break* terhadap perulangan.

Untuk menghitung rata-rata jarak program pertama kali akan menghitung jumlah dari nilai jarak *Euclidean distance* antara vektor dokumen *i* dan *j* yang dapat dilihat pada baris 10. Kemudian untuk memperoleh nilai rata-rata jarak maka program akan memproses baris 14 dimana hasil jumlah tadi dibagi dengan nilai kombinasi. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.13.

Source Code 5.13 Hitung Rata-rata Jarak

```
1  private void setMeanDist() {
2          int comb = 0;
3          double sum = 0;
4
5          for (int i = 0; i < euclideanDist.length; i++) {
6                  for (int j = 0; j < euclideanDist[0].length; j++) {
7                          if (i == j) {
8                                  break;
9                          }
10                         sum = sum + euclideanDist[i][j];
11                         comb++;
12                 }
13         }
14         MeanDist = (double) sum / comb;
15 }
```

5.1.3.3. Hitung Densitas Dokumen

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* diberi nama *setDensity* memiliki fungsi untuk menghitung densitas dokumen. Baris 2-11 terdapat perulangan *for* dalam *nested loop* untuk menghitung densitas dokumen. Perhitungan densitas dokumen terdapat pada baris 6-8 dimana densitas ini dapat diperoleh dengan menggunakan persamaan densitas. Densitas dokumen ini disimpan dalam sebuah variabel *density* bertipe *array integer* yang terdapat pada baris 10. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.14.

Source Code 5.14 Hitung Densitas Dokumen

```
1  private void setDensity() {
2          for (int i = 0; i < euclideanDist.length; i++) {
3                  int dens = 0;
4
5                  for (int j = 0; j < euclideanDist[0].length; j++) {
6                          if (MeanDist - euclideanDist[i][j] >= 0) {
7                                  dens = dens + 1;
8                          }
9                  }
10                 density[i] = dens;
11         }
12 }
```

5.1.3.4. Hitung Rata-rata Densitas

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setMeanDens* yang memiliki fungsi untuk menghitung nilai rata-rata densitas dokumen. Langkah pertama untuk menghitung nilai rata-rata densitas adalah menghitung jumlah densitas keseluruhan dokumen yang terdapat pada baris 4-6. Kemudian rata-rata densitas ini dapat diperoleh dengan cara hasil penjumlahan densitas dibagi jumlah dokumen yang dapat dilihat pada baris 7. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.15.

Source Code 5.15 Hitung Rata-rata Densitas

```
1  private void setMeanDens() {
2         int sumDens = 0;
3
4         for (int i = 0; i < density.length; i++) {
5               sumDens = sumDens + density[i];
6         }
7         MeanDens = (double) sumDens / density.length;
8  }
```

5.1.3.5. Menentukan Dokumen dengan Densitas Tertinggi

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setHighDensData* yang memiliki fungsi untuk menentukan dokumen mana yang memiliki nilai densitas tertinggi. Langkah pertama yang dilakukan pada proses ini adalah menghitung nilai α dikalikan rata-rata densitas yang disimpan pada variabel *alphaMeanDens* bertipe *double*. Nilai atau variabel ini digunakan sebagai batasan dalam menentukan mana dokumen yang memiliki densitas tertinggi jika nilai densitas dokumen lebih besar dari nilai ini. Hal ini dapat dilihat pada kondisional *if* yang terdapat pada baris 8 dan 17.

Langkah selanjutnya adalah melakukan inisialisasi sebuah variabel *highDensData* bertipe *array double* dua dimensi yang terdapat pada baris 13-14. Variabel ini digunakan untuk menyimpan nilai vektor normalisasi *wf.idf* dokumen yang memiliki nilai densitas tertinggi. Pada baris 16-24 terdapat perulangan *for* dalam *nested loop* yang digunakan sebagai proses menentukan dokumen yang memiliki densitas tertinggi. Vektor dokumen densitas tertinggi ini disimpan pada variabel *highDensData* yang terjadi pada baris 19-20. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.16.

Source Code 5.16 Menentukan Dokumen dengan Densitas Tertinggi

```
1  private void setHighDensData() {
2         alphaMeanDens = alpha * MeanDens;
3         higherDensity = new ArrayList();
4         indexHigherDensity = new ArrayList();
5         int index = 0;
6
7         for (int j = 0; j < density.length; j++) {
8               if (density[j] >= alphaMeanDens) {
9                     higherDensity.add(density[j]);
10                    indexHigherDensity.add(j);
11              }
12        }
```

Source Code 5.16 (lanjutan)

```
13          highDensData = new
14  double[Norm_WF_IDF.length][higherDensity.size()];
15
16          for (int j = 0; j < density.length; j++) {
17                  if (density[j] >= alphaMeanDens) {
18                          for (int k = 0; k < Norm_WF_IDF.length; k++) {
19                                  highDensData[k][index] =
20  Norm_WF_IDF[k][j];
21                          }
22                          index++;
23                  }
24          }
25  }
```

5.1.3.6. Menentukan *Centroid* Awal Klaster

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *setInitialCentroid* yang memiliki fungsi untuk menentukan *centroid* awal klaster. Langkah pertama dalam proses ini adalah menentukan *centroid* awal klaster pertama dimana *centroid* merupakan dokumen yang memiliki nilai densitas tertinggi. Pemilihan *centroid* awal klaster pertama ini terjadi pada baris 4-7, dimana vektor *centroid* disimpan pada sebuah variabel *centroid* bertipe *array double* dua dimensi.

Untuk pemilihan *centroid* awal klaster kedua dan seterusnya terjadi pada baris 10-33 dengan menggunakan perulangan *for* dalam *nested loop*. Dimana *centroid* ini diperoleh dengan mencari jarak terjauh dari vektor *centroid* yang sudah terpilih. Hal ini dapat dilihat pada baris 17-25 dimana terdapat kondisional *if* untuk mengecek vektor dokumen mana yang memiliki jarak terjauh terhadap *centroid* yang sudah terpilih. Vektor *centroid* klaster kedua dan seterusnya ini disimpan pada variabel yang sama pada *centroid* klaster pertama yang dapat dilihat 29-31. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.17.

Source Code 5.17 Menentukan *Centroid* Awal Klaster

```
1   private void setInitialCentroid() {
2           indexCentroid = new ArrayList();
3
4           for (int i = 0; i < highDensData.length; i++) {
5                   centroid[i][0] =
6   highDensData[i][getIndexMax(higherDensity)];
7           }
8           indexCentroid.add(getIndexMax(higherDensity));
9
10          for (int x = 1; x < k; x++) {
11                  double max = 0;
12                  int index = 0;
13
14                  for (int i = 0; i < x; i++) {
15                          for (int j = 0; j < highDensData[0].length;
16  j++) {
17                                  if (!isCentroid(indexCentroid, j)) {
18                                          double dist = (double) 1 -
19  cosineSimilarity(centroid, highDensData, i, j);
20
```

Source Code 5.17 (lanjutan)

```
21                                      if (dist > max) {
22                                              max = dist;
23                                              index = j;
24                                      }
25                              }
26                      }
27              }
28
29              for (int i = 0; i < highDensData.length; i++) {
30                      centroid[i][x] = highDensData[i][index];
31              }
32              indexCentroid.add(index);
33      }
34  }
```

5.1.3.7. *K-Means*

Implementasi program pada proses ini ditulis dalam sebuah *method* bertipe *void* yang diberi nama *doKMeans* yang memiliki fungsi untuk melakukan pengelompokan dokumen dengan menggunakan *centroid* awal klaster yang sudah terpilih. Program akan mengeksekusi sebuah baris untuk memanggil dan menjalankan sebuah *method doKMeans* yang ada pada kelas *KMeans*. Pada *method doKMeans* yang terdapat pada kelas *KMeans,* program akan menjalankan metode *K-Means* dalam mengelompokkan dokumen dengan menggunakan *centroid* awal klaster yang sudah terpilih. Pada baris 2 terdapat inisialisasi sebuah variabel *iteration* bertipe *integer* yang digunakan untuk mengetahui jumlah iterasi proses *K-Means*.

Proses *K-Means* terjadi pada baris 4-12 dengan menggunakan perulangan *do-while*, dimana perulangan akan berhenti jika kondisi sudah dinyatakan konvergen. Kondisi konvergen yang digunakan adalah jika nilai *centroid* sebelumnya dan *centroid* baru tidak terdapat perubahan yang dapat dilihat pada baris 12. Terdapat kondisional *if* pada baris 5-7 dimana jika nilai iterasi sudah 2 dan seterusnya maka nilai *centroid* sebelumnya akan diperbaharui dengan *centroid* yang sekarang. Pada baris 8 program akan menjalankan *method determineCluster* untuk menentukan posisi klaster pada masing-masing dokumen. Pada baris 9 program akan menjalankan *method calculateNewCentroid* untuk menghitung nilai *centroid* baru. Secara lebih lengkap implementasi program pada proses ini dapat dilihat pada Source Code 5.18.

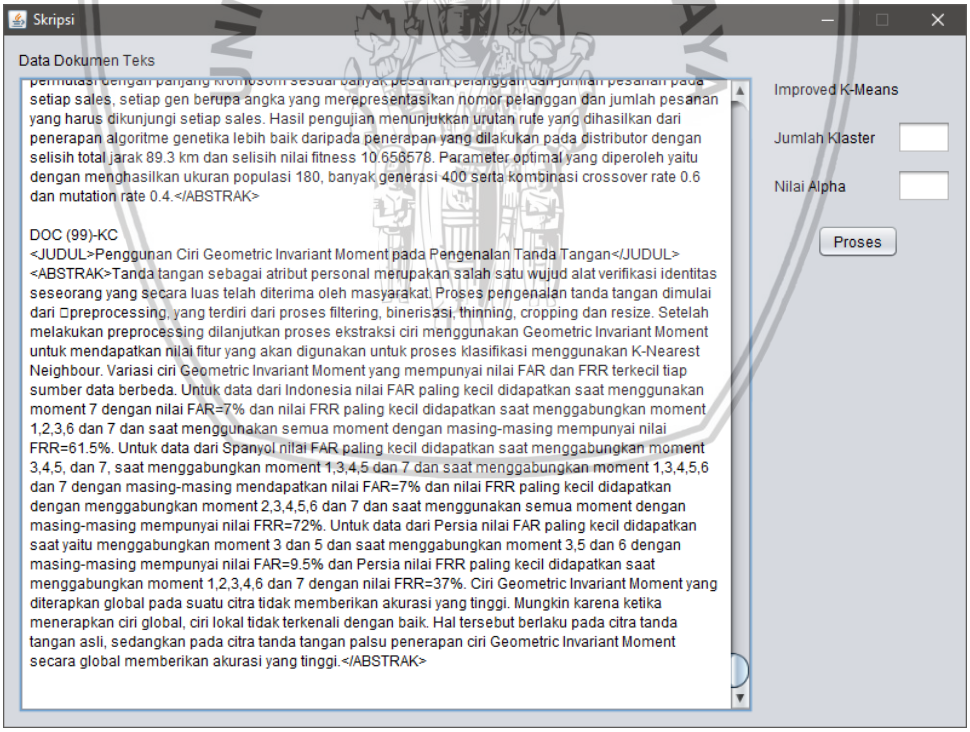
Source Code 5.18 *K-Means*

```
1   public void doKMeans() {
2           int iteration = 1;
3
4           do {
5                   if (iteration > 1) {
6                           oldCentroid = getNewCentroid();
7                   }
8                   determineCluster();
9                   calculateNewCentroid();
10
11                  iteration++;
12          } while (!isConvergen(getNewCentroid(), getOldCentroid()));
13  }
```

5.2 Implementasi Antarmuka Pengguna

Implementasi antarmuka pengguna (*user interface*) dalam penelitian ini dilakukan dengan menggunakan Java JFrame. Java JFrame merupakan sebuah tool desainer yang bersifat *graphic user interface* (GUI) yang sudah tersedia pada Netbeans IDE 8.1. Dalam implementasi antarmuka pengguna (*user interface*), antarmuka dibedakan menjadi dua bagian yakni antarmuka utama ketika program pertama kali dijalankan dan antarmuka lanjutan ketika melakukan pengelompokan dokumen. Antarmuka lanjutan terdiri dari beberapa *tab* dalam menampilkan antarmuka untuk proses pengelompokan dokumen seperti praproses teks, *vector space model*, dan *improved k-means* dan evaluasi.

Pada antarmuka utama ketika program pertama kali dijalankan terdapat sebuah *text area* untuk menampilkan seluruh data dokumen teks yang akan dilakukan pengelompokan dengan menggunakan metode *improved k-means.* Terdapat juga dua buah *text field* yang masing-masing berfungsi sebagai tempat masukan jumlah klaster dan nilai α dalam melakukan pengelompokan dokumen dengan menggunakan metode *improved k-means.* Proses pengelompokan dokumen baru dilakukan ketika *text field* sudah diberi nilai dan *button* Proses ditekan. Antarmuka utama ketika program pertama kali dijalankan dapat dilihat pada Gambar 5.28.

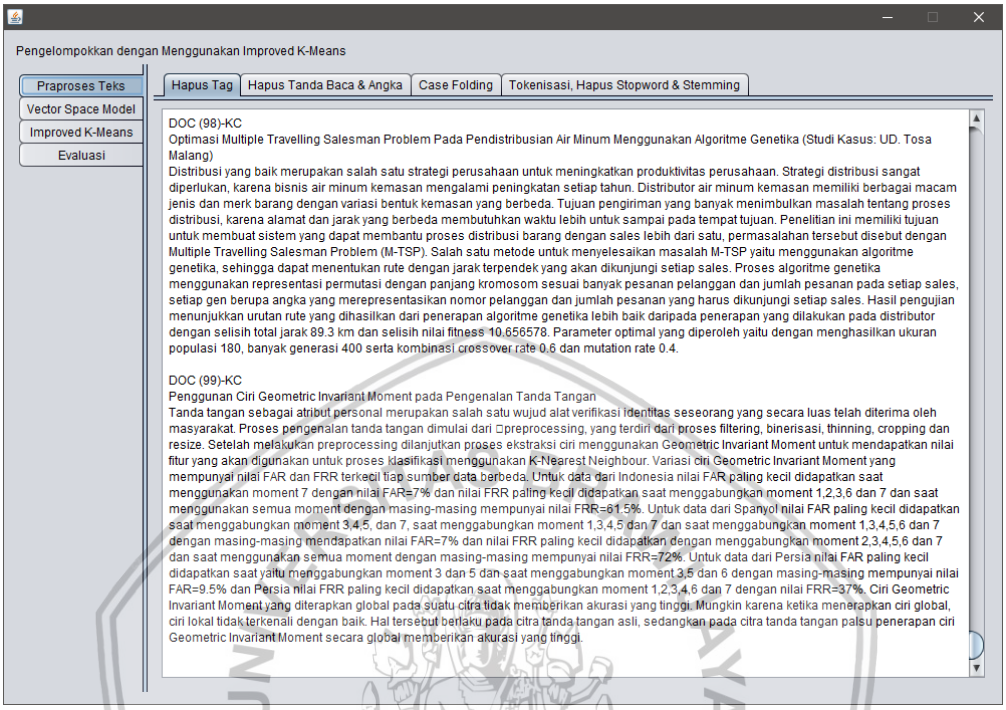


Gambar 5.28 Antarmuka Utama Ketika Program Dijalankan

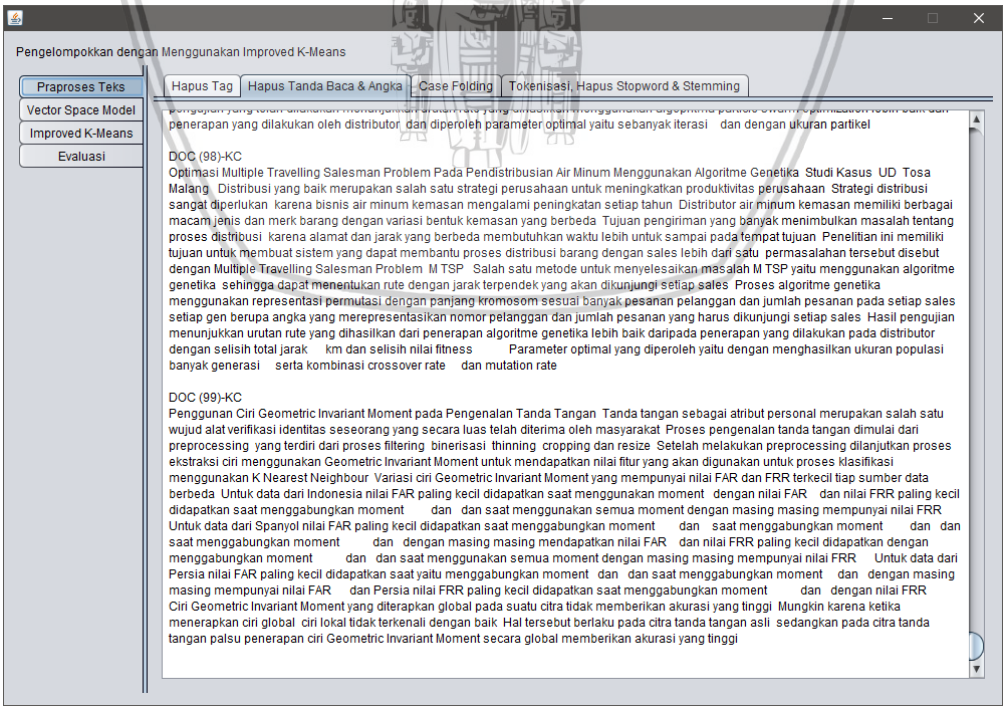
5.2.1 Antarmuka Praproses Teks

Tab pertama pada antarmuka lanjutan merupakan antarmuka praproses teks yang menampilkan hasil dari proses menghapus *tag* JUDUL dan ABSTRAK, menghapus tanda baca dan angka, melakukan *case folding*, tokenisasi, menghapus

stopword dan melakukan *stemming*. Pada antarmuka proses menghapus *tag* JUDUL dan ABSTRAK terdapat sebuah *text area* untuk menampilkan dokumen teks yang sudah dilakukan penghapusan *tag* JUDUL dan ABSTRAK. Antarmuka proses menghapus *tag* JUDUL dan ABSTRAK dapat dilihat pada Gambar 5.29.

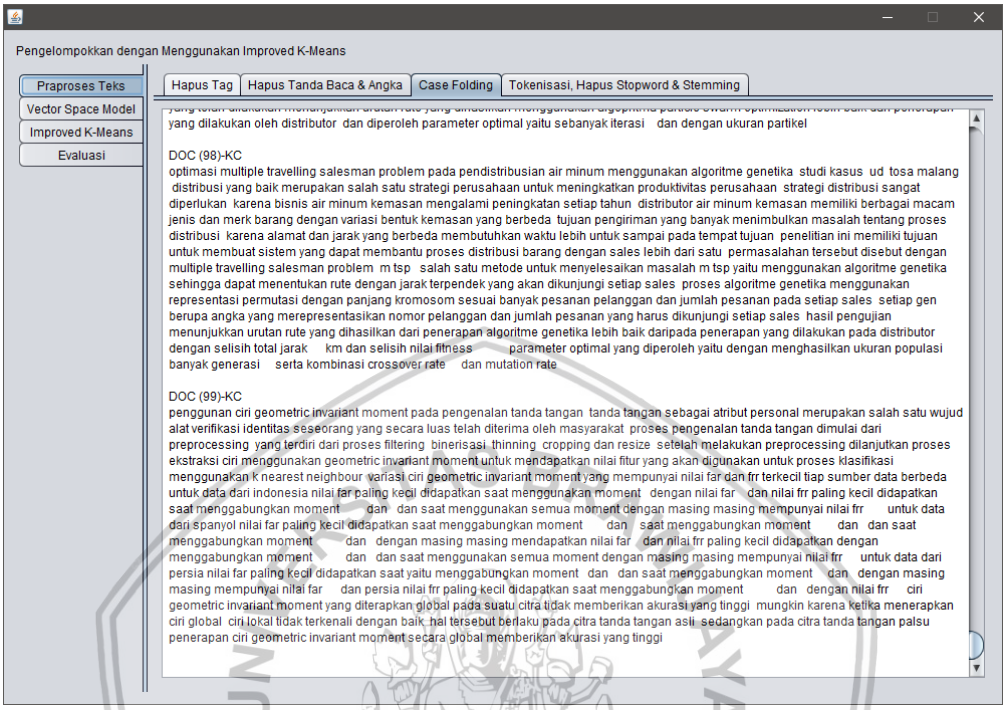


Gambar 5.29 Antarmuka Hapus *Tag* JUDUL dan ABSTRAK

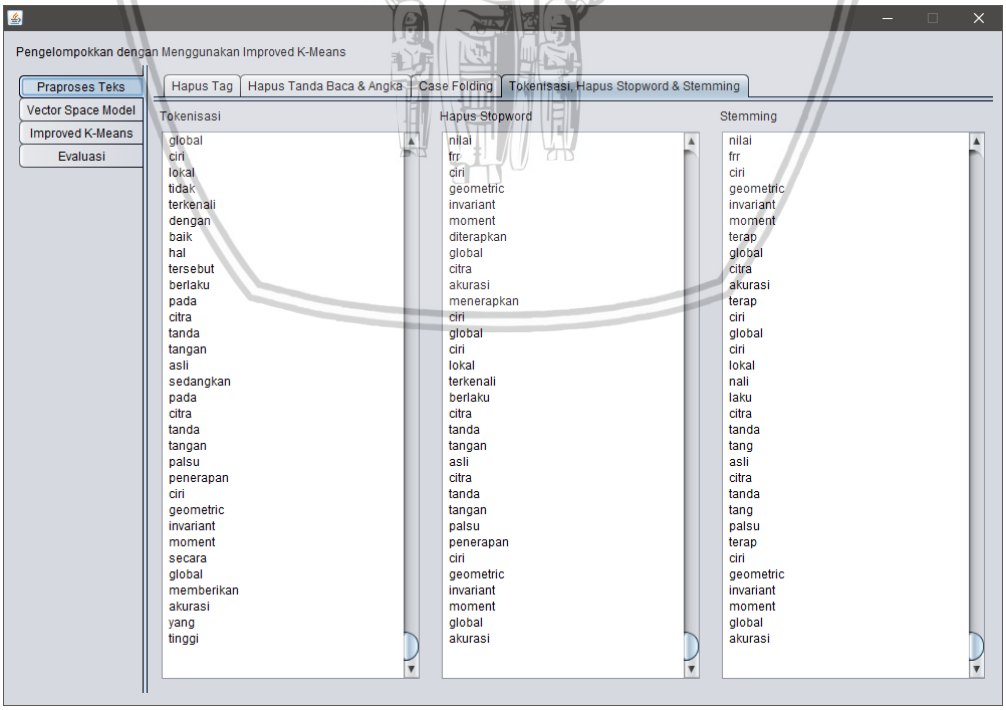


Gambar 5.30 Antarmuka Hapus Tanda Baca dan Angka

Pada antarmuka proses menghapus tanda baca dan angka terdapat sebuah *text area* untuk menampilkan dokumen teks yang sudah dilakukan penghapusan tanda baca dan angka pada dokumen teks. Antarmuka proses menghapus tanda baca dan angka dapat dilihat pada Gambar 5.30.



Gambar 5.31 Antarmuka *Case Folding*



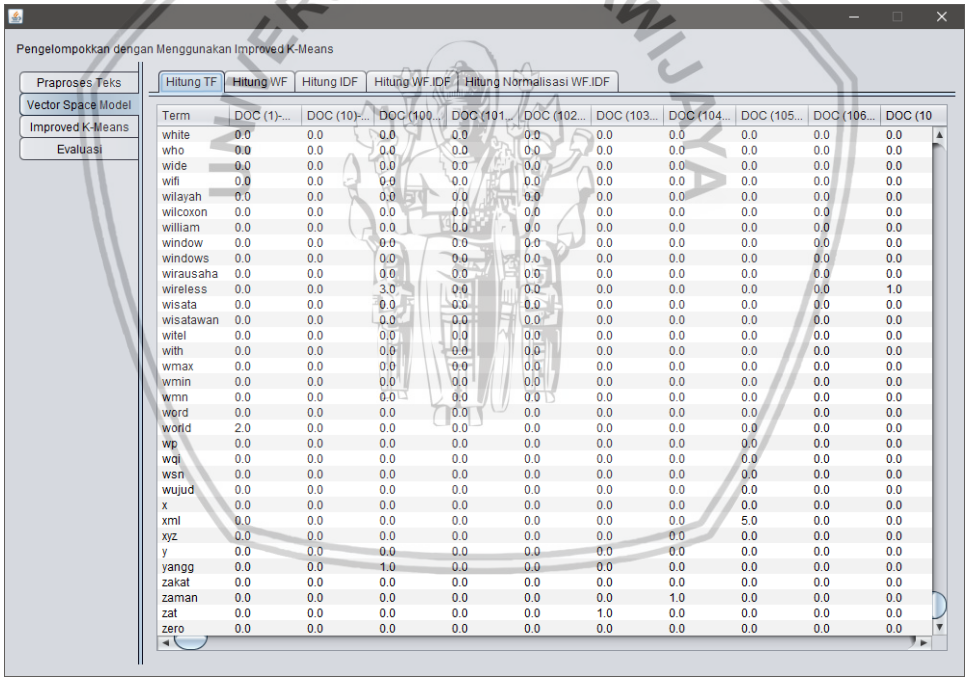
Gambar 5.32 Antarmuka Tokenisasi, Hapus *Stopword,* dan *Stemming*

Pada antarmuka proses melakukan *case folding* terdapat sebuah *text area* untuk menampilkan dokumen teks yang sudah dilakukan *case folding* dengan cara mengubah seluruh karakter menjadi huruf kecil. Antarmuka proses melakukan *case folding* dapat dilihat pada Gambar 5.31.

Proses melakukan tokenisasi, menghapus *stopword,* dan melakukan *stemming* ditempatkan pada satu bagian antarmuka. Pada antarmuka ini terdapat tiga *text area* yang masing-masing menampilkan dokumen teks yang sudah dilakukan tokenisasi, menghapus *stopword,* dan *stemming*. Antarmuka proses melakukan tokenisasi, menghapus *stopword,* dan melakukan *stemming* dapat dilihat pada Gambar 5.32.

5.2.2 Antarmuka *Vector Space Model*

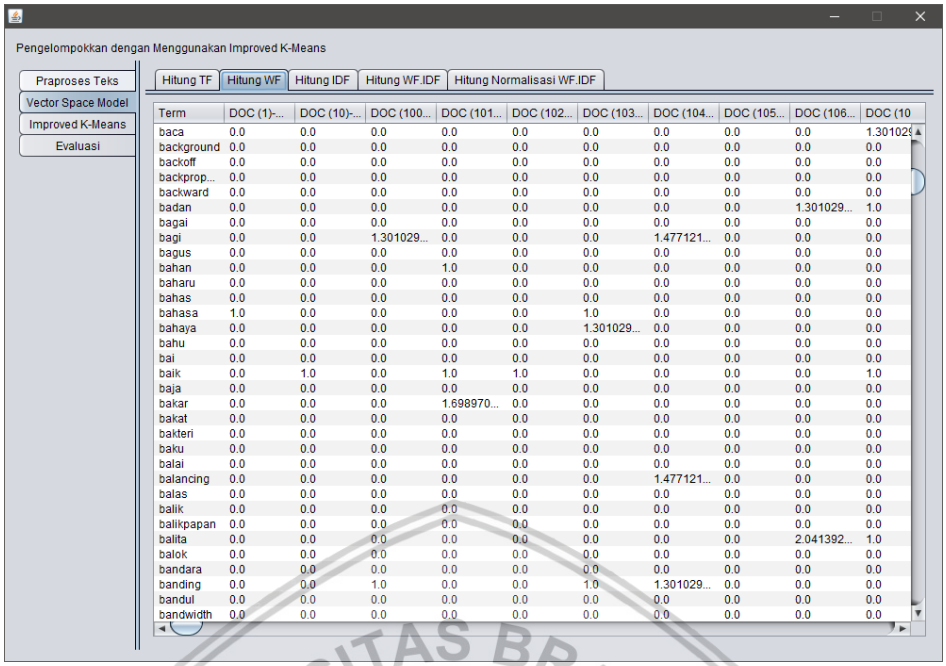
Tab kedua pada antarmuka lanjutan merupakan antarmuka *vector space model* atau pembobotan kata yang menampilkan hasil dari proses menghitung *tf, wf, idf*, *wf.idf* dan normalisasi *wf.idf*. Pada antarmuka proses menghitung *tf* terdapat sebuah *table* untuk menampilkan nilai *tf*. Antarmuka proses menghitung *tf* masing-masing dokumen dapat dilihat pada Gambar 5.33.



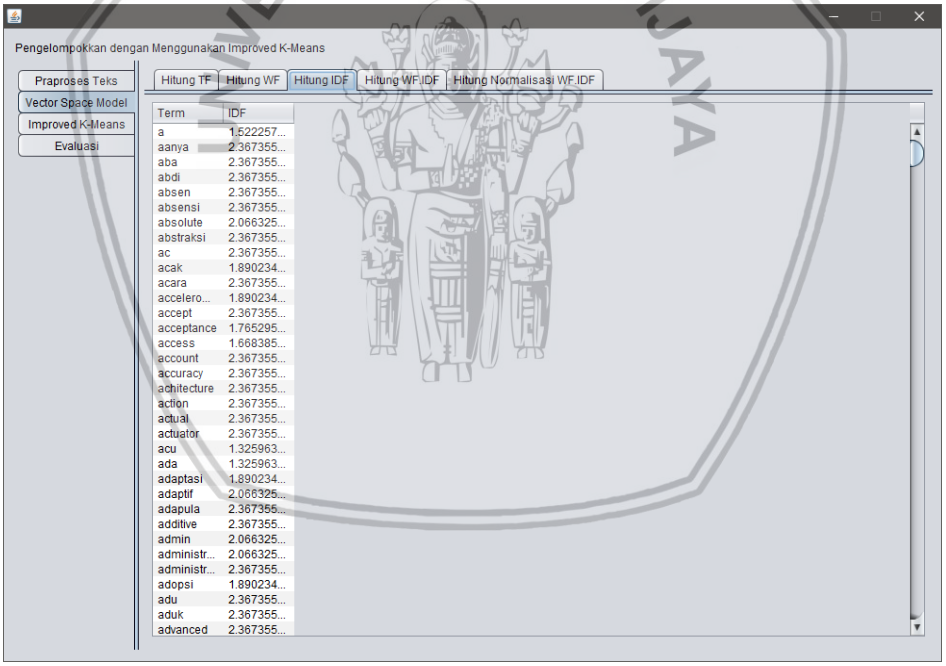
Gambar 5.33 Antarmuka Hitung *tf*

Pada antarmuka proses menghitung *wf* terdapat sebuah *table* untuk menampilkan nilai *wf* (bobot *tf*) pada masing-masing dokumen. Antarmuka proses menghitung *wf* masing-masing dokumen dapat dilihat pada Gambar 5.34.

Pada antarmuka proses menghitung *idf* terdapat sebuah *table* untuk menampilkan nilai *idf* pada masing-masing kata (*term*). Antarmuka proses menghitung *idf* masing-masing kata (*term*) dapat dilihat pada Gambar 5.35.



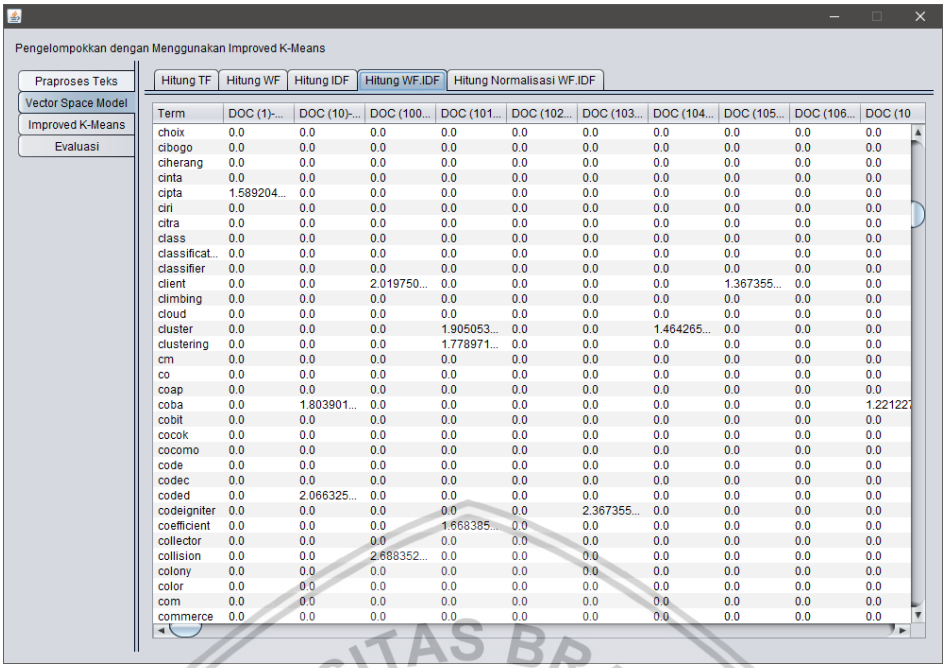
Gambar 5.34 Antarmuka Hitung *wf*



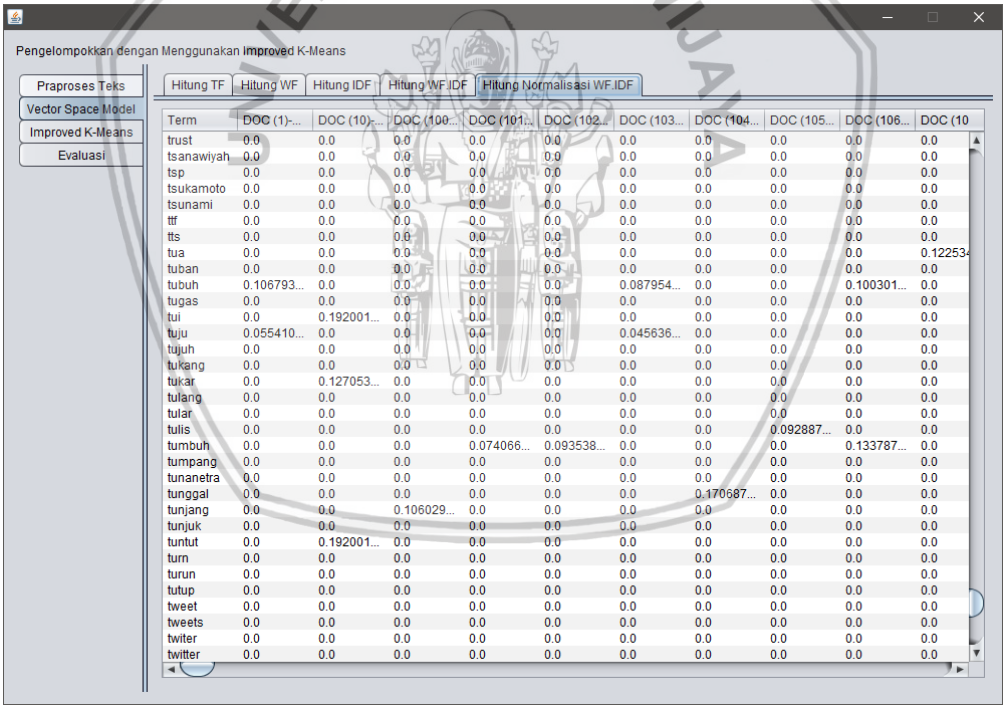
Gambar 5.35 Antarmuka Hitung i*df*

Pada antarmuka proses menghitung *wf.idf* terdapat sebuah *table* untuk menampilkan nilai pembobotan *wf.idf*. Antarmuka proses menghitung *wf.idf* masing-masing dokumen dapat dilihat pada Gambar 5.36.

Pada antarmuka proses menghitung normalisasi *wf.idf* terdapat sebuah *table* untuk menampilkan nilai normalisasi pembobotan *wf.idf* pada masing-masing dokumen. Antarmuka proses menghitung normalisasi *wf.idf* masing-masing dokumen dapat dilihat pada Gambar 5.37.



Gambar 5.36 Antarmuka Hitung *wf.idf*

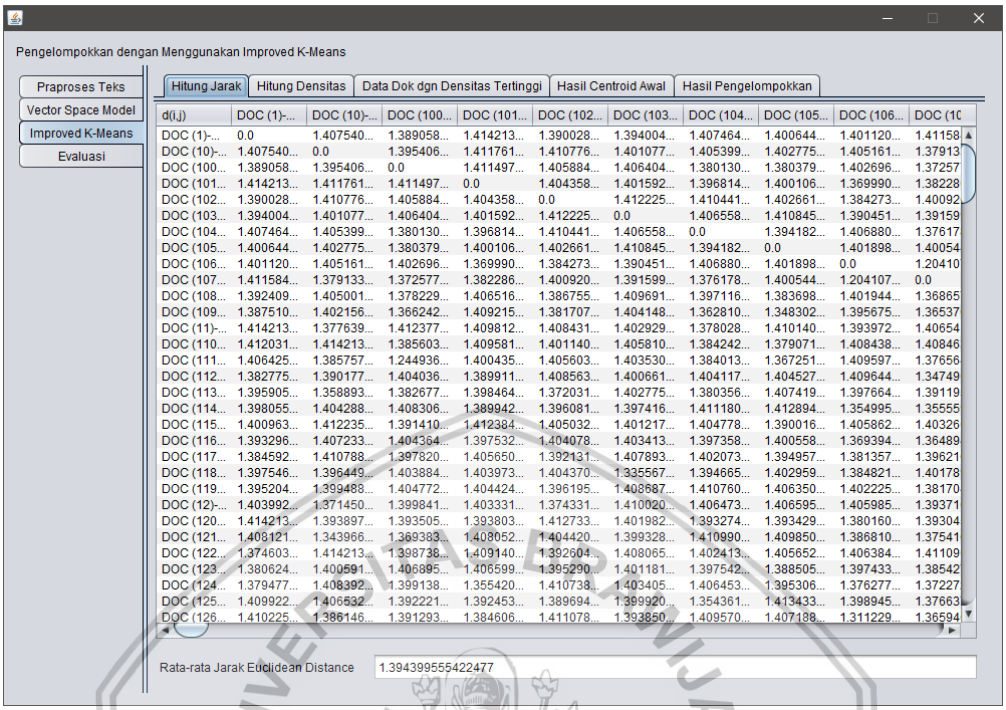


Gambar 5.37 Antarmuka Hitung Normalisasi *wf.idf*

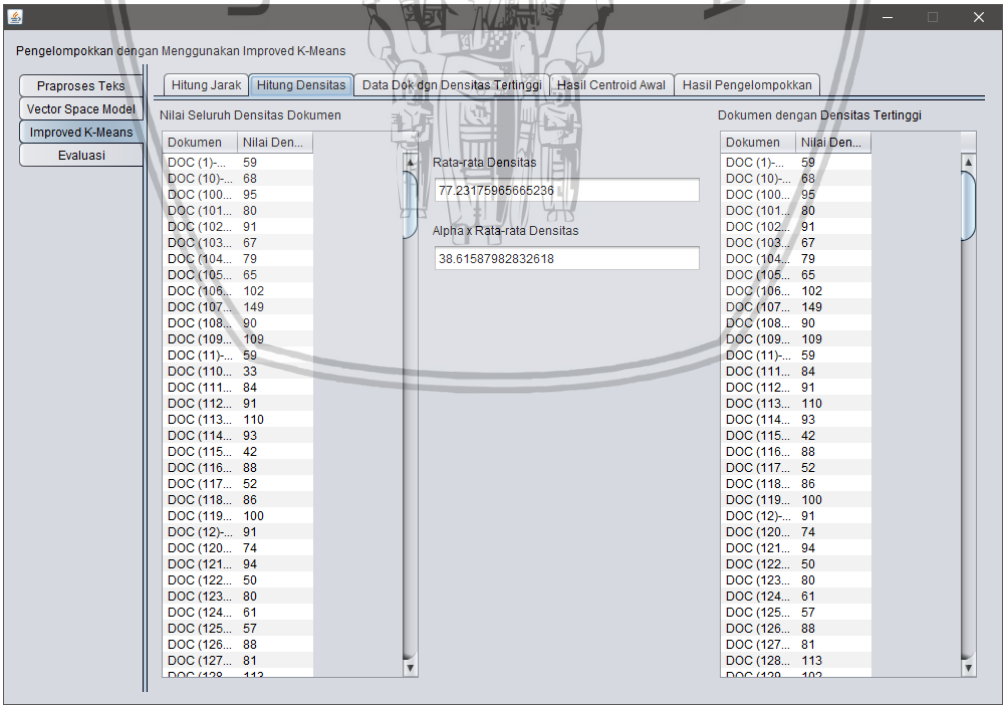
5.2.3 Antarmuka *Improved K-Means*

Tab ketiga pada antarmuka lanjutan merupakan antarmuka *improved k-means* yang menampilkan hasil dari proses menghitung jarak dan rata-rata jarak, menghitung densitas dan rata-rata densitas, penentuan *centroid* awal klaster dan melakukan pengelompokan dengan *k-means*. Pada antarmuka proses menghitung jarak dan rata-rata jarak terdapat sebuah *table* untuk menampilkan jarak tiap

dokumen dan *text area* untuk menampilkan nilai rata-rata jarak. Antarmuka proses menghitung jarak dan rata-rata jarak dapat dilihat pada Gambar 5.38.



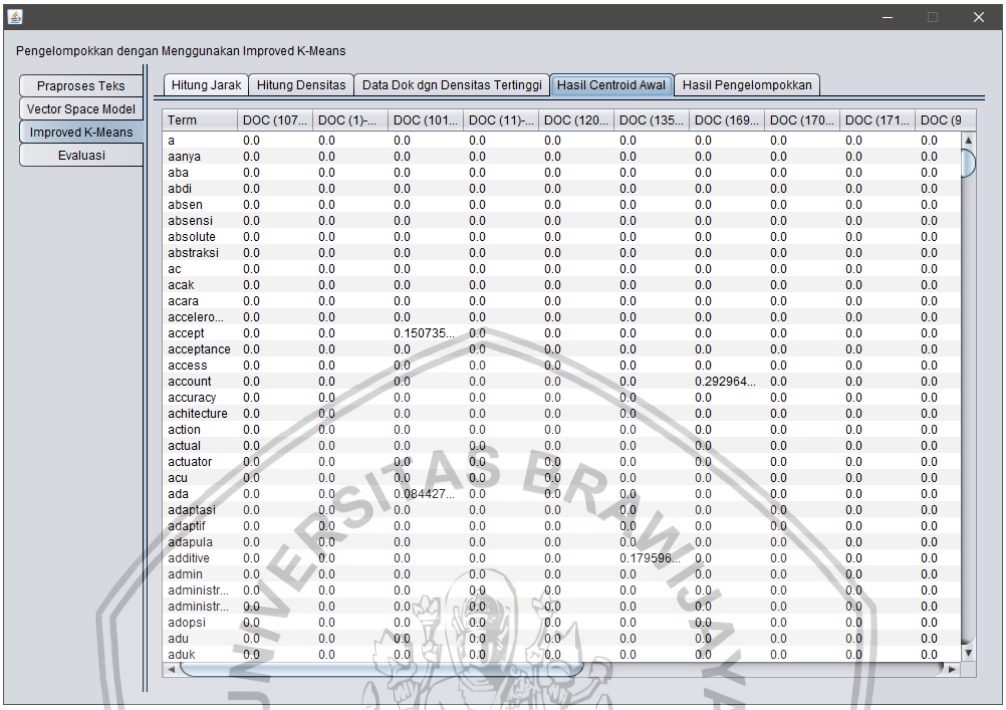
Gambar 5.38 Antarmuka Hitung Jarak dan Rata-rata Jarak



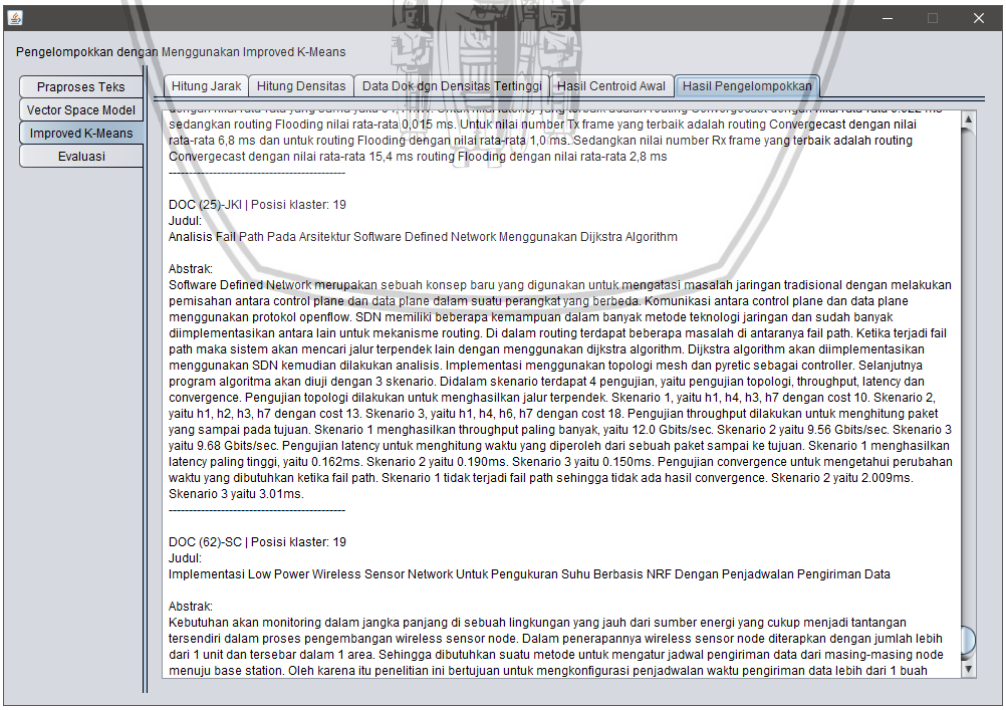
Gambar 5.39 Antarmuka Hitung Densitas dan Rata-rata Densitas

Pada antarmuka proses menghitung densitas dan rata-rata densitas terdapat dua buah *table* yang masing-masing berfungsi untuk menampilkan nilai densitas dokumen dan dokumen yang memiliki densitas tertinggi. Pada antarmuka ini juga

terdapat dua buah *text field* yang masing-masing berfungsi untuk menampilkan nilai rata-rata densitas dan hasil perkalian α dengan rata-rata densitas. Antarmuka proses menghitung densitas dan rata-rata densitas dapat dilihat pada Gambar 5.39.



Gambar 5.40 Antarmuka *Centroid* Awal Klaster



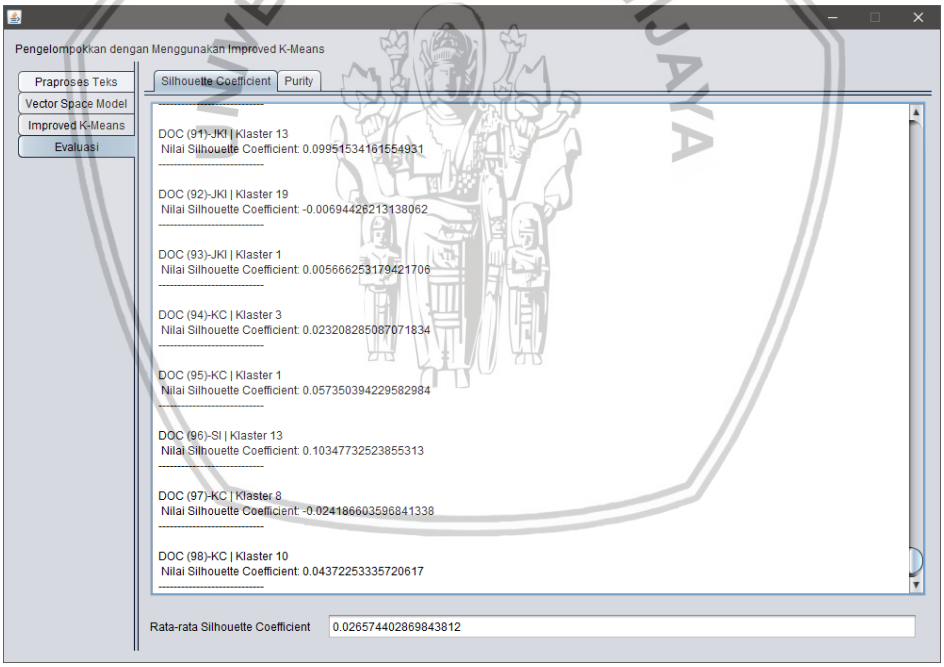
Gambar 5.41 Antarmuka Hasil Pengelompokan Dokumen

Pada antarmuka proses menentukan *centroid* awal klaster terdapat sebuah *table* untuk menampilkan hasil *centroid* awal klaster yang akan digunakan dalam melakukan pengelompokan. Antarmuka hasil *centroid* awal klaster yang akan digunakan dalam melakukan pengelompokan dapat dilihat pada Gambar 5.40.

Pada antarmuka proses melakukan pengelompokan dengan *k-means* terdapat sebuah *text area* untuk menampilkan hasil pengelompokan dokumen teks. Antarmuka hasil pengelompokan dokumen teks dapat dilihat pada Gambar 5.41.

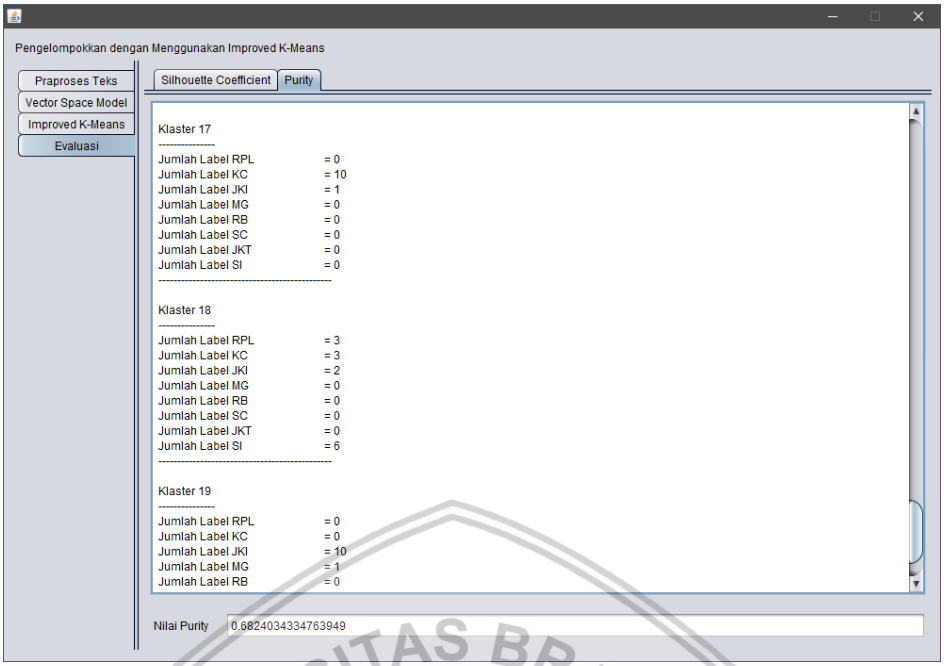
5.2.4 Antarmuka Evaluasi

Tab keempat pada antarmuka lanjutan merupakan antarmuka evaluasi atau pengujian hasil pengelompokan yang menampilkan hasil dari proses evaluasi atau pengujian dengan menggunakan *silhouette coefficient* dan *purity*. Pada antarmuka proses evaluasi atau pengujian dengan menggunakan *silhouette coefficient* terdapat sebuah *text area* untuk menampilkan nilai *silhouette coefficient* dari masing-masing dokumen. Pada antarmuka ini juga terdapat sebuah *text field* yang digunakan untuk menampilkan nilai rata-rata *silhouette coefficient*. Antarmuka proses evaluasi atau pengujian dengan menggunakan *silhouette coefficient* dapat dilihat pada Gambar 5.42.



Gambar 5.42 Antarmuka Evaluasi dengan *Silhouette Coefficient*

Pada antarmuka proses evaluasi atau pengujian dengan menggunakan *purity* terdapat sebuah *text area* untuk menampilkan jumlah distribusi atau sebaran label pada masing-masing klaster. Pada antarmuka ini juga terdapat sebuah *text field* yang digunakan untuk menampilkan nilai *purity*. Antarmuka proses evaluasi atau pengujian dengan menggunakan *purity* dapat dilihat pada Gambar 5.43.



Gambar 5.43 Antarmuka Evaluasi dengan *Purity*

BAB 6 PENGUJIAN

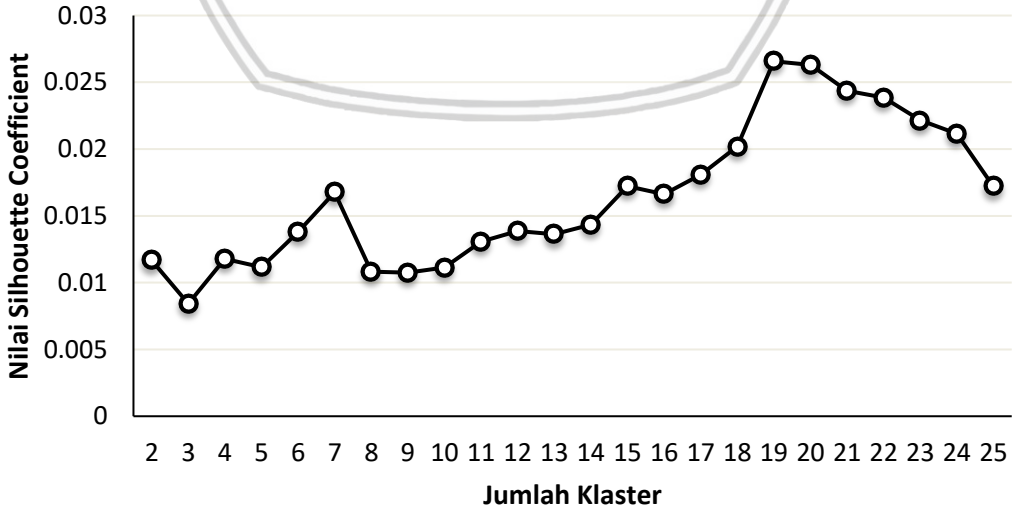
6.1 Evaluasi Pengaruh Jumlah Klaster pada Pengelompokan

Evaluasi yang pertama kali dilakukan dalam penelitian ini adalah melakukan pengujian pengaruh jumlah klaster terhadap nilai *silhouette coefficient* dan *purity*. Hal ini dilakukan untuk mengetahui apakah jumlah klaster dalam melakukan pengelompokan dokumen berpengaruh terhadap menghasilkan nilai *silhouette coefficient* dan *purity* yang bertambah tinggi atau rendah. Jumlah data yang digunakan dalam melakukan evaluasi ini adalah sebanyak 233 dokumen.

6.1.1 Evaluasi dengan *Silhouette Coefficient*

Untuk mengetahui kualitas dari hasil pengelompokan dokumen J-PTIIK maka dalam penelitian ini dilakukan evaluasi atau pengujian dengan menggunakan teknik *silhouette coefficient*. Pengujian dalam penelitian ini dilakukan secara berulang untuk mengetahui nilai *silhouette coefficient* optimal pada nilai k = 2 hingga nilai k = 25. Untuk masing-masing nilai k juga ditentukan batas terendah dan tertinggi dari nilai α dalam menentukan *centroid* awal klaster. Nilai α yang diberikan adalah α = 0,40 hingga α = 0,95 dengan interval 0,05. Hasil evaluasi atau pengujian dengan menggunakan teknik *silhouette coefficient* secara lengkap dapat dilihat pada Tabel 6.28.

Melalui hasil ini dapat diperoleh kesimpulan bahwa dalam menentukan *centroid* awal klaster nilai α optimal terdapat pada nilai 0,50. Hal ini dapat dilihat pada Gambar 6.44 dimana nilai *silhouette coefficient* terus mengalami peningkatan yang dimulai pada k = 2 hingga k = 19 dan mulai mengalami penurunan signifikan pada k = 20. Sehingga berdasarkan Tabel 6.28 dan Gambar 6.44, dapat diketahui bahwa nilai *silhouette coefficient* optimal diperoleh pada nilai k = 19 dan nilai α = 0,50 dengan nilai *silhouette coefficient* sebesar 0,026574.



Gambar 6.44 Hasil Evaluasi *Silhouette Coefficient* pada α = 0,50

Tabel 6.28 Hasil Evaluasi dengan Menggunakan *Silhouette Coefficient*

k	*Silhouette Coefficient*											
	α = 0,40	**α = 0,45**	**α = 0,50**	**α = 0,55**	**α = 0,60**	**α = 0,65**	**α = 0,70**	**α = 0,75**	**α = 0,80**	**α = 0,85**	**α = 0,90**	**α = 0,95**
2	0,011666	0,011666	0,011666	0,011666	0,011666	0,011666	0,011666	0,011666	0,009017	0,009017	0,009017	0,009017
3	0,008427	0,008427	0,008427	0,008427	0,008427	0,008427	0,008427	0,008427	0,013164	0,013164	0,013164	0,013164
4	0,011792	0,011792	0,011792	0,011792	0,011792	0,011792	0,011792	0,011792	0,008181	0,008181	0,008181	0,008181
5	0,011154	0,011154	0,011154	0,011154	0,011154	0,011154	0,011154	0,011154	0,0062	0,0062	0,0062	0,008368
6	0,013784	0,013784	0,013784	0,013784	0,013784	0,013784	0,013784	0,013784	0,004549	0,004549	0,004549	0,012772
7	0,016778	0,016778	0,016778	0,016778	0,016778	0,016778	0,016778	0,016778	0,008271	0,008271	0,008271	0,014935
8	0,010827	0,010827	0,010827	0,010827	0,010827	0,010827	0,010827	0,010827	0,006425	0,006425	0,006425	0,013595
9	0,010749	0,010749	0,010749	0,010749	0,010749	0,010749	0,010749	0,010749	0,009404	0,009404	0,009404	0,013697
10	0,011125	0,011125	0,011125	0,011125	0,011125	0,011125	0,011125	0,011125	0,006239	0,006239	0,006239	0,013392
11	0,013076	0,013076	0,013076	0,013076	0,013076	0,011149	0,010304	0,008461	0,008194	0,008194	0,008194	0,010997
12	0,013878	0,013878	0,013878	0,013878	0,013878	0,012188	0,010162	0,008391	0,008913	0,008913	0,00604	0,004695
13	0,012225	0,012225	0,013621	0,013621	0,013621	0,011054	0,018209	0,013011	0,007028	0,007028	0,005347	0,004103
14	0,013746	0,013746	0,014357	0,014357	0,014357	0,012394	0,015985	0,008568	0,005624	0,005624	0,002419	0,002699
15	0,01149	0,01149	0,017231	0,01369	0,01369	0,015348	0,014393	0,00802	0,007054	0,007054	0,002226	0,005951
16	0,011965	0,011965	0,016631	0,01494	0,01494	0,021685	0,013845	0,008955	0,009963	0,009963	0,001626	0,005964
17	0,015925	0,015925	0,018077	0,016171	0,016171	0,020869	0,012239	0,005564	0,004859	0,004859	0,003926	0,004381

Tabel 6.28 Hasil Evaluasi dengan Menggunakan *Silhouette Coefficient* (lanjutan)

k	*Silhouette Coefficient*											
	α = 0,40	**α = 0,45**	**α = 0,50**	**α = 0,55**	**α = 0,60**	**α = 0,65**	**α = 0,70**	**α = 0,75**	**α = 0,80**	**α = 0,85**	**α = 0,90**	**α = 0,95**
18	0,014333	0,014333	0,020192	0,024388	0,024388	0,019439	0,01139	0,005423	0,005591	0,007125	0,004933	0,005192
19	0,014926	0,014926	0,026574	0,024081	0,023369	0,01665	0,010054	0,005647	0,007524	0,003679	0,003671	0,005589
20	0,016513	0,016513	0,026309	0,023106	0,023504	0,015081	0,012267	0,006448	0,007298	0,003698	0,003174	0,006089
21	0,017416	0,017416	0,024371	0,023485	0,021895	0,012737	0,007756	0,003572	0,009201	0,003391	0,003314	0,003241
22	0,021018	0,021018	0,023859	0,021889	0,02084	0,012273	0,00645	0,003147	0,005855	0,003007	0,003817	0,002761
23	0,021419	0,021419	0,022151	0,020855	0,016717	0,010143	0,005316	0,002335	0,005315	0,007129	0,004099	4,81E-04
24	0,022745	0,022745	0,02115	0,016819	0,013587	0,013092	0,006653	0,004862	0,005274	0,006472	0,004237	0,002198
25	0,022435	0,022435	0,017202	0,013764	0,011585	0,01161	0,007243	0,007442	0,0054	0,005812	0,003992	0,002384

Pengujian selanjutnya yang dilakukan pada bagian ini adalah menguji dengan jumlah klaster hingga sebanyak jumlah dokumen J-PTIIK yang memiliki densitas tertinggi. Jumlah dokumen J-PTIIK dengan densitas tertinggi yang digunakan adalah pada saat nilai α = 0,50, sebab nilai α = 0,50 merupakan nilai yang optimal digunakan dalam menentukan *centroid* awal klaster. Sehingga jumlah klaster yang memungkinkan untuk dilakukan pengelompokan adalah sebanyak 224. Pengujian *silhouette coefficient* dilakukan pada jumlah klaster 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 dan 224. Hasil pengujian *silhouette coefficient* dengan jumlah klaster sebanyak jumlah dokumen J-PTIIK dengan densitas tertinggi dapat dilihat pada Tabel 6.29.

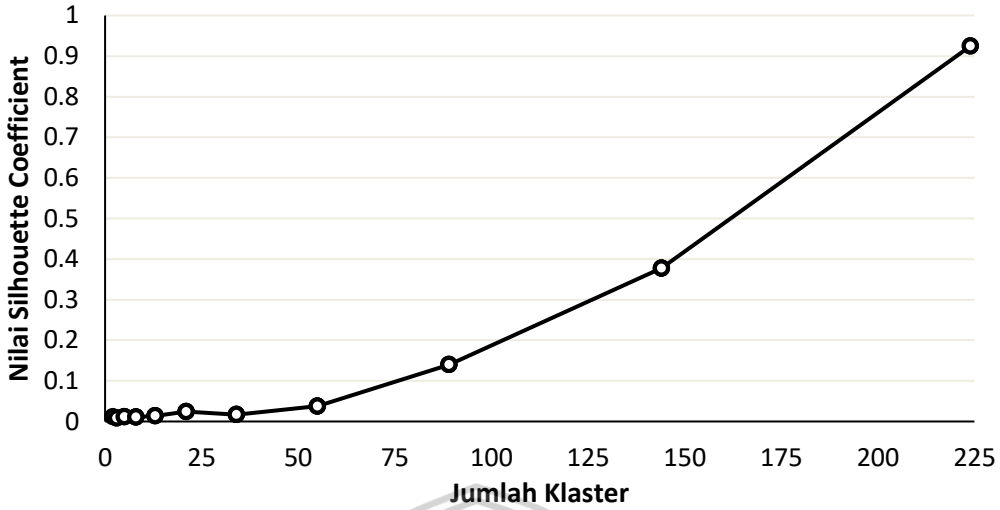
Tabel 6.29 Hasil Pengujian *Silhouette Coefficient* dengan Jumlah Klaster Sebanyak Jumlah Dokumen J-PTIIK dengan Densitas Tertinggi

k	*Silhouette Coefficient*	*k*	*Silhouette Coefficient*
2	0,011666	34	0,016767
3	0,008427	55	0,037887
5	0,011154	89	0,139739
8	0,010827	144	0,377638
13	0,013621	224	0,924896
21	0,024371		

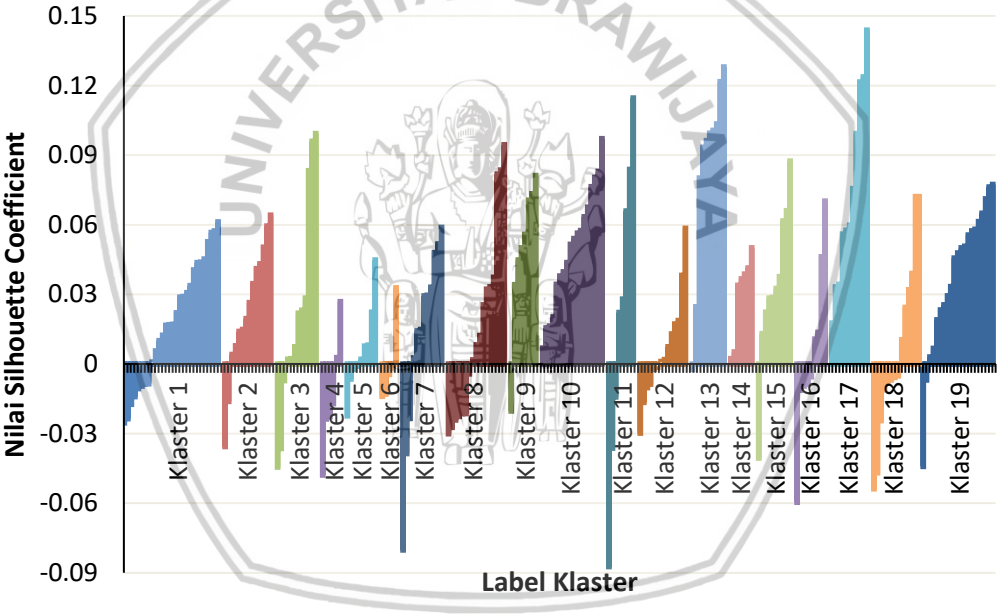
Tabel 6.29 dapat ditampilkan ke dalam bentuk grafik untuk melihat pola pengaruh jumlah klaster terhadap nilai *silhouette coefficient.* Grafik ini dapat dilihat pada Gambar 6.45. Berdasarkan Gambar 6.45, dapat diperoleh kesimpulan bahwa nilai *silhouette coefficient* terus mengalami peningkatan seiring dengan bertambahnya jumlah klaster. Khususnya ketika jumlah klaster mencapai nilai 224, nilai *silhouette coefficient* yang dihasilkan mendekati nilai 1. Hal ini disebabkan karena 233 dokumen J-PTIIK dikelompokkan ke dalam 224 klaster, sehingga terdapat banyak klaster yang hanya memiliki 1 anggota data.

Nilai *silhouette coefficient* pada saat nilai *k* = 19 dan α = 0,50 yang optimal ini berada pada rentang nilai *silhouette coefficient* yang kurang dari 0,25. Jika nilai *silhouette coefficient* kurang dari 0,25, maka kualitas hasil pengelompokan dikatakan tidak memiliki struktur klaster yang baik (Muca, Kutrolli, & Kutrolli, 2015). Nilai *silhouette coefficient* pada *k* = 19 dan α = 0,50 memiliki nilai yang kurang dari 0,25 disebabkan oleh data yang digunakan dalam penelitian ini merupakan multi-dimensi *bag of words*. Sehingga dalam penggunaan teknik *silhouette coefficient* jarak yang diperoleh antar dokumen cukup berdekatan.

Hal ini dapat dilihat melalui Gambar 6.46 dimana sebaran nilai *silhouette coefficient* masing-masing dokumen pada tiap klaster berada pada nilai lebih besar dari 0 dan sebagian kecil lainnya berada pada nilai lebih kecil dari 0. Jika nilai *silhouette coefficient* dokumen lebih kecil dari 0, maka dapat dikatakan dokumen ditempatkan pada klaster yang kurang tepat.



Gambar 6.45 Hasil Pengujian *Silhouette Coefficient* dengan Jumlah Klaster Sebanyak Jumlah Dokumen J-PTIIK dengan Densitas Tertinggi



Gambar 6.46 Nilai *Silhouette Coefficient* Dokumen pada k = 19 dan α = 0,50

6.1.2 Evaluasi dengan *Purity*

Salah satu cara lain dalam melakukan evaluasi atau pengujian terhadap hasil pengelompokan dokumen J-PTIIK adalah dengan menggunakan evaluasi eksternal seperti *purity*. Dalam melakukan evaluasi menggunakan *purity*, masing-masing dokumen J-PTIIK harus memiliki label. Pemberian label pada dokumen J-PTIIK dilakukan secara manual oleh penulis. Dalam penelitian ini penamaan label dilakukan berdasarkan keminatan pada J-PTIIK yang mengacu pada Pedoman Akademik FILKOM Universitas Brawijaya TA 2016-2020. Berikut adalah hasil penamaan label yang dilakukan berdasarkan keminatan pada dokumen J-PTIIK:

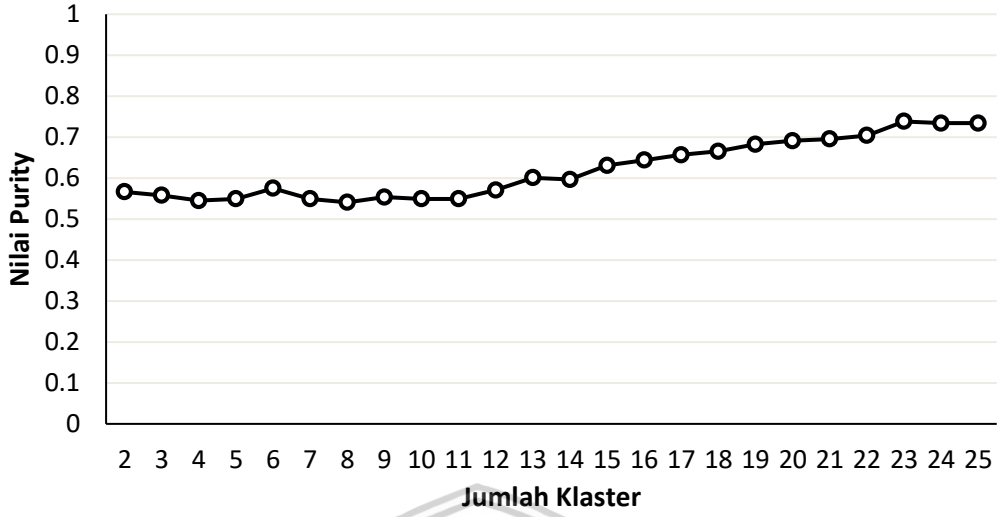
1. Rekayasa Perangkat Lunak (RPL)
2. Komputasi Cerdas (KC)
3. Jaringan Komputer Informatika (JKI)
4. *Mobile* Dan *Game* (MG)
5. Robotika (RB)
6. Sistem Cerdas (SC)
7. Jaringan Komputer Teknik (JKT)
8. Sistem Informasi (SI)

Evaluasi dengan menggunakan *purity* dilakukan pada jumlah klaster 2 hingga 25 dan nilai α = 0,50. Sebab α pada nilai 0,50 merupakan nilai optimal yang diperoleh dalam menentukan *centroid* awal klaster seperti yang telah dibahas pada sub bab sebelumnya. Hasil evaluasi dengan menggunakan *purity* pada nilai *k* = 2 hingga *k* = 25 dan nilai α = 0,50 dapat dilihat pada Tabel 6.30.

Tabel 6.30 Hasil Evaluasi dengan Menggunakan *Purity*

k	*Purity*	*k*	*Purity*
2	0,566524	12	0,570815
3	0,55794	13	0,600858
4	0,545064	14	0,596567
5	0,549356	15	0,630901
6	0,575107	16	0,643777
7	0,549356	17	0,656652
8	0,540773	18	0,665236
9	0,553648	19	0,682403
10	0,549356	20	0,690987
11	0,549356	21	0,695279
22	0,703863	24	0,733906
23	0,738197	25	0,733906

Berdasarkan Tabel 6.30 dapat diketahui bahwa nilai *purity* optimal diperoleh pada saat nilai *k* = 23 dengan nilai *purity* sebesar 0,738197. Hasil ini juga memberikan kesimpulan bahwa semakin banyak jumlah klaster maka nilai *purity* akan semakin baik. Hal ini dapat dilihat pada Gambar 6.47 dimana nilai *purity* yang terus meningkat hingga pada nilai *k* = 23.



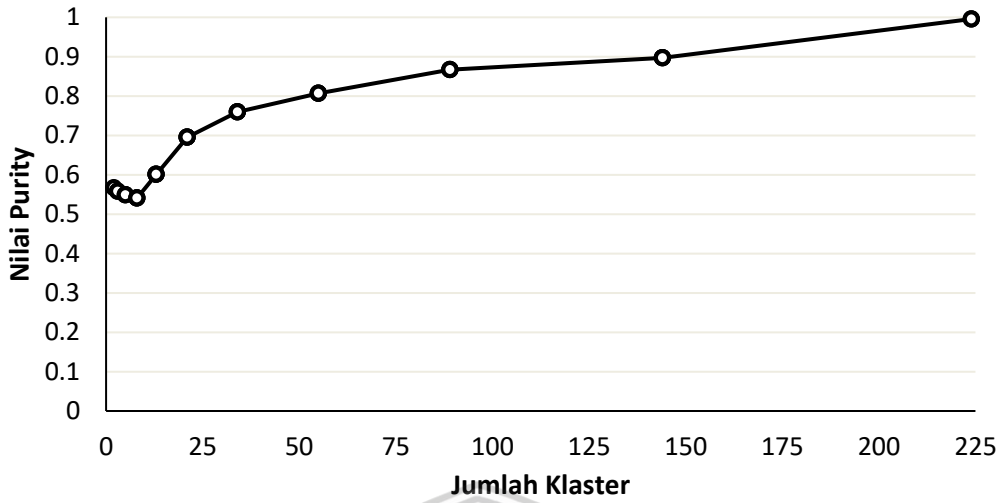
Gambar 6.47 Hasil Evaluasi dengan Menggunakan *Purity*

Pengujian selanjutnya yang dilakukan pada bagian ini adalah menguji dengan jumlah klaster hingga sebanyak jumlah dokumen J-PTIIK yang memiliki densitas tertinggi. Jumlah dokumen J-PTIIK dengan densitas tertinggi yang digunakan adalah pada saat nilai α = 0,50, sebab nilai α = 0,50 merupakan nilai yang optimal digunakan dalam menentukan *centroid* awal klaster. Sehingga jumlah klaster yang memungkinkan untuk dilakukan pengelompokan adalah sebanyak 224. Pengujian *purity* dilakukan pada jumlah klaster 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 dan 224. Hasil pengujian *purity* dengan jumlah klaster sebanyak jumlah dokumen J-PTIIK dengan densitas tertinggi dapat dilihat pada Tabel 6.31.

Tabel 6.31 Hasil Pengujian *Purity* dengan Jumlah Klaster Sebanyak Jumlah Dokumen J-PTIIK dengan Densitas Tertinggi

k	*Purity*	*k*	*Purity*
2	0.566524	34	0.759657
3	0.55794	55	0.806867
5	0.549356	89	0.866953
8	0.540773	144	0.896996
13	0.600858	224	0.995708
21	0.695279		

Tabel 6.31 dapat ditampilkan ke dalam bentuk grafik untuk melihat pola pengaruh jumlah klaster terhadap nilai *purity.* Grafik ini dapat dilihat pada Gambar 6.48. Berdasarkan Gambar 6.48, dapat diperoleh kesimpulan bahwa nilai *purity* terus mengalami peningkatan seiring dengan bertambahnya jumlah klaster. Khususnya ketika jumlah klaster mencapai nilai 224, nilai *purity* yang dihasilkan mendekati nilai 1. Hal ini disebabkan karena 233 dokumen J-PTIIK dikelompokkan ke dalam 224 klaster, sehingga terdapat banyak klaster yang hanya memiliki 1 anggota data.



Gambar 6.48 Hasil Pengujian *Purity* dengan Jumlah Klaster Sebanyak Jumlah Dokumen J-PTIIK dengan Densitas Tertinggi

Pada bagian ini juga dilakukan pemberian label secara manual terhadap hasil pengelompokan dokumen J-PTIIK, dimana pemberian label dilakukan berdasarkan topik jurnal yang ada pada tiap klaster. Pemberian label berdasarkan topik jurnal pada tiap klaster diperoleh dengan cara melihat topik jurnal yang terdapat pada dokumen yang menjadi *centroid* akhir dari proses pengelompokan dokumen J-PTIIK. Hasil pemberian label berdasarkan topik jurnal secara manual pada saat nilai *k* = 19 dapat dilihat pada Tabel 6.32.

Tabel 6.32 Hasil *Labeling* Manual Berdasarkan Topik Jurnal pada Tiap Klaster

Klaster	Label Berdasarkan Topik Jurnal
1	Klasifikasi KNN
2	Mixed Reality
3	Pengelompokan Fuzzy
4	Genetika
5	Sistem Pendukung Keputusan
6	Sistem Rekomendasi
7	Genetika
8	Particle Swarm Optimization
9	Profile Matching
10	Optimasi Genetika

Klaster	Label Berdasarkan Topik Jurnal
11	Citra Digital
12	Sistem Online
13	Analisis Structural Equation Modeling
14	Optimasi Rute
15	Internet of Things
16	Perancangan Alat
17	Text Mining
18	Evaluasi QEF
19	Analisis Wireless Sensor Network

6.2 Evaluasi Pengaruh Jumlah Data pada Pengelompokan

Evaluasi berikutnya yang dilakukan dalam penelitian ini adalah melakukan pengujian pengaruh jumlah data terhadap nilai *silhouette coefficient* dan *purity*. Hal ini dilakukan untuk mengetahui apakah jumlah data dalam melakukan pengelompokan dokumen berpengaruh terhadap menghasilkan nilai *silhouette coefficient* dan *purity* yang bertambah tinggi atau rendah. Jumlah klaster yang digunakan dalam melakukan evaluasi ini adalah pada nilai k = 19 dan k = 23. Sebab pada jumlah klaster ini diperoleh nilai *silhouette coefficient* dan *purity* optimal dalam melakukan pengelompokan 233 dokumen J-PTIIK.

6.2.1 Evaluasi dengan *Silhouette Coefficient*

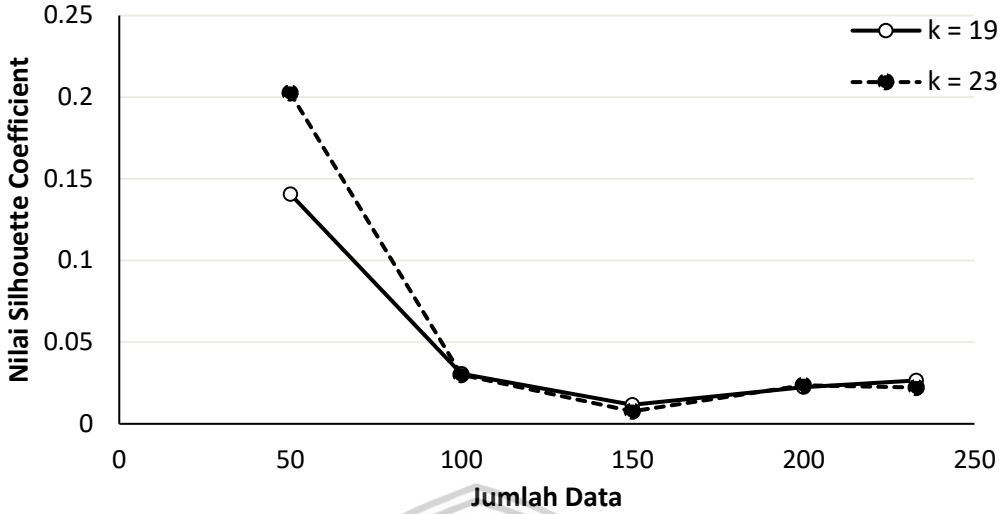
Evaluasi atau pengujian dalam penelitian ini dilakukan pada jumlah data yang berbeda, antara lain 50, 100, 150, 200 dan 233 dokumen J-PTIIK. Hasil evaluasi atau pengujian pengaruh jumlah data pada pengelompokan dengan menggunakan teknik *silhouette coefficient* secara lengkap dapat dilihat pada Tabel 6.33.

Tabel 6.33 Hasil Evaluasi Pengaruh Jumlah Data Terhadap *Silhouette Coefficient*

Jumlah Data	*Silhouette Coefficient*	
	k = 19	k = 23
50	0,140668	0,202518
100	0,030562	0,030004
150	0,011652	0,007669
200	0,022373	0,023581
233	0,026574	0,022151

Berdasarkan Tabel 6.33 dapat diketahui bahwa pada saat jumlah data dokumen J-PTIIK sebanyak 50 dokumen, nilai *silhouette coefficient* yang diperoleh cukup baik pada saat nilai k = 19 dan k = 23. Hal ini disebabkan karena jumlah data yang kecil sebanyak 50 dokumen dilakukan pengelompokan dengan menggunakan metode *improved k-means* ke dalam 19 dan 23 klaster. Sehingga jarak antar dokumen pada tiap klaster tidak cukup dekat.

Hal yang berbeda terjadi pada saat jumlah data dokumen J-PTIIK sebanyak 100 dan 150 dokumen, dimana nilai *silhouette coefficient* mengalami penurunan. Hal ini dapat dilihat pada Gambar 6.49. Nilai *silhouette coefficient* kembali mengalami peningkatan pada saat jumlah data dokumen J-PTIIK sebanyak 200 dan 233 dokumen. Sehingga dapat diperoleh kesimpulan bahwa semakin banyak jumlah data atau lebih dari 150 dokumen J-PTIIK yang digunakan dalam melakukan pengelompokan pada nilai k = 19 dan k = 23 maka nilai *silhouette coefficient* akan semakin baik.



Gambar 6.49 Hasil Evaluasi Pengaruh Jumlah Data Terhadap *Silhouette Coefficient*

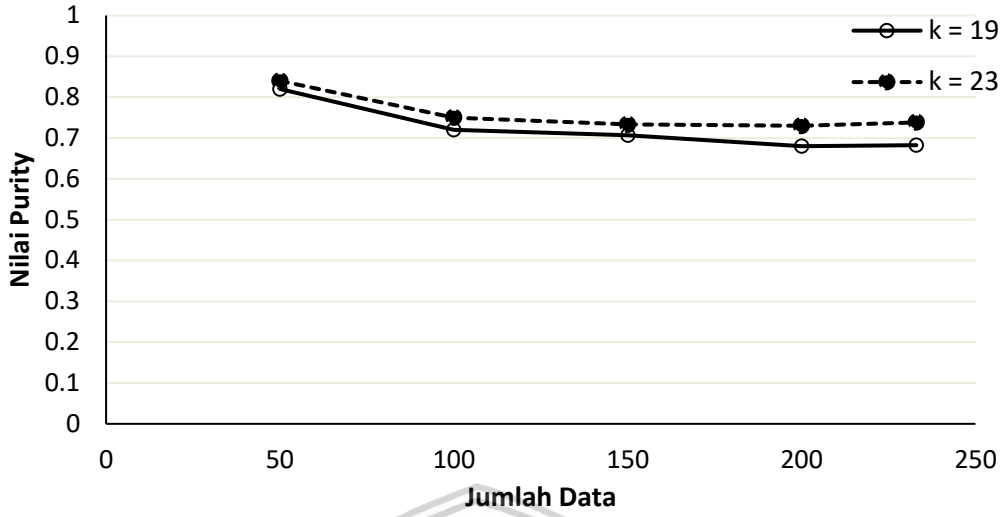
6.2.2 Evaluasi dengan *Purity*

Evaluasi atau pengujian dalam penelitian ini dilakukan pada jumlah data yang berbeda, antara lain 50, 100, 150, 200 dan 233 dokumen J-PTIIK. Hasil evaluasi atau pengujian pengaruh jumlah data pada pengelompokan dengan menggunakan *purity* secara lengkap dapat dilihat pada Tabel 6.34.

Tabel 6.34 Hasil Evaluasi Pengaruh Jumlah Data Terhadap *Purity*

Jumlah Data	*Silhouette Coefficient*	
	***k* = 19**	***k* = 23**
50	0,82	0,84
100	0,72	0,75
150	0,706667	0,733333
200	0,68	0,73
233	0,682403	0,738197

Berdasarkan Tabel 6.34 dapat diketahui bahwa semakin banyak jumlah data dokumen J-PTIIK yang digunakan dalam melakukan pengelompokan dengan menggunakan metode *improved k-means* pada nilai *k* = 19 dan *k* = 23, maka nilai *purity* yang diperoleh akan semakin kecil. Hal ini disebabkan karena nilai *purity* diperoleh dengan cara menghitung jumlah label yang paling banyak muncul pada tiap klaster, sehingga pertambahan jumlah data mempengaruhi nilai *purity*. Grafik penurunan nilai *purity* ini dapat dilihat pada Gambar 6.50.



Gambar 6.50 Hasil Evaluasi Pengaruh Jumlah Data Terhadap *Purity*

6.3 Perbandingan Evaluasi Metode *Improved K-Means dan K-Means*

Perbandingan evaluasi atau pengujian dalam penelitian ini dilakukan untuk mengetahui apakah metode *improved k-means* memiliki nilai *silhouette coefficient* dan *purity* yang lebih baik bila dibandingkan dengan metode *k-means.* Sebab pemilihan *centroid* awal klaster dalam metode *k-means* dilakukan secara acak, sehingga dapat menghasilkan nilai *silhouette coefficient* dan *purity* yang selalu berbeda ketika dilakukan pengelompokan. Hal ini berbeda dengan menggunakan metode *improved k-means* dimana pemilihan *centroid* awal yang dihasilkan selalu tetap.

6.3.1 *Silhouette Coefficient*

Dalam melakukan perbandingan ini, metode *k-means* dilakukan sebanyak 7 kali percobaan dalam melakukan pengelompokan dokumen pada nilai *k* = 2 hingga *k* = 25. Untuk menentukan nilai *silhouette coefficient* maka ditentukan dengan memperoleh nilai rata-rata *silhouette coefficient* dari 7 percobaan metode *k-means* yang dilakukan. Hasil evaluasi atau pengujian dengan menggunakan *silhouette coefficient* pada ketujuh percobaan pengelompokan dokumen dengan metode *k-means* dapat dilihat pada Tabel 6.35.

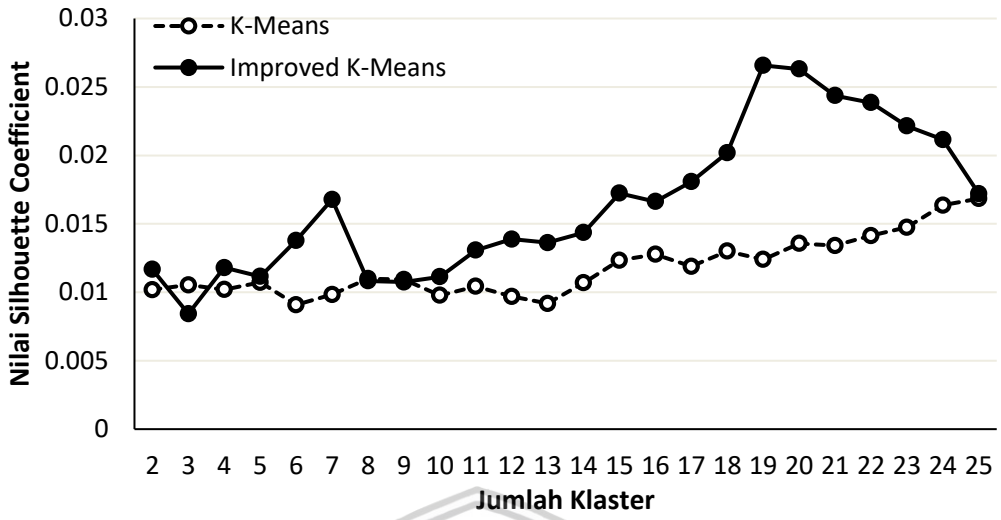
Tabel 6.35 Evaluasi *Silhouette Coefficient* pada Metode *K-Means*

k	***Silhouette Coefficient***							
	Perc. 1	**Perc. 2**	**Perc. 3**	**Perc. 4**	**Perc. 5**	**Perc. 6**	**Perc. 7**	**Rata-rata**
2	0,008398	0,013184	0,007883	0,008213	0,010037	0,011209	0,012349	0,010182
3	0,013084	0,011377	0,007131	0,007234	0,007137	0,013712	0,01409	0,010538
4	0,013617	0,01281	0,005614	0,007987	0,012453	0,007307	0,011595	0,010197
5	0,016694	0,010967	0,004165	0,009457	0,014155	0,008872	0,010746	0,010722

Tabel 6.35 (lanjutan)

k	*Silhouette Coefficient*							
	Perc. 1	**Perc. 2**	**Perc. 3**	**Perc. 4**	**Perc. 5**	**Perc. 6**	**Perc. 7**	**Rata-rata**
6	0,013678	0,001879	0,006607	0,007991	0,013696	0,010838	0,008892	0,009083
7	0,012902	0,002809	0,008357	0,011434	0,012603	0,009927	0,010833	0,009838
8	0,01244	0,005136	0,006005	0,015898	0,012033	0,010837	0,014568	0,010988
9	0,01508	0,006418	0,005009	0,017132	0,0099	0,011031	0,011955	0,010932
10	0,014501	0,006856	0,004615	0,014871	0,004735	0,00963	0,013215	0,009775
11	0,013456	0,005117	0,009238	0,01452	0,008304	0,011518	0,010813	0,010424
12	0,014362	0,003151	0,011157	0,013248	0,007331	0,010346	0,008286	0,009697
13	0,01157	0,003714	0,014132	0,012972	0,005463	0,005984	0,01044	0,009182
14	0,011435	0,009371	0,012003	0,01549	0,006904	0,008059	0,011578	0,010691
15	0,01061	0,011739	0,015811	0,016521	0,007647	0,008277	0,015782	0,012341
16	0,006662	0,011575	0,014772	0,021576	0,008546	0,009948	0,016288	0,012767
17	0,007323	0,012066	0,015823	0,020169	0,004652	0,00988	0,013332	0,011892
18	0,011818	0,011955	0,015333	0,020019	0,007178	0,010605	0,014074	0,012997
19	0,0132	0,009995	0,014821	0,016289	0,005854	0,01334	0,013249	0,012392
20	0,014142	0,012988	0,013673	0,016368	0,00738	0,014756	0,015658	0,013566
21	0,01325	0,012349	0,013575	0,017808	0,009149	0,012667	0,015088	0,013412
22	0,012221	0,010915	0,016975	0,020566	0,008892	0,013878	0,015508	0,014136
23	0,010594	0,012122	0,016238	0,018958	0,013592	0,015638	0,01604	0,01474
24	0,013752	0,013693	0,018364	0,019625	0,014297	0,017566	0,017199	0,016357
25	0,01477	0,013456	0,01779	0,019731	0,017097	0,018855	0,016191	0,016841

Nilai rata-rata *silhouette coefficient* yang terdapat pada Tabel 6.35 kemudian dibandingkan dengan nilai *silhouette coefficient* pada metode *improved k-means* untuk nilai *k* = 2 hingga *k* = 25 dan α = 0,50. Perbandingan ini ditampilkan dalam bentuk grafik perbandingan nilai *silhouette coefficient* antara metode *improved k-means* dan *k-means* yang dapat dilihat pada Gambar 6.51. Berdasarkan Gambar 6.51 ini dapat diperoleh kesimpulan bahwa penggunaan metode *improved k-means* dalam melakukan pengelompokan dokumen memiliki kualitas hasil klaster yang lebih baik bila dibandingkan dengan penggunaan metode *k-means.* Metode *improved k-means* memiliki nilai rata-rata *silhouette coefficient* pada *k* = 2 hingga *k* = 25 sebesar 0,016457654. Sedangkan metode *k-means* hanya memiliki nilai rata-rata *silhouette coefficient* sebesar 0,011820563.



Gambar 6.51 Perbandingan Evaluasi *Silhouette Coefficient* Metode *Improved K-Means* dan *K-Means*

Hal ini disebabkan oleh pemilihan *centroid* awal klaster dalam metode *improved k-means* dilakukan secara terstruktur melalui algoritme pemilihan *centroid* yang sudah dibahas pada bab sebelumnya. Sehingga setiap kali pengelompokan dilakukan dengan menggunakan metode *improved k-means* selalu menghasilkan klaster yang sama. Hal yang berbeda terjadi ketika pengelompokan dokumen dilakukan dengan menggunakan metode *k-means,* maka pengelompokan menghasilkan klaster yang berbeda setiap pengelompokan dokumen dilakukan. Hasil klaster yang berbeda ini tentu berpengaruh terhadap nilai rata-rata *silhouette coefficient* ketujuh percobaan pada metode *k-means.* Nilai yang dihasilkan memiliki nilai *silhouette coefficient* yang cenderung lebih kecil bila dibandingkan penggunaan metode *improved k-means.*

6.3.2 *Purity*

Dalam melakukan perbandingan ini, metode *k-means* dilakukan sebanyak 7 kali percobaan dalam melakukan pengelompokan dokumen pada nilai *k* = 2 hingga *k* = 25. Untuk menentukan nilai *purity* maka ditentukan dengan memperoleh nilai rata-rata *purity* dari 7 percobaan metode *k-means* yang dilakukan. Hasil evaluasi atau pengujian dengan menggunakan *purity* pada ketujuh percobaan pengelompokan dokumen dengan metode *k-means* dapat dilihat pada Tabel 6.36.

Tabel 6.36 Evaluasi *Purity* pada Metode *K-Means*

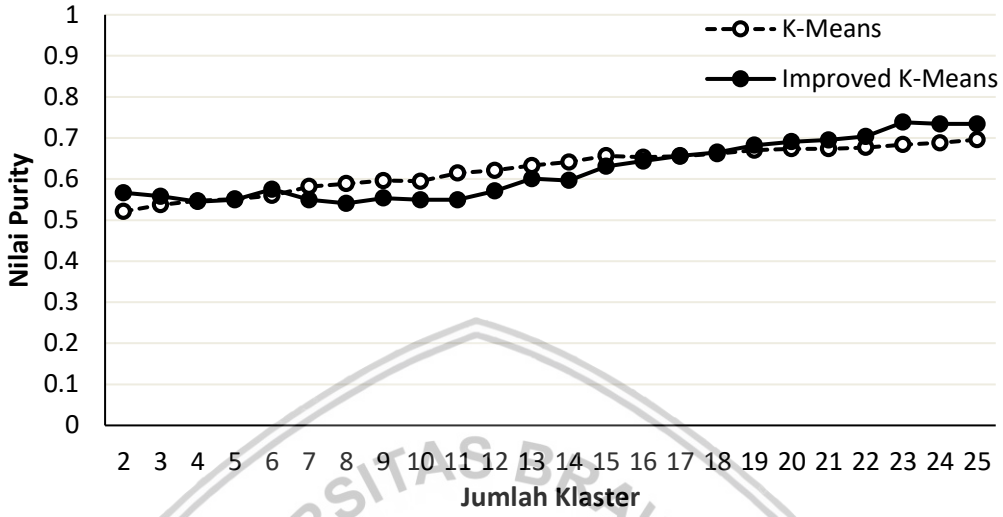
k	*Purity*							
	Perc. 1	**Perc. 2**	**Perc. 3**	**Perc. 4**	**Perc. 5**	**Perc. 6**	**Perc. 7**	**Rata-rata**
2	0,51073	0,55794	0,51073	0,51073	0,51073	0,51073	0,536481	0,521153
3	0,51073	0,55794	0,519313	0,51073	0,55794	0,540773	0,562232	0,537094
4	0,562232	0,549356	0,51073	0,51073	0,592275	0,536481	0,566524	0,546904

Tabel 6.36 (lanjutan)

k	*Purity*							
	Perc. 1	**Perc. 2**	**Perc. 3**	**Perc. 4**	**Perc. 5**	**Perc. 6**	**Perc. 7**	**Rata-rata**
5	0,570815	0,536481	0,532189	0,527897	0,592275	0,540773	0,55794	0,551196
6	0,583691	0,523605	0,549356	0,545064	0,587983	0,55794	0,570815	0,559779
7	0,596567	0,549356	0,570815	0,60515	0,600858	0,549356	0,600858	0,581852
8	0,592275	0,540773	0,566524	0,626609	0,60515	0,575107	0,613734	0,588596
9	0,630901	0,55794	0,549356	0,643777	0,609442	0,570815	0,609442	0,595953
10	0,639485	0,592275	0,549356	0,639485	0,55794	0,566524	0,618026	0,594727
11	0,652361	0,587983	0,609442	0,652361	0,575107	0,596567	0,626609	0,614347
12	0,67382	0,579399	0,639485	0,656652	0,575107	0,596567	0,626609	0,621091
13	0,690987	0,592275	0,67382	0,648069	0,596567	0,596567	0,630901	0,632741
14	0,695279	0,639485	0,660944	0,665236	0,60515	0,60515	0,618026	0,641324
15	0,695279	0,648069	0,690987	0,665236	0,618026	0,622318	0,656652	0,656652
16	0,678112	0,656652	0,652361	0,67382	0,630901	0,626609	0,652361	0,652974
17	0,686695	0,665236	0,660944	0,665236	0,626609	0,635193	0,648069	0,655426
18	0,708155	0,690987	0,660944	0,656652	0,630901	0,639485	0,643777	0,661557
19	0,716738	0,708155	0,660944	0,656652	0,630901	0,67382	0,643777	0,670141
20	0,72103	0,716738	0,665236	0,656652	0,630901	0,682403	0,643777	0,67382
21	0,716738	0,712446	0,652361	0,660944	0,652361	0,678112	0,643777	0,67382
22	0,725322	0,703863	0,656652	0,669528	0,648069	0,678112	0,656652	0,676885
23	0,733906	0,716738	0,669528	0,665236	0,660944	0,682403	0,656652	0,68363
24	0,746781	0,72103	0,682403	0,660944	0,660944	0,678112	0,665236	0,687922
25	0,746781	0,72103	0,678112	0,652361	0,682403	0,678112	0,712446	0,695892

Nilai rata-rata *purity* yang terdapat pada Tabel 6.36 kemudian dibandingkan dengan nilai *purity* pada metode *improved k-means* untuk nilai k = 2 hingga k = 25 dan α = 0,50. Perbandingan ini ditampilkan dalam bentuk grafik perbandingan nilai *purity* antara metode *improved k-means* dan *k-means* yang dapat dilihat pada Gambar 6.52. Berdasarkan Gambar 6.52 ini dapat diperoleh kesimpulan bahwa penggunaan metode *improved k-means* dalam melakukan pengelompokan dokumen memiliki nilai *purity* yang lebih baik pada saat nilai k = 17 hingga k = 25 bila dibandingkan dengan penggunaan metode *k-means*. Tetapi secara keseluruhan, metode *improved k-means* memiliki nilai rata-rata *purity* yang lebih buruk bila dibandingkan metode *k-means.* Metode *improved k-means* hanya

memiliki nilai rata-rata *purity* pada *k* = 2 hingga *k* = 25 sebesar 0,619992847. Sedangkan metode *k-means* memiliki nilai rata-rata *purity* sebesar 0,623978132.



Gambar 6.52 Perbandingan Evaluasi *Purity* Metode *Improved K-Means* dan *K-Means*

6.4 Analisis Hasil Evaluasi

Berdasarkan evaluasi yang dilakukan, nilai *silhouette coefficient* mengalami peningkatan jika jumlah klaster terus bertambah. Akan tetapi pada beberapa jumlah klaster tertentu, nilai *silhouette coefficient* mengalami penurunan yang cukup signifikan. Nilai *silhouette coefficient* yang tinggi menunjukkan bahwa nilai terkecil rata-rata jarak suatu dokumen J-PTIIK dengan semua dokumen J-PTIIK lain yang berada pada klaster berbeda cukup besar. Sementara *silhouette coefficient* yang rendah menunjukkan bahwa nilai terkecil rata-rata jarak suatu dokumen J-PTIIK dengan semua dokumen J-PTIIK lain yang berada pada klaster berbeda lebih besar. Pada nilai *purity* terus mengalami peningkatan seiring dengan bertambahnya jumlah klaster. Hal ini menunjukkan pada saat jumlah klaster semakin besar, jumlah label terbanyak pada tiap klaster tentu juga akan semakin besar sehingga menghasilkan *purity* yang terus meningkat.

Berdasarkan evaluasi pengaruh jumlah data pada pengelompokan, pada saat jumlah data yang digunakan terlalu sedikit maka nilai *silhouette coefficient* yang dihasilkan akan tinggi. Hal ini disebabkan karena jumlah data yang sedikit dikelompokkan dengan jumlah klaster mendekati setengah dari jumlah data yakni 19 dan 23. Sehingga nilai terkecil rata-rata jarak suatu dokumen J-PTIIK dengan semua dokumen J-PTIIK lain yang berada pada klaster berbeda cukup besar dan nilai *silhouette coefficient* yang dihasilkan cukup tinggi. Sementara pada saat jumlah data yang digunakan mengalami peningkatan maka nilai *silhouette coefficient* yang dihasilkan cenderung stabil pada rentang nilai 0,01 hingga 0,03.

Sementara pada evaluasi *purity*, penggunaan jumlah data yang semakin besar menyebabkan nilai *purity* yang dihasilkan semakin kecil. Hal ini menunjukkan pada saat jumlah data semakin besar, jumlah label terbanyak pada tiap klaster tentu juga akan terbagi menjadi semakin kecil sehingga menghasilkan *purity* yang terus menurun. Hasil evaluasi juga menunjukkan metode *improved k-means* memiliki nilai *silhouette coefficient* yang lebih baik bila dibandingkan dengan penggunaan metode *k-means.* Hal ini disebabkan karena pada penggunaan metode *k-means, centroid* awal klaster selalu berbeda setiap kali pengelompokan dilakukan. Sehingga hasil pengelompokan selalu berbeda tiap proses *k-means* dilakukan. Hal ini yang menyebabkan nilai *silhouette coefficient* dari metode *k-means* lebih kecil dari metode *improved k-means.*

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan oleh penulis maka dapat diperoleh kesimpulan sebagai berikut:

1. Metode *improved k-means* yang digunakan dalam penelitian ini dapat digunakan untuk mengelompokkan dokumen J-PTIIK. Adapun tahap dalam melakukan pengelompokan dokumen ini dimulai dari melakukan praproses teks dokumen, melakukan pembobotan kata (*vector space model*), dan melakukan pengelompokan dokumen dengan menggunakan metode *improved k-means.*
2. Nilai *silhouette coefficient* optimal dalam melakukan evaluasi atau pengujian diperoleh pada saat k = 19 dan α = 0,50 dengan nilai *silhouette coefficient* sebesar 0,026574.
3. Nilai *purity* optimal dalam melakukan evaluasi atau pengujian diperoleh pada saat k = 23 dan α = 0,50 dengan nilai *purity* sebesar 0,738197.
4. Metode *improved k-means* memiliki nilai *silhouette coefficient* yang lebih baik bila dibandingkan dengan penggunaan metode *k-means*, dengan nilai rata-rata *silhouette coefficient* pada *improved k-means* sebesar 0,016457654 dan *k-means* sebesar 0,011820563.
5. Metode *improved k-means* memiliki nilai *purity* yang lebih buruk bila dibandingkan dengan penggunaan metode *k-means*, dengan nilai rata-rata *purity* pada *improved k-means* sebesar 0,619992847 dan *k-means* sebesar 0,623978132.

7.2 Saran

Saran dari penulis yang dapat diberikan terhadap penelitian lebih lanjut atau penelitian lainnya dalam bidang pengelompokan atau klasterisasi adalah sebagai berikut:

1. Pemilihan *centroid* awal klaster dapat menggunakan metode *improved k-means* yang lain atau menggunakan metode pemilihan *centroid* awal klaster yang lainnya.
2. Proses pemberian label secara otomatis dapat dilakukan pada akhir proses pengelompokan dokumen J-PTIIK.

DAFTAR PUSTAKA

A., V.K. & Aghila, G., 2010. *Text Mining Process, Techniques and Tools: an Overview.* International Journal of Information Technology and Knowledge Management, Vol. 2, No. 2, pp. 613-622

Al-Azzawy, D.S. & Al-Rufaye, F.M.L., 2017. *Arabic Words Clustering by Using K-Means Algorithm.* Annual Conference on New Trends in Information & Communication Technology Applications, pp. 263-267

Asian, J., 2007. *Effective Techniques for Indonesian Text Retrieval*. PhD. RMIT University. Tersedia melalui: <https://researchbank.rmit.edu.au/view/rmit:6312/Asian.pdf>

Asian, J., Williams, H.E., & Tahaghoghi, S.M.M., 2005. *Stemming Indonesian*. 28th Australasian Computer Science Conference, Conferences in Research and Practice in Information Technology, Vol. 38

Çakir, M.U. & Güldamlasroglu, S., 2016. *Text Mining Analysis in Turkish Language Using Big Data Tools.* IEEE 40th Annual Computer Software and Applications Conference, pp. 614-618. Tersedia melalui: <ieeexplore.ieee.org>

Chaimontree, S., Atkinson, K., & Coenen, F., 2010. *Best Clustering Configuration Metrics: Towards Multiagent Based Clustering.* [online] Tersedia di: <https://cgi.csc.liv.ac.uk/~katie/adma10.pdf> [Diakses 11 Agustus 2017]

Chandrawanshi, V.S., Tripathi, R.K., & Khan, N.U., 2016. *A Comprehensive Study on K-means Algorithms Initialization Techniques for Wireless Sensor Network*. [online] Tersedia di: <ieeexplore.ieee.org> [Diakses 14 Agustus 2017]

Chayangkoon, N. & Srivihok, A., 2016. *Two Step Clustering Model for K-Means Algorithm.* ICNCC'16, pp. 213-217. Kyoto: ACM.

Dang, S. & Ahmad, P.H., 2014. *Text Mining: Techniques and its Application.* International Journal of Engineering & Technology Innovations, Vol. 1, Issue 4, pp. 22-25. Tersedia melalui: <www.researchgate.net>

Deepa, M. & Revathy, P., 2012. *Validation of Document Clustering based on Purity and Entropy measures.* International Journal of Advanced Research in Computer and Communication Engineering, Vol. 1, Issue 3, pp. 147-152

Karimov, J. & Ozbayoglu, M., 2015. *Clustering Quality Improvement of k-means using a Hybrid Evoluationary Model.* Procedia Computer Science 61, Publication 5, pp. 38-45. Tersedia melalui: <www.sciencedirect.com>

Kaur, S. & Kalra, S., 2016. *Disease Prediction using Hybrid K-means and Support Vector Machine*. [online] Tersedia di: <ieeexplore.ieee.org> [Diakses 14 Agustus 2017]

KM, S. & Reddy, T.H., 2016. *Text Mining: An Improvised Feature Based Model Approach.* 2nd International Conference on Applied and Theoretical

Computing and Communication Technology, pp. 38-42. Tersedia melalui: <ieeexplore.ieee.org>

Kumar, L. & Bhatia, P.K., 2013. *Text Mining: Concepts, Process and Applications.* Journal of Global Research in Computer Science, Vol. 4, No. 3, pp. 36-39. Tersedia melalui: <www.jgrcs.info>

Lee, D.L., Chuang, H., & Seamons, K., 1997. *Document Ranking and the Vector-Space Model*. [online] Tersedia di: <ieeexplore.ieee.org> [Diakses 11 Agustus 2017]

Lu, C., Shi, Y., Chen, Y., Bao, S. & Tang, L., 2016. *Data Mining Applied to Oil Well Using K-means and DBSCAN*. 7th International Conference on Cloud Computing and Big Data, pp. 37-40

Manning, C.D., Raghavan, P., & Schutze, H., 2009. *Scoring, term weighting and the vector space model*. [online] Tersedia di: <nlp.stanford.edu/IR-book/pdf/06vect.pdf> [Diakses 16 Agustus 2017]

Muca, M., Kutrolli, G., & Kutrolli, M., 2015. *A Proposed Algorithm for Determining The Optimal Number of Clusters.* European Scientific Journal, Vol 11, No. 36, pp. 112-120

Poomagal, S. & Hamsapriya, T., 2011. *Optimized K-Means Clustering with Intelligent Initial Centroid Selection for Web Search Using URL and Tag Contents.* Sogndal: ACM.

Rahman, M.A., Islam, M.Z., & Bossomaier, T., 2015. *ModEx and Seed-Detective: Two novel techniques for high quality clustering by using good initial seeds in K-Means.* Journal of King Saud University – Computer and Information Science, pp. 113-128. Tersedia melalui: <www.sciencedirect.com>

Reddy, D. & Jana, P.K., 2012. *Initialization for K-means clustering using Voronoi diagram.* Procedia Technology 4, pp. 395-400. Tersedia melalui: <www.sciencedirect.com>

Repository Universitas Sumatera Utara. *Kajian Teoritis Jurnal Ilmiah.* [online] Tersedia di: <http://repository.usu.ac.id/bitstream/handle/123456789/45666/Chapter%20II.pdf> [Diakses 31 Juli 2017]

Sabthami, J., Thirumoorthy, K., & Muneeswaran, K., 2016. *Multi-View Clustering of Clinical Documents Based on Conditions and Medical Responses of Patients*. [online] Tersedia di: <ieeexplore.ieee.org> [Diakses 14 Agustus 2017]

Sahu, L. & Mohan, B.R., 2014. *An Improved K-means Algorithm Using Modified Cosine Distance Measure for Document Clustering Using Mahout with Hadoop.* [online] Tersedia di: <ieeexplore.ieee.org> [Diakses 14 Agustus 2017]

Shen, S. & Meng, Z., 2012. *Optimization of Initial Centroids for K-Means Algorithm Based on Small World Network.* IIP 2012, IFIP AICT 385, pp. 87-96.

Shoolihah, A., Furqon, M.T., & Widodo, A.W., 2017. *Implementasi Metode Improved K-Means untuk Mengelompokkan Titik Panas Bumi.* Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol. 1, No. 11, hlm. 1270-1276.

Singh, J.N. & Dwivedi, S.K., 2012. *Analysis of Vector Space Model in Information Retrieval.* National Conference on Communication Technologies & its impact on Next Generation Computing, pp. 14-18

Soucy, P. & Mineau, G.W., 2005. *Beyond TFIDF Weighting for Text Categorization in the Vector Space Model.* [online] Tersedia di: <https://ijcai.org/Proceedings/05/Papers/0304.pdf> [Diakses 11 Agustus 2017]

Sutariya, A. & Amin, K., 2013. *An Improvement in K-means Clustering Algorithm.* International Journal of Engineering Research & Technology, Vol. 2, Issue 1

Turney, P.D. & Pantel, P., 2010. *From Frequency to Meaning: Vector Space Models of Semantics.* Journal of Artificial Intelligence Research 37, pp. 141-188

Vijayarani, S., Ilamathi, J., & Nithya, 2011. *Preprocessing Techniques for Text Mining – An Overview*. International Journal of Computer Science & Communication Networks, Vol. 5, No. 1, pp. 7-16

Xiong, C., Hua, Z., Lv, Ke. & Li, X., 2016. *An Improved K-means text clustering algorithm By Optimizing initial cluster centers.* 7th International Conference on Cloud Computing and Big Data, pp. 265-268. Tersedia melalui: <ieeexplore.ieee.org>

Zoubi, M.B.A. & Rawi, M.A., 2008. *An Efficient Approach for Computing Silhouette Coefficient*. Journal of Computer Science, Vol. 4, No. 3, pp. 252-255

LAMPIRAN A VEKTOR JUMLAH FREKUENSI *TERM* PADA DOKUMEN

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
acceptance	0	0	0	0	0	0	1
alat	0	0	0	0	1	0	0
algorithm	0	1	0	0	0	0	0
algoritma	0	0	1	0	0	0	0
analisa	0	0	0	0	0	1	0
analisis	0	0	0	0	0	1	1
and	0	0	0	0	0	0	1
aplikasi	0	0	0	0	0	1	0
arduino	0	0	0	0	1	0	0
baik	0	1	0	0	0	0	0
banding	0	0	0	0	0	1	0
basis	1	0	0	0	0	0	0
cakup	0	0	0	0	0	1	0
canny	0	0	0	1	0	0	0
citra	0	0	0	1	0	0	0
coba	0	1	0	0	0	0	0
coded	0	1	0	0	0	0	0
danau	0	0	0	2	0	0	0
dapat	0	1	0	0	0	0	0
darat	0	0	0	1	0	0	0
dc	0	0	0	0	1	0	0
deteksi	0	0	0	1	0	0	0
dimensi	0	0	0	0	1	0	0
fitur	0	0	0	1	0	0	0
flexible	0	1	0	0	0	0	0
genetic	0	1	0	0	0	0	0
genetika	0	0	1	0	0	0	0
gerak	1	0	0	0	0	0	0
ghost	2	0	0	0	0	0	0
guna	2	0	0	0	0	0	1
hadap	0	0	1	0	0	0	0
hc	0	0	0	0	1	0	0
hidup	0	0	0	1	0	0	0
indonesia	0	0	0	0	0	0	1
interactive	1	0	0	0	0	0	0
investasi	0	0	1	0	0	0	0
investor	0	0	1	0	0	0	0
jadwal	0	1	0	0	0	0	0

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
job	0	1	0	0	0	0	0
kali	0	0	1	0	0	0	0
kendali	1	0	0	0	0	0	0
kinect	1	0	0	0	0	0	0
mahasiswa	0	0	0	0	0	0	1
manusia	0	0	0	1	0	0	0
metode	0	0	0	1	0	0	0
metodologi	0	0	0	0	0	1	0
mixed	2	0	0	0	0	0	0
modal	0	0	1	0	0	0	0
motor	0	0	0	0	1	0	0
of	0	0	0	0	0	0	2
optimal	0	0	1	0	0	0	0
pasar	0	0	1	0	0	0	0
penetrasi	0	0	0	0	0	1	0
penetration	0	0	0	0	0	1	0
pepper	2	0	0	0	0	0	0
peran	0	0	0	1	0	0	0
peta	0	0	0	0	1	0	0
portofolio	0	0	1	0	0	0	0
problem	0	1	0	0	0	0	0
rancang	0	0	0	0	1	0	0
real	0	1	0	0	0	0	0
reality	2	0	0	0	0	0	0
responden	0	0	0	0	0	0	1
ruang	0	0	0	0	1	0	0
rugi	0	0	1	0	0	0	0
rumit	0	1	0	0	0	0	0
saham	0	0	1	0	0	0	0
salah	1	0	0	0	0	0	0
satelit	0	0	0	1	0	0	0
selesai	0	2	0	0	0	0	0
sensor	0	0	0	0	1	0	0
shop	0	1	0	0	0	0	0
siap	0	0	0	0	0	1	0
sistem	0	0	0	0	1	0	0
solusi	0	1	0	0	0	0	0
sr	0	0	0	0	1	0	0
system	2	0	0	0	0	0	0
tahap	0	0	0	0	0	1	0
tangan	1	0	0	0	0	0	0
technology	0	0	0	0	0	0	1

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
teknologi	1	0	0	0	0	0	0
teliti	0	0	0	0	0	0	1
tentu	0	0	1	0	0	0	0
tepi	0	0	0	1	0	0	0
terima	0	0	0	0	0	0	1
tes	0	0	0	0	0	2	0
testing	0	0	0	0	0	1	0
theory	0	0	0	0	0	0	1
tingkat	0	0	2	0	0	0	0
tool	0	0	0	0	0	1	0
uji	0	0	0	0	0	2	0
ultrasonik	0	0	0	0	2	0	0
unified	0	0	0	0	0	0	1
uno	0	0	0	0	1	0	0
untung	0	0	1	0	0	0	0
use	0	0	0	0	0	0	1
utaut	0	0	0	0	0	0	1
web	0	0	0	0	0	1	0
windows	0	0	0	0	0	0	1

LAMPIRAN B HASIL PERHITUNGAN *WF*

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
acceptance	0	0	0	0	0	0	1
alat	0	0	0	0	1	0	0
algorithm	0	1	0	0	0	0	0
algoritma	0	0	1	0	0	0	0
analisa	0	0	0	0	0	1	0
analisis	0	0	0	0	0	1	1
and	0	0	0	0	0	0	1
aplikasi	0	0	0	0	0	1	0
arduino	0	0	0	0	1	0	0
baik	0	1	0	0	0	0	0
banding	0	0	0	0	0	1	0
basis	1	0	0	0	0	0	0
cakup	0	0	0	0	0	1	0
canny	0	0	0	1	0	0	0
citra	0	0	0	1	0	0	0
coba	0	1	0	0	0	0	0
coded	0	1	0	0	0	0	0
danau	0	0	0	1,30103	0	0	0
dapat	0	1	0	0	0	0	0
darat	0	0	0	1	0	0	0
dc	0	0	0	0	1	0	0
deteksi	0	0	0	1	0	0	0
dimensi	0	0	0	0	1	0	0
fitur	0	0	0	1	0	0	0
flexible	0	1	0	0	0	0	0
genetic	0	1	0	0	0	0	0
genetika	0	0	1	0	0	0	0
gerak	1	0	0	0	0	0	0
ghost	1,30103	0	0	0	0	0	0
guna	1,30103	0	0	0	0	0	1
hadap	0	0	1	0	0	0	0
hc	0	0	0	0	1	0	0
hidup	0	0	0	1	0	0	0
indonesia	0	0	0	0	0	0	1
interactive	1	0	0	0	0	0	0
investasi	0	0	1	0	0	0	0
investor	0	0	1	0	0	0	0
jadwal	0	1	0	0	0	0	0
job	0	1	0	0	0	0	0

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
kali	0	0	1	0	0	0	0
kendali	1	0	0	0	0	0	0
kinect	1	0	0	0	0	0	0
mahasiswa	0	0	0	0	0	0	1
manusia	0	0	0	1	0	0	0
metode	0	0	0	1	0	0	0
metodologi	0	0	0	0	0	1	0
mixed	1,30103	0	0	0	0	0	0
modal	0	0	1	0	0	0	0
motor	0	0	0	0	1	0	0
of	0	0	0	0	0	0	1,30103
optimal	0	0	1	0	0	0	0
pasar	0	0	1	0	0	0	0
penetrasi	0	0	0	0	0	1	0
penetration	0	0	0	0	0	1	0
pepper	1,30103	0	0	0	0	0	0
peran	0	0	0	1	0	0	0
peta	0	0	0	0	1	0	0
portofolio	0	0	1	0	0	0	0
problem	0	1	0	0	0	0	0
rancang	0	0	0	0	1	0	0
real	0	1	0	0	0	0	0
reality	1,30103	0	0	0	0	0	0
responden	0	0	0	0	0	0	1
ruang	0	0	0	0	1	0	0
rugi	0	0	1	0	0	0	0
rumit	0	1	0	0	0	0	0
saham	0	0	1	0	0	0	0
salah	1	0	0	0	0	0	0
satelit	0	0	0	1	0	0	0
selesai	0	1,30103	0	0	0	0	0
sensor	0	0	0	0	1	0	0
shop	0	1	0	0	0	0	0
siap	0	0	0	0	0	1	0
sistem	0	0	0	0	1	0	0
solusi	0	1	0	0	0	0	0
sr	0	0	0	0	1	0	0
system	1,30103	0	0	0	0	0	0
tahap	0	0	0	0	0	1	0
tangan	1	0	0	0	0	0	0
technology	0	0	0	0	0	0	1
teknologi	1	0	0	0	0	0	0

Kata	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
teliti	0	0	0	0	0	0	1
tentu	0	0	1	0	0	0	0
tepi	0	0	0	1	0	0	0
terima	0	0	0	0	0	0	1
tes	0	0	0	0	0	1,30103	0
testing	0	0	0	0	0	1	0
theory	0	0	0	0	0	0	1
tingkat	0	0	1,30103	0	0	0	0
tool	0	0	0	0	0	1	0
uji	0	0	0	0	0	1,30103	0
ultrasonik	0	0	0	0	1,30103	0	0
unified	0	0	0	0	0	0	1
uno	0	0	0	0	1	0	0
untung	0	0	1	0	0	0	0
use	0	0	0	0	0	0	1
utaut	0	0	0	0	0	0	1
web	0	0	0	0	0	1	0
windows	0	0	0	0	0	0	1

LAMPIRAN C HASIL PERHITUNGAN *IDF*

Kata	df_t	idf_t
acceptance	1	0,845098
alat	1	0,845098
algorithm	1	0,845098
algoritma	1	0,845098
analisa	1	0,845098
analisis	2	0,544068
and	1	0,845098
aplikasi	1	0,845098
arduino	1	0,845098
baik	1	0,845098
banding	1	0,845098
basis	1	0,845098
cakup	1	0,845098
canny	1	0,845098
citra	1	0,845098
coba	1	0,845098
coded	1	0,845098
danau	1	0,845098
dapat	1	0,845098
darat	1	0,845098
dc	1	0,845098
deteksi	1	0,845098
dimensi	1	0,845098
fitur	1	0,845098
flexible	1	0,845098
genetic	1	0,845098
genetika	1	0,845098
gerak	1	0,845098
ghost	1	0,845098
guna	2	0,544068
hadap	1	0,845098
hc	1	0,845098
hidup	1	0,845098
indonesia	1	0,845098
interactive	1	0,845098
investasi	1	0,845098
investor	1	0,845098
jadwal	1	0,845098
job	1	0,845098

Kata	df_t	idf_t
kali	1	0,845098
kendali	1	0,845098
kinect	1	0,845098
mahasiswa	1	0,845098
manusia	1	0,845098
metode	1	0,845098
metodologi	1	0,845098
mixed	1	0,845098
modal	1	0,845098
motor	1	0,845098
of	1	0,845098
optimal	1	0,845098
pasar	1	0,845098
penetrasi	1	0,845098
penetration	1	0,845098
pepper	1	0,845098
peran	1	0,845098
peta	1	0,845098
portofolio	1	0,845098
problem	1	0,845098
rancang	1	0,845098
real	1	0,845098
reality	1	0,845098
responden	1	0,845098
ruang	1	0,845098
rugi	1	0,845098
rumit	1	0,845098
saham	1	0,845098
salah	1	0,845098
satelit	1	0,845098
selesai	1	0,845098
sensor	1	0,845098
shop	1	0,845098
siap	1	0,845098
sistem	1	0,845098
solusi	1	0,845098
sr	1	0,845098
system	1	0,845098
tahap	1	0,845098

Kata	df_t	idf_t
tangan	1	0,845098
technology	1	0,845098
teknologi	1	0,845098
teliti	1	0,845098
tentu	1	0,845098
tepi	1	0,845098
terima	1	0,845098
tes	1	0,845098
testing	1	0,845098
theory	1	0,845098
tingkat	1	0,845098

Kata	df_t	idf_t
tool	1	0,845098
uji	1	0,845098
ultrasonik	1	0,845098
unified	1	0,845098
uno	1	0,845098
untung	1	0,845098
use	1	0,845098
utaut	1	0,845098
web	1	0,845098
windows	1	0,845098

LAMPIRAN D HASIL PERHITUNGAN *WF.IDF*

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
acceptance	0	0	0	0	0	0	0,845098
alat	0	0	0	0	0,845098	0	0
algorithm	0	0,845098	0	0	0	0	0
algoritma	0	0	0,845098	0	0	0	0
analisa	0	0	0	0	0	0,845098	0
analisis	0	0	0	0	0	0,544068	0,544068
and	0	0	0	0	0	0	0,845098
aplikasi	0	0	0	0	0	0,845098	0
arduino	0	0	0	0	0,845098	0	0
baik	0	0,845098	0	0	0	0	0
banding	0	0	0	0	0	0,845098	0
basis	0,845098	0	0	0	0	0	0
cakup	0	0	0	0	0	0,845098	0
canny	0	0	0	0,845098	0	0	0
citra	0	0	0	0,845098	0	0	0
coba	0	0,845098	0	0	0	0	0
coded	0	0,845098	0	0	0	0	0
danau	0	0	0	1,099498	0	0	0
dapat	0	0,845098	0	0	0	0	0
darat	0	0	0	0,845098	0	0	0
dc	0	0	0	0	0,845098	0	0
deteksi	0	0	0	0,845098	0	0	0
dimensi	0	0	0	0	0,845098	0	0
fitur	0	0	0	0,845098	0	0	0
flexible	0	0,845098	0	0	0	0	0
genetic	0	0,845098	0	0	0	0	0
genetika	0	0	0,845098	0	0	0	0
gerak	0,845098	0	0	0	0	0	0
ghost	1,099498	0	0	0	0	0	0
guna	0,707849	0	0	0	0	0	0,544068
hadap	0	0	0,845098	0	0	0	0
hc	0	0	0	0	0,845098	0	0
hidup	0	0	0	0,845098	0	0	0
indonesia	0	0	0	0	0	0	0,845098
interactive	0,845098	0	0	0	0	0	0
investasi	0	0	0,845098	0	0	0	0
investor	0	0	0,845098	0	0	0	0
jadwal	0	0,845098	0	0	0	0	0
job	0	0,845098	0	0	0	0	0

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
kali	0	0	0,845098	0	0	0	0
kendali	0,845098	0	0	0	0	0	0
kinect	0,845098	0	0	0	0	0	0
mahasiswa	0	0	0	0	0	0	0,845098
manusia	0	0	0	0,845098	0	0	0
metode	0	0	0	0,845098	0	0	0
metodologi	0	0	0	0	0	0,845098	0
mixed	1,099498	0	0	0	0	0	0
modal	0	0	0,845098	0	0	0	0
motor	0	0	0	0	0,845098	0	0
of	0	0	0	0	0	0	1,099498
optimal	0	0	0,845098	0	0	0	0
pasar	0	0	0,845098	0	0	0	0
penetrasi	0	0	0	0	0	0,845098	0
penetration	0	0	0	0	0	0,845098	0
pepper	1,099498	0	0	0	0	0	0
peran	0	0	0	0,845098	0	0	0
peta	0	0	0	0	0,845098	0	0
portofolio	0	0	0,845098	0	0	0	0
problem	0	0,845098	0	0	0	0	0
rancang	0	0	0	0	0,845098	0	0
real	0	0,845098	0	0	0	0	0
reality	1,099498	0	0	0	0	0	0
responden	0	0	0	0	0	0	0,845098
ruang	0	0	0	0	0,845098	0	0
rugi	0	0	0,845098	0	0	0	0
rumit	0	0,845098	0	0	0	0	0
saham	0	0	0,845098	0	0	0	0
salah	0,845098	0	0	0	0	0	0
satelit	0	0	0	0,845098	0	0	0
selesai	0	1,099498	0	0	0	0	0
sensor	0	0	0	0	0,845098	0	0
shop	0	0,845098	0	0	0	0	0
siap	0	0	0	0	0	0,845098	0
sistem	0	0	0	0	0,845098	0	0
solusi	0	0,845098	0	0	0	0	0
sr	0	0	0	0	0,845098	0	0
system	1,099498	0	0	0	0	0	0
tahap	0	0	0	0	0	0,845098	0
tangan	0,845098	0	0	0	0	0	0
technology	0	0	0	0	0	0	0,845098
teknologi	0,845098	0	0	0	0	0	0

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
teliti	0	0	0	0	0	0	0,845098
tentu	0	0	0,845098	0	0	0	0
tepi	0	0	0	0,845098	0	0	0
terima	0	0	0	0	0	0	0,845098
tes	0	0	0	0	0	1,099498	0
testing	0	0	0	0	0	0,845098	0
theory	0	0	0	0	0	0	0,845098
tingkat	0	0	1,099498	0	0	0	0
tool	0	0	0	0	0	0,845098	0
uji	0	0	0	0	0	1,099498	0
ultrasonik	0	0	0	0	1,099498	0	0
unified	0	0	0	0	0	0	0,845098
uno	0	0	0	0	0,845098	0	0
untung	0	0	0,845098	0	0	0	0
use	0	0	0	0	0	0	0,845098
utaut	0	0	0	0	0	0	0,845098
web	0	0	0	0	0	0,845098	0
windows	0	0	0	0	0	0	0,845098

LAMPIRAN E HASIL PERHITUNGAN NORMALISASI *WF.IDF*

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
acceptance	0	0	0	0	0	0	0,253823
alat	0	0	0	0	0,260885	0	0
algorithm	0	0,252436	0	0	0	0	0
algoritma	0	0	0,252436	0	0	0	0
analisa	0	0	0	0	0	0,251579	0
analisis	0	0	0	0	0	0,161965	0,16341
and	0	0	0	0	0	0	0,253823
aplikasi	0	0	0	0	0	0,251579	0
arduino	0	0	0	0	0,260885	0	0
baik	0	0,252436	0	0	0	0	0
banding	0	0	0	0	0	0,251579	0
basis	0,241367	0	0	0	0	0	0
cakup	0	0	0	0	0	0,251579	0
canny	0	0	0	0,280688	0	0	0
citra	0	0	0	0,280688	0	0	0
coba	0	0,252436	0	0	0	0	0
coded	0	0,252436	0	0	0	0	0
danau	0	0	0	0,365183	0	0	0
dapat	0	0,252436	0	0	0	0	0
darat	0	0	0	0,280688	0	0	0
dc	0	0	0	0	0,260885	0	0
deteksi	0	0	0	0,280688	0	0	0
dimensi	0	0	0	0	0,260885	0	0
fitur	0	0	0	0,280688	0	0	0
flexible	0	0,252436	0	0	0	0	0
genetic	0	0,252436	0	0	0	0	0
genetika	0	0	0,252436	0	0	0	0
gerak	0,241367	0	0	0	0	0	0
ghost	0,314026	0	0	0	0	0	0
guna	0,202168	0	0	0	0	0	0,16341
hadap	0	0	0,252436	0	0	0	0
hc	0	0	0	0	0,260885	0	0
hidup	0	0	0	0,280688	0	0	0
indonesia	0	0	0	0	0	0	0,253823
interactive	0,241367	0	0	0	0	0	0
investasi	0	0	0,252436	0	0	0	0
investor	0	0	0,252436	0	0	0	0
jadwal	0	0,252436	0	0	0	0	0
job	0	0,252436	0	0	0	0	0

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
kali	0	0	0,252436	0	0	0	0
kendali	0,241367	0	0	0	0	0	0
kinect	0,241367	0	0	0	0	0	0
mahasiswa	0	0	0	0	0	0	0,253823
manusia	0	0	0	0,280688	0	0	0
metode	0	0	0	0,280688	0	0	0
metodologi	0	0	0	0	0	0,251579	0
mixed	0,314026	0	0	0	0	0	0
modal	0	0	0,252436	0	0	0	0
motor	0	0	0	0	0,260885	0	0
of	0	0	0	0	0	0	0,330232
optimal	0	0	0,252436	0	0	0	0
pasar	0	0	0,252436	0	0	0	0
penetrasi	0	0	0	0	0	0,251579	0
penetration	0	0	0	0	0	0,251579	0
pepper	0,314026	0	0	0	0	0	0
peran	0	0	0	0,280688	0	0	0
peta	0	0	0	0	0,260885	0	0
portofolio	0	0	0,252436	0	0	0	0
problem	0	0,252436	0	0	0	0	0
rancang	0	0	0	0	0,260885	0	0
real	0	0,252436	0	0	0	0	0
reality	0,314026	0	0	0	0	0	0
responden	0	0	0	0	0	0	0,253823
ruang	0	0	0	0	0,260885	0	0
rugi	0	0	0,252436	0	0	0	0
rumit	0	0,252436	0	0	0	0	0
saham	0	0	0,252436	0	0	0	0
salah	0,241367	0	0	0	0	0	0
satelit	0	0	0	0,280688	0	0	0
selesai	0	0,328427	0	0	0	0	0
sensor	0	0	0	0	0,260885	0	0
shop	0	0,252436	0	0	0	0	0
siap	0	0	0	0	0	0,251579	0
sistem	0	0	0	0	0,260885	0	0
solusi	0	0,252436	0	0	0	0	0
sr	0	0	0	0	0,260885	0	0
system	0,314026	0	0	0	0	0	0
tahap	0	0	0	0	0	0,251579	0
tangan	0,241367	0	0	0	0	0	0
technology	0	0	0	0	0	0	0,253823
teknologi	0,241367	0	0	0	0	0	0

wf.idf	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7
teliti	0	0	0	0	0	0	0,253823
tentu	0	0	0,252436	0	0	0	0
tepi	0	0	0	0,280688	0	0	0
terima	0	0	0	0	0	0	0,253823
tes	0	0	0	0	0	0,327311	0
testing	0	0	0	0	0	0,251579	0
theory	0	0	0	0	0	0	0,253823
tingkat	0	0	0,328427	0	0	0	0
tool	0	0	0	0	0	0,251579	0
uji	0	0	0	0	0	0,327311	0
ultrasonik	0	0	0	0	0,33942	0	0
unified	0	0	0	0	0	0	0,253823
uno	0	0	0	0	0,260885	0	0
untung	0	0	0,252436	0	0	0	0
use	0	0	0	0	0	0	0,253823
utaut	0	0	0	0	0	0	0,253823
web	0	0	0	0	0	0,251579	0
windows	0	0	0	0	0	0	0,253823

LAMPIRAN F *CENTROID* AWAL KLASTER 1 DAN KLASTER 2

Kata	Doc7	Doc6	Kata	Doc7	Doc6
acceptance	0,253823	0	kali	0	0
alat	0	0	kendali	0	0
algorithm	0	0	kinect	0	0
algoritma	0	0	mahasiswa	0,253823	0
analisa	0	0,251579	manusia	0	0
analisis	0,16341	0,161965	metode	0	0
and	0,253823	0	metodologi	0	0,251579
aplikasi	0	0,251579	mixed	0	0
arduino	0	0	modal	0	0
baik	0	0	motor	0	0
banding	0	0,251579	of	0,330232	0
basis	0	0	optimal	0	0
cakup	0	0,251579	pasar	0	0
canny	0	0	penetrasi	0	0,251579
citra	0	0	penetration	0	0,251579
coba	0	0	pepper	0	0
coded	0	0	peran	0	0
danau	0	0	peta	0	0
dapat	0	0	portofolio	0	0
darat	0	0	problem	0	0
dc	0	0	rancang	0	0
deteksi	0	0	real	0	0
dimensi	0	0	reality	0	0
fitur	0	0	responden	0,253823	0
flexible	0	0	ruang	0	0
genetic	0	0	rugi	0	0
genetika	0	0	rumit	0	0
gerak	0	0	saham	0	0
ghost	0	0	salah	0	0
guna	0,16341	0	satelit	0	0
hadap	0	0	selesai	0	0
hc	0	0	sensor	0	0
hidup	0	0	shop	0	0
indonesia	0,253823	0	siap	0	0,251579
interactive	0	0	sistem	0	0
investasi	0	0	solusi	0	0
investor	0	0	sr	0	0
jadwal	0	0	system	0	0
job	0	0	tahap	0	0,251579

Kata	Doc7	Doc6
tangan	0	0
technology	0,253823	0
teknologi	0	0
teliti	0,253823	0
tentu	0	0
tepi	0	0
terima	0,253823	0
tes	0	0,327311
testing	0	0,251579
theory	0,253823	0
tingkat	0	0

Kata	Doc7	Doc6
tool	0	0,251579
uji	0	0,327311
ultrasonik	0	0
unified	0,253823	0
uno	0	0
untung	0	0
use	0,253823	0
utaut	0,253823	0
web	0	0,251579
windows	0,253823	0

LAMPIRAN G *CENTROID* BARU KLASTER 1 DAN KLASTER 2 PADA ITERASI PERTAMA DAN KEDUA

Kata	Klaster 1	Klaster 2
acceptance	0,042304	0
alat	0,043481	0
algorithm	0,042073	0
algoritma	0,042073	0
analisa	0	0,251579
analisis	0,027235	0,161965
and	0,042304	0
aplikasi	0	0,251579
arduino	0,043481	0
baik	0,042073	0
banding	0	0,251579
basis	0,040228	0
cakup	0	0,251579
canny	0,046781	0
citra	0,046781	0
coba	0,042073	0
coded	0,042073	0
danau	0,060864	0
dapat	0,042073	0
darat	0,046781	0
dc	0,043481	0
deteksi	0,046781	0
dimensi	0,043481	0
fitur	0,046781	0
flexible	0,042073	0
genetic	0,042073	0
genetika	0,042073	0
gerak	0,040228	0
ghost	0,052338	0
guna	0,06093	0
hadap	0,042073	0
hc	0,043481	0
hidup	0,046781	0
indonesia	0,042304	0
interactive	0,040228	0
investasi	0,042073	0
investor	0,042073	0
jadwal	0,042073	0

Kata	Klaster 1	Klaster 2
job	0,042073	0
kali	0,042073	0
kendali	0,040228	0
kinect	0,040228	0
mahasiswa	0,042304	0
manusia	0,046781	0
metode	0,046781	0
metodologi	0	0,251579
mixed	0,052338	0
modal	0,042073	0
motor	0,043481	0
of	0,055039	0
optimal	0,042073	0
pasar	0,042073	0
penetrasi	0	0,251579
penetration	0	0,251579
pepper	0,052338	0
peran	0,046781	0
peta	0,043481	0
portofolio	0,042073	0
problem	0,042073	0
rancang	0,043481	0
real	0,042073	0
reality	0,052338	0
responden	0,042304	0
ruang	0,043481	0
rugi	0,042073	0
rumit	0,042073	0
saham	0,042073	0
salah	0,040228	0
satelit	0,046781	0
selesai	0,054738	0
sensor	0,043481	0
shop	0,042073	0
siap	0	0,251579
sistem	0,043481	0
solusi	0,042073	0
sr	0,043481	0

Kata	Klaster 1	Klaster 2
system	0,052338	0
tahap	0	0,251579
tangan	0,040228	0
technology	0,042304	0
teknologi	0,040228	0
teliti	0,042304	0
tentu	0,042073	0
tepi	0,046781	0
terima	0,042304	0
tes	0	0,327311
testing	0	0,251579
theory	0,042304	0

Kata	Klaster 1	Klaster 2
tingkat	0,054738	0
tool	0	0,251579
uji	0	0,327311
ultrasonik	0,05657	0
unified	0,042304	0
uno	0,043481	0
untung	0,042073	0
use	0,042304	0
utaut	0,042304	0
web	0	0,251579
windows	0,042304	0