KLASIFIKASI DOKUMEN ABSTRAK SKRIPSI BERDASARKAN FOKUS PENELITIAN DI BIDANG KOMPUTASI CERDAS MENGGUNAKAN BM25 DAN *K-NEAREST NEIGHBOR*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Deri Hendra Binawan
NIM: 155150200111036

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KLASIFIKASI DOKUMEN ABSTRAK SKRIPSI BERDASARKAN FOKUS PENELITIAN DI BIDANG KOMPUTASI CERDAS MENGGUNAKAN BM25 DAN *K-NEAREST NEIGHBOR*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Deri Hendra Binawan
NIM: 155150200111036

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Januari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I

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

Dosen Pembimbing II

Putra Pandu Adikara, S.Kom., M.Kom.
NIP: 19850725 200812 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

Apabila ternyata di dalam 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, 2 Januari 2019

Deri Hendra Binawan

NIM: 155150200111036

PRAKATA

Puji syukur penulis panjatkan atas kehadirat Tuhan Yang Maha Esa, Yang telah melimpahkan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penelitian yang berjudul "Klasifikasi Dokumen Abstrak Skripsi Berdasarkan Fokus Penelitian di Bidang Komputasi Cerdas Menggunakan BM25 dan *K-Nearest Neighbor*".

Penelitian ini merupakan syarat untuk memenuhi sebagian persyaratan kurikulum pada Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya. Pada kesempatan ini penulis ingin menyampaikan rasa terima kasih kepada pihak-pihak yang telah membantu selama penyusunan laporan penelitian ini, antara lain:

1. Ibu Indriati, S.T., M.Kom selaku Dosen Pembimbing I yang dengan tulus ikhlas membimbing dan membantu penulis dalam menyelesaikan penelitian ini.
2. Bapak Putra Pandu Adikara, S.Kom., M.Kom selaku Dosen Pembimbing II yang dengan sabar membimbing dan mengarahkan penulis dalam penyusunan laporan penelitian ini.
3. Bapak Agus Wahyu Widodo, S.T., M.Sc selaku Ketua Program Studi Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Achmad Basuki, S.T., MMG., Ph.D., selaku Dosen Pembimbing Akademik yang telah membantu penulis selama proses perkuliahan.
6. Keluarga penulis, terutama bapak, ibu, kakak, dan ketiga adik penulis yang dengan seluruh kemampuan dan keikhlasannya membantu penulisan dalam segala hal, serta selalu memberikan doa dan motivasi kepada penulis dalam menyelesaikan skripsi ini.
7. Khalisma Frinta, Indriya Dewi Onantya, Dwi Suci Ariska Yanti, Gantara Hadi, dan Fifi selaku pemberi semangat dan sebagai tempat berbagi keluh kesah penulis selama proses perkuliahan dan pengerjaan skripsi.

Penulis menyadari penelitian ini masih banyak kekurangan dan jauh dari kata sempurna, untuk itu penulis mengharpkan kritik dan saran yang bersifat membangun. Semoga penelitian ini dapat bermanfaat bagi pihak yang membutuhkan

Malang, 2 Januari 2019

Penulis

hendraderi48@gmail.com

ABSTRAK

Deri Hendra Binawan, Klasifikasi Dokumen Abstrak Skripsi Berdasarkan Fokus Penelitian di Bidang Komputasi Cerdas Menggunakan BM25 dan *K-Nearest Neighbor*

Pembimbing: Indriati, S.T., M.Kom., dan Putra Pandu Adikara, S.Kom., M.Kom.

Salah satu proses yang dapat dilakukan di *text mining* adalah pengelompokan dokumen teks. Permasalahan yang berkaitan dengan pengelompokan dokumen teks ditemukan di perguruan tinggi khususnya di ruang baca Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB). Permasalahannya adalah belum adanya proses pengelompokan dokumen skripsi secara otomatis. Pengelompokan dokumen skripsi di ruang baca FILKOM UB masih belum tertata sesuai dengan fokus penelitian yang ada. Pengelompokan tersebut dapat diselesaikan menggunakan metode BM25 dan *K-Nearest Neighbor.* Proses yang dilakukan adalah melakukan *pre-processing* dokumen, perhitungan *score* BM25 tiap dokumen, dan proses klasifikasi menggunakan algoritme *K-Nearest Neighbor.* Proses pengujian pada penelitian ini menggunakan k-fold sebanyak 10. Setiap pengujian menggunakan data testing sebanyak 31 dokumen dan data training sebanyak 300 dokumen. Hasil rata-rata yang diperoleh pada tiap pengujian menghasilkan hasil terbaik pada nilai *k*=11 dengan nilai *f-measure* sebesar 0,9092, *recall* sebesar 0,9087, dan *precision* sebesar 0,9265, sedangkan untuk nilai *k* yang semakin besar, hasil *f-measure, recall*, dan *precision* mengalami penurunan. Oleh karena itu, metode BM25 dan *K-Nearest Neighbor* dapat diterapkan pada proses klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang Komputasi Cerdas dan untuk hasil yang klasifikasi pada penelitian ini berada pada nilai *k* yang kecil, untuk nilai *k* yang semakin besar proses klasifikasi berjalan kurang maksimal.

Kata kunci: klasifikasi, *Text Mining*, BM25, *K-Nearest Neighbor*, dokumen abstrak skripsi

ABSTRACT

Deri Hendra Binawan, Classification of the Abstract Thesis Document Based on Research Focus in the Field of Intelligent Computing Using BM25 and K-Nearest Neighbor

Supervisors: Indriati, S.T., M.Kom., and Putra Pandu Adikara, S.Kom., M.Kom.

One of the process that can be implemented in text mining is categorizing text documents. Problems that related the categorizing text documents are found in universities, especially in the reading room of the Faculty of Computer Science, Universitas Brawijaya (FILKOM UB). There is no process for categorizing thesis documents automatically is one of the problem. The thesis documents categorization in FILKOM UB's reading room is still not organized according to the focus of the existing research. The categorization is completed using the BM25 and K-Nearest Neighbor methods. The process was done is pre-processing text document, calculate the BM25 score of each document, then classify them using the K-Nearest Neighbor algorithm. The testing process in this research uses 10 k-fold. Each test used 31 testing documents and 300 training documents. The average results obtained in each test produced the best results at the value of k=11 with a f-measure value is 0.9092, recall is 0.9087, and precision is 0.9265. The greater the value of k cause the classification process runs less optimally because it produces a smaller f-measure value.

Keywords: classification, Text Mining, BM25, K-Nearest Neighbor, thesis abstract document

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

Bagian penelitian ini terdiri dari beberapa komponen yang tersusun secara sistematis. Pendahuluan merupakan bab yang pertama dengan menjelaskan mengenai latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, dan batasan masalah.

1.1 Latar Belakang

Perkembangan data berupa teks telah mencapai jumlah yang begitu besar. Hal ini dikarenakan dunia teknologi informasi yang terdiri dari data teks berkembang sangat pesat. Di zaman sekarang, berbagai macam media *online* memiliki informasi yang penting, sehingga sangat potensial untuk diambil informasinya. Akan tetapi data berupa teks memiliki kelemahan. Salah satu kelemahan tersebut adalah data bersifat tidak terstruktur sehingga sangat banyak memuat *noise*. Oleh karena itu, peran *text mining* sangat dibutuhkan. Dengan menerapkan teknik *text mining,* maka akan didapatkan pola-pola data dan ekstraksi informasi dari pengetahuan yang potensial dari data berupa teks (Feldman dan Sanger, 2006).

Fakultas Ilmu Komputer Universitas Brawijaya atau yang biasa disebut FILKOM UB merupakan salah satu fakultas di Universitas Brawijaya yang didirikan dan diresmikan pada Januari 2015. FILKOM UB saat ini memiliki enam program studi, salah satunya adalah program studi S1 Teknik Informatika. Program studi S1 Teknik Informatika adalah program studi yang memiliki tujuan untuk menyajikan pemanfaatan teknologi informasi sesuai dengan prinsip keilmuan dan kerekayasaan. Di dalam program studi S1 Teknik Informatika dibagi menjadi empat peminatan yaitu Rekayasa Perangkat Lunak (RPL), Komputasi Berbasis Jaringan (KBJ), Komputasi Cerdas (KC), dan Multimedia, Game, dan Mobile (MGM) (FILKOM, 2017b). Peminatan Komputasi Cerdas (KC) adalah peminatan yang menekankan pada kemampuan lulusan dalam menerapkan, memanipulasi, dan menganalisis data berupa teks, numerik, dan citra pada berbagai bidang aplikasi sesuai dengan metode sistem cerdas yang digunakan.

Salah satu proses yang dapat diterapkan di *text mining* adalah pengelompokan dokumen teks. Permasalahan yang berkaitan dengan pengelompokan dokumen teks ditemukan di perguruan tinggi khususnya di ruang baca Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB). Permasalahannya adalah belum adanya proses pengelompokan dokumen skripsi secara otomatis. Pengelompokan dokumen skripsi di ruang baca FILKOM UB masih belum tertata sesuai dengan fokus penelitian yang ada. Pengelompokan dokumen skripsi tersebut masih berdasarkan nomor urut masuk dokumen. Oleh karena itu, perlu adanya pengelompokan dokumen skripsi sesuai dengan fokus penelitian yang ada di FILKOM UB khususnya peminatan Komputasi Cerdas. Hal tersebut diharapkan dapat membantu mahasiswa untuk mencari dokumen referensi berupa dokumen skripsi yang relevan dengan penelitian yang mereka teliti. Oleh karena itu, dalam penelitian ini akan dilakukan proses pengelompokan dokumen abstrak skripsi di bidang Komputasi Cerdas. Dalam hal ini, dokumen yang digunakan berupa

dokumen abstrak skripsi dari mahasiswa progam studi S1 Teknik Informatika khususnya peminatan Komputasi Cerdas. Dokumen tersebut akan diklasifikasikan berdasarkan fokus penelitian di bidang Komputasi Cerdas.

Klasifikasi teks adalah proses untuk menentukan suatu kelas tertentu dari sebuah dokumen teks. Algoritme yang sering digunakan dalam klasifikasi teks adalah *K-Nearest Neighbor* (KNN). KNN merupakan salah satu dari banyak algoritme klasifikasi teks yang dapat memberikan kinerja dan hasil yang akurat (Suharno, Fauzi dan Perdana, 2017). Penelitian yang dilakukan oleh (Bagaskoro, Fauzi dan Adikara, 2018) menunjukkan bahwa *K-Nearest Neighbor* memiliki kinerja yang baik dengan hasil akurasi yang diperoleh sebesar 90% dengan nilai *k* sebesar 5. Penelitian lainnya yang menggunakan algoritme *K-Nearest Neighbor* adalah (Nurjanah, Perdana dan Fauzi, 2017) yang miliki hasil akurasi sebesar 82,50%. Akan tetapi algoritme *K-Nearest Neighbor* juga memiliki kelemahan. Penelitian yang dilakukan oleh (Claudy, Perdana dan Fauzi, 2018) menunjukkan bahwa kinerja algoritme *K-Nearest Neighbor* tergantung pada jumlah data *training* yang digunakan. Pada penelitian (Claudy, Perdana dan Fauzi, 2018) menggunakan sebanyak 80 data *tranning* dengan hasil akurasi yang diperoleh sebesar 66%. Dengan melihat hasil penelitian sebelumnya dapat disimpulkan bahwa penggunaan algoritme *K-Nearest Neighbor* memang baik dan cocok digunakan pada klasifikasi teks dengan jumlah data *training* yang sesuai.

Di dalam *text mining* terdapat proses pembobotan dan pemeringkatan dokumen. Salah satu proses pemeringkatan dokumen dapat menggunakan BM25. Metode BM25 sendiri adalah metode pemeringkatan yang digunakan untuk mengurutkan hasil kecocokan terhadap dokumen-dokumen, berdasarakan kata kunci yang dicari. Di dalam metode BM25 terdapat proses pembobotan yang memiliki kinerja lebih baik daripada pembobotan TF-IDF (Yang dkk., 2012). Penelitian yang dilakukan oleh (Yang dkk., 2012) menunjukan bahwa kinerja *term weighting* BM25 lebih baik daripada pembobotan TF-IDF yang diterapkan pada deteksi duplikasi laporan *bugs.* Pada penelitian tersebut menghasilkan akurasi sebesar 90%. Penelitian selanjutnya dilakukan oleh (Aramaki dkk., 2006) menunjukkan bahwa metode BM25 dan KNN pada klasifikasi status merokok seorang pasien memiliki hasil akurasi terbaik yaitu 88,97% dengan mengunakan lima kelas dan nilai *k* sebesar 10.

Berdasarkan permasalahan di atas, fokus penelitian adalah mengklasifikasikan dokumen abstrak skripsi berdasarkan penelitian di bidang Komputasi Cerdas menggunakan BM25 dan *K-Nearest Neighbor* (KNN). BM25 digunakan sebagai pembobotan dan pemeringkatan dokumen, sedangkan KNN digunakan sebagai klasifikasi. Diharapkan pada penelitian ini dapat mengetahui kinerja sistem klasifikasi dokumen teks dengan menggunakan BM25 dan KNN dan dapat digunakan sebagai acuan untuk membangun model klasifikasi dokumen skripsi pada Fakultas Ilmu Komputer Universitas Brawijaya.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dijelaskan, maka didapatkan rumusan masalahnya sebagi berikut.

1. Bagaimanakah rancangan algoritme yang menggunakan metode BM25 dan *K-Nearest Neighbor* pada klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang Komputasi Cerdas?
2. Bagaimana hasil akurasi *Precision, Recall*, dan *F-Measure* yang diperoleh pada klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang Komputasi Cerdas mengunakan BM25 dan *K-Nearest Neighbor*?
3. Bagaimana pengaruh nilai *k* pada klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang Komputasi Cerdas mengunakan BM25 dan *K-Nearest Neighbor*?

1.3 Tujuan

Berdasarkan rumusan masalah yang sudah dijelaskan, maka didapatkan tujuan penelitian ini sebagai berikut.

1. Merancang algoritme untuk klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang Komputasi Cerdas menggunakan BM25 dan *K-Nearest Neighbor*.
2. Mengetahui hasil akurasi *Precision, Recall*, dan *F-Measure* yang diperoleh pada klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang Komputasi Cerdas mengunakan BM25 dan *K-Nearest Neighbor*.
3. Mengetahui pengaruh nilai *k* pada klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang Komputasi Cerdas mengunakan BM25 dan *K-Nearest Neighbor*.

1.4 Manfaat

Penelitian ini diharapkan dapat memberikan manfaat. Manfaat penelitian yang diharapkan sebagai berikut.

1. Membantu pengguna untuk mengelompokkan dokumen skripsi secara otomatis.
2. Memberikan acuan untuk membangun model klasifikasi dokumen skripsi pada Fakultas Ilmu Komputer Universitas Brawijaya.
3. Membatu pengguna mencari dokumen skripsi yang relevan dengan bidang riset yang diteliti khususnya peminatan Komputasi Cerdas.

1.5 Batasan Masalah

Untuk menghindari terjadinya perluasan masalah, maka perlu dilakukan suatu batasan masalah agar tidak terjadi hal tersebut. Batasan masalah penelitian sebagai berikut.

1. Dokumen yang digunakan adalah dokumen abstrak skripsi Fakultas Ilmu Komputer Universitas Brawijaya khususnya peminatan Komputasi Cerdas yang diambil dari ruang baca FILKOM dan jurnal J-PTIIK.
2. Dokumen yang digunakan berjumlah 331 dokumen. Dokumen tersebut dibagi menjadi dua yaitu, data *training* adalah 300 dokumen dan data *testing* 31 dokumen.
3. Pengklasifikasian dibagi menjadi lima kelas yaitu Text Mining (TM), Data Mining (DM), Sistem Pakar (SP), Pengolahan Citra Digital (PCD), dan Sistem Pendukung Keputusan (SPK).
4. Melibatkan pakar di bidang Komputasi Cerdas. Pakar tersebut berasal dari dosen Fakultas Ilmu Komputer Universitas Brawijaya.

1.6 Sistematika Pembahasan

Sistematika penulisan ini bertujuan untuk memberikan suatu gambaran tentang penyusunan skripsi secara garis besar. Gambaran tersebut meliputi beberapa bab sebagai berikut.

BAB I PENDAHULUAN

Bab ini menguraikan latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, serta sistematika penulisan.

BAB II LANDASAN KEPUSTAKAAN

Bab ini menguraikan teori-teori yang digunakan sebagai referensi dalam penyusunan laporan penelitian. Pada bagian landasan kepustakaan terdiri dari kajian pustaka dan dasar teori penelitian. Kajian pustaka membahas mengenai penelitian-penelitian sebelumnya sedangkan pada bagian dasar teori digunakan untuk memperkuat latar belakang penelitian.

BAB III METODOLOGI

Bab ini menjelaskan mengenai langkah kerja yang dilakukan dalam penelitian ini. Tahap metodologi meliputi, tipe penelitian, lokasi penelitian, partisipan penelitian, peralatan pendukung, pengumpulan data, perancangan algoritme, teknik penerapan algoritme, hasil, pembahasan, dan penutup.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab ini menguraikan perancangan sebuah sistem yang menjadi objek studi kasus pembuatan sistem. Pada bagian ini meliputi perancangan algoritme, perancangan pengujian, manualisasi, dan implementasi sistem.

BAB V PENGUJIAN DAN ANALISIS

Bab ini bertujuan untuk melaporkan hasil penelitian. Bab ini menyajikan pula data yang mendukung hasil penelitian. Selain itu pada bab ini, menguraikan makna dari hasil yang diperoleh dan juga untuk menjawab masalah penelitian.

BAB VI PENUTUP

Bab ini menguraikan kesimpulan yang telah diperoleh dari hasil dan pembahasan serta saran-saran untuk mengembangkan penelitian ini lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Bagian ini menjelaskan mengenai penelitian yang pernah dilakukan sebelumnya yang berhubungan dengan permasalahan pada penelitian ini. Pada bab ini juga dijabarkan mengenai teori pendukung penelitian ini seperti *text mining*, metode BM25 dan *K-Nearest Neighbor* dan dasar teori mengenai dokumen abstrak skripsi.

2.1 Kajian Pustaka

Pada subbab ini dijelaskan mengenai kajian pustaka yang didapatkan pada penelitian sebelumnya. Penelitian-penelitian sebelumnya berkaitan dengan penelitian yang diangkat oleh peneliti.

Penelitian yang dilakukan oleh Claudy, Perdana, dan Fauzi (2018) dengan menerapkan algoritme *K-Nearest Neighbor* (KNN). Penelitian ini menjelaskan tentang klasifikasi dokumen Twitter. Klasifikasi tersebut bertujuan untuk mengetahui karakter calon karyawan. Pada proses klasifikasi dokumen Twitter dibagi menjadi 4 kelas sesuai dengan konsep Myers-Bringgs Type Indicator (MBTI) yaitu Artisan, Guardian, Idealist, dan Rasional. Data yang digunakan pada penelitian ini sejumlah 160 data *tweet* dari calon karyawan. Data *tweet* tersebut dibagi menjadi dua dengan pembagian 80 data untuk data *training* dan 80 data untuk data *testing*. Persentase hasil yang didapatkan pada pengujian adalah 66%. Hasil ini diperoleh yaitu data uji benar sejumlah 53 data dan data uji yang salah sejumlah 27 data. Hal tersebut menunjukkan bahwa kinerja algoritme *K-Nearest Neighbor* dipengaruhi oleh banyaknya jumlah data *training* yang digunakan pada sistem (Claudy, Perdana dan Fauzi, 2018).

Penelitian lain juga dilakukan oleh Bagaskoro, Fauzi, dan Adikara (2018) dengan menerapkan algoritme *K-Nearest Neighbor* (KNN) sebagai metode untuk klasifikasi *tweets* pada berita Twitter. Pada penelitian ini dilakukan dengan melalui beberapa tahap yaitu *pre-processing,* ekspansi kata pada data uji, dan dilakukan proses klasifikasi menggunakan algoritme *K-Nearest Neighbor*. Penelitian ini dijelaskan juga bahwa dengan menggunakan data uji sebanyak 105 data dan data latih sebanyak 400 data, proses klasifikasi menggunakan algoritme *K-Nearest Neighbor* berjalan baik. Tanpa ekspansi kata, diperoleh hasil akurasi sebesar 90% dengan nilai *k* sebesar 5 sedangkan dengan menambahkan ekspansi kata, diperoleh akurasi sebesar 92% dengan nilai *k* sebesar 5 dan *threshold* sebesar 0,5 (Bagaskoro, Fauzi dan Adikara, 2018).

Penelitian yang dilakukan oleh Nurjanah, Perdana, dan Fauzi (2017) dengan menerapkan algoritme *K-Nearest Neighbor* untuk analisis sentimen tayangan televisi. Objek yang digunakan berupa opini masyarakat pada media sosial Twitter. Pada penelitian ini digunakan data sejumlah 400 data *tweet.* Data tersebut digunakan pada proses klasifikasi untuk menentukan opini masyarakat bersifat positif maupun negatif. Persentase hasil yang diperoleh sebesar 82,50% (Nurjanah, Perdana dan Fauzi, 2017).

Penelitian lain juga dilakukan oleh Suharno, Fauzi, dan Perdana (2017) dengan menerapkan algoritme *K-Nearest Neighbor* (KNN). Pada penelitian ini menjelaskan tentang proses klasifikasi teks bahasa Indonesia. Data yang digunakan berupa dokumen pengaduan SAMBAT *online*. Proses yang dilakukan pada penelitian ini meliputi beberapa bagian yaitu *pre-processing*, seleksi fitur mengunakan *Chi-Square*, *term weighting* mengunakan TF-IDF, dan klasifikasi menggunakan KNN. Data yang digunakan sejumlah 204 data dengan rasio pembagian yaitu 20% untuk data uji dan 80% untuk data latih. Hasil yang diperoleh pada penelitian ini sebesar 78% pada nilai *k* sebesar 5 (Suharno, Fauzi dan Perdana, 2017).

Penelitian yang dilakukan oleh Yang dkk. (2012) dengan menerapkan *term weighting* BM25. Pada penelitian ini menjelaskan tentang proses deteksi duplikasi laporan *bugs* dengan menggunakan *term weighting* BM25. Hasil yang diperoleh pada penelitian ini bahwa kinerja *term weighting* BM25 lebih efektif dan baik dari pada metode TF-IDF untuk duplikasi laporan *bugs.* Pada penelitian tersebut menghasilkan akurasi sebesar 90% (Yang dkk., 2012).

Penelitian lain juga dilakukan oleh Aramaki dkk. (2006) dengan menggunakan kombinasi metode BM25 dan *K-Nearest Neighbor* (KNN). Pada penelitian ini dijelaskan bahwa metode BM25 dan KNN memiliki kinerja yang baik dalam proses klasifikasi status pasien perokok. Pada proses klasifikiasi ini mengunakan lima kelas yaitu Current Smoker, Smoker, Past Smoker, Non-Smoker, dan Unknown. Pada penelitian ini menghasil akurasi sistem sebesar 88,87% dengan nilai *k* sebesar 10 (Aramaki dkk., 2006).

Berdasarkan beberapa penelitian yang dilakukan sebelumnya, maka pada penelitian ini, peneliti akan melakukan klasifikasi dokumen abstrak skripsi berdasarkan penelitian di bidang Komputasi Cerdas. Penelitian ini nantinya akan menggunakan BM25 sebagai pemeringkatan dokumen dan *K-Nearest Neighbor* sebagai klasifikasi. Diharapkan pada penelitian ini dapat mengetahui kinerja sistem klasifikasi dokumen dengan menggunakan metode BM25 dan KNN sehingga dapat digunakan sebagai acuan untuk membangun model klasifikasi dokumen skripsi pada Fakultas Ilmu Komputer Universitas Brawijaya.

2.2 Dokumen Abstrak Skripsi

Skripsi merupakan sebuah karya ilmiah bagi mahasiswa program sarjana (S1). Skripsi merupakan wujud dari penerapan pengetahuan dan teknologi yang didasarkan pada kaidah ilmiah dalam minat studi tertentu. Skripsi juga merupakan hasil sebuah penelitian yang disusun dengan sistematis dan dengan kaidah yang baku. Selain itu skripsi harus memenuhi unsur kesesuaian dengan bidang minat mahasiswa pada salah satu program studi yang ditempuh (FILKOM, 2017a).

Di dalam dokumen skripsi terdapat bagian abstrak. Abstrak adalah bagian yang menguraikan secara singkat intisari skripsi. Pada bagian abstrak berfungsi untuk mendapatkan gambaran secara akurat mengenai intisari skripsi. Oleh

karena itu, pada bagian abstrak harus memberikan suatu gambaran yang jelas mengenai beberapa hal berikut (FILKOM, 2017a).

1. Mengenai penelitian latar belakang penelitian, masalah penelitian, dan tujuan penelitian.
2. Mengenai metode yang digunakan.
3. Hasil dan pembahasan penelitian.
4. Kesimpulan dari penelitian.

2.3 Komputasi Cerdas FILKOM UB

Fakultas Ilmu Komputer atau disingkat FILKOM UB merupakan salah satu fakultas di Universitas Brawijaya yang didirikan pada tahun 2015. Fakultas Ilmu Komputer memiliki tujuan untuk menghasilkan lulusan yang kompeten di bidang teknologi informasi dan ilmu komputer. Pada saat ini FILKOM UB memiliki enam program studi yaitu S1 Sistem Informasi, S1 Teknik Informatika, S1 Pendidikan Teknologi Informasi, S1 Sistem Komputer, S1 Teknologi Informasi dan S2 Ilmu Komputer (FILKOM, 2017b).

Program studi S1 Teknik Informatika pada FILKOM UB memiliki tujuan untuk menyajikan pemanfaatan teknologi informasi dalam proses identifikasi masalah, pengolahan data dan pengambilan keputusan sesuai dengan prinsip keilmuan dan kerekayasaan. Dalam menunjang hal tersebut, Teknik Informatika FILKOM UB memiliki 4 bidang peminatan yaitu Rekaya Perangkat Lunak (RPL), Komputasi Berbasis Jaringan (KBJ), Komputasi Cerdas (KC), dan Multimedia, Game dan Mobile (MGM) (FILKOM, 2017b).

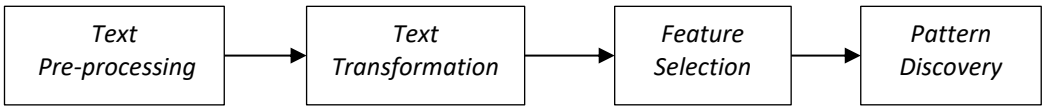
Komputasi Cerdas (KC) adalah bidang minat yang menekankan pada kemampuan lulusan dalam menerapkan, memanipulasi dan menganalisis data berupa *text*, numerik dan citra pada berbagai bidang aplikasi sesuai dengan metode sistem cerdas yang digunakan. Dalam menunjang hal tersebut, peminatan Komputasi Cerdas memiliki beberapa mata kuliah pilihan, diantaranya adalah *text mining*, data *mining*, sistem pakar, pengolahan citra digital, sistem pendukung keputusan, algoritme evolusi, logika *fuzzy*, dan lain sebagainya (FILKOM, 2017b). Mata kuliah tersebut bertujuan untuk menunjang penelitian mahasiswa program studi S1 Teknik Informatika.

2.4 *Text Mining*

Text mining merupakan salah satu teknik yang menerapkan konsep data *mining* untuk mencari pola di dalam kumpulan teks. *Text mining* dapat diartikan juga sebagai penambangan data yang berupa teks. *Text mining* memiliki manfaat untuk memperoleh pola-pola data dan ekstraksi informasi dari pengetahuan yang potensial dari data teks. Perbedaan mendasar antar *text mining* dan data *mining* yaitu *text mining* memproses data teks yang sifat data tersebut tidak terstruktur, sedangkan data mining yaitu mengolah data yang bersifat terstruktur. Oleh karena itu, di dalam *text mining* terdapat tahapan *pre-procesing* teks. Tahapan tersebut

digunakan untuk mengubah teks dari tidak terstruktur menjadi terstruktur (Feldman dan Sanger, 2006).

Tahapan *text mining* dibagi menjadi 4 tahapan utama, yaitu pemrosesan awal teks (*text pre-processing*), transformasi teks (*text transformation*), pemilihan fitur (*feature selection*) dan penemuan pola (*pattern discovery*) (Feldman dan Sanger, 2006). Tahapan *text mining* ditunjukkan pada Gambar 2.1



Gambar 2.1 Tahapan *Text Mining*

2.5 *Text Pre-processing*

Text pre-processing adalah tahapan awal untuk menyiapkan *dataset* untuk mempermudah pemrosesan data teks dan juga untuk mendapatkan kinerja yang tinggi (Feldman dan Sanger, 2006). *Text pre-processing* digunakan untuk mengubah data teks dari tidak terstruktur menjadi terstruktur. Proses ini juga bertujuan untuk meningkatkan kualitas data, meningkatkan efisiensi dalam proses penambangan data teks. Tahapan *text pre-processing* terdiri dari *case folding*, *tokenization*, *stopword*, dan *stemming*. Tahapan tersebut dapat dilihat pada Gambar 2.2.

Gambar 2.2 Tahapan *Text Pre-processing*

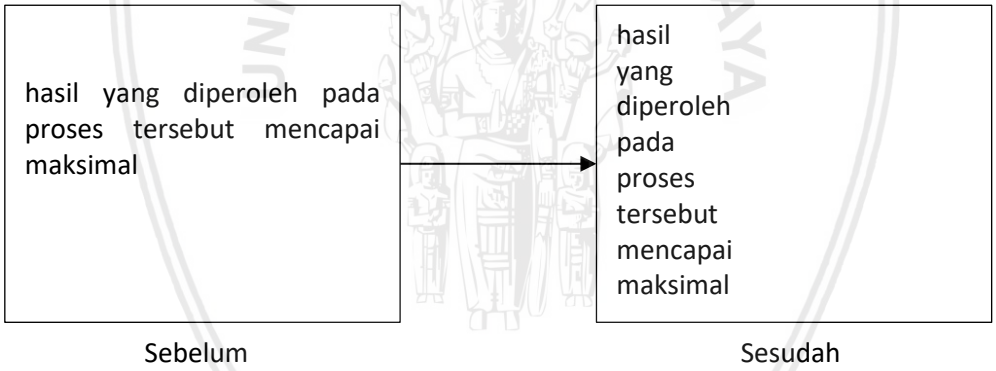
2.5.1 *Case Folding*

Case folding adalah tahap awal pada *pre-processing* yang digunakan untuk mengubah semua karakter pada dokumen menjadi huruf besar atau huruf kecil (Feldman dan Sanger, 2006). Proses *case folding* dilakukan karena dokumen teks tidak konsisten dalam penggunaan huruf kapital . Proses *case folding* dilakukan untuk mengkorversi keseluruhan dokumen teks menjadi karakter huruf kecil, sehingga dengan melibatkan proses tersebut dapat memudahkan saat pengolahan teks pada tahap berikutnya. Proses *case folding* ditunjukkan pada Gambar 2.3.

Gambar 2.3 Proses *Case Folding*

2.5.2 *Tokenization*

Tokenization adalah tahap yang diproses setelah tahap *case folding.* Tahap ini merupakan proses yang bertujuan untuk memecah *string* input bedasarkan tiap kata yang merangkainya. Pemecahan tersebut dilakukan dengan pemisah seperti karakter *whitespace* (spasi*,* tabulasi*,* dan *newline*). Selain itu pada *tokenization* juga melakukan penghilangan karakter lain selain alfabet seperti “”:/@^*&.,<>!+ dan lain sebagainya (Feldman dan Sanger, 2006). Proses *tokenization* ditunjukkan pada Gambar 2.4.

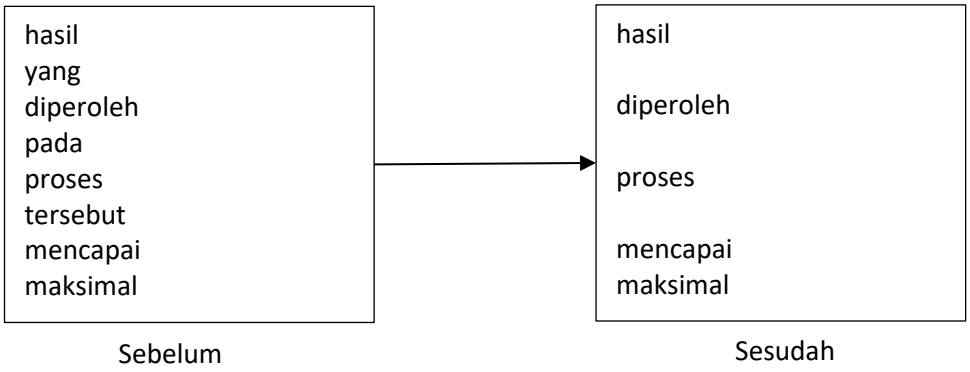


Gambar 2.4 Proses *Tokenization*

2.5.3 *Stopword Removal*

Stopword removal atau biasa disebut dengan *filtering* adalah teknik penghilangan kata tidak penting dari hasil *tokenization*. *Stopword* adalah kata-kata yang tidak penting, yang mana kata tersebut tidak memiliki makna,sehingga dibuang dengan pendekatan *bag-of-words.* Tujuan dari adanya *stopword removal* adalah untuk mempercepat pemrosesan teks dan menghemat penyimpanan. (Feldman dan Sanger, 2006).

Pada penelitian ini kamus *stopword* menggunakan kamus *stopword* Tala. *Stopword* Tala diolah dengan melakukan penambahan kata pada *stopword list* karena kata dianggap tidak penting dan tidak mengandung kata *sentiment* (Tala, 2003). Proses *stopword removal* ditunjukkan pada Gambar 2.5.

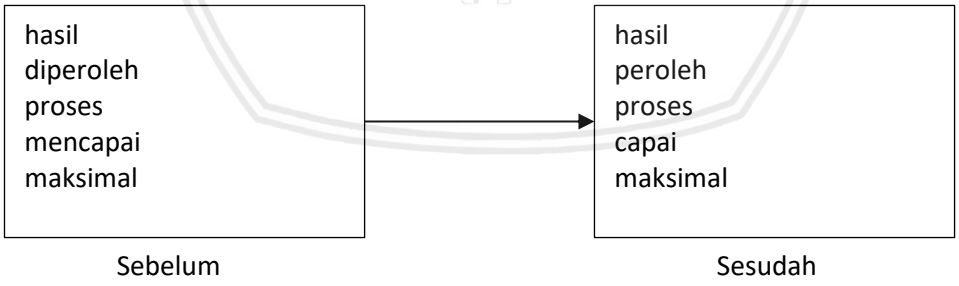


Gambar 2.5 Proses *Stopword Removal*

2.5.4 *Stemming*

Stemming adalah proses pengubahan kata dari kata berimbuhan menjadi kata dasarnya. Proses *stemming* pada teks berbahasa Indonesia meliputi identifikasi dan pengapusan *suffix*, *prefix*, dan *confix* sehingga dapat menjadi bentuk kata dasar. Pada penelitian ini menggunakan algoritme *stemming* Nazief Adriani. Algoritme Nazief Adriani merupakan *morphologi* yang luas, yang menggabungkan maupun melakukan atau tidak melakukan rangkuman *affixed* yang terdiri dari *suffix*, *prefix*, dan *confix* (Adriani dkk., 2007).

Pada penelitian ini menggunakan library Sastrawi. Sastrawi *stemmer* merupakan *library* sederhana yang mudah untuk digunakan. *Library* ini menerapkan algoritme Nazief Adriani (Adriani dkk., 2007). Proses *stemming* ditunjukkan pada Gambar 2.6.



Gambar 2.6 Proses *Stemming*

2.6 BM25

Di dalam *text mining* terdapat proses pembobotan dan pemeringkatan dokumen. Salah satu proses pemeringkatan dokumen dapat menggunakan BM25. Metode BM25 sendiri adalah metode pemeringkatan yang digunakan untuk mengurutkan hasil kecocokan terhadap dokumen-dokumen, berdasarkan kata

kunci yang dicari. Di dalam metode BM25 terdapat proses pembobotan yang memiliki kinerja lebih baik daripada pembobotan TF-IDF (Yang dkk., 2012). Persamaan metode BM25 ditunjukkan pada Persamaan 2.1.

$$BM25\ Score\ (q,d) = \sum_{i=1}^{|q|} idf(q_i).\frac{tf(q_i,d)\ .\ (k_1+1)}{tf(q_i,d) + k_i\ .(1-b+b\ .\ \frac{|d|}{dl_{avg}})} \quad (2.1)$$

Keterangan:

$idf(qi)$: nilai *invers document frequency* pada *term query i*

$tf(q_i,d)$: jumlah frekuensi *term query i* pada dokumen *j*

k_1 : 1,2 ≤ k_1≤ 2,0

b : 0,5 ≤ b ≤ 0,8

dl_{avg} : rata-rata panjang semua dokumen

$|d|$: panjang dokumen

Pada Persamaan 2.1 terdapat nilai IDF. Nilai tersebut dapat ditunjukkan pada Persamaan 2.2.

$$idf(q_i) = \log(\frac{N - df(q_i) + 0{,}5}{df(q_i) + 0{,}5}) \quad (2.2)$$

Keterangan:

$idf(qi)$: nilai *invers document frequency* pada *term query i*

N : total dokumen dalam koleksi

$df(qi)$: jumlah dokumen yang terdapat *term query i*

2.7 *Pattern Discovery*

Pattern discovery adalah suatu tahap di bidang *text mining* untuk menemukan suatu pola dari kumpulan data teks. Pada tahap ini biasanya menggunakan teknik data *mining*. Oleh karena itu, untuk menemuan pola data, teknik data *mining* sering digunakan pada proses *text mining*. Proses data *mining* yang sering dilakukan adalah klasifikasi (Feldman dan Sanger, 2006).

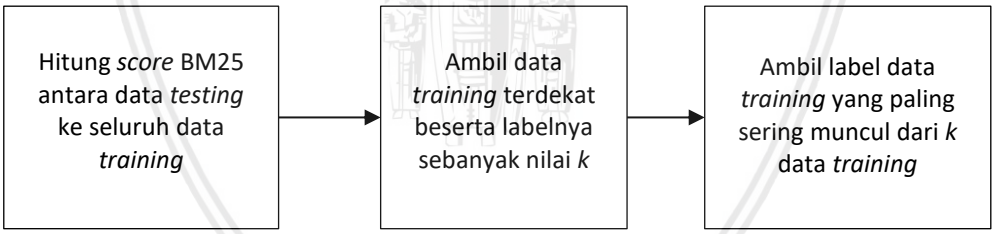
2.8 Klasifikasi Teks

Klasifikasi teks adalah proses atau teknik dalam data *mining* yang bertujuan untuk menentukan suatu dokumen teks masuk kelas tertentu. Pada klasifikasi terdapat 2 metode dasar yaitu *Supervised Text Classification dan Unsupervised Text Classification*. *Unsupervised Text Classification* adalah metode klasifikasi dokumen yang sebelumnya tidak memiliki informasi mengenai kelas dokumen tersebut, sedangkan *Supervised Text Classification* adalah metode klasifikasi dokumen yang sebelumnya sudah mengetahui kelas dokumen tersebut (Feldman dan Sanger, 2006).

2.9 Algoritme *K-Nearest Neighbor*

K-Nearest Neighbor (KNN) adalah algoritme yang digunakan sebagai klasifikasi dokumen. Cara kerja metode *K-Nearest Neighbor* adalah menghitung jarak kedekatan suatu dokumen dengan dokumen lainnya. Pada penelitian ini, perhitungan jarak kedekatan dokumen dengan dokumen lainnya menggunakan nilai dari BM25.

Pada klasifikasi *K-Nearest Neighbor* akan dilakukan perhitungan jarak menggunakan BM25 antara data *testing* dengan data *training.* Setelah itu diambil data sebanyak *k* data *training* terdekat dengan label data tersebut. Label yang paling banyak muncul dari *k* data *training* akan menjadi kelas label di data *testing*. Tahap *K-Nearest Neighbor* dapat ditunjukkan pada Gambar 2.7.



Gambar 2.7 Tahapan *K-Nearest Neighbor*

2.10 Evaluasi

Evaluasi adalah proses pengukuran kinerja sistem klasifikasi. Tujuan perhitungan evaluasi untuk mengetahui nilai persentase kebenaran dari hasil klasifikasi dokumen abstrak skripsi bedasarkan penelitian di bidang komputasi cerdas. Pengukuran kinerja dapat dilakukan dengan mengunakan *confusion matrix*. Pada *confusion matrix* terdapat empat istilah sebagai gambar hasil klasifikasi, yiatu *True Positive* (TP), *False Positive* (FP), *True Negatif (*TN), dan *False Negative* (FN) (Powers, 2007). *Confusion matrix* dapat dilihat pada Tabel 2.1.

Tabel 2.1 *Confusion Matrix*

		Nilai Prediksi	
		True	*False*
Nilai Sebenarnya	*True*	*True Positive* (TP)	*False Negative* (FN)
	False	*False Positive* (FP)	*True Negative* (TN)

Keterangan:

True Positive (TP) : jumlah data positif yang terdeteksi benar

False Positive (FP) : jumlah data negatif yang terdeteksi data positif

True Negative (TN) : jumlah data negatif yang terdeteksi benar

False Negative (FN) : jumlah data positif terdeteksi data negatif

Pengukuran kinerja sistem dilakukan dengan menghitung *precision*, *recall, f-measure,* dan *accuracy*. *Precision* adalah tingkat ketepatan antara informasi yang diperoleh pengguna dengan jawaban yang dihasilkan oleh sistem (George dan Rothschild, 2005). Persamaan *precision* dapat ditunjukan pada Persamaan 2.3

$$precision = \frac{tp_i}{tp_i + fp_i} \quad (2.4)$$

Keterangan:

$precision$: persentase nilai ketepatan sistem melakukan prediksi

tp_i : jumlah data positif yang terdeteksi benar

fp_i : jumlah data negatif yang terdeteksi data positif

Recall adalah tingkat keberhasilan suatu sistem dalam menemukan kembali sebuah informasi (George dan Rothschild, 2005). Persamaan *recall* ditunjukkan pada Persamaan 2.4.

$$recall = \frac{tp_i}{tp_i + fn_i} \quad (2.4)$$

Keterangan:

$precision$: persentase nilai kebenaran sistem melakukan prediksi

tp_i : jumlah data positif yang terdeteksi benar

fn_i : jumlah data positif terdeteksi data negatif

Persamaan *f-measure* dapat ditunjukkan pada Persamaan 2.5 dan Persamaan 2.6. Persamaan 2.6 merupakan persamaan turunan dari Persamaan 2.5 (Chinchor, 1992).

$$f - measure = \frac{(\beta^2 + 1)\ precision\ \times recall}{(\beta^2 \times precision) + recall} \quad (2.5)$$

β adalah parameter yang mengontrol keseimbangan antara *precision* dan *recall*. Ketika β = 1 maka *f- measure* setara dengan *harmonic mean*, sehingga *f-measure* adalah kombinasi *precision* dan *recall* sebagai *harmonic mean* (George dan Rothschild, 2005). Persamaan *harmonic mean* ditunjukkan pada Persamaan 2.6.

$$f - measure = 2 \times \frac{precision\ \times recall}{precision + recall} \quad (2.6)$$

Keterangan:

$f - measure$: persentase dari kombinasi *precision* dan *recall*

$precision$: persentase nilai ketepatan sistem melakukan prediksi

$recall$: persentase nila kebenaran sistem melakukan prediksi

Accuracy adalah tingkat ketepatan sistem dalam memprediksi hasil klasifikasi (George dan Rothschild, 2005). Persamaan *accuracy* dapat ditunjukkan pada Persamaan 2.7.

$$accuracy = \frac{tp_i + tn_i}{tp_i + tn_i + fp_i + fn_i} \quad (2.7)$$

Keterangan:

tp_i : jumlah data positif yang terdeteksi benar

fn_i : jumlah data positif terdeteksi data negatif

fp_i : jumlah data negatif yang terdeteksi data positif

tn_i : jumlah data negatif yang terdeteksi benar

BAB 3 METODOLOGI

Pada bagian ini dibahas mengenai metode yang digunakan dalam penelitian ini. Tahap metodologi meliputi tipe penelitian, lokasi penelitian, partisipan penelitian, peralatan pendukung, pengumpulan data, statistik data, perancangan algoritme, teknik penerapan algoritme, dan teknik analisis data.

3.1 Tipe Penelitian

Tipe penelitian ini adalah penelitian non-implementatif analitik. Pada tipe penelitian ini nantinya akan menjelaskan mengenai hubungan antar elemen dalam objek penelitian dengan masalah tertentu yang sedang diteliti. Produk utama yang dihasilkan adalah hasil analisis dengan menjawab pertanyaan-pertanyaan penelitian.

3.2 Lokasi Penelitian

Lokasi penelitian berada di laboratorium riset Fakultas Ilmu Komputer Universitas Brawijaya (FILKOM UB) yang berada Jalan Veteran No. 8 Malang.

3.3 Partisipan Penelitian

Penelitian ini akan melibatkan partisipan berupa pakar di bidang Komputasi Cerdas. Pakar tersebut berasal dari dosen Fakultas Ilmu Komputer Universitas Brawijaya. Pakar tersebut nantinya akan membantu penelitian ini berupa pelabelan dokumen abstrak skripsi kedalam kelas yang sudah ditentukan sebelumnya.

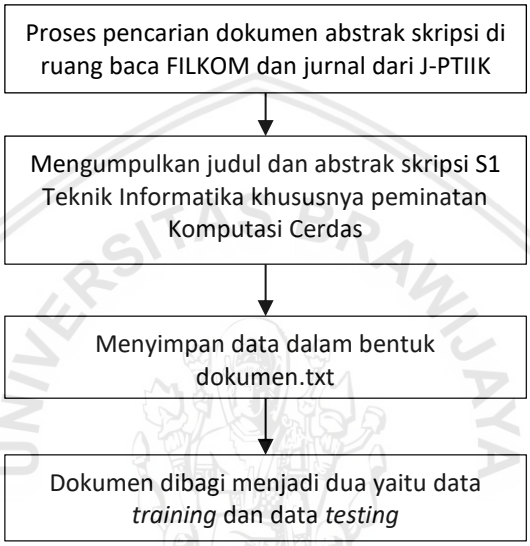
3.4 Peralatan Pendukung

Pada tahap ini dilakukan analisis peralatan pendukung yang dapat membantu penelitian. Beberapa peralatan pendukung yang diperlukan dalam penelitian ini sebagai berikut.

1. Peralatan pendukung perangkat keras meliputi:
    a. RAM 6 GB
    b. Prosesor Intel Core i-3
    c. Sistem Operasi Windows 10 Pro 64-bit
2. Peralatan pendukung perangkat lunak meliputi:
    a. Python 3.6, digunakan sebagai bahasa pemrograman untuk membangun sistem.
    b. Spyder 3.2.6

3.5 Pengumpulan Data

Data mengenai dokumen abstrak skripsi diperoleh dari kumpulan dokumen skripsi S1 Teknik Informatika FILKOM UB dan jurnal dari J-PTIIK. Data yang digunakan berupa dokumen abstrak skripsi di bidang Komputasi Cerdas. Data yang diperoleh disimpan dalam bentuk dokumen.txt dan dibagi menjadi dua bagian yaitu data *training* dan data *testing*. Data tersebut kemudian diproses menggunakan BM25 dan *K-Nearest Neighbor* (KNN) untuk klasifikasi dokumen abstrak skripsi bedasarkan fokus penelitian di bidang Komputasi Cerdas. Diagram pengumpulan data ditunjukkan pada Gambar 3.1.



Gambar 3.1 Pengumpulan Data

3.6 Statistik Data

Data yang digunakan adalah data pada tahun 2017 sampai 2018, yang terdiri dari 3 Volume yaitu Volume 1 No. 1-12, Volume 2 No. 1-12, dan Volume 3 No. 1. Data diambil pada tanggal 5 Oktober 2018 dari J-PTIIK. Data tersebut diperoleh sebanyak 331 data. Berdasarkan *keyword* yang telah ditentukan sebelumnya, data tersebut terdiri dari dokumen Sistem Pakar, dokumen Sistem Pendukung Keputusan, dokumen Text Mining, dokumen Data Mining, dan dokumen Pengolahan Citra Digital. Statistik data dapat dilihat pada Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Statistik Data

Keyword	Jumlah Data	Fokus Penelitian
Sistem Pakar, Diagnosis, *Dempster Shafer*, *Certainty Factor*	75	Sistem Pakar (SP)
Citra Digital, HSV, Piksel,	32	Pengolahan Citra Digital (PCD)

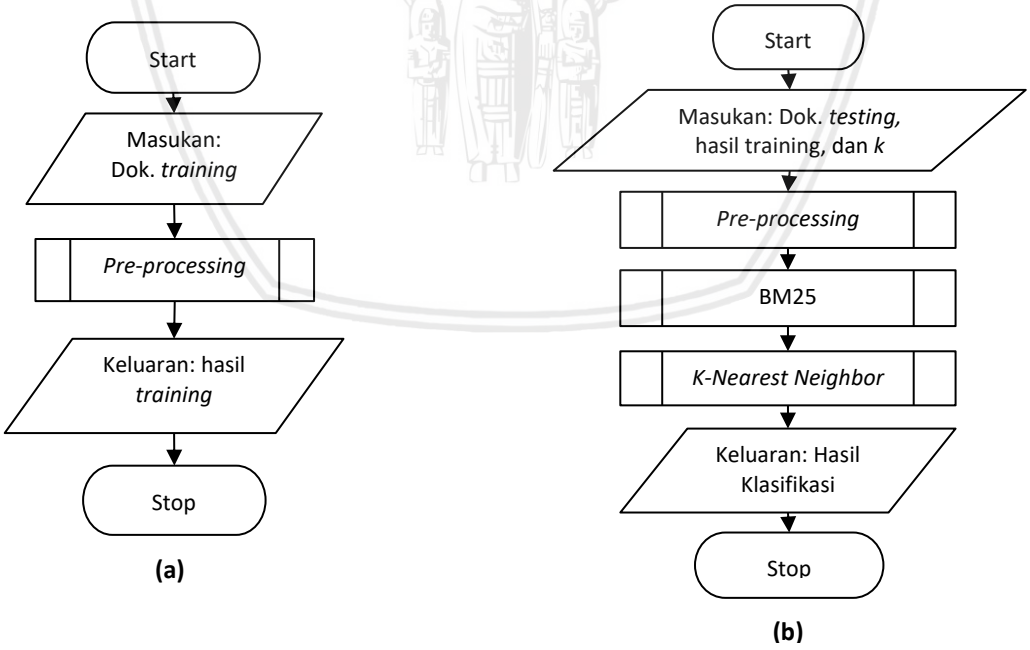
Tabel 3.2 Statistik Data (Lanjutan)

Keyword	Jumlah Data	Fokus Penelitian
Sistem Pendukung Keputusan, Sistem Rekomendasi, *Profile Matching*, AHP, TOPSIS, SAW, *Promethee*	67	Sistem Pendukung Keputusan (SPK)
Peringkasan, Analisis Sentimen, NER, TF-IDF, Dokumen, Teks, Twitter, Artikel, *Query Expansion*	67	Text Mining (TM)
Klasifikasi, Prediksi, Peramalan, Apriori, *Clustering*	90	Data Mining (DM)

Berdasarkan statistik data yang telah dibuat, maka pada penelitian ini menggunakan lima kelas atau kategori. Lima kelas tersebut yaitu Sistem Pakar (SP), Sistem Pendukung Keputusan (SPK), Text Mining (TM), Data Mining (DM), dan Pengenalan Citra Digital (PCD).

3.7 Perancangan Algoritme

Pada tahap ini dilakukan perancangan algoritme untuk dijadikan dasar saat implementasi. Pada penelitian ini sistem dapat memproses data *training* dan data *testing* sesuai dengan tahapan klasifikasi teks. Alur algoritme dapat ditunjukkan pada Gambar 3.2.



Gambar 3.2 Alur Algoritme, (a) Alur Data *Training,* (b) Alur Data *Testing*

3.8 Teknik Penerapan Algoritme

Pada tahap teknik penerapan algoritme dilakukan dengan menggunakan bahasa pemrograman Python dan *library* pendukung. Teknik penerapan algoritme sebagai berikut.

1. Pembuatan kode program *pre-processing*
2. Pembuatan kode program BM25
3. Pembuatan kode program klasifikasi (*K-Nearest Neighbor*)

3.9 Teknik Analisis Data

Pada tahap analisis data berfungsi untuk menjelaskan dan menganalisis dari hasil yang diperoleh. Pada Tahap ini berfungsi juga untuk menjawab masalah penelitian. Sehingga nantinya akan dibahas mengenai hasil akurasi (*Precision, Recall*, dan *F-measure*) yang diperoleh dan pengaruh nilai *k* pada klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang komputasi cerdas menggunakan BM25 dan KNN.

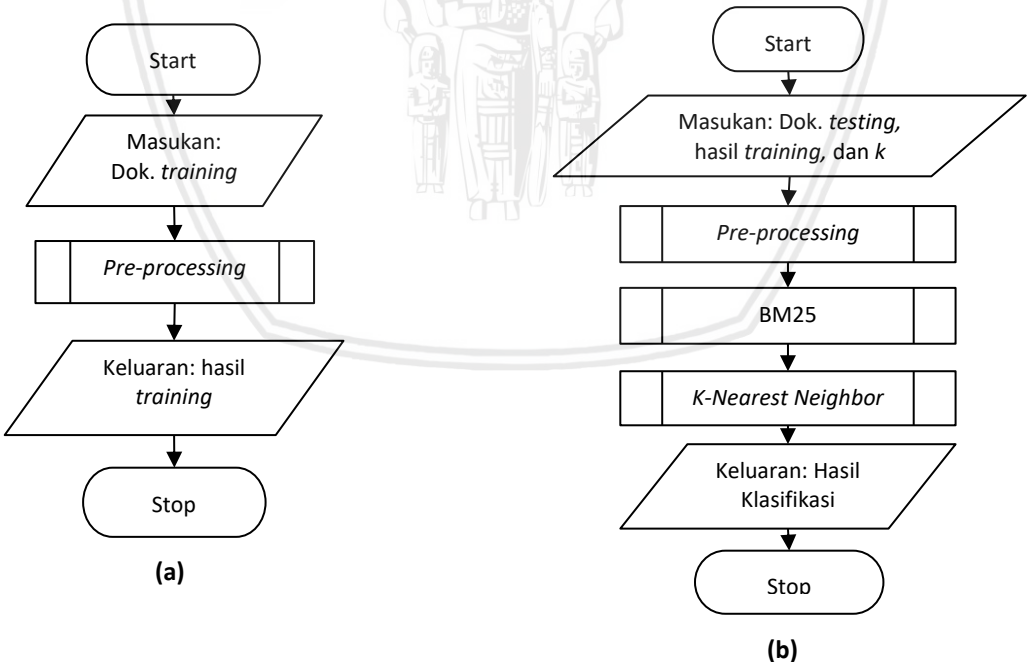
BAB 4 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini menjelaskan mengenai perancangan yang berkaitan dengan sistem yang akan dibuat pada penelitian ini, seperti deskripsi permasalahan, deskipsi umum sistem, alur algoritme, manualisasi dan perancangan evaluasi sistem.

4.1 Deskripsi Umum Sistem

Sistem yang dikembangkan bertujuan untuk mengkategorikan dokumen skripsi sesuai dengan fokus penelitian di bidang Komputasi Cerdas. Algoritme yang digunakan adalah BM25 dan *K-Nearest Neighbor* (KNN). Dokumen yang digunakan adalah dokumen abstrak skripsi khususnya di bidang Komputasi Cerdas yang dibagi menjadi dua yaitu data *training* dan data *testing*.

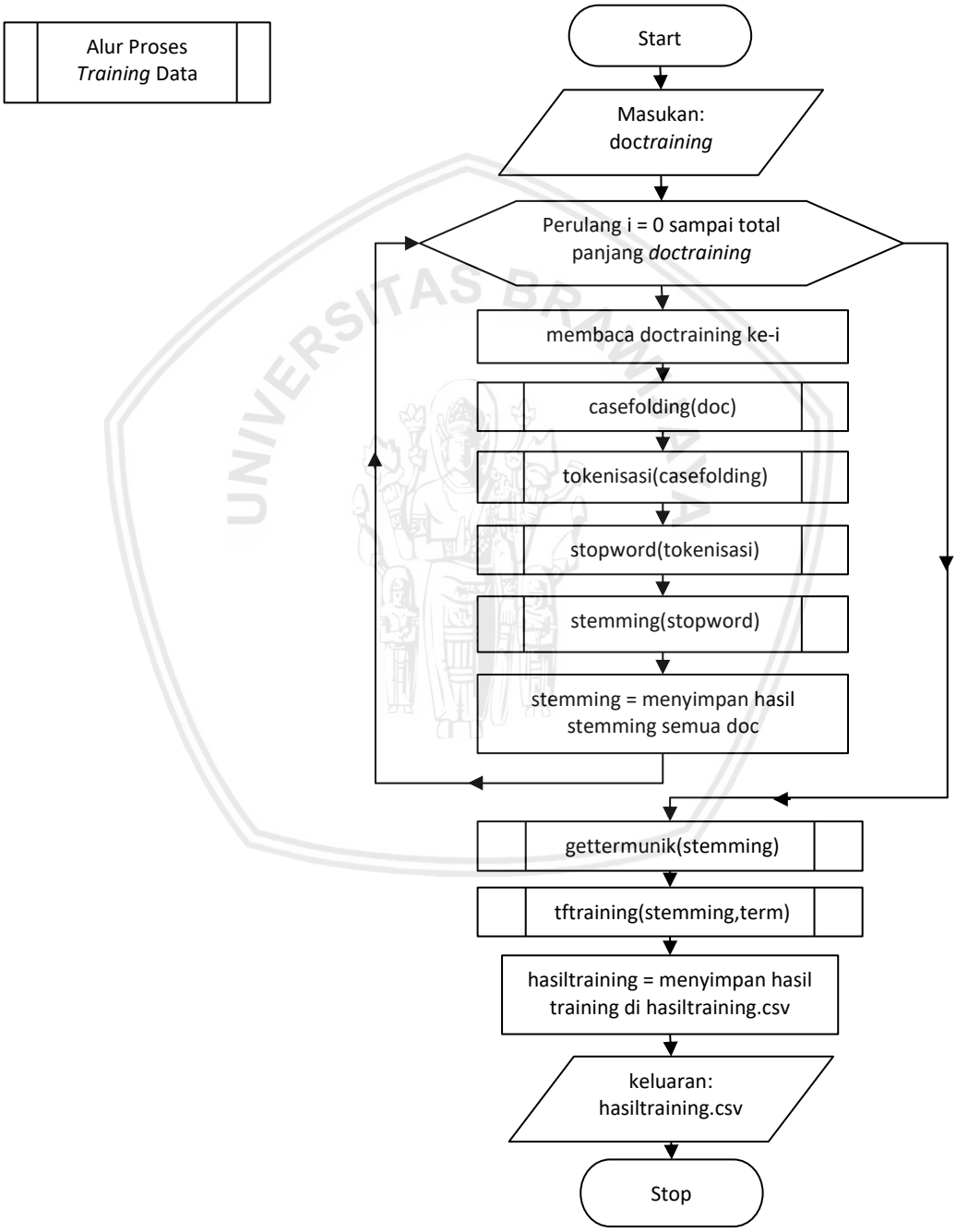
Dokumen yang diperoleh tersebut akan diproses pada *pre-processing* teks. Kemudian setiap kata hasil *pre-processing* teks akan dilakukan perhitungan frekuensi dan nilai IDF. Setelah didapatkan frekuensi tiap kata dan nilai IDF, maka dihitung *score* BM25 tiap dokumen *training* terhadap dokumen *testing* yang dimasukkan. Setelah mendapat *score* tiap dokumen *training* terhadap dokumen *testing* maka dilakukan proses klasifikasi dengan menggunakan algortime KNN. Proses klasifikasi tersebut dilakukan dengan mengurutkan dan menyeleksi sebanyak nilai *k.* Deskripsi umum sistem ditunjukkan pada Gambar 4.1.



Gambar 4.1 Deskripsi Umum Sistem, (a) Alur Data *Training*, (b) Alur Data *Testing*

4.2 Alur Proses *Training* Data

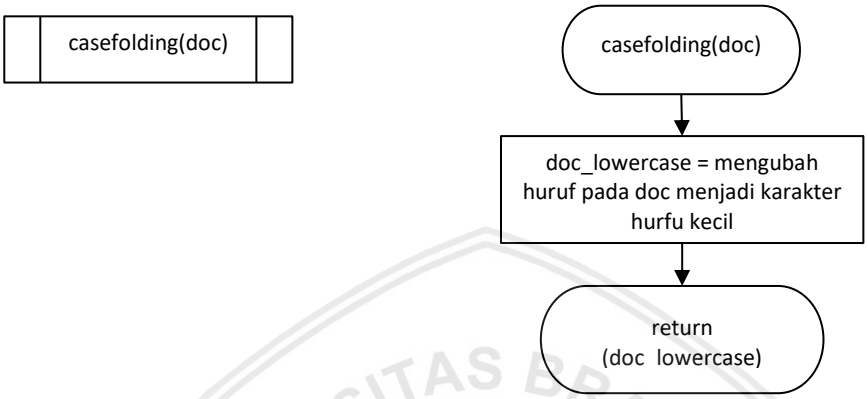
Alur proses *training* data merupakan alur untuk memproses atau mengolah data *training*. Data *training* tersebut berupa dokumen abstrak skripsi di bidang Komputasi Cerdas. Alur proses *training* data yaitu melakukan *pre-processing* dokumen *training*. *Pre-processing* teks tersebut terdiri dari proses *case folding*, *tokenization*, *stopword removal*, dan *stemming*. Alur proses training data ditunjukkan pada Gambar 4.2.



Gambar 4.2 Alur Proses *Training* Data

4.2.1 Alur Proses *Case Folding*

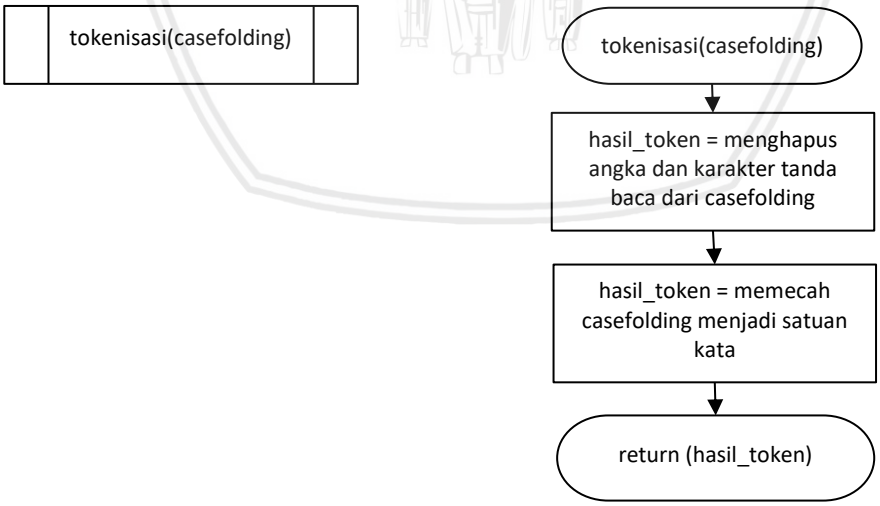
Case folding merupakan alur proses pertama pada *pre-processing* teks. Pada alur ini, seluruh karakter huruf pada dokumen akan diubah menjadi karakter huruf kecil. Tujuannya adalah untuk dapat memudahkan pengolahan teks pada proses berikutnya. Alur proses *case folding* ditunjukkan pada Gambar 4.3.



Gambar 4.3 Alur Proses *Case Folding*

4.2.2 Alur Proses Tokenisasi

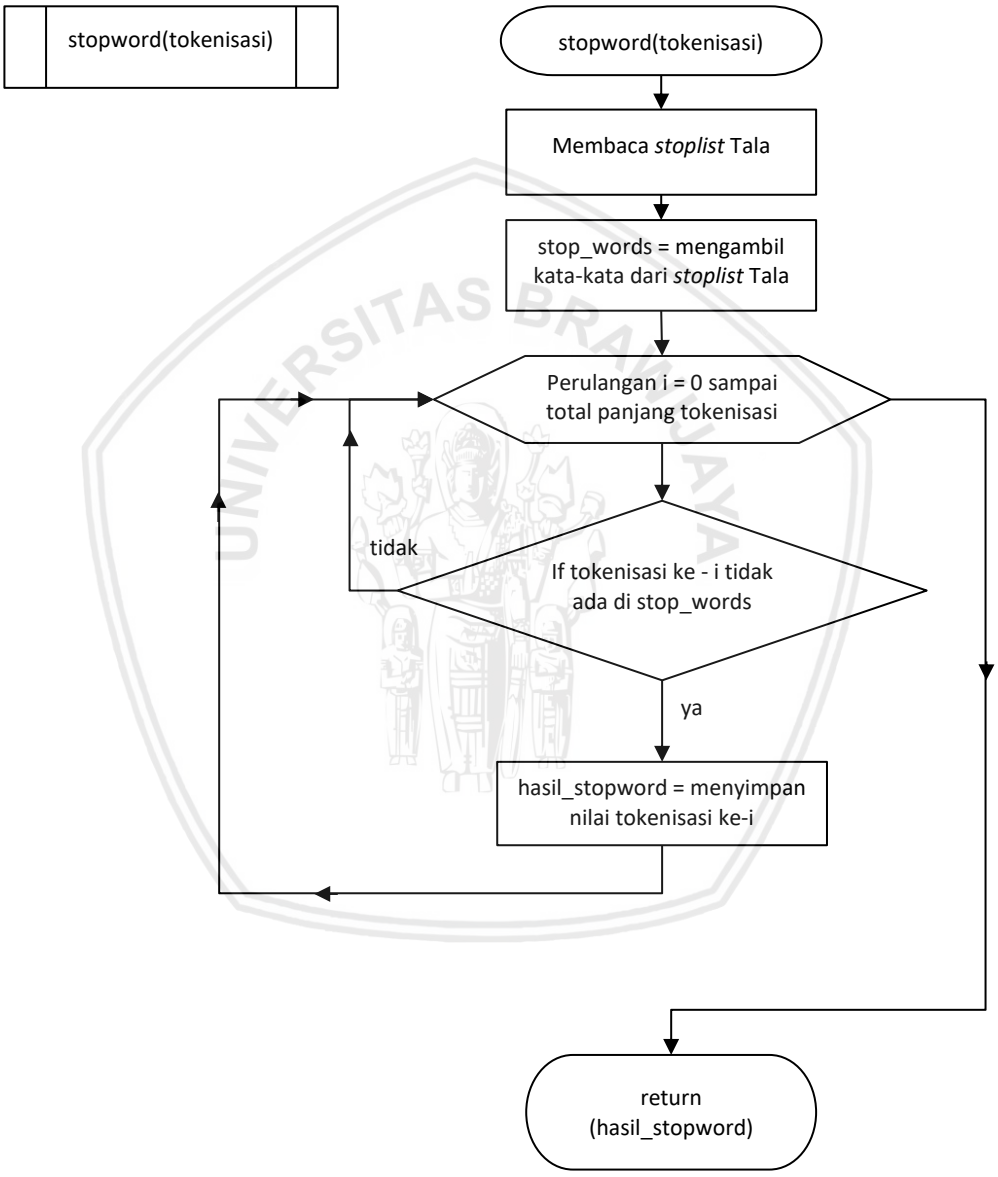
Tokenisasi merupakan tahap dalam *text pre-processing*. Tokenisasi dipakai pada *training* data yang berguna untuk memecah kalimat dalam dokumen *training* menjadi satuan kata. Selain itu juga karakter angka dan tanda baca akan dihilangkan dari dokumen pada tahap ini. Alur proses Tokenisasi ditunjukkan Gambar 4.4.



Gambar 4.4 Alur Proses Tokenisasi

4.2.3 Alur Proses *Stopword Removal*

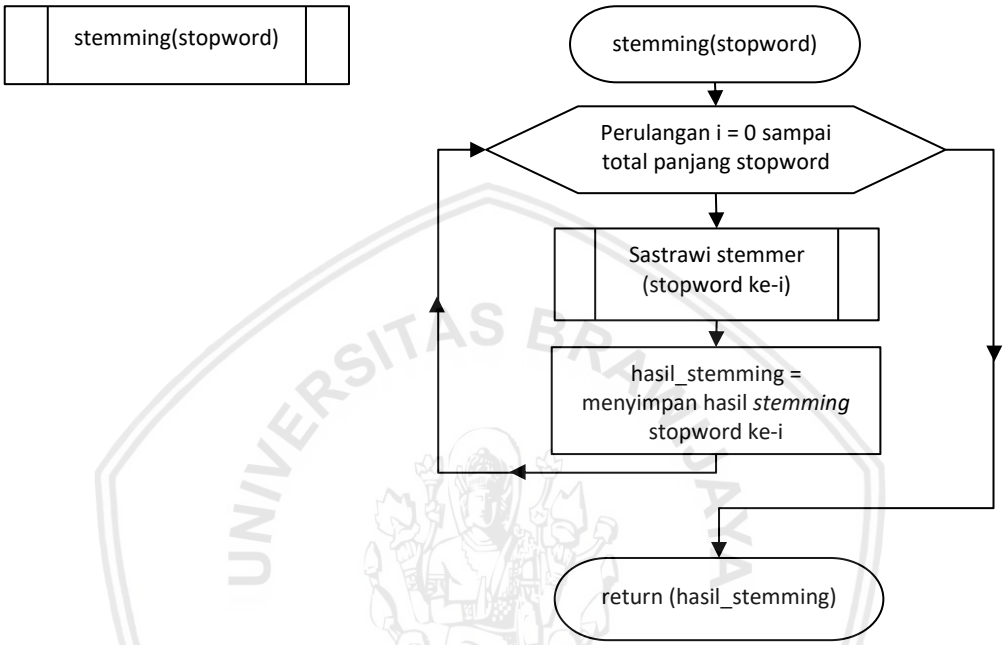
Stopword removal merupakan salah satu tahap dalam *text pre-processing*. *Stopword removal* dipakai pada proses *training* data yang berguna untuk menghapus kata-kata yang tidak penting pada dokumen *training*. Alur proses *stopword removal* ditunjukkan pada Gambar 4.5.



Gambar 4.5 Alur Proses *Stopword Removal*

4.2.4 Alur Proses *Stemming*

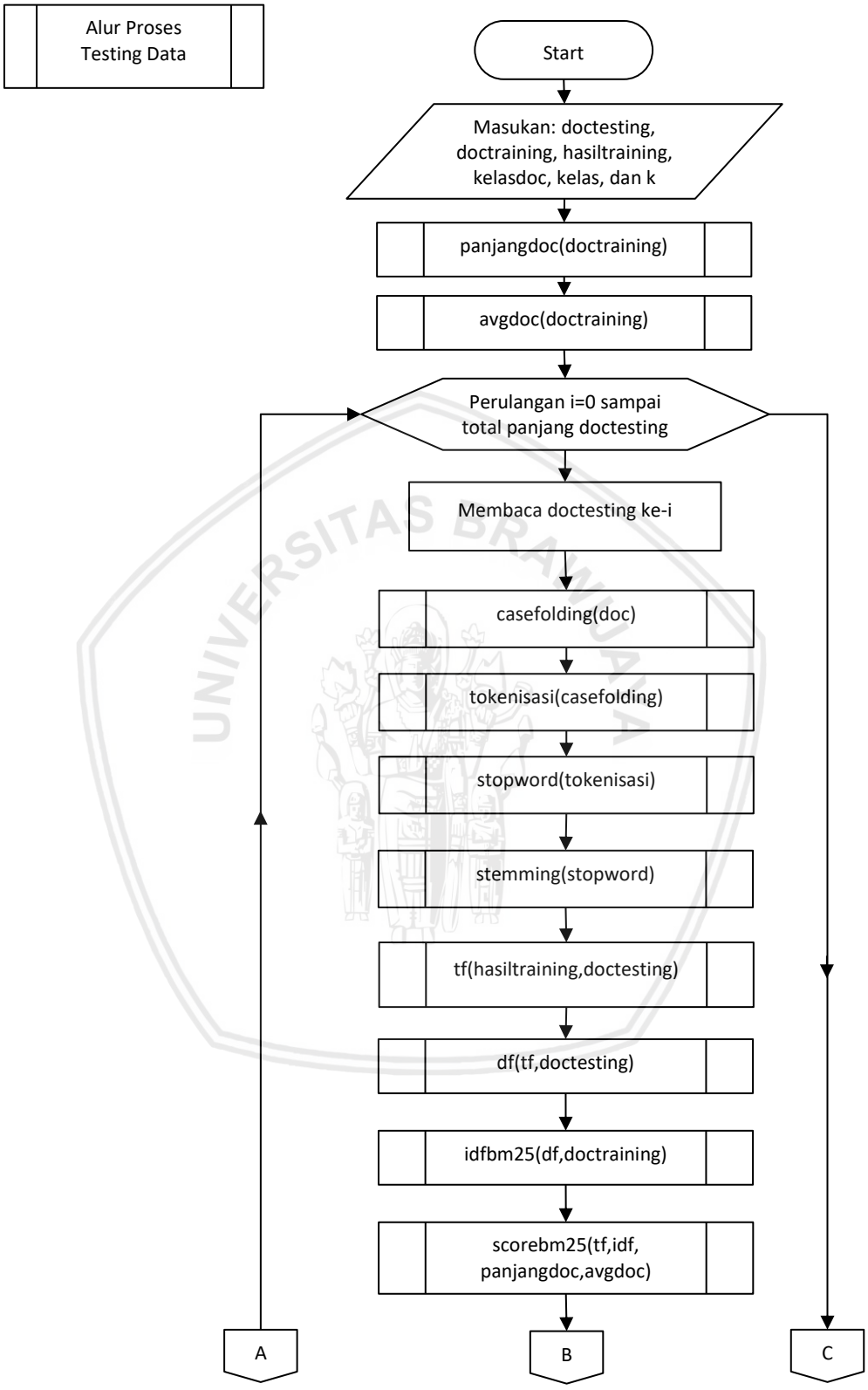
Stemming merupakan tahap terakhir dari *pre-processing* teks*.* Proses *stemming* dipakai pada *training* data yang berguna untuk mengubah kata yang berimbuhan menjadi kata dasarnya. Pada proses *stemming* menggunakan *library* pendukung yaitu sastrawi *stemmer.* Alur proses *stemming* ditunjukkan pada Gambar 4.6.



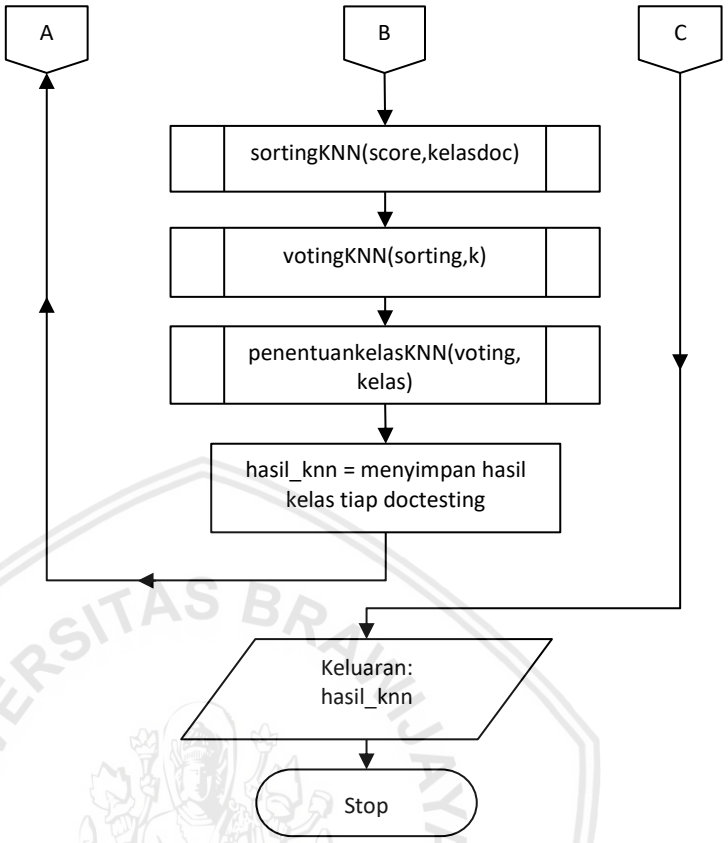
Gambar 4.6 Alur Proses *Stemming*

4.3 Alur Proses *Testing* Data

Alur proses *testing* data merupakan alur untuk memproses atau mengolah data *testing*. Data *testing* tersebut berupa dokumen abstrak skripsi di bidang Komputasi Cerdas. Alur proses *testing* data terdiri dari *pre-processing* dokumen *testing*, menghitung BM25 tiap dokumen *training*, dan klasifikasi dokumen testing menggunakan *K-Nearest Neighbor* (KNN). *Pre-processing* tersebut terdiri dari proses *case folding*, *tokenization*, *stopword removal*, dan *stemming*. Metode BM25 terdiri dari proses perhitungan panjang dokumen, rata-rata dokumen, TF, DF, IDF, dan *Score* BM25 tiap dokumen *training*. Setelah mendapatkan *score* BM25 tiap dokumen *training,* dilanjutkan dengan klasifikasi dokumen *testing* menggunakan algoritme KNN. Algoritme KNN terdiri dari mengurutkan *score* BM25, mengambil *score* BM25 sebanyak nilai *k,* dan menentukan kelas dokumen *testing*. Alur proses Testing Data ditunjukkan pada Gambar 4.7 dan Gambar 4.8.



Gambar 4.7 Alur Proses *Testing* Data



Gambar 4.8 Alur Proses Testing Data (Lanjutan)

4.3.1 Alur Proses *Case Folding*

Case folding merupakan alur proses pertama pada *pre-processing* teks. Pada alur ini, seluruh karakter huruf pada dokumen akan diubah menjadi karakter huruf kecil. Tujuannya adalah untuk dapat memudahkan pengolahan teks pada proses berikutnya. Alur proses *case folding* dapat dilihat pada Gambar 4.3.

4.3.2 Alur Proses Tokenisasi

Tokenisasi merupakan tahap dalam *text pre-processing.* Tokenisasi dipakai pada *testing* data yang berguna untuk memecah kalimat dalam dokumen *testing* menjadi satuan kata. Selain itu juga karakter angka dan tanda baca akan dihilangkan dari dokumen pada tahap ini. Alur proses Tokenisasi dapat dilihat pada Gambar 4.4.

4.3.3 Alur Proses *Stopword Removal*

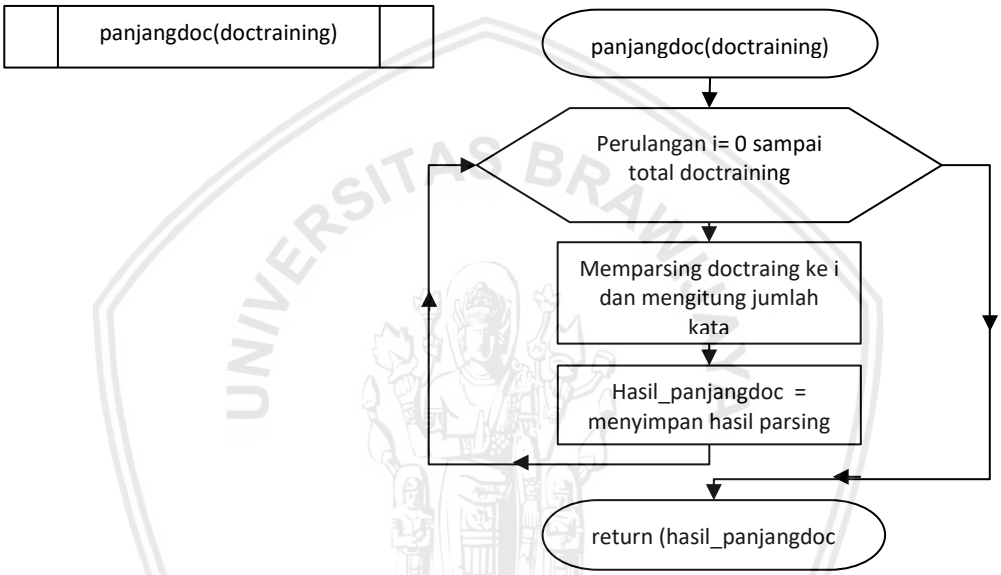
Stopword removal merupakan salah satu tahap dalam *text pre-processing*. *Stopword removal* dipakai pada proses *testing* data yang berguna untuk menghapus kata-kata yang tidak penting pada dokumen *testing*. Alur proses *stopword removal* dapat dilihat pada Gambar 4.5.

4.3.4 Alur Proses *Stemming*

Stemming merupakan tahap terakhir dari *pre-processing* teks. Proses *stemming* dipakai pada *training* data yang berguna untuk mengubah kata yang berimbuhan menjadi kata dasarnya. Pada proses *stemming* menggunakan *library* pendukung yaitu sastrawi *stemmer.* Alur proses *stemming* dapat dilihat pada Gambar 4.6.

4.3.5 Alur Proses Panjang Dokumen

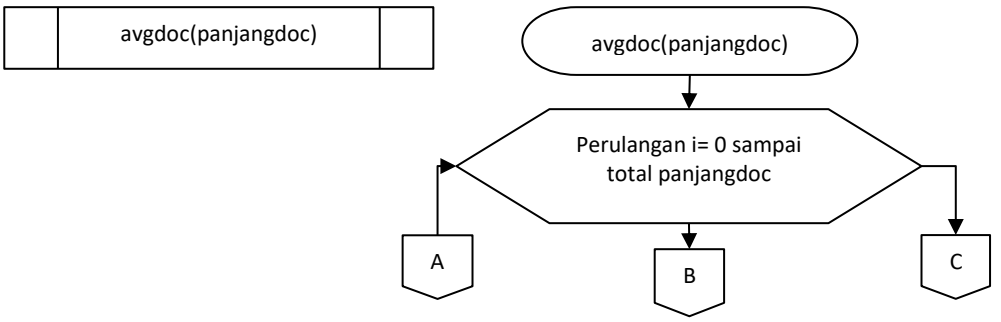
Tahap ini digunakan untuk menghitung panjang dokumen tiap dokumen *training.* Alur proses panjang dokumen dlihat pada Gambar 4.9.



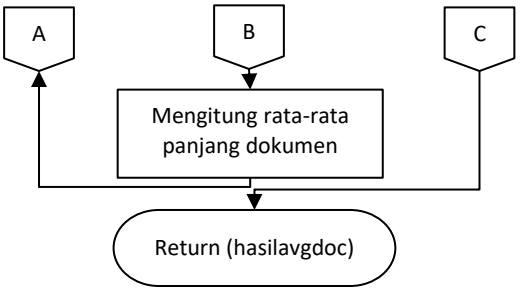
Gambar 4.9 Alur Proses Panjang Dokumen

4.3.6 Alur Proses Rata-Rata Panjang Dokumen

Tahap ini digunakan untuk menghitung rata-rata panjang seluruh dokumen *training.* Alur proses rata-rata panjang dokumen dlihat pada Gambar 4.10 dan Gambar 4.11.



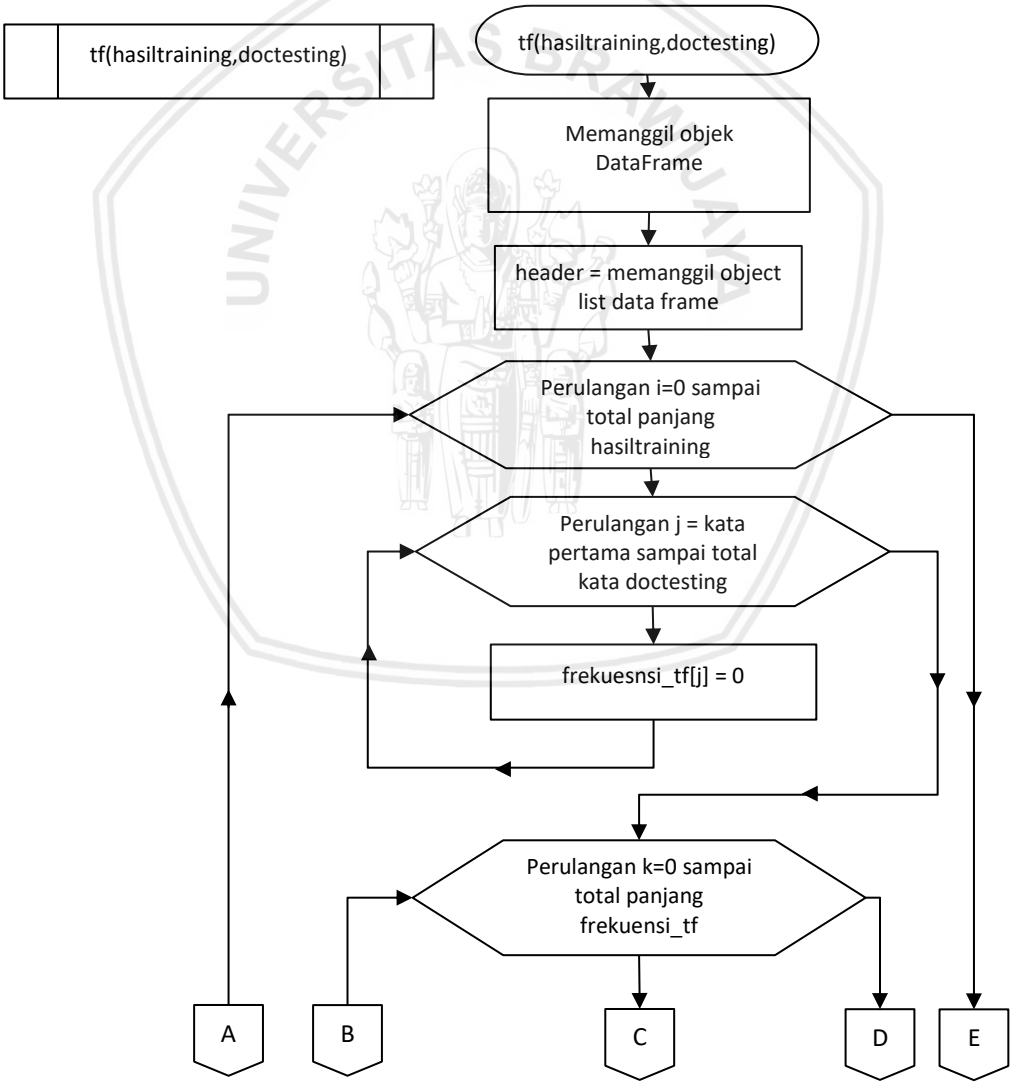
Gambar 4.10 Alur Proses Rata-Rata Panjang Dokumen



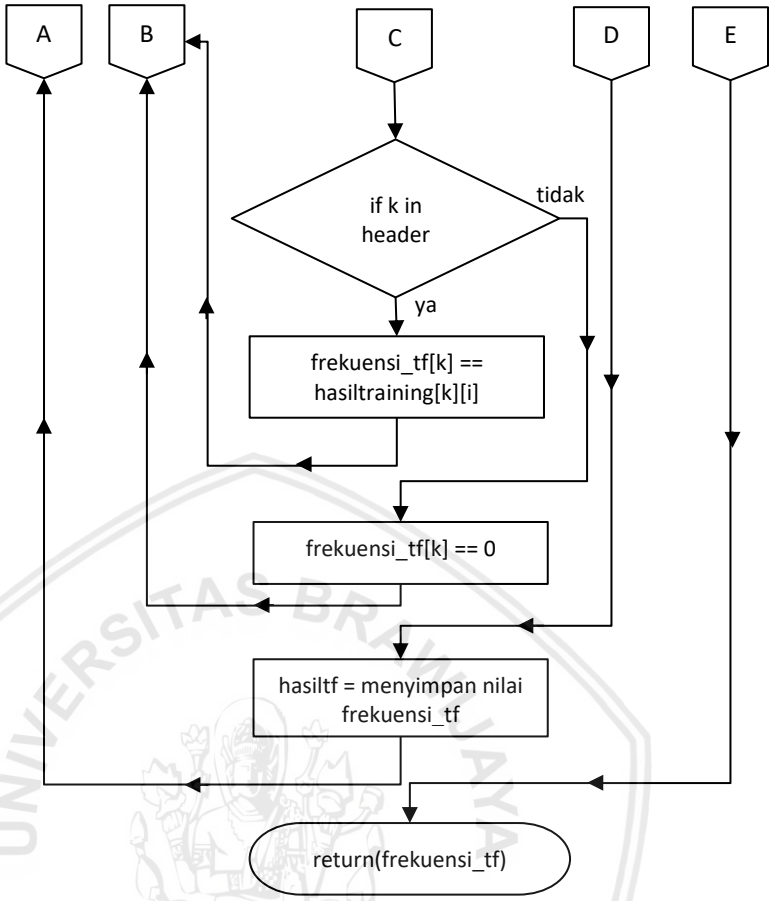
Gambar 4.11 Alur Proses Rata-Rata Panjang Dokumen (Lanjutan)

4.3.7 Alur Proses TF

Hitung nilai TF merupakan proses pertama pada metode BM25. Dalam perhitungan ini, kata yang sering muncul memiliki frekuensi yang tinggi pada dokumen. Alur proses TF ditunjukkan pada Gambar 4.12 dan Gambar 4.13.



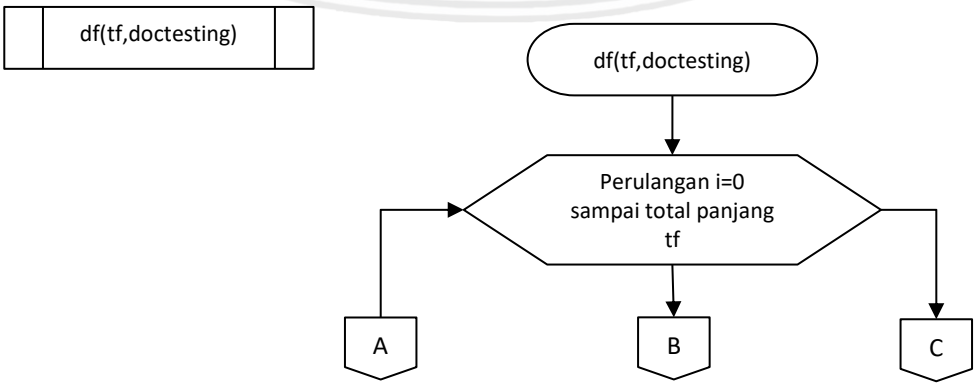
Gambar 4.12 Alur Proses TF



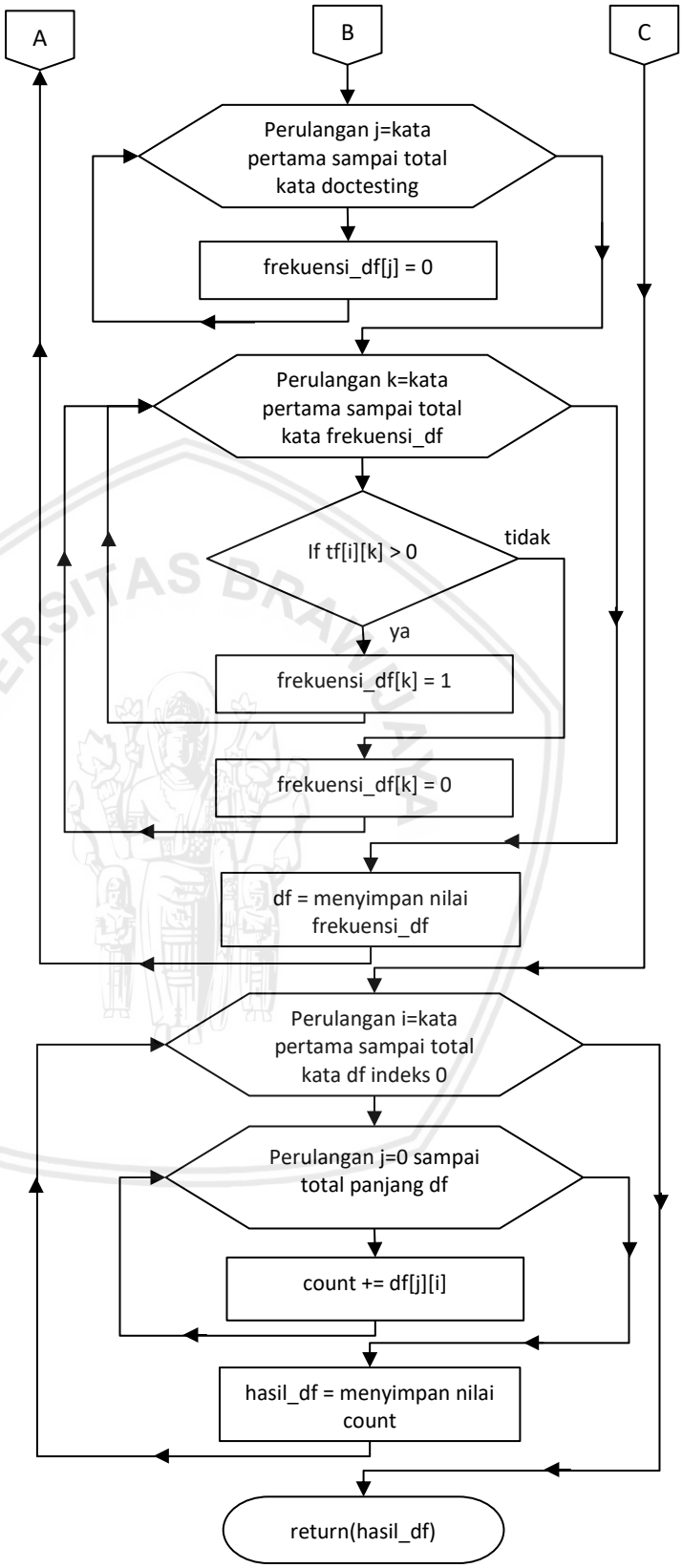
Gambar 4.13 Alur Proses TF (Lanjutan)

4.3.8 Alur Proses DF

Hitung nilai DF merupakan langkah setelah menghitung nilai TF. DF adalah jumlah dokumen *training* yang muncul term *query*. Alur proses DF ditunjukkan pada Gambar 4.14 dan Gambar 4.15.



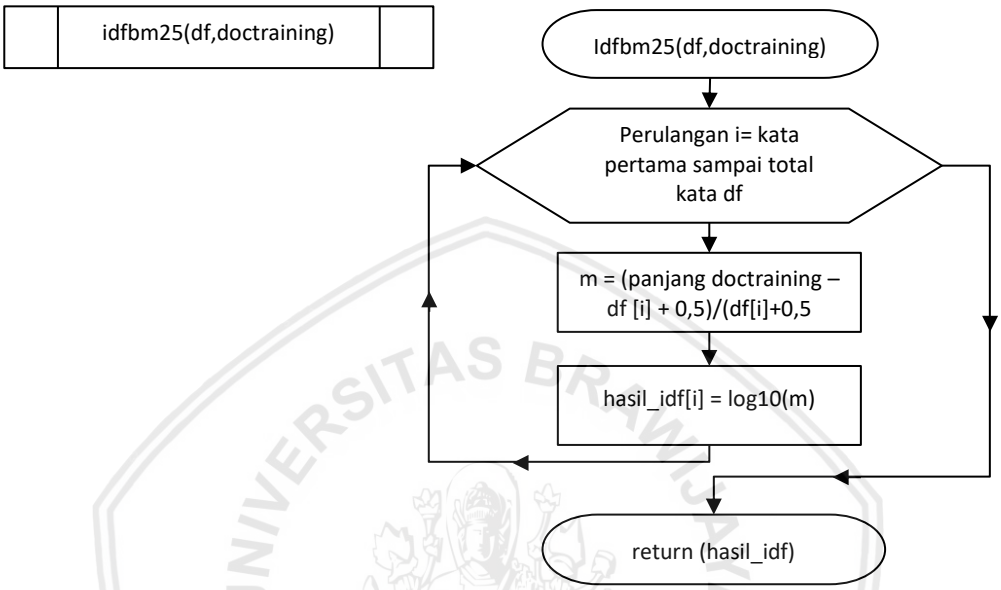
Gambar 4.14 Alur Proses DF



Gambar 4.15 Alur Proses DF (Lanjutan)

4.3.9 Alur Proses IDF

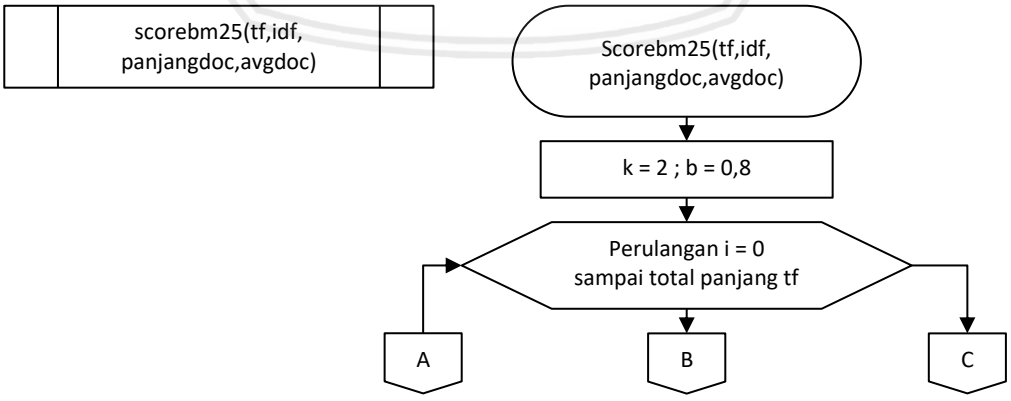
Hitung nilai IDF merupakan langkah setelah hitung nilai DF. Dalam perhitungan ini, kata yang sering muncul pada dokumen *training* memiliki nilai IDF yang kecil sedangkan yang jarang mucul di dokumen *training* memiliki nilai IDF besar. Alur proses IDF ditunjukkan pada Gambar 4.16.



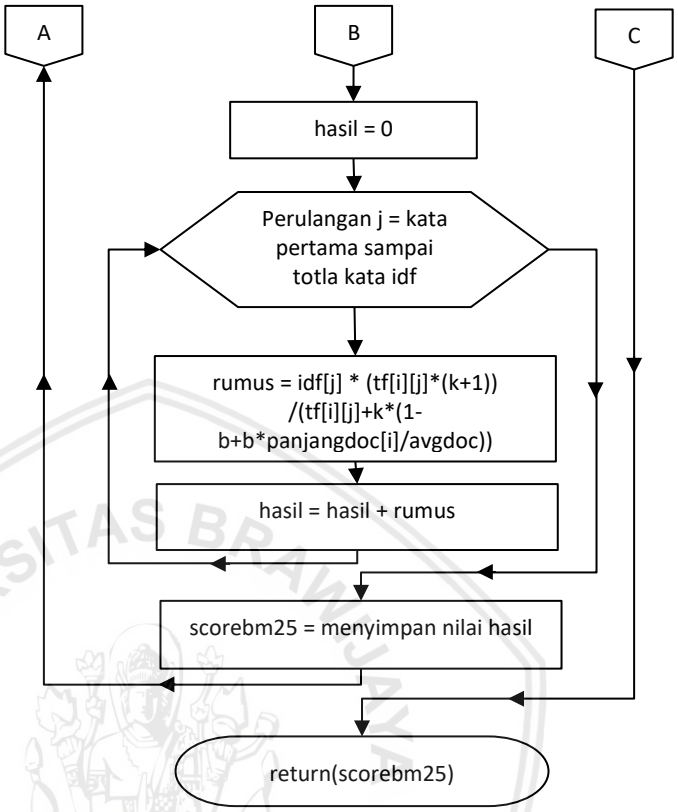
Gambar 4.16 Alur Proses IDF

4.3.10 Alur Proses *Score* BM25

Hitung *score* BM25 merupakan langkah terakhir dalam metode BM25. Proses tersebut merupakan proses pemeringkatan hasil dari kecocokan dokumen-dokumen, berdasarkan *query* yang dicari. Proses *score* BM25 ditunjukkan pada Gambar 4.17 dan Gambar 4.18.



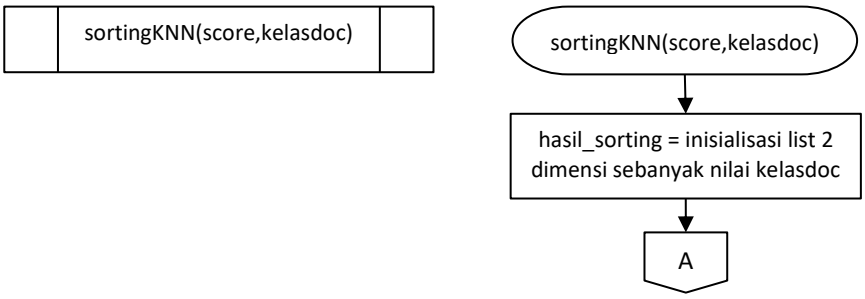
Gambar 4.17 Alur Proses *Score* BM25



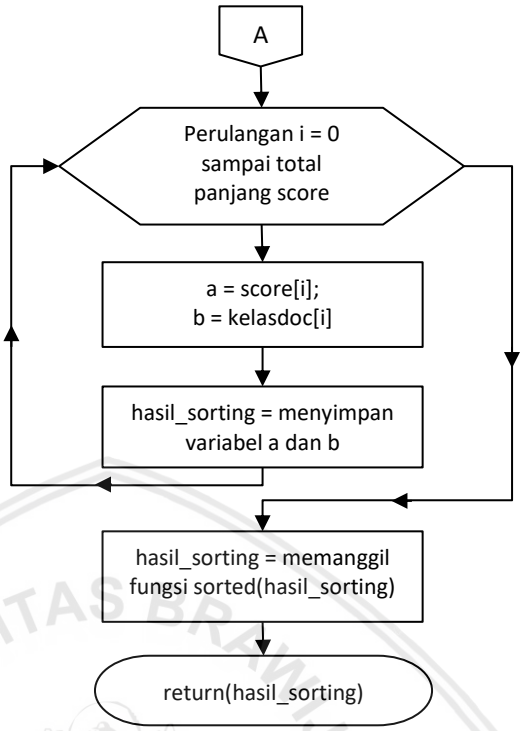
Gambar 4.18 Alur Proses *Score* BM25 (Lanjutan)

4.3.11 Alur Proses Pengurutan *Score* BM25

Langkah pertama yang dilakukan pada proses K-Nearest Neighbor adalah mengurutkan. Proses pengurutan dilakukan dari nilai yang terbesar sampai yang nilai yang terkecil pada *score* BM25. Alur proses pengurutan *score* BM25 ditunjukkan pada Gambar 4.19 dan Gambar 4.20.



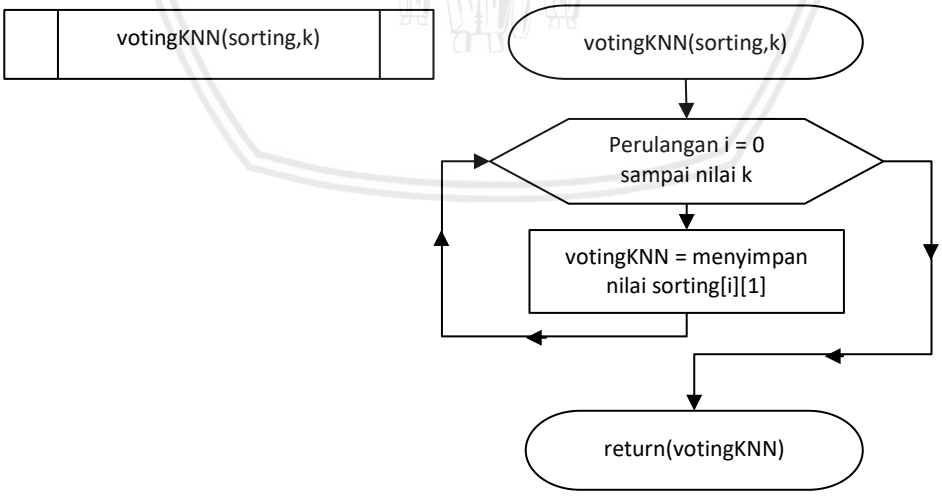
Gambar 4.19 Alur Pengurutan *Score* BM25



Gambar 4.20 Alur Pengurutan *Score* BM25 (Lanjutan)

4.3.12 Alur Proses Seleksi Sebanyak Nilai *K*

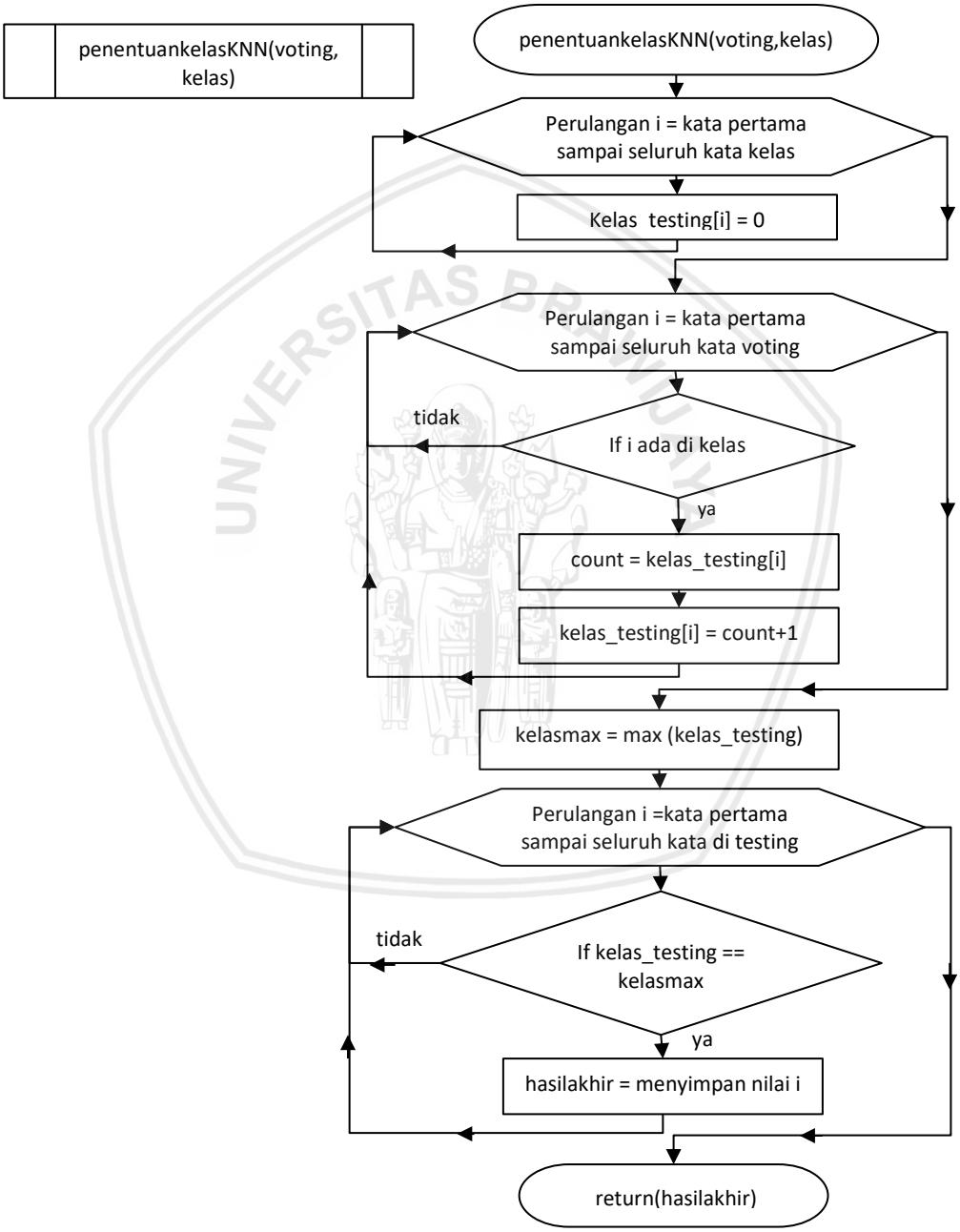
Langkah berikutnya adalah menyeleksi *score* BM25 tertinggi sebanyak nilai *k*. Proses tersebut adalah prose pengambil *score* BM25 yang paling tinggi sejumlah nilai *k.* Alur proses seleksi sebanyak nilai *k* ditunjukkan pada Gambar 4.21.



Gambar 4.21 Alur Proses Seleksi Sebanyak Nilai *K*

4.3.13 Alur Proses Penentuan Kelas Berdasarkan Hasil Seleksi Nilai *K*

Penentuan kelas berdasarkan hasil seleksi nilai *k* merupakan tahap terakhir dalam proses klasifikasi *K-Nearest Neighbor* (KNN). Pada tahap ini, akan dipilih kelas dokumen *testing* berdasarkan banyak kelas yang muncul pada tahap seleksi nilai *k*. Kelas yang paling banyak muncul pada tahap seleksi nilai *k* maka dijadikan kelas pada dokumen *testing*. Alur proses penentuan kelas berdasarkan hasil seleksi nilai *k* ditunjukkan pada Gambar 4.22.



Gambar 4.22 Alur Proses Penentuan Kelas Berdasarkan Hasil Seleksi *K*

4.4 Manualisasi

Pada tahap ini, dokumen abstrak skripsi yang akan dilakukan klasifikasi berdasarkan fokus penelitian di bidang Komputasi Cerdas menggunakan algoritme *K-Nearest Neighbor* (KNN). Dokumen-dokumen tersebut diambil dari jurnal J-PTIIK dan kemudian dibagi menjadi dua jenis yaitu data *training* dan data *testing*. Data *training* adalah dokumen yang telah memiliki kelas dokumen yang jelas. Sementara data *testing* adalah dokumen akan ditentukan kelasnya menggunakan algoritme KNN. Dalam manualisasi berikut sampel data yang digunakan berupa judul dan nantinya data yang digunakan sebagai penelitian adalah judul dan abstrak skripsi. Sampel data *training* dan data *testing* untuk manualisasi dapat dilihat pada Tabel 4.1 dan Tabel 4.2.

Tabel 4.1 Sampel Data *Training*

Dokumen	Isi Dokumen	Kelas
D1	Sistem Pakar Klasifikasi Permasalahan Berdasar AUM Menggunakan FIS Tsukamoto	SP
D2	Penggunaan Ciri Geometric Invariant Moment pada pengenalan tanda tangan	PCD
D3	Pemodelan Sistem Pakar Diagnosa Penyakit Tanaman Apel Manalagi Dengan Metode Backward Chaining Menggunakan Certainty Factor	SP
D4	Sistem Rekomendasi Pemilihan Sekolah Menengah Atas Sederajat Kota Malang Menggunakan Metode Ahp electre dan Topsis	SPK
D5	Sistem Pendukung Keputusan Penentuan Tingkat Keparahan Autis Menggunakan Metode Fuzzy K-Nearest Neighbor	SPK
D6	Klasifikasi Teks Bahasa Indonesia pada Dokumen Pengaduan Sambat Online menggunakan Metode K-Nearest Neighbor dan Chi-Square	TM
D7	Peringkasan Teks Otomatis Pada Artikel Berita Kesehatan Menggunakan K-Nearest Neighbor Berbasis Fitur Statistik	TM
D8	Sistem Monitoring Cairan Infus Terpusat Menggunakan Pengolahan Citra Digital	PCD
D9	Implementasi Metode Improved K-Means untuk Mengelompokkan Titik Panas Bumi	DM
D10	Implementasi Algoritme Modified K-Nearest Neighbor untuk Klasifikasi Penyakit Demam	DM

Tabel 4.2 Sampel Data *Testing*

Dokumen	Isi Dokumen
Q1	Sistem Pakar Diagnosa Penyakit Pada Tanaman Bawang Putih

4.4.1 *Case Folding*

Case folding merupakan alur proses pertama pada *pre-processing* teks. Pada alur ini, seluruh karakter huruf pada dokumen akan diubah menjadi karakter huruf kecil. Tujuannya adalah untuk dapat memudahkan pengolahan teks pada proses berikutnya. Hasil manualisasi *case folding* dapat dilihat pada Tabel 4.3 dan Tabel 4.4.

Tabel 4.3 *Case Folding* Pada Data *Training*

Dok	Sebelum *Case Folding*	Sesudah *Case Folding*
D1	Sistem Pakar Klasifikasi Permasalahan Berdasar AUM Menggunakan FIS Tsukamoto	sistem pakar klasifikasi permasalahan berdasar aum menggunakan fis tsukamoto
D2	Penggunaan Ciri Geometric Invariant Moment pada pengenalan tanda tangan	penggunaan ciri geometric invariant moment pada pengenalan tanda tangan
D3	Pemodelan Sistem Pakar Diagnosa Penyakit Tanaman Apel Manalagi Dengan Metode Backward Chaining Menggunakan Certainty Factor	pemodelan sistem pakar diagnosa penyakit tanaman apel manalagi dengan metode backward chaining menggunakan certainty factor
D4	Sistem Rekomendasi Pemilihan Sekolah Menengah Atas Sederajat Kota Malang Menggunakan Metode Ahp electre dan Topsis	sistem rekomendasi pemilihan sekolah menengah atas sederajat kota malang menggunakan metode ahp electre dan topsis
D5	Sistem Pendukung Keputusan Penentuan Tingkat Keparahan Autis Menggunakan Metode Fuzzy K-Nearest Neighbor	sistem pendukung keputusan penentuan tingkat keparahan autis menggunakan metode fuzzy k-nearest neighbor
D6	Klasifikasi Teks Bahasa Indonesia pada Dokumen Pengaduan Sambat Online menggunakan Metode K-Nearest Neighbor dan Chi-Square	klasifikasi teks bahasa indonesia pada dokumen pengaduan sambat online menggunakan metode k-nearest neighbor dan chi-square
D7	Peringkasan Teks Otomatis Pada Artikel Berita Kesehatan Menggunakan K-Nearest Neighbor Berbasis Fitur Statistik	peringkasan teks otomatis pada artikel berita kesehatan menggunakan k-nearest neighbor berbasis fitur statistik
D8	Sistem Monitoring Cairan Infus Terpusat Menggunakan Pengolahan Citra Digital	sistem monitoring cairan infus terpusat menggunakan pengolahan citra digital
D9	Implementasi Metode Improved K-Means untuk Mengelompokkan Titik Panas Bumi	implementasi metode improved k-means untuk mengelompokkan titik panas bumi
D10	Implementasi Algoritme Modified K-Nearest Neighbor untuk Klasifikasi Penyakit Demam	implementasi algoritme modified k-nearest neighbor untuk klasifikasi penyakit demam

Tabel 4.4 *Case Folding* Pada Data *Testing*

Dok	Sebelum *Case Folding*	Sesudah *Case Folding*
Q1	Sistem Pakar Diagnosa Penyakit Pada Tanaman Bawang Putih	sistem pakar diagnosa penyakit pada tanaman bawang putih

4.4.2 *Tokenization*

Tokenisasi merupakan tahap dalam *text pre-processing.* Tokenisasi dipakai pada *testing* data yang berguna untuk memecah kalimat dalam dokumen *testing* menjadi satuan kata. Selain itu juga karakter angka dan tanda baca akan dihilangkan dari dokumen pada tahap ini. Hasil manualisasi *tokenization* dapat dilihat pada Tabel 4.5, Tabel 4.6, dan Tabel 4.7.

Tabel 4.5 *Tokenization* Pada Data *Training*

Dok	Sebelum *Tokenization*	Sesudah *Tokenization*
D1	sistem pakar klasifikasi permasalahan berdasar aum menggunakan fis tsukamoto	[‘sistem’, ‘pakar’, ‘klasifikasi’, ‘permasalahan’, ’berdasar’, ‘aum’, ‘menggunakan’, ‘fis’, ‘tsukamoto’]
D2	penggunaan ciri geometric invariant moment pada pengenalan tanda tangan	[‘penggunaan’, ‘ciri’, ‘geometric’, ‘invariant’, ‘moment’, ‘pada’, ‘pengenalan’, ‘tanda’, ‘tangan’]
D3	pemodelan sistem pakar diagnosa penyakit tanaman apel manalagi dengan metode backward chaining menggunakan certainty factor	[‘pemodelan’, ‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘tanaman’, ‘apel’, ‘manalagi’, ‘dengan’, ‘metode’, ‘backward’, ‘chaining’, ‘menggunakan’, ‘certainty’, ‘factor’]
D4	sistem rekomendasi pemilihan sekolah menengah atas sederajat kota malang menggunakan metode ahp electre dan topsis	[‘sistem’, ‘rekomendasi’, ‘pemilihan’, ‘sekolah’, ‘menengah’, ‘atas’, ‘sederajat’, ‘kota’, ‘malang’, ‘menggunakan’, ‘metode’, ‘ahp’, ‘electre’, ‘dan’, ‘topsis’]
D5	sistem pendukung keputusan penentuan tingkat keparahan autis menggunakan metode fuzzy k-nearest neighbor	[‘sistem’, ‘pendukung’, ‘keputusan’, ‘penentuan’, ‘tingkat’, ‘keparahan’, ‘autis’, ‘menggunakan’, ‘metode’, ‘fuzzy’, ‘k’, ‘nearest’, ‘neighbor’]
D6	klasifikasi teks bahasa indonesia pada dokumen pengaduan sambat online menggunakan metode k-nearest neighbor dan chi-square	[‘klasifikasi’, ‘teks’, ‘bahasa’, ‘indonesia’, ‘pada’, ‘dokumen’, ‘pengaduan’, ‘sambat’, ‘online’, ‘menggunakan’, ‘metode’, ‘k’, ‘nearest’, ‘neighbor’, ‘dan’, ‘chi’, ‘square’]
D7	peringkasan teks otomatis pada artikel berita kesehatan menggunakan k-nearest neighbor berbasis fitur statistik	[‘peringkasan’, ‘teks’, ‘otomatis’, ‘pada’, ‘artikel’, ‘berita’, ‘kesehatan’, ‘menggunakan’, ‘k’, ‘nearest’, ‘neighbor’, ‘berbasis’, ‘fitur’, ‘statistik’]
D8	sistem monitoring cairan infus terpusat menggunakan pengolahan citra digital	[‘sistem’, ‘monitoring’, ‘cairan’, ‘infus’, ‘terpusat’, ‘menggunakan’, ‘pengolahan’, ‘citra’, ‘digital’]

Tabel 4.6 *Tokenization* Pada Data *Training* (Lanjutan)

Dok	Sebelum *Tokenization*	Sesudah *Tokenization*
D9	implementasi metode improved k-means untuk mengelompokkan titik panas bumi	[‘implementasi’, ‘metode’, ‘improved’, ‘k’, ‘means’, ‘untuk’, ‘mengelompokkan’, ‘titik’, ‘panas’, ‘bumi’]
D10	implementasi algoritme modified k-nearest neighbor untuk klasifikasi penyakit demam	[‘implementasi’, ‘algoritme’, ‘modified’, ‘k’, ‘nearest’, ‘neighbor’, ‘untuk’, ‘klasifikasi’, ‘penyakit’, ‘demam’]

Tabel 4.7 *Tokenization* Pada Data *Testing*

Dok	Sebelum *Tokenization*	Sesudah *Tokenization*
Q1	Sistem Pakar Diagnosa Penyakit Pada Tanaman Bawang Putih	[‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘pada’, ‘tanaman’, ‘bawang’, ‘putih’]

4.4.3 *Stopword Removal*

Stopword removal merupakan salah satu tahap dalam *text pre-processing*. *Stopword removal* dipakai pada proses dokumen yang berguna untuk menghapus kata-kata yang tidak penting pada dokumen. *Stoplist* menggunakan *stoplist* Tala Hasil manualisasi *stopword removal* dapat dilihat pada Tabel 4.8, Tabel 4.9, dan Tabel 4.10.

Tabel 4.8 *Stopword Removal* Pada Data *Training*

Dok	Sebelum *Stopword Removal*	Sesudah *Stopword Removal*
D1	[‘sistem’, ‘pakar’, ‘klasifikasi’, ‘permasalahan’, ’berdasar’, ‘aum’, ‘menggunakan’, ‘fis’, ‘tsukamoto’]	[‘sistem’, ‘pakar’, ‘klasifikasi’, ‘permasalahan’, ’berdasar’, ‘aum’, ‘fis’, ‘tsukamoto’]
D2	[‘penggunaan’, ‘ciri’, ‘geometric’, ‘invariant’, ‘moment’, ‘pada’, ‘pengenalan’, ‘tanda’, ‘tangan’]	[‘penggunaan’, ‘ciri’, ‘geometric’, ‘invariant’, ‘moment’, ‘pengenalan’, ‘tanda’, ‘tangan’]
D3	[‘pemodelan’, ‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘tanaman’, ‘apel’, ‘manalagi’, ‘dengan’, ‘metode’, ‘backward’, ‘chaining’, ‘menggunakan’, ‘certainty’, ‘factor’]	[‘pemodelan’, ‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘tanaman’, ‘apel’, ‘manalagi’, ‘metode’, ‘backward’, ‘chaining’, ‘certainty’, ‘factor’]
D4	[‘sistem’, ‘rekomendasi’, ‘pemilihan’, ‘sekolah’, ‘menengah’, ‘sederajat’, ‘kota’, ‘malang’, ‘menggunakan’, ‘metode’, ‘ahp’, ‘electre’, ‘dan’, ‘topsis’]	[‘sistem’, ‘rekomendasi’, ‘pemilihan’, ‘sekolah’, ‘menengah’, ‘sederajat’, ‘kota’, ‘malang’, ‘metode’, ‘ahp’, ‘electre’, ‘topsis’]
D5	[‘sistem’, ‘pendukung’, ‘keputusan’, ‘penentuan’, ‘tingkat’, ‘keparahan’, ‘autis’, ‘menggunakan’, ‘metode’, ‘fuzzy’, ‘k’, ‘nearest’, ‘neighbor’]	[‘sistem’, ‘pendukung’, ‘keputusan’, ‘penentuan’, ‘tingkat’, ‘keparahan’, ‘autis’, ‘metode’, ‘fuzzy’, ‘k’, ‘nearest’, ‘neighbor’]

Tabel 4.9 *Stopword Removal* Pada Data *Training* (Lanjutan)

Dok	Sebelum *Stopword Removal*	Sesudah *Stopword Removal*
D6	[‘klasifikasi’, ‘teks’, ‘bahasa’, ‘indonesia’, ‘pada’, ‘dokumen’, ‘pengaduan’, ‘sambat’, ‘online’, ‘menggunakan’, ‘metode’, ‘k’, ‘nearest’, ‘neighbor’, ‘dan’, ‘chi’, ‘square’]	[‘klasifikasi’, ‘teks’, ‘bahasa’, ‘indonesia’, ‘dokumen’, ‘pengaduan’, ‘sambat’, ‘online’, ‘metode’, ‘k’, ‘nearest’, ‘neighbor’, ‘chi’, ‘square’]
D7	[‘peringkasan’, ‘teks’, ‘otomatis’, ‘pada’, ‘artikel’, ‘berita’, ‘kesehatan’, ‘menggunakan’, ‘k’, ‘nearest’, ‘neighbor’, ‘berbasis’, ‘fitur’, ‘statistik’]	[‘peringkasan’, ‘teks’, ‘otomatis’, ‘artikel’, ‘berita’, ‘kesehatan’, ‘k’, ‘nearest’, ‘neighbor’, ‘berbasis’, ‘fitur’, ‘statistik’]
D8	[‘sistem’, ‘monitoring’, ‘cairan’, ‘infus’, ‘terpusat’, ‘menggunakan’, ‘pengolahan’, ‘citra’, ‘digital’]	[‘sistem’, ‘monitoring’, ‘cairan’, ‘infus’, ‘terpusat’, ‘pengolahan’, ‘citra’, ‘digital’]
D9	[‘implementasi’, ‘metode’, ‘improved’, ‘k’, ‘means’, ‘untuk’, ‘mengelompokkan’, ‘titik’, ‘panas’, ‘bumi’]	[‘implementasi’, ‘metode’, ‘improved’, ‘k’, ‘means’, ‘mengelompokkan’, ‘titik’, ‘panas’, ‘bumi’]
D10	[‘implementasi’, ‘algoritme’, ‘modified’, ‘k’, ‘nearest’, ‘neighbor’, ‘untuk’, ‘klasifikasi’, ‘penyakit’, ‘demam’]	[‘implementasi’, ‘algoritme’, ‘modified’, ‘k’, ‘nearest’, ‘neighbor’, ‘klasifikasi’, ‘penyakit’, ‘demam’]

Tabel 4.10 *Stopword Removal* Pada Data *Testing*

Dok	Sebelum *Stopword Removal*	Sesudah *Stopword Removal*
Q1	[‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘pada’, ‘tanaman’, ‘bawang’, ‘putih’]	[‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘tanaman’, ‘bawang’, ‘putih’]

4.4.4 *Stemming*

Stemming merupakan tahap terakhir dari *pre-processing* teks. Proses *stemming* dipakai pada dokuemn yang berguna untuk mengubah kata yang berimbuhan menjadi kata dasarnya. Hasil manualisasi *stemming* dapat dilihat pada Tabel 4.11, Tabel 4.12, dan Tabel 4.13.

Tabel 4.11 *Stemming* Pada Data *Training*

Dok	Sebelum *Stemming*	Sesudah *Stemming*
D1	[‘sistem’, ‘pakar’, ‘klasifikasi’, ‘permasalahan’, ’berdasar’, ‘aum’, ‘fis’, ‘tsukamoto’]	[‘sistem’, ‘pakar’, ‘klasifikasi’, ‘masalah’, ‘dasar’, ‘aum’, ‘fis’, ‘tsukamoto’]
D2	[‘penggunaan’, ‘ciri’, ‘geometric’, ‘invariant’, ‘moment’, ‘pengenalan’, ‘tanda’, ‘tangan’]	[‘guna’, ‘ciri’, ‘geometric’, ‘invariant’, ‘moment’, ‘kenal’, ‘tanda’, ‘tangan’]
D3	[‘pemodelan’, ‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘tanaman’, ‘apel’, ‘manalagi’, ‘metode’, ‘backward’, ‘chaining’, ‘certainty’, ‘factor’]	[‘model’, ‘sistem’, ‘pakar’, ‘diagnosa’, ‘sakit’, ‘tanam’, ‘apel’, ‘manalagi’, ‘metode’, ‘backward’, ‘chaining’, ‘certainty’, ‘factor’]

Tabel 4.12 *Stemming* Pada Data *Training* (Lanjutan)

Dok	Sebelum *Stemming*	Sesudah *Stemming*
D4	[‘sistem’, ‘rekomendasi’, ‘pemilihan’, ‘sekolah’, ‘menengah’, ‘atas’, ‘sederajat’, ‘kota’, ‘malang’, ‘metode’, ‘ahp’, ‘electre’, ‘topsis’]	[‘sistem’, ‘rekomendasi’, ‘pilih’, ‘sekolah’, ‘tengah’, ‘atas’, ‘derajat’, ‘kota’, ‘malang’, ‘metode’, ‘ahp’, ‘electre’, ‘topsis’]
D5	[‘sistem’, ‘pendukung’, ‘keputusan’, ‘penentuan’, ‘tingkat’, ‘keparahan’, ‘autis’, ‘metode’, ‘fuzzy’, ‘k’, ‘nearest’, ‘neighbor’]	[‘sistem’, ‘dukung’, ‘putus’, ‘tentu’, ‘tingkat’, ‘parah’, ‘autis’, ‘metode’, ‘fuzzy’, ‘k’, ‘nearest’, ‘neighbor’]
D6	[‘klasifikasi’, ‘teks’, ‘bahasa’, ‘indonesia’, ‘dokumen’, ‘pengaduan’, ‘sambat’, ‘online’, ‘metode’, ‘k’, ‘nearest’, ‘neighbor’, ‘chi’, ‘square’]	[‘klasifikasi’, ‘teks’, ‘bahasa’, ‘indonesia’, ‘dokumen’, ‘adu’, ‘sambat’, ‘online’, ‘metode’, ‘k’, ‘nearest’, ‘neighbor’, ‘chi’, ‘square’]
D7	[‘peringkasan’, ‘teks’, ‘otomatis’, ‘artikel’, ‘berita’, ‘kesehatan’, ‘k’, ‘nearest’, ‘neighbor’, ‘berbasis’, ‘fitur’, ‘statistik’]	[‘ringkas’, ‘teks’, ‘otomatis’, ‘artikel’, ‘berita’, ‘sehat’, ‘k’, ‘nearest’, ‘neighbor’, ‘basis’, ‘fitur’, ‘statistik’]
D8	[‘sistem’, ‘monitoring’, ‘cairan’, ‘infus’, ‘terpusat’, ‘pengolahan’, ‘citra’, ‘digital’]	[‘sistem’, ‘monitoring’, ‘cair’, ‘infus’, ‘pusat’, ‘olah’, ‘citra’, ‘digital’]
D9	[‘implementasi’, ‘metode’, ‘improved’, ‘k’, ‘means’, ‘mengelompokkan’, ‘titik’, ‘panas’, ‘bumi’]	[‘implementasi’, ‘metode’, ‘improved’, ‘k’, ‘means’, ‘kelompok’, ‘titik’, ‘panas’, ‘bumi’]
D10	[‘implementasi’, ‘algoritme’, ‘modified’, ‘k’, ‘nearest’, ‘neighbor’, ‘klasifikasi’, ‘penyakit’, ‘demam’]	[‘implementasi’, ‘algoritme’, ‘modified’, ‘k’, ‘nearest’, ‘neighbor’, ‘klasifikasi’, ‘sakit’, ‘demam’]

Tabel 4.13 *Stemming* Pada Data *Testing*

Dok	Sebelum *Stemming*	Sesudah *Stemming*
Q1	[‘sistem’, ‘pakar’, ‘diagnosa’, ‘penyakit’, ‘pada’, ‘tanaman’, ‘bawang’, ‘putih’]	[‘sistem’, ‘pakar’, ‘diagnosa’, ‘sakit’, ‘tanam’, ‘bawang’, ‘putih’]

4.4.5 Hitung Nilai TF

Setelah *pre-processing text* berakhir, maka dilanjutkan dengan mencari nilai TF pada tiap dokumen *training.* TF adalah frekuensi banyak *term* yang sama dalam satu dokumen. *Term* yang digunakan adalah *term* dari data *testing*. Hasil hitung manualisasi nilai TF dapat dilihat pada Tabel 4.14 dan Tabel 4.15.

Tabel 4.14 Hitung Nilai TF

Term Query	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
sistem	1	0	1	1	1	0	0	1	0	0
pakar	1	0	1	0	0	0	0	0	0	0
diagnosa	0	0	1	0	0	0	0	0	0	0

Term Query	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
sakit	0	0	1	0	0	0	0	0	0	1

Tabel 4.15 Hitung Nilai TF (Lanjutan)

Term Query	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
tanam	0	0	1	0	0	0	0	0	0	0
bawang	0	0	0	0	0	0	0	0	0	0
putih	0	0	0	0	0	0	0	0	0	0

4.4.6 Hitung Nilai DF dan IDF

Setelah menghitung nilai TF, dilanjutkan dengan menghitung nilai IDF. Pada tahap hitung nilai IDF, terdapat pula menghitung nilai DF. Nilai DF adalah frekuensi banyaknya *term* yang sama muncul di beberapa dokumen, sedangkan IDF adalah nilai kebalikan dari DF. Persamaan nilai IDF dapat dilihat pada Persamaan 2.2. Hasil hitung manualisasi nilai IDF dapat dilihat pada Tabel 4.16. Contoh perhitungan nilai IDF pada "*term query* pakar" sebagai berikut.

$$idf(q_{pakar}) = \log(\frac{N - df(q_{pakar}) + 0{,}5}{df(q_{pakar}) + 0{,}5})$$

$$idf(q_{pakar}) = \log\left(\frac{10 - 2 + 0{,}5}{2 + 0{,}5}\right) = log\frac{8{,}5}{2{,}5} = log3{,}4 = \ 0{,}531$$

Tabel 4.16 Hitung Nilai DF dan IDF

Term Query	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	DF	IDF
sistem	1	0	1	1	1	0	0	1	0	0	5	0
pakar	1	0	1	0	0	0	0	0	0	0	2	0,531
diagnosa	0	0	1	0	0	0	0	0	0	0	1	0,802
sakit	0	0	1	0	0	0	0	0	0	1	2	0,531
tanam	0	0	1	0	0	0	0	0	0	0	1	0,802
bawang	0	0	0	0	0	0	0	0	0	0	0	1,322
putih	0	0	0	0	0	0	0	0	0	0	0	1,322

4.4.7 Hitung *Score* BM25

Setelah mengihitung nilai IDF, dilanjutkan dengan menghitung *score* BM25 tiap dokumen *training*. Pada perhitungan *score* BM25, dilakukan pula perhitungan panjang dokumen *training* dan rata-rata panjang dokumen *training*. Perhitungan *score* BM25 dapat dilihat pada Persamaan 2.1. Hasil hitung manualisasi panjang

dokumen, rata-rata panjang dokumen, dan *score* BM25 tiap dokumen *training* dapat dilihat pada Tabel 4.17, Tabel 4.18, dan Tabel 4.19.

Tabel 4.17 Panjang Dokumen *Training* dan Rata-Rata Panjang Dokumen *Training*

Keterangan	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
Panjang Dokumen	9	9	15	15	12	15	13	9	9	9
Rata-Rata	11,5									

Contoh perhitung *score* BM25 antara sebagai berikut.

$$BM25\ Score\ (Q1, D1) = \sum_{i=1}^{|q|} idf(Q_1).\frac{tf(Q_1, D1)\ .\ (k_1 + 1)}{tf(Q_1, D1) + k_i\ .\ (1 - b + b\ .\ \frac{|d|}{dl_{avg}})}$$

$$\begin{aligned} BM25\ Score\ (Q1, D1) &= 0 * \frac{1 * (2+1)}{1 + 2 * \left(1 - 0{,}8 + 0{,}8 * \frac{9}{11{,}5}\right)} + 0{,}531 \\ &* \frac{1 * (2+1)}{1 + 2 * \left(1 - 0{,}8 + 0{,}8 * \frac{9}{11{,}5}\right)} + 0{,}801 \\ &* \frac{0 * (2+1)}{0 + 2 * \left(1 - 0{,}8 + 0{,}8 * \frac{9}{11{,}5}\right)} + 0{,}531 \\ &* \frac{0 * (2+1)}{0 + 2 * \left(1 - 0{,}8 + 0{,}8 * \frac{9}{11{,}5}\right)} + 0{,}801 \\ &* \frac{0 * (2+1)}{0 + 2 * \left(1 - 0{,}8 + 0{,}8 * \frac{9}{11{,}5}\right)} + 1{,}322 \\ &* \frac{0 * (2+1)}{0 + 2 * \left(1 - 0{,}8 + 0{,}8 * \frac{9}{11{,}5}\right)} + 1{,}322 \\ &* \frac{0 * (2+1)}{0 + 2 * \left(1 - 0{,}8 + 0{,}8 * \frac{9}{11{,}5}\right)} = 0{,}60118107 \end{aligned}$$

Tabel 4.18 *Score* BM25 Tiap Dokumen *Training*

Dokumen *Training*	*Score* BM25	Kategori Kelas
D1	0,60118107	SP
D2	0	PCD
D3	2,293882224	SP
D4	0	SPK
D5	0	SPK
D6	0	TM

Tabel 4.19 *Score* BM25 Tiap Dokumen *Training* (Lanjutan)

Dokumen *Training*	*Score* BM25	Kategori Kelas
D7	0	TM
D8	0	PCD
D9	0	DM
D10	0,60118107	DM

4.4.8 Pengurutan dan Seleksi Sebanyak Nilai K

Score BM25 tiap dokumen *training* akan diurutkan mulai dari nilai terbesar sampai nilai terkecil. Nilai hasil *score* BM25 merupakan pemeringkatan dokumen *training*. Hasil manualisasi pengurutan *score* BM25 tiap dokumen *training* dapat dilihat pada Tabel 4.20.

Tabel 4.20 Pengurutan *Score* BM25

Dokumen *Training*	*Score* BM25	Kategori Kelas
D3	2,293882224	SP
D1	0,60118107	SP
D10	0,60118107	DM
D2	0	PCD
D4	0	SPK
D5	0	SPK
D6	0	TM
D7	0	TM
D8	0	PCD
D9	0	DM

Setelah dilakukan pengurutan dari terbesar sampai terkecil, maka dilakukan seleksi *score* BM25 beserta labelnya sebanyak nilai *k*. Hasil manualisasi seleksi *score* BM25 sebanyak nilai *k* dapat dilihat pada Tabel 4.21.

Tabel 4.21 Seleksi *Score* BM25 Sebanyak Nilai *K*

Dokumen *Training*	*Score* BM25	Kategori Kelas
D3	2,293882224	SP
D1	0,60118107	SP
D10	0,60118107	DM

4.4.9 Penentuan Kelas Dokumen *Testing* Berdasarkan Seleksi Nilai *K*

Pada tahap ini dilakukan proses pemilihan kelas dokumen untuk dokumen *testing* berdasarkan dokumen *training* yang terseleksi. Proses pemilihan kelas

dilakukan dengan cara dicari kelas pada dokumen *training* yang paling banyak muncul dan kelas dokumen yang paling banyak muncul akan dijadikan kelas pada dokumen *testing*. Hasil manualisasi penentuan kelas dokumen testing dapat dilihat pada Tabel 4.22.

Tabel 4.22 Penentuan Kelas Dokumen *Testing*

Dokumen *Training* Terseleksi	*Score* BM25	Kategori Kelas
D3	2,293882224	SP
D1	0,60118107	SP
D10	0,60118107	DM
Dokumen *Testing*	Kategori Kelas Terpilih	
Q1	SP	

4.5 Perancangan Evaluasi Sistem

Evaluasi sistem digunakan untuk mengukur kinerja dari algoritme BM25 dan *K-Nearest Neighbor*. Evaluasi sistem digunakan adalah dengan cara menghitung *cofusion matrix* tiap kelas, nilai *precision*, nilai *recall*, dan nilai *f-measure* dengan nilai *k* yang berbeda-beda. Perancangan evaluasi sistem dapat dilihat pada Tabel 4.23, Tabel 4.24, Tabel 4.25, Tabel 4.26, Tabel 4.27, dan Tabel 4.28.

Tabel 4.23 *Confusion Matrix* Kelas SP

		Nilai Prediksi	
		SP	Bukan SP
Nilai Sebenarnya	SP	*True Positive* (TP)	*False Negative* (FN)
	Bukan SP	*False Positive* (FP)	*True Negative* (TN)

Tabel 4.24 *Confusion Matrix* Kelas SPK

		Nilai Prediksi	
		SPK	Bukan SPK
Nilai Sebenarnya	SPK	*True Positive* (TP)	*False Negative* (FN)
	Bukan SPK	*False Positive* (FP)	*True Negative* (TN)

Tabel 4.25 *Confusion Matrix* Kelas DM

		Nilai Prediksi	
		DM	Bukan DM
Nilai Sebenarnya	DM	*True Positive* (TP)	*False Negative* (FN)
	Bukan DM	*False Positive* (FP)	*True Negative* (TN)

Tabel 4.26 *Confusion Matrix* Kelas TM

		Nilai Prediksi	
		TM	Bukan TM
Nilai Sebenarnya	TM	*True Positive* (TP)	*False Negative* (FN)
	Bukan TM	*False Positive* (FP)	*True Negative* (TN)

Tabel 4.27 *Confusion Matrix* Kelas PCD

		Nilai Prediksi	
		PCD	Bukan PCD
Nilai Sebenarnya	PCD	*True Positive* (TP)	*False Negative* (FN)
	Bukan PCD	*False Positive* (FP)	*True Negative* (TN)

Tabel 4.28 Perancangan Evaluasi Sistem

Nilai *k*	*Precision*	*Recall*	*F-measure*
2			
3			
5			
7			
9			
11			
13			
15			
20			
50			
100			
150			

4.6 Penarikan Kesimpulan

Penarikan kesimpulan digunakan untuk merangkum hasil yang diperoleh dari pengujian yang nantinya akan digunakan sebagai acuan untuk membuat saran. Kesimpulan dan saran digunakan untuk membantu pengembangan penelitian lebih lanjut.

4.7 Implementasi Sistem

Pada tahap ini dilakukan untuk menerapakan sebuah perancangan yang telah dibuat pada pada bab sebelumya. Beberapa proses yang akan diterapkan dalam bahasa pemrograman python diantaranya *pre-processing text*, BM25, dan klasifikasi *K-Nearest Neighbor*.

4.8 Implementasi *Pre-processing Text*

Pre-processing text adalah tahapan paling awal pada text dalam pembuatan sistem pada penelitian ini. *Pre-processing* bertujuan untuk mendapatkan fitur kata yang nantinya akan digunakan pada proses BM25 dan klasifikasi *K-Nearest Neighbor*. Tahapan *pre-processing* text terdiri *casefolding*, *tokenization*, *stopword removal*, dan *stemming*.

4.8.1 Implementasi *Casefolding*

Casefolding adalah tahapan paling awal dalam *pre-processing text*. *Case folding* digunakan untuk mengubah seluruh karakter huruf pada seluruh dokumen menjadi karakter huruf kecil. Implementasi *casefolding* ditunjukkan pada Kode Program 4.1.

```
Algoritme: casefolding
1  def casefolding(doc):
2      doc_lowercase = doc.lower()
3      return doc_lowercase
```

Kode Program 4.1 Implementasi *Casefolding*

Penjelasan potongan Kode Program 4.1 yaitu pada baris 2 merupakan proses pengubahan setiap karakter pada dokumen menjadi karakter huruf kecil.

4.8.2 Implementasi *Tokenization*

Tokenization adalah tahap yang dilakukan setelah *casefolding*. Tahap ini yaitu menghapus karakter tanda baca dan angka pada seluruh dokumen serta memecah kalimat pada dokumen menjadi satuan kata. Implementasi *Tokenization* ditunjukkan pada Kode Program 4.2.

```
Algoritme: tokenisasi
1  def tokenisasi(casefolding):
2      hasiltoken = []
3      hasil_token = re.sub('[^a-z]',' ',casefolding)
4      hasil_token = hasil_token.split(' ')
5      kosong =''
6      for i in hasil_token:
7          if i != kosong:
8              hasiltoken.append(i)
9      return hasiltoken
```

Kode Program 4.2 Implementasi *Tokenization*

Berikut ini penjelasan potongan Kode Program 4.2.

1. Pada baris 3 merupakan proses penggantian karakter tanda baca dan angka menjadi spasi.

2. Pada baris 4 merupakan proses pemecahan kalimat menjadi satuan kata.
3. Pada baris 6-8 merupakan proses perulangan untuk mengecek apabila ada himpunan kosong maka tidak akan disimpan pada variabel *hasiltoken*.

4.8.3 Implementasi *Stopword Removal*

Stopword adalah tahap yang dilakuakan setelah *tokenization*. Pada tahap ini dilakukan prose penghilangan kata yang tidak memiliki makna penting. Implementasi *stopword* ditunjukkan pada Kode Program 4.3

```
Algoritme: stopword
1  def stopword(token):
2      hasil_stopword = []
3      with open("stop_words.txt") as s:
4          stopword = s.read().replace('\n',' ')
5      stop_words = stopword.split(' ')
6      for i in token:
7          if i not in stop_words:
8              hasil_stopword.append(i)
9      return hasil_stopword
```

Kode Program 4.3 Implementasi *Stopword Removal*

Berikut ini penjelasan Kode Program 4.3.

1. Pada baris 3-4 merupakan proses pembaca berkas *stop_words.txt* dengan mengganti karakter "\n" menjadi spasi.
2. Pada baris 5 merupakan proses pemecahan isi dari berkas *stop_words.txt* menjadi satuan kata.
3. Pada proses 6-8 merupakan proses perulangan untuk mengecek apakah kata pada variabel token terdapat pada variabel *stop_words*. Jika kata tersebut tidak ada pada variabel *stop_words*, maka kata tersebut akan disimpan pada variabel *hasil_stopword*.

4.8.4 Implementasi *Stemming*

Stemming adalah tahap yang dilakukan setelah *stopword*. Pada tahap ini yaitu mengubah kata yang berimbuhan menjadi kata dasarnya. Pada implementasi ini menggunakan *library* Sastrawi. Implementasi *Stemming* ditunjukkan pada Kode Program 4.4.

```
Algoritme: stemming
1  def stemming(stopword):
2      hasil_stemming =[]
3      for i in range(len(stopword)):
4          factory = StemmerFactory()
5          stemmer = factory.create_stemmer()
6          stemming = stemmer.stem(stopword[i])
7          hasil_stemming.append(stemming)
8      return hasil_stemming
```

Kode Program 4.4 Implementasi *Stemming*

Berikut ini penjelasan potongan Kode Program 4.4.

1. Pada baris 4 merupakan pemanggilan fungsi *StemmerFactory.*

2. Pada baris 5 merupakan proses penggunaan *library* Sastrawi.
3. Pada baris 6 merupakan proses pengubahan kata berimbuhan menjadi kata dasarnya.

4.9 Implementasi BM25

BM25 adalah metode yang digunakan untuk melakukan pemeringkatan hasil dari kecocokan dokumen-dokuemen berdasarkan *query* yang dicari. Pada tahap ini terdiri dari proses TF, IDF, dan *score* BM25.

4.9.1 Implementasi TF

Hitung nilai TF merupakan tahap awal pada metode BM25. Dalam tahap ini, kata yang sering muncul memiliki frekuensi yang tinggi pada dokumen. Implementasi TF ditunjukkan pada Kode Program 4.5.

Algoritme: tf

```
1  def tf(hasiltraining,doctesting):
2      hasiltf = []
3      d = pd.DataFrame(hasiltraining)
4      header = list(d)
5      for i in range(len(hasiltraining)):
6          frekuensi_tf = {}
7          for j in doctesting:
8              frekuensi_tf[j] = 0
9          for k in frekuensi_tf:
10             if k in header:
11                 frekuensi_tf[k] = hasiltraining[k][i]
12             else:
13                 frekuensi_tf[k] = 0
14         hasiltf.append(frekuensi_tf)
15     return hasiltf
```

Kode Program 4.5 Implementasi TF

Berikut ini penjelasan potongan Kode Program 4.5.

1. Pada baris 3 merupakan pemanggilan fungsi pandas *DataFrame.*
2. Pada baris 2 merupakan pemanggilan fungsi *list* dari pandas *DataFrame.*
3. Pada baris 5 merupakan proses perulangan yang dimulai dari *i* = 0 sampai total panjang variabel *hasiltraining*.
4. Pada baris 6-7 merupakan proses pembuatan *dictionary* pada *doctesting* dengan nama variabel *frekuensi_tf* dan memberikan nilai awalan 0 pada setiap *keys*.
5. Pada baris 9-13 merupakan proses perulangan dengan mengecek apakah setiap kata pada variabel *frekuensi _tf* terdapat pada variabel *header*. Jika kata tersebut ada, maka nilai *frekuensi_tf* indeks ke *k* sama dengan nilai *hasiltraining* pada baris *k* kolom *i*. Jika kata tersebut tidak ada di variabel *header* maka, nilai variabel *frekuensi_tf* indeks ke *k* sama dengan 0.

4.9.2 Implementasi DF

Hitung nilai DF merupakan langkah setelah menghitung nilai TF. DF adalah jumlah dokumen *training* yang muncul term *query.* Implementasi DF ditunjukkan pada Kode Program 4.6.

```
Algoritme: df
 1  def df(tf,doctesting):
 2      df = []
 3      hasil_df = {}
 4      for i in range(len(tf)):
 5          frekuensi_df = {}
 6          for j in doctesting:
 7              frekuensi_df[j] = 0
 8          for k in frekuensi_df:
 9              if tf[i][k] > 0:
10                  frekuensi_df[k] = 1
11              else:
12                  frekuensi_df[k] = 0
13          df.append(frekuensi_df)
14      for i in df[0]:
15          count=0
16          for j in range(len(df)):
17              count = count + df[j][i]
18          hasil_df[i] = count
19      return hasil_df
```

Kode Program 4.6 Implementasi DF

Berikut ini penjelasan potongan Kode Program 4.6.

1. Pada baris 4 merupakan proses perulangan yang dimulai dari *i* = 0 sampai total panjang variabel *tf*.
2. Pada baris 5-7 merupakan proses pembuatan *dictionary* pada *doctesting* dengan nama variabel *frekuensi_df* dan memberikan nilai awalan 0 pada setiap *keys*.
3. Pada baris 8-12 merupakan proses perulangan dengan mengecek apakah nilai variabel *tf* baris *i* kolom *j* lebih besar 0. Jika iya maka nilai *frekuensi_df* indeks ke *k* sama dengan 1. Jika tidak maka nilai variabel *frekuensi_df* indeks ke *k* sama dengan 0.
4. Pada baris 14-17 merupakan proses perulangan dengan menjumlahkan variabel *count* dengan variabel *df* baris *j* kolom *i*.
5. Pada baris 18 merupakan mengisi nilai pada *key i* dengan nilai dari variabel *count.*

4.9.3 Implementasi IDF

Hitung nilai IDF merupakan langkah setelah hitung nilai DF. Dalam tahap ini, kata yang banyak muncul dalam banyak dokumen *training* akan memiliki bobot yang lebih sedikit daripada kata yang sedikit muncul dalam banyak dokumen *training.* Implemetasi IDF ditunjukkan pada Kode Program 4.7.

```
Algoritme: idfbm25
1  def idfbm25(df,doctraining):
2      hasil_idf={}
3      for i in df:
4          m = ((len(doctraining)-df[i]+0.5)/(df[i]+0.5))
5          hasil_idf[i] = math.log10(m)
6      return hasil_idf
```

Kode Program 4.7 Implementasi IDF

Penjelasan potongan Kode Program 4.7 yaitu pada baris 3-5 merupakan proses perulangan sebanyak kata yang ada di variabel *df* dan dilakukan proses perhitungan *idf* dengan rumus panjang dokumen training dikurangi nilai *df* pada kata *i* ditambah 0,5 lalu dibagi dengan nilai *df* pada kata *i* ditambah 0,5. Hasil tersebut lalu dilakukan operasi *log*.

4.9.4 Implementasi Panjang Dokumen

Hitung panjanh doc merupakan langkah menghitung panjang dokumen *training*. Implementasi panjang dokumen ditunjukkan pada Kode Program 4.8.

```
Algoritme: panjangdoc
1  def panjangdoc(doctraining):
2      hasil_panjangdoc = []
3      for i in range(len(doctraining)):
4          with open(doctraining[i],'r') as r:
5              a = r.read().replace('\n',' ')
6          panjangdoc = a.split(' ')
7          hasil_panjangdoc.append(len(panjangdoc))
8          r.close()
9      return hasil_panjangdoc
```

Kode Program 4.8 Implementasi Panjang Dokumen

Berikut ini penjelasan potongan Kode Program 4.8.

1. Pada baris 3-5 merupakan perulangan sebanyak *doctraining* dan dilakukan pembacaan file.
2. Pada baris 6 merupakan proses memparsing *doctraining* ke *i*

4.9.5 Implementasi Rata-Rata Panjang Dokumen

Hitung rata-rata panjang dokumen merupakan langkah menghitung rata-rata panjang dokumen *training*. Implementasi rata-rata panjang dokumen ditunjukkan pada Kode Program 4.9.

```
Algoritme: avgdoc
1  def avgdoc(panjangdoc):
2      avgdoc = 0
3      hasil_avgdoc = 0
4      for i in range(len(panjangdoc)):
5          avgdoc = avgdoc + panjangdoc[i]
6      hasil_avgdoc = avgdoc / len(panjangdoc)
7      return hasil_avgdoc
```

Kode Program 4.9 Implementasi Rata-Rata Panjang Dokumen

Berikut ini penjelasan potongan Kode Program 4.9.

1. Pada baris 4 merupakan perulangan sebanyak *panjangdoc.*
2. Pada baris 5-6 merupakan proses menghitung rata-rata panjang dokumen.

4.9.6 Implementasi *Score* BM25

Hitung *score* BM25 merupakan langkah terakhir dalam metode BM25. Proses tersebut merupakan proses pemeringkatan hasil dari kecocokan dokumen-dokumen, berdasarkan *query* yang dicari. Implementasi *score* BM25 ditunjukkan pada Kode Program 4.10.

```
Algoritme: scorebm25
 1  def scorebm25(tf,idf,panjangdoc,avgdoc):
 2      k = 2
 3      b = 0.8
 4      scorebm25 =[]
 5      for i in range(len(tf)):
 6          hasil = 0
 7          for j in idf:
 8              rumus = idf[j]*((tf[i][j]*(k+1))/(tf[i][j]+k*(1-
 9  b+b*panjangdoc[i]/avgdoc)))
10              hasil = hasil+rumus
11          scorebm25.append(hasil)
12      return scorebm25
```

Kode Program 4.10 Implementasi *Score* BM25

Berikut ini penjelasan potongan Kode Program 4.10.

1. Pada baris 2-3 merupakan inisialisasi variabel *k* sama dengan 2 dan variabel *b* sama dengan 0,8.
2. Pada baris 5 merupakan perulangan yang dimulai dari *i* sama dengan 0 sampai total panjang variabel *tf*.
3. Pada baris 7-10 merupakan proses perhitungan rumus IDF.

4.10 Implementasi *K-Nearest Neigbor*

Klasifikasi *K-Nearest Neighbor* merupakan proses mengelompokkan dokumen *testing* berdasarkan kelas yang terpilih. Proses-proses yang ada didalamnya yaitu pengurutan *score* BM25, seleksi sebanyak nilai *k*, dan penentuan kelas berdasarkan hasil seleksi nilai *k*.

4.10.1 Implementasi Pengurutan *Score* BM25

Langkah pertama yang dilakukan pada proses *K-Nearest Neighbor* adalah mengurutkan *score* BM25. Proses pengurutan dilakukan secara *descending* pada *score* BM25. Pengurutan *score* BM25 ditunjukkan pada Kode Program 4.11.

```
Algoritme: sortingKNN
1  def sortingKNN(score,kelasdoc):
2      hasil_sorting=[[]for k in range (len(kelasdoc))]
3      for i in range(len(score)):
4          a = score[i]
5          b = kelasdoc[i]
6          hasil_sorting[i].append(a)
7          hasil_sorting[i].append(b)
8      hasil_sorting = sorted(hasil_sorting,reverse=True)
9      return hasil_sorting
```

Kode Program 4.11 Implementasi Pengurutan *Score* BM25

Berikut ini penjelasan potongan Kode Program 4.11.

1. Pada baris 2 merupakan perulangan untuk membuat *list* dua dimensi dengan indeks sebanyak total panjang variabel *kelasdoc*.
2. Pada baris 3-7 merupakan proses perulangan dengan dilakukan pengambilan nilai pada variabel *score* indeks *i* dan nilai pada variabel *kelasdoc* indeks *i*. Nilai tersebut pada variabel *hasil_sorting* indeks *i*.
3. Pada baris 8 merupakan proses pemanggilan fungsi *sorted* yang bertujuan untuk mengurutkan nilai dari variabel *hasil_sorting* dari nilai yang terbesar sampai nilai yang terkecil.

4.10.2 Implementasi Seleksi Sebanyak Nilai *K*

Langkah berikutnya adalah menyeleksi *score* BM25 tertinggi sebanyak nilai *k*. Proses tersebut adalah proses pengambilan *score* BM25 yang paling tinggi sejumlah nilai *k.* Implementasi seleksi sebanyak nilai *k* ditunjukkan pada Kode Program 4.12.

```
Algoritme: votingKNN
1  def votingKNN(sorting,k):
2      votingKNN=[]
3      for i in range(k):
4          votingKNN.append(sorting[i][1])
5      return votingKNN
```

Kode Program 4.12 Seleksi Sebanyak Nilai *K*

Penjelasan potongan Kode Program 4.12 dan Kode Program 4.12 yaitu pada baris 3-4 merupakan proses perulangan sepanjang nilai variabel *k* dengan dilakukan pengambilan nilai pada variabel *sorting* baris *i* kolom satu dan disimpan pada variabel *votingKNN*.

4.10.3 Implementasi Penentuan Kelas Berdasarkan Hasil Seleksi Nilai *K*

Penentuan kelas berdasarkan hasil seleksi nilai *k* merupakan tahap terakhir dalam proses klasifikasi *K-Nearest Neighbor* (KNN). Pada tahap ini, akan dipilih kelas dokumen *testing* berdasarkan banyak kelas yang muncul pada tahap seleksi nilai *k*. Kelas yang paling banyak muncul pada tahap seleksi nilai *k* maka dijadikan kelas pada dokumen *testing.* Implementasi penentuan kelas berdasarkan hasil seleksi nilai *k* ditunjukkan pada Kode Program 4.13.

```
Algoritme: unweightedpenentuankelasKNN
 1  def unweightedpenentuankelasKNN(voting,kelas):
 2      kelas_testing = {}
 3      hasilakhir =[]
 4      for i in kelas:
 5          kelas_testing[i]=0
 6      for i in voting:
 7          if i in kelas:
 8              count = kelas_testing[i]
 9              kelas_testing[i] = count+1
10      kelasmax = max(kelas_testing['SP'],kelas_testing['SPK'],
11      kelas_testing['PCD'],kelas_testing['DM'],
12      kelas_testing['TM'])
13      for i in kelas_testing:
14          if kelas_testing[i]==kelasmax:
15              hasilakhir.append(i)
16      return hasilakhir[0]
```

Kode Program 4.13 Implementasi Penentuan Kelas Berdasarkan Seleksi Nilai *K*

Berikut ini penjelasan potongan Kode Program 4.13.

1. Pada baris baris 4-6 merupakan proses pembuatan *dictionary* sebanyak variabel *kelas* dengan memberi nilai awalan 0 pada semua *keys*.
2. Pada baris 6-9 merupakan proses perulangan dengan mengecek jika kata variabel *i* ada pada variabel *kelas* maka *kelas_testing* indeks *i* ditambah 1.
3. Pada baris 10-12 merupakan proses pemanggilan fungsi *max* yang bertujuan untuk mencari nilai tertinggi pada variabel *kelas_testing*.
4. Pada baris 13-15 merupakan proses perulangan dengan mengecek apakah nilai *kelas_testing* indeks *i* sama dengan nilai variabel *kelasmax*. Jika iya maka nilai variabel *i* akan disimpan pada variabel *hasilakhir*.
5. Pada baris 16 merupakan mengembalikan nilai *hasilakhir* indeks 0.

4.11 Implementasi *Training* Data

Training data merupakan alur untuk memproses atau mengolah data *training*. Data *training* tersebut berupa dokumen abstrak skripsi di bidang Komputasi Cerdas. Implementasi *training* data ditunjukkan pada Kode Program 4.14 dan Kode Program 4.15.

```
Algoritme: trainingdata
 1  def trainingdata(doctraining):
 2      hasil_training = []
 3      for j in range(len(doctraining)):
 4          with open(doctraining[j],'r') as r:
 5              readtraining = r.read().replace('\n',' ')
 6          casefoldingtraining = casefolding(readtraining)
 7          tokentraining = tokenisasi(casefoldingtraining)
 8          stopwordtraining = stopword(tokentraining)
 9          stemmingtraining = stemming(stopwordtraining)
```

Kode Program 4.14 Implementasi *Training* Data

```
Algoritme: trainingdata
10      hasil_training.append(stemmingtraining)
11      r.close()
12      term = gettermunik(hasil_training)
13      tf = tftraining(hasil_training,term)
14      with open('hasiltraining.csv', 'w') as csvfile:
15          writer = csv.DictWriter(csvfile, fieldnames=tf[0])
16          writer.writeheader()
17          for data in tf:
18              writer.writerow(data)
19      csvfile.close()
20      return 'hasiltraining.csv'
```

Kode Program 4.15 Implementasi *Training* Data (Lanjutan)

Berikut ini penjelasan potongan Kode Program 4.14 dan Kode Program 4.15.

1. Pada baris 4-5 merupakan proses pembaca dokumen *training.*
2. Pada baris 6-9 merupakan proses *pre-processing text* pada dokumen *training*.
3. Pada baris 12 merupakan pemanggilan fungsi *gettermunik*.
4. Pada baris 13 merupakan pemanggilan fungsi *tftraining*.
5. Pada baris 14-19 merupakan proses penyimpanan hasil *training* data pada berkas *hasiltraining.csv*

4.12 Implementasi *Testing* Data

Alur proses *testing* data merupakan alur untuk memproses atau mengolah data *testing*. Data *testing* tersebut berupa dokumen abstrak skripsi di bidang Komputasi Cerdas. Implementasi *testing* data ditunjukkan pada Kode Program 4.16 dan Kode Program 4.17.

```
Algoritme: testingdata
1   def testingdata(hasiltraining,k,doctesting):
2       hasil_knn =[]
3       kelas = ['SP','SPK','PCD','DM','TM']
4       doctraining = [f for f in glob.glob("D:\DATA
5       PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER
6       7\Skripsi\Program\datatraining\*.txt")]
7       bacakelasdoc = bacacsv("kelastraining.csv")
8       ambilkelasdoc = bacaKolom(bacakelasdoc,0))
9       hasil_training = pd.read_csv(hasiltraining)
10      panjang_doc = panjangdoc(doctraining)
11      avg_doc = avgdoc(panjang_doc)
12      for i in range(len(doctesting)):
13          with open(doctesting[i],'r') as r:
14              readtesting = r.read().replace('\n',' ')
15          casefoldingtesting = casefolding(readtesting)
16          tokentesting = tokenisasi(casefoldingtesting)
17          stopwordtesting = stopword(tokentesting
```

Kode Program 4.16 Implementasi *Testing* Data

```
Algoritme: testingdata
 18          stemmingtesting = stemming(stopwordtesting)
 19          tf_testing = tf(hasil_training,stemmingtesting)
 20          df_testing = df(tf_testing,stemmingtesting)
 21          idf_testing = idfbm25(df_testing,doctraining)
 22          score_bm25 = scorebm25(tf_testing,idf_testing,
 23          panjang_doc,avg_doc)
 24          sorting = sortingKNN(score_bm25,ambilkelasdoc)
 25          voting = votingKNN(sorting,k)
 26          penentuankelas = unweightedpenentuankelasKNN
 27          (voting,kelas)
 28          hasil_knn.append(penentuankelas)
 29          r.close()
 30      print(hasil_knn)
 31      return hasil_knn
```

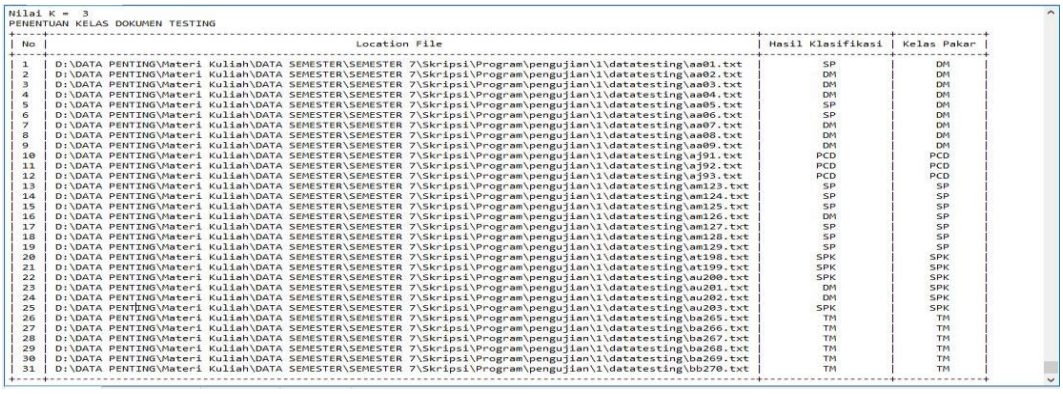
Kode Program 4.17 Implementasi *Testing* Data (Lanjutan)

Berikut ini penjelasan potongan Kode Program 4.16 dan Kode Program 4.17.

1. Pada baris 1-11 merupakan pembacaan berkas *doctesting, kelas,* dan *hasiltraning*.
2. Pada baris 12-13 merupakan proses pemanggilan fungsi *panjangdoc* dan *avgdoc.*
3. Pada baris 17-20 merupakan proses *pre-processing text* pada dokumen *testing*.
4. Pada baris 21-25 merupakan proses BM25.
5. Pada baris 26-29 merupakan proses klasifikasi *K-Nearest Neighbor*.
6. Pada baris 32 merupakan proses mencetak variabel *hasil_knn.*
7. Pada baris 33 merupakan proses pengembalian variabel *hasil_knn*.

4.13 Hasil Sistem

Berikut ini adalah hasil dari sistem yang telah dibuat pada penelitian ini yang ditunjukkan pada Gambar 4.23.



Nilai K = 3
PENENTUAN KELAS DOKUMEN TESTING

No	Location File	Hasil Klasifikasi	Kelas Pakar
1	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa01.txt	SP	DM
2	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa02.txt	DM	DM
3	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa03.txt	DM	DM
4	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa04.txt	DM	DM
5	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa05.txt	SP	DM
6	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa06.txt	SP	DM
7	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa07.txt	DM	DM
8	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa08.txt	DM	DM
9	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aa09.txt	DM	DM
10	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aj91.txt	PCD	PCD
11	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aj92.txt	PCD	PCD
12	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\aj93.txt	PCD	PCD
13	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\am123.txt	SP	SP
14	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\am124.txt	SP	SP
15	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\am125.txt	SP	SP
16	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\am126.txt	DM	SP
17	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\am127.txt	SP	SP
18	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\am128.txt	SP	SP
19	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\am129.txt	SP	SP
20	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\at198.txt	SPK	SPK
21	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\at199.txt	SPK	SPK
22	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\au200.txt	SPK	SPK
23	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\au201.txt	DM	SPK
24	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\au202.txt	DM	SPK
25	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\au203.txt	SPK	SPK
26	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\ba265.txt	TM	TM
27	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\ba266.txt	TM	TM
28	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\ba267.txt	TM	TM
29	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\ba268.txt	TM	TM
30	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\ba269.txt	TM	TM
31	D:\DATA PENTING\Materi Kuliah\DATA SEMESTER\SEMESTER 7\Skripsi\Program\pengujian\1\datatesting\bb270.txt	TM	TM

Gambar 4.23 Hasil Sistem

BAB 5 PENGUJIAN DAN ANALISIS

Bab ini menjelaskan tentang pengujian dan pembahasan terhadap sistem yang telah dibuat. Pengujian tersebut meliputi pengujian akurasi, *precision, recall,* dan *f-measure.*

5.1 *Precision*, *Recall*, dan *F-Measure*

Untuk mengetahui hasil *precision*, *recall*, dan *f-measure* sistem, maka dilakukan proses pengujian terhadap semua data. Pengujian tersebut menggunakan pengujian *k-fold* sebanyak 10 kali pengujian. Data uji yang digunakan setiap pengujian sebanyak 31 dokumen dengan komposisi kelas DM sebanyak 9 dokumen, kelas PCD sebanyak 3 dokumen, kelas SP sebanyak 7 dokumen, kelas SPK sebanyak 6 dokumen, dan kelas TM sebanyak 6 dokumen. Setiap pengujian *k-fold* menggunakan data uji yang berbeda-beda.

5.1.1 Pengujian *K-Fold* 1

Pengujian *k-fold* 1 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 1 dapat dilihat pada Tabel 5.1.

Tabel 5.1 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 1

k	*Precision*	*Recall*	*F-measure*
2	0,852	0,816	0,818
3	0,867	0,838	0,843
5	0,867	0,838	0,843
7	0,890	0,860	0,867
9	0,867	0,838	0,843
11	0,867	0,838	0,843
13	0,867	0,838	0,843
15	0,867	0,838	0,843
20	0,867	0,838	0,843
50	0,877	0,794	0,820
100	0,666	0,683	0,661
150	0,638	0,660	0,638

Pada Tabel 5.1 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 1. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=7 dengan nilai *f-measure* sebesar 0,867, *recall* sebesar 0,860, dan *precision* sebesar 0,890. Sedangkan nilai *f-measure* terendah

berada pada nilai *k*=150 dengan nilai *f-measure* sebesar 0,638, *recall* sebesar 0,660, dan *precision* sebesar 0,638.

5.1.2 Pengujian *K-Fold* 2

Pengujian *k-fold* 2 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 2 dapat dilihat pada Tabel 5.2.

Tabel 5.2 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 2

k	*Precision*	*Recall*	*F-measure*
2	0,946	0,911	0,920
3	0,942	0,900	0,913
5	0,971	0,933	0,945
7	0,946	0,911	0,920
9	0,946	0,911	0,920
11	0,946	0,911	0,920
13	0,946	0,911	0,920
15	0,946	0,911	0,920
20	0,946	0,911	0,920
50	0,933	0,844	0,848
100	0,711	0,778	0,738
150	0,656	0,716	0,681

Pada Tabel 5.2 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 2. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=5 dengan nilai *f-measure* sebesar 0,945, *recall* sebesar 0,933, dan *precision* sebesar 0,971. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=150 dengan nilai *f-measure* sebesar 0,681, *recall* sebesar 0,716, dan *precision* sebesar 0,656.

5.1.3 Pengujian *K-Fold* 3

Pengujian *k-fold* 3 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 3 dapat dilihat pada Tabel 5.3.

Tabel 5.3 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 3

k	*Precision*	*Recall*	*F-measure*
2	0,975	0,978	0,975
3	0,975	0,978	0,975
5	0,975	0,978	0,975

k	*Precision*	*Recall*	*F-measure*
7	0,971	0,933	0,945
9	1,000	1,000	1,000
11	1,000	1,000	1,000
13	1,000	1,000	1,000
15	1,000	1,000	1,000
20	1,000	1,000	1,000
50	0,980	0,933	0,949
100	0,955	0,867	0,876
150	0,739	0,800	0,767

Pada Tabel 5.3 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 3. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=9, *k*=11, *k*=13, dan *k*=15 dengan nilai *f-measure* sebesar 1, *recall* sebesar 1, dan *precision* sebesar 1. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=150 dengan nilai *f-measure* sebesar 0,767, *recall* sebesar 0,800, dan *precision* sebesar 0,739.

5.1.4 Pengujian *K-Fold* 4

Pengujian *k-fold* 4 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 4 dapat dilihat pada Tabel 5.4.

Tabel 5.4 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 4

k	*Precision*	*Recall*	*F-measure*
2	0,911	0,900	0,897
3	0,953	0,944	0,946
5	0,924	0,878	0,891
7	0,924	0,878	0,891
9	0,902	0,844	0,864
11	0,902	0,856	0,866
13	0,897	0,856	0,866
15	0,896	0,822	0,846
20	0,902	0,844	0,864
50	0,675	0,711	0,688
100	0,569	0,627	0,592
150	0,592	0,622	0,599

Pada Tabel 5.4 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 4. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=3 dengan nilai *f-measure* sebesar 0,946, *recall* sebesar 0,944, dan *precision* sebesar 0,953. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=100 dengan nilai *f-measure* sebesar 0,592, *recall* sebesar 0,627, dan *precision* sebesar 0,569.

5.1.5 Pengujian *K-Fold* 5

Pengujian *k-fold* 5 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 5 dapat dilihat pada Tabel 5.5.

Tabel 5.5 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 5

k	*Precision*	*Recall*	*F-measure*
2	0,908	0,900	0,888
3	0,925	0,933	0,918
5	0,925	0,933	0,918
7	0,950	0,956	0,946
9	0,906	0,822	0,827
11	0,950	0,956	0,946
13	0,906	0,822	0,827
15	0,906	0,822	0,827
20	0,906	0,822	0,827
50	0,906	0,822	0,827
100	0,681	0,760	0,716
150	0,688	0,771	0,720

Pada Tabel 5.5 Tabel 5.1menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 5. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=7, 11 dengan nilai *f-measure* sebesar 0,946, *recall* sebesar 0,956, dan *precision* sebesar 0,950. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=100 dengan nilai *f-measure* sebesar 0,716, *recall* sebesar 0,760, dan *precision* sebesar 0,681.

5.1.6 Pengujian *K-Fold* 6

Pengujian *k-fold* 6 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 6 dapat dilihat pada Tabel 5.6.

Tabel 5.6 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 6

k	*Precision*	*Recall*	*F-measure*
2	0,871	0,878	0,857
3	0,866	0,867	0,852
5	0,921	0,933	0,916
7	0,921	0,933	0,916
9	0,918	0,927	0,919
11	0,943	0,956	0,944
13	0,921	0,933	0,916
15	0,864	0,871	0,864
20	0,886	0,900	0,888
50	0,921	0,933	0,916
100	0,883	0,789	0,801
150	0,665	0,722	0,691

Pada Tabel 5.6 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 6. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=11 dengan nilai *f-measure* sebesar 0,944, *recall* sebesar 0,956, dan *precision* sebesar 0,943. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=150 dengan nilai *f-measure* sebesar 0,691, *recall* sebesar 0,722, dan *precision* sebesar 0,665.

5.1.7 Pengujian *K-Fold* 7

Pengujian *k-fold* 7 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 7 dapat dilihat pada Tabel 5.7.

Tabel 5.7 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 7

k	*Precision*	*Recall*	*F-measure*
2	0,868	0,827	0,839
3	0,820	0,727	0,736
5	0,844	0,760	0,767
7	0,881	0,783	0,795
9	0,846	0,760	0,766
11	0,846	0,760	0,766
13	0,875	0,827	0,841
15	0,875	0,827	0,841
20	0,911	0,849	0,870

k	*Precision*	*Recall*	*F-measure*
50	0,911	0,849	0,870
100	0,661	0,716	0,681
150	0,645	0,694	0,664

Pada Tabel 5.7Tabel 5.1 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 1. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=13 dan 15 dengan nilai *f-measure* sebesar 0,841, *recall* sebesar 0,827, dan *precision* sebesar 0,875. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=100 dengan nilai *f-measure* sebesar 0,664, *recall* sebesar 0,694, dan *precision* sebesar 0,645.

5.1.8 Pengujian *K-Fold* 8

Pengujian *k-fold* 8 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 8 dapat dilihat pada Tabel 5.8.

Tabel 5.8 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 8

k	*Precision*	*Recall*	*F-measure*
2	0,921	0,956	0,931
3	0,918	0,927	0,919
5	0,918	0,927	0,919
7	0,918	0,927	0,919
9	0,943	0,956	0,944
11	0,913	0,922	0,916
13	0,913	0,922	0,916
15	0,913	0,922	0,916
20	0,975	0,978	0,975
50	0,975	0,978	0,975
100	0,924	0,844	0,849
150	0,735	0,800	0,765

Pada Tabel 5.8Tabel 5.1 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 8. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=20 dan 50 dengan nilai *f-measure* sebesar 0,975, *recall* sebesar 0,978, dan *precision* sebesar 0,975. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=150 dengan nilai *f-measure* sebesar 0,765, *recall* sebesar 0,800, dan *precision* sebesar 0,735.

5.1.9 Pengujian *K-Fold* 9

Pengujian *k-fold* 9 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 9 dapat dilihat pada Tabel 5.9.

Tabel 5.9 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 9

k	*Precision*	*Recall*	*F-measure*
2	0,971	0,978	0,973
3	0,975	0,978	0,975
5	0,953	0,944	0,946
7	0,935	0,878	0,897
9	0,935	0,878	0,897
11	0,953	0,944	0,946
13	0,953	0,944	0,946
15	0,953	0,944	0,946
20	0,975	0,978	0,975
50	0,921	0,889	0,896
100	0,673	0,756	0,711
150	0,636	0,711	0,667

Pada Tabel 5.9 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 9. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=3 dan 20 dengan nilai *f-measure* sebesar 0,975, *recall* sebesar 0,978, dan *precision* sebesar 0,975. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=150 dengan nilai *f-measure* sebesar 0,667, *recall* sebesar 0,711, dan *precision* sebesar 0,636.

5.1.10 Pengujian *K-Fold* 10

Pengujian *k-fold* 10 dilakukan dengan data latih sebanyak 300 dokumen dan data uji sebanyak 31 dokumen. Hasil pengujian *k-fold* 10 dapat dilihat pada Tabel 5.10

Tabel 5.10 *Precision*, *Recall*, dan *F-Measure* pada *K-Fold* 10

k	*Precision*	*Recall*	*F-measure*
2	0,927	0,916	0,920
3	0,912	0,887	0,893
5	0,927	0,916	0,920
7	0,927	0,916	0,920
9	0,944	0,944	0,944

k	*Precision*	*Recall*	*F-measure*
11	0,944	0,944	0,944
13	0,944	0,944	0,944
15	0,944	0,944	0,944
20	0,917	0,922	0,918
50	0,925	0,889	0,893
100	0,681	0,756	0,714
150	0,681	0,756	0,714

Pada Tabel 5.10 menunjukkan hasil *precision*, *recall*, *f-measure*, dan *accuracy* terhadap nilai *k* pada pengujian *k-fold* 10. Pada pengujian ini, nilai *f-measure* tertinggi berada pada nilai *k*=9, *k*=11, *k*=13, dan *k*=15 dengan nilai *f-measure* sebesar 0,944, *recall* sebesar 0,944, dan *precision* sebesar 0,944. Sedangkan nilai *f-measure* terendah berada pada nilai *k*=100 dan 150 dengan nilai *f-measure* sebesar 0,714, *recall* sebesar 0,756, dan *precision* sebesar 0,681.

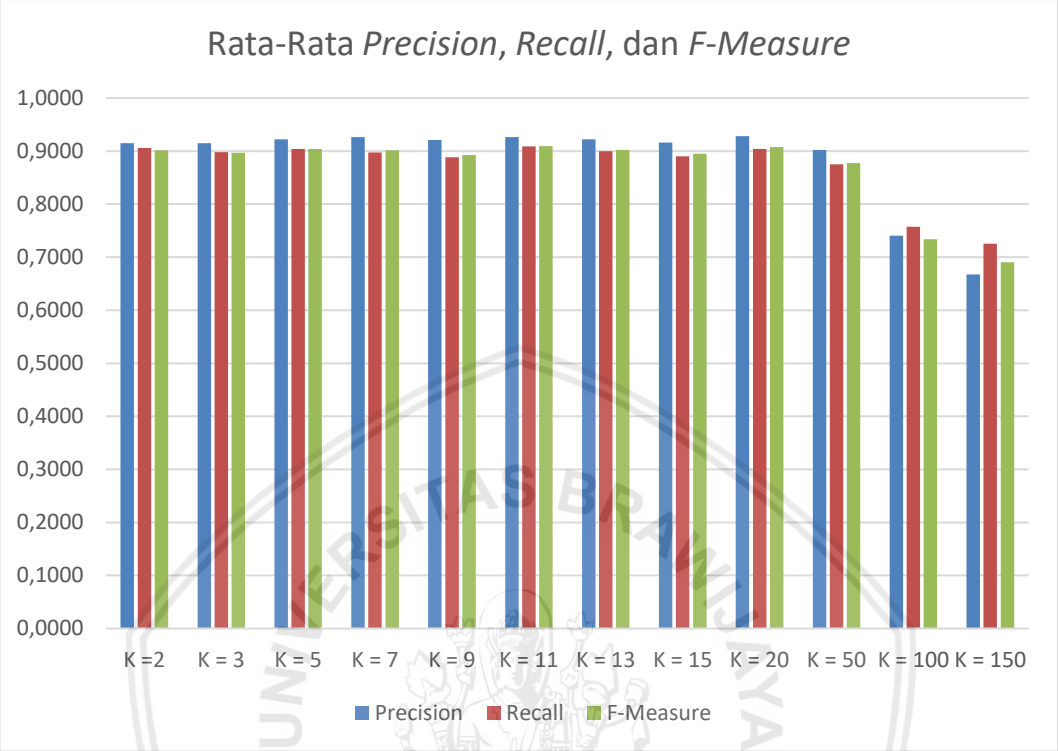
5.1.11 Rata-rata *Precision*, *Recall*, dan *F-Measure*

Pada Tabel 5.11 ditunjukkan rata-rata dari hasil perhitungan *precision*, *recall*, dan *f-measure* untuk tiap nilai *k.*

Tabel 5.11 Rata-Rata *Precision*, *Recall*, dan *F-Measure*

k	*Precision*	*Recall*	*F-measure*
2	0,9152	0,9059	0,9017
3	0,9152	0,8979	0,8970
5	0,9225	0,9041	0,9039
7	0,9264	0,8975	0,9015
9	0,9207	0,8881	0,8924
11	0,9265	0,9087	0,9092
13	0,9223	0,8998	0,9020
15	0,9164	0,8903	0,8948
20	0,9284	0,9043	0,9079
50	0,9025	0,8754	0,8776
100	0,7405	0,7574	0,7340
150	0,6676	0,7252	0,6904

Dari Tabel 5.11 didapatkan bahwa rata-rata *f-measure* terbaik berada pada nilai *k* sebesar 11 dengan nilai *f-measure* sebesar 0,9092. Grafik rata-rata nilai *percision*, *recall* dan *f-measure* tiap nilai k dapat ditunjukkan pada Gambar 5.1.



Gambar 5.1 Rata-Rata *Precision*, *Recall*, dan *F-Measure*

5.2 Pengujian dengan Selisih Nilai *K* Linear

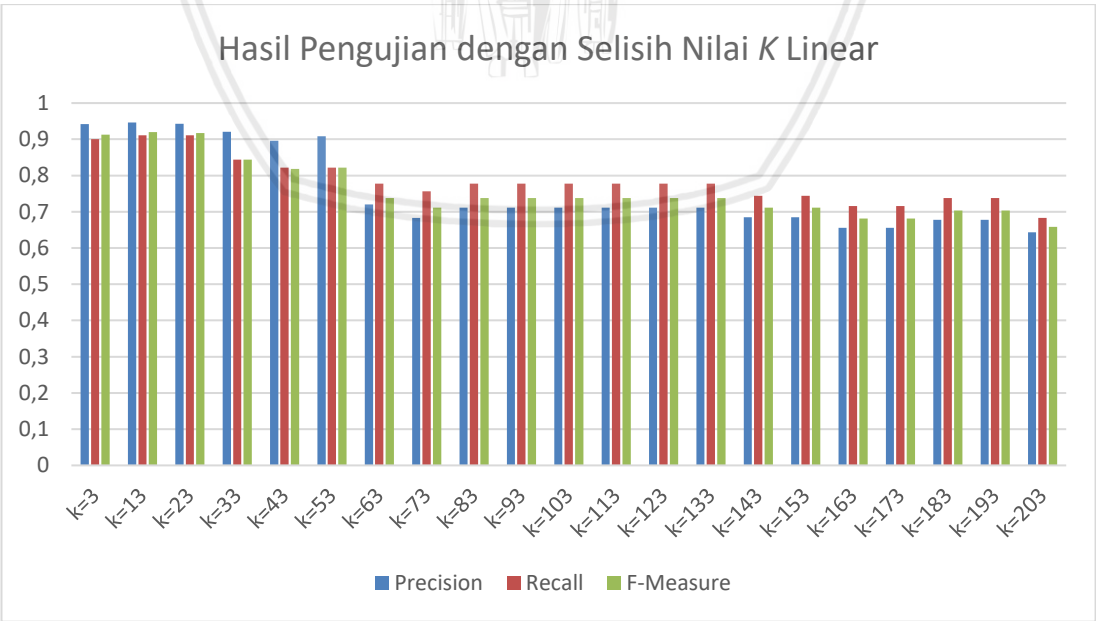
Pada pengujian ini, dilakukan dengan menggunakan selisih nilai *k* yang linear. Data yang digunakan pada pengujian ini yaitu data *training* berjumlah 300 dokumen dan data *testing* berjumlah 31 dokumen, sedangkan nilai *k* yang digunakan yaitu dimulai dari *k*=3, *k*=13, *k*=23, sampai *k*=203. Selisih nilai *k* satu dengan nilai *k* berikutnya sebesar 10. Hasil pengujian dengan selisih nilai *k* linear ditunjukkan pada Tabel 5.12.

Tabel 5.12 Hasil Pengujian dengan Selisih Nilai *K* Linear

k	*Precision*	*Recall*	*F-measure*
3	0,942	0,900	0,913
13	0,946	0,911	0,920
23	0,943	0,911	0,917
33	0,921	0,844	0,844
43	0,896	0,822	0,818
53	0,908	0,822	0,822

k	*Precision*	*Recall*	*F-measure*
63	0,720	0,778	0,738
73	0,683	0,756	0,711
83	0,711	0,778	0,738
93	0,711	0,778	0,738
103	0,711	0,778	0,738
113	0,711	0,778	0,738
123	0,711	0,778	0,738
133	0,711	0,778	0,738
143	0,685	0,744	0,711
153	0,685	0,744	0,711
163	0,656	0,716	0,681
173	0,656	0,716	0,681
183	0,678	0,738	0,703
193	0,678	0,738	0,703
203	0,643	0,683	0,658

Dari Tabel 5.12 didapatkan bahwa hasil *f-measure* terbaik berada pada nilai *k* sebesar 13 dengan nilai *f-measure* sebesar 0,920. Grafik pengujian dengan selisih nilai *k* linear dapat ditunjukkan pada Gambar 5.2.



Gambar 5.2 Hasil Pengujian dengan Selisih Nilai *K* Linear

5.3 Analisis

Dari hasil keseluruhan evaluasi tiap pengujian, dapat diketahui bahwa faktor yang mempengaruhi klasifikasi dengan algortime *K-Nearest Neighbor* adalah faktor nilai *k*. Besar kecilnya nilai *k* dapat memengaruhi hasil klasifikasi *K-Nearest Neighbor*. Pada nilai *k*=1 sampai nilai *k*=20, nilai *precision*, *recall*, dan *f-measure* mengalami fluktuatif dengan rentang nilai yang kecil, sedangkan untuk nilai *k* yang semakin besar, nilai *precision*, *recall*, dan *f-measure* mengalami penurunan. Penurunan tersebut terjadi karena banyak kesalahan klasifikasi pada kelas PCD dan DM.

Kelas PCD mengalami kesalahan pada proses klasifikasi karena data latih yang digunakan terlalu sedikit dibandingkan dengan kelas yang lainnya. Sedangkan pada kelas DM mengalami kesalahan pada proses klasifikasi karena dokumen kelas DM memiliki kemiripan isi dengan dokumen SP, sehingga hal tersebut mengganggu proses klasifikasi. Hasil terbaik yang didapatkan pada keseluruhan evaluasi berada pada nilai *k*=11 dengan nilai *f-measure* sebesar 0,9092. Pada nilai *k*=11 metode BM25 dapat memeringkatkan dokumen dari yang relevan sampai yang tidak relevan, sedangkan pada hasil keseluruhan evaluasi penelitian ini, hasil terendah berada pada nilai *k* yang semakin besar.

Oleh karena itu, pada penelitian ini dapat kita tarik kesimpulan bahwa metode BM25 dapat melakukan pemeringkatan dokumen dari yang relevan sampai dokumen yang tidak relevan, sedangkan untuk algoritme *K-Nearest Neighbor* dapat berjalan baik apabila dokumen yang digunakan dalam data latih tiap kelas berjumlah banyak dan semakin kecil nilai *k* yang digunakan, sistem berjalan dengan baik, sedangkan semakin besar nilai *k* yang digunakan, sistem berjalan kurang baik.

BAB 6 PENUTUP

Bab ini menjelaskan mengenai kesimpulan dan saran terhadap penelitian ini. Kesimpulan dan saran digunakan untuk membantu pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Berikut ini adalah kesimpulan yang diperoleh dari hasil penelitian Klasifikasi Dokumen Abstrak Skripsi Berdasarkan Fokus Penelitian Di Bidang Komputasi Cerdas Menggunakan BM25 Dan *K-Nearest Neighbor*.

1. Metode BM25 dan K-Nearest Neighbor dapat diterapkan dalam penelitian klasifikasi dokumen abstrak skripsi berdasarkan fokus penelitian di bidang komputasi cerdas. Proses yang dilalui dokumen abstrak skripsi ada beberapa tahapan, mulai dari *pre-processing,* perhitungan BM25, dan klasifikasi dengan algoritme *K-Nearest Neighbor.*
2. Hasil yang diperoleh pada proses evaluasi, didapatkan hasil terbaik rata-rata *f-measure* berada pada nilai *k* sebesar 11 dengan nilai *f-measure* sebesar 0,9092.
3. Besar kecil nilai *k* memengaruhi hasil klasifikasi pada penelitian ini. Semakin kecil nilai *k* yang digunakan, sistem berjalan dengan baik, sedangkan semakin besar nilai *k* yang digunakan sistem berjalan kurang baik.

6.2 Saran

Berikut ini adalah saran yang diperoleh dari hasil penelitian Klasifikasi Dokumen Abstrak Skripsi Berdasarkan Fokus Penelitian Di Bidang Komputasi Cerdas Menggunakan BM25 Dan *K-Nearest Neighbor.*

1. Proses klasifikasi dengan menggunakan algoritme *K-Nearest Neighbor* seharusnya menggunakan data latih tiap kelas berjumlah banyak. Hal ini sangat mempengaruhi kinerja dari algoritme *K-Nearest Neighbor.*
2. Pada penelitian ini, besar kecil nilai *k* sangat memengaruhi hasil dari klasifikasi menggunakan algoritme *K-Nearest Neighbor,* sehingga diperlukan metode khusus yang dapat mengoptimal nilai *k.* Hal tersebut dapat menggunakan algoritme *Improved K-Nearest Neighbor.*

DAFTAR REFERENSI

Adriani, M., Asian, J., Nazief, B. dan Williams, H.E., 2007. Stemming Indonesian : A Confi x-Stripping Approach. *ACM Transactions on Asian Language Information Processing (TALIP)*, 6(4), hal.1–33.

Aramaki, E., Imai, T., Miyo, K. dan Ohe, K., 2006. Patient Status Classification by using Rule based Sentence Extraction and BM25-kNN based Classifier. *i2b2 Workshop on Challenges in Natural Language Processing for Clinical Data*. [daring] Tersedia pada: <http://luululu.com/paper/2006-i2b2/i2b2-smoking.pdf>.

Bagaskoro, G.N., Fauzi, M.A. dan Adikara, P.P., 2018. Penerapan Klasifikasi Tweets Pada Berita Twitter Menggunakan Metode K-Nearest Neighbor Dan Query Expansion Berbasis Distributional Semantic. 2(10), hal.3849–3855.

Chinchor, N., 1992. MUC-4 evaluation metrics. *Science Applications International Corporation, San Diego*, hal.22–29.

Claudy, Y.I., Perdana, R.S. dan Fauzi, M.A., 2018. Klasifikasi Dokumen Twitter Untuk Mengetahui Karakter Calon Karyawan Menggunakan Algoritme K-Nearest Neighbor (KNN). 2(February), hal.2761–2765.

Feldman, R. dan Sanger, J., 2006. *The Text Mining Handbook*. [daring] Tersedia pada: <https://www.cambridge.org/core/product/identifier/9780511546914/type/book>.

FILKOM, 2017a. *Panduan Skripsi Fakultas Ilmu Komputer Universitas Brawijaya*.

FILKOM, 2017b. *Pedoman Akademik Fakultas Ilmu Komputer Universitas Brawijaya*. [daring] Tersedia pada: <http://filkom.ub.ac.id/page/read_/Y2E1NWIzODhmZDdiYzU3NmQ5YzNmMjI2ZmEwMjI0ZTc=>.

George, H. dan Rothschild, A., 2005. Agreement , the F-Measure , and Reliability in Information Retrieval. *Journal of the American …*, [daring] hal.296–298. Tersedia pada: <https://jamia.oxfordjournals.org/content/12/3/296.full>.

Nurjanah, W.E., Perdana, R.S. dan Fauzi, M.A., 2017. Analisis Sentimen Terhadap Tayangan Televisi Berdasarkan Opini Masyarakat pada Media Sosial Twitter menggunakan Metode K-Nearest Neighbor dan Pembobotan Jumlah Retweet. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, [daring] 1(12), hal.1750–1757. Tersedia pada: <http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/631>.

Powers, D.M.W., 2007. Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation. *School of Informatics and Engineering Flinders University of South Australia*, (December).

Suharno, C.F., Fauzi, M.A. dan Perdana, R.S., 2017. Klasifikasi Teks Bahasa Indonesia pada Dokumen Pengaduan Sambat Online Menggunakan Metode K-Nearest Neighbors dan Chi-Square. *Systemic: Information System and Informatics Journal*, 3(1), hal.25–32.

Tala, F.Z., 2003. A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. *Master of Logic Project Institute for Logic, Language and Computation Universiteit van Amsterdam*.

Yang, C., Du, H., Wu, S. dan Chen, I., 2012. Duplication Detection for Software Bug Reports based on BM25 Term Weighting.