TUGAS AKHIR – KI141502

IMPLEMENTASI KLASIFIKASI OPINI REVIEW FILM MENGGUNAKAN SUPPORT VECTOR MACHINE DENGAN PARTICLE SWARM OPTIMIZATION

ADDIEN HANIEFARDY
NRP 5112 100 127

Dosen Pembimbing
Diana Purwitasari, S.Kom., M.Sc.
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – KI141502

IMPLEMENTASI KLASIFIKASI OPINI REVIEW FILM MENGGUNAKAN SUPPORT VECTOR MACHINE DENGAN PARTICLE SWARM OPTIMIZATION

ADDIEN HANIEFARDY
NRP 5112 100 127

Dosen Pembimbing
Diana Purwitasari, S.Kom., M.Sc.
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

IMPLEMENTATION OF OPINION REVIEW FILM CLASSIFICATION USING SUPPORT VECTOR MACHINE WITH PARTICLE SWARM OPTIMIZATION

ADDIEN HANIEFARDY
NRP 5112 100 127

Advisor
Diana Purwitasari, S.Kom., M.Sc.
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI KLASIFIKASI OPINI REVIEW FILM MENGGUNAKAN SUPPORT VECTOR MACHINE DENGAN PARTICLE SWARM OPTIMIZATION

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
ADDIEN HANIEFARDY
NRP : 5112 100 127

Disetujui oleh Dosen Pembimbing Tugas Akhir :

Diana Purwitasari, S.Kom., M.Sc. ..... (Pembimbing 1)
NIP: 197804102003122001

Dr. Eng. Chastine Fatichah, S.Kom., M.Kom ..... (Pembimbing 2)
NIP: 197512202001122002

SURABAYA
JUNI 2016

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI KLASIFIKASI OPINI REVIEW FILM MENGGUNAKAN SUPPORT VECTOR MACHINE DENGAN PARTICLE SWARM OPTIMIZATION

Nama Mahasiswa : ADDIEN HANIEFARDY
NRP : 5112 100 127
Jurusan : Teknik Informatika ITS
Dosen Pembimbing I : Diana Purwitasari, S.Kom., M.Sc.
Dosen Pembimbing II : Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Abstrak

Pertumbuhan pesat jumlah pengguna internet dan jaringan sosial telah membuka jalan untuk mengakses dan menganalisis opini para penggunanya. Analisis maksud opini apakah termasuk positif atau negatif merupakan hal yang mudah bagi manusia. Namun, analisis semua opini secara manual menjadi hal yang mustahil karena jumlah opini yang meningkat secara besar-besaran. Oleh karena itu, penggalian opini diperlukan untuk menganalisis opini secara otomatis dan dapat mengambil informasi yang berguna.

Pada tugas akhir ini, sistem yang diimplementasikan berupa sistem yang mampu melakukan klasifikasi opini review film. Sistem menganalisis apakah opini termasuk opini posistif atau negatif. Tahap pertama adalah ekstraksi fitur dengan pemrosesan teks. Tahap ini memroses dokumen menjadi fitur, dimulai dari case folding, tokenization, stopwords removal, dan pembobotan TF-IDF. Tahap kedua adalah klasifikasi menggunakan metode SVM. Namun, untuk meminimalkan jumlah fitur yang sangat banyak dilakukan penambahan proses seleksi fitur dengan metode PSO. Tahap ketiga adalah evaluasi. Kinerja sistem dievaluasi dengan menggunakan metode confusion matrix.

Uji coba pada tugas akhir ini menggunakan data opini review film berjumlah 2000 dokumen. Untuk metode validasi, sistem menggunakan metode k-fold cross validation. Hasil uji coba menyimpulkan bahwa sistem dapat melakukan klasifikasi opini review film cukup baik. Nilai akurasi klasifikasi SVM dengan nilai

k pada cross validation sama dengan 5 sebesar 50.40% meningkat menjadi 60.85% setelah ditambahkan seleksi fitur menggunakan metode PSO. Penggunaan sentimen untuk pemilihan fitur juga sangat berpengaruh.

Kata kunci: opini review film, analisis sentimen, klasifikasi teks, support vector machine, particle swarm optimization.

IMPLEMENTATION OF OPINION REVIEW FILM CLASSIFICATION USING SUPPORT VECTOR MACHINE WITH PARTICLE SWARM OPTIMIZATION

Name : ADDIEN HANIEFARDY
NRP : 5112 100 127
Major : Informatics Department - ITS
Supervisor I : Diana Purwitasari, S.Kom., M.Sc.
Supervisor II : Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Abstract

The rapid growth of Internet users and social networking has opened the way to access and analyze the opinions of its users. Analysis of whether the opinion is positif or negatif is an easy thing for humans. However, the analysis of all manually opinion becomes impossible because of the number of opinions increased massively. Therefore, opinion mining is required for automatically analyze the opinion and can retrieve useful information

In this thesis, the system is implemented in the form of a system that is able to classify the opinions of movie reviews. System analyzes whether the opinion includes positive or negative opinion. The first stage is the extraction of features with text processing. This phase process the document into a feature, starting from folding case, tokenization, stopwords removal and TF-IDF weighting. The second stage is classification using SVM method. However, to minimize the number of features that are very much, the addition of feature selection using PSO method is needed. The third stage is evaluation. Performance of system was evaluated using the confusion matrix.

The trial in this thesis using a 2000 opinion movie review documents. For validation of methods, systems using the k-fold cross validation. The trial results concluded that the system can classify opinion of review films quite well. SVM classification accuracy value with k for cross validation equal to 5 at 50.40% increase to 60.85% after being added feature selection using the

method of PSO. Use of sentiment for feature selection is also very influential.

Keywords: movie review opinion, sentiment analysis, text classification, support vector machine, particle swarm optimization.

KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Segala puji dan syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"*Implementasi Klasifikasi Opini Review Film menggunakan Support Vector Machine dengan Particle Swarm Optimization*"**.

Dalam pelaksanaan tugas akhir ini tentu penulis sebagai makhluk sosial tidak dapat menyelesaikannya tanpa bantuan dari pihak lain. Tanpa mengurangi rasa hormat, penulis memberikan penghargaan serta ucapan terima kasih yang sebesar-besarnya kepada:

1. Ibu dan ayah, yang senantiasa memberikan semangat, dukungan dan doa agar penulis dapat menyelesaikan tugas akhir dengan tepat waktu.
2. Kakak saya, mbak Nuris, yang telah bersedia menjadi tempat untuk saya berdiskusi tentang apapun yang berkaitan atau tidak dengan penyelesaian tugas akhir saya ini.
3. Ibu Diana Purwitasari, S.Kom, M.Sc., selaku dosen pembimbing tugas akhir pertama yang telah membimbing dengan penuh kesabaran dan dukungan, memotivasi dan memberikan banyak masukan dalam pengerjaan tugas akhir ini.
4. Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku dosen pembimbing tugas akhir kedua yang telah membimbing, memotivasi dengan penuh senyuman dan memberikan banyak masukan dalam pengerjaan tugas akhir ini.
5. Bapak dan Ibu dosen Teknik Informatika ITS yang telah mengajarkan banyak ilmu berharga kepada saya selama 4 tahun ini.

6. Bapak dan Ibu karyawan Teknik Informatika ITS atas berbagai bantuan yang telah diberikan kepada saya selama masa perkuliahan.
7. Teman-teman administrator Laboratorium Komputasi Cerdas dan Visi yang telah membantu memfasilitasi saya dalam pengerjaan tugas akhir ini di laboratorium KCV.
8. Para sahabat KCV yang selalu menemani hari-hari saya, Mustofa, Yunan, Andrew, Ghozie, Jono, Fika, Yudha.
9. Teman-teman Ikatan Mahasiswa Muhammadiyah Sepuluh Nopember yang telah membersamai saya dalam berjuang membangun ikatan ini dan selalu menyemangati saya untuk bisa lulus tepat waktu. Banyak sekali momen-momen yang tidak terlupakan bersama kalian.
10. Teman-teman Teknik Informatika ITS angkatan 2012, yang telah memberikan warna-warni kehidupan mahasiswa saya sejak mahasiswa baru hingga lulus.
11. Pihak-pihak lain yang tidak sempat saya sebutkan, yang telah membantu kelancaran pengerjaan tugas akhir ini.

Penulis sangat berharap bahwa apa yang dihasilkan dari tugas akhir ini dapat memberikan manfaat bagi semua pihak, khususnya bagi diri penulis sendiri dan seluruh *civitas academica* Teknik Informatika ITS, serta bagi agama, bangsa, dan negara. Tak ada manusia yang sempurna sekalipun penulis berusaha sebaik mungkin dalam menyelesaikan tugas akhir ini. Karena itu, penulis memohon maaf apabila terdapat kesalahan, kekurangan, maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun sangat diharapkan oleh penulis untuk dapat disampaikan untuk perbaikan selanjutnya.

Surabaya, Juli 2016

Addien Haniefardy

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini membahas garis besar penyusunan tugas akhir yang terdiri dari 6 sub bab. Sub bab pertama membahas tentang latar belakang pengerjaan tugas akhir. Sub bab kedua membahas tentang rumusan permasalahan. Sub bab ketiga membahas tentang batasan permasalahan. Sub bab keempat membahas tentang tujuan pengerjaan tugas akhir. Sub bab kelima membahas tentang metodologi pengerjaan tugas akhir meliputi penyusunan proposal tugas akhir, studi literatur, implementasi, uji coba dan evaluasi, dan penyusunan buku tugas akhir. Sub bab terakhir membahas tentang sistematika penulisan buku tugas akhir.

1.1. Latar Belakang

Pertumbuhan pesat jumlah pengguna internet dan jaringan sosial telah membuka jalan untuk mengakses opini para penggunanya. Analisis maksud opini apakah termasuk positif atau negatif merupakan hal yang mudah bagi manusia [1]. Namun, tugas menganalisis semua opini secara manual menjadi hal yang mustahil karena jumlah opini yang meningkat secara besar-besaran. Oleh karena itu, penggalian opini diperlukan untuk menganalisis opini secara otomatis dan mengambil informasi yang berguna. Penggalian opini merupakan bidang studi yang menganalisis opini pengguna internet terhadap entitas seperti produk, jasa, organisasi, individu, masalah, peristiwa dan topik. Salah satu permasalahan yang akan diangkat dalam tugas akhir ini adalah analisis opini review film.

Banyak pendekatan yang dikembangkan untuk menganalisis sentimen opini seperti pendekatan mesin pembelajaran dan pendekatan semantik. Metode yang akan diimplementasikan dalam tugas akhir ini adalah teknik mesin pembelajaran *Support Vector Machine* (SVM). SVM merupakan salah satu algoritma klasifikasi linier yang memiliki prinsip utama untuk menentukan

pemisah linear terbaik dalam ruang pencarian yang dapat memisahkan dua kelas yang berbeda. Algoritma SVM memiliki beberapa kelebihan. Salah satunya adalah mampu mengidentifikasi *hyperplane* terpisah yang memaksimalkan margin antara dua kelas. SVM termasuk metode yang kuat untuk minimalisasi resiko. Namun, dalam pengolahan data teks, SVM memiliki kekurangan berupa penggunaan jumlah fitur yang sangat banyak. Ini mengakibatkan nilai akurasi yang dihasilkan tidak maksimal. Untuk mengatasi kekurangan tersebut, dibutuhkan minimalisasi jumlah fitur untuk dapat meningkatkan nilai akurasi pada klasifikasi SVM. Algoritma *Partical Swarm Optimization* (PSO) memiliki potensi untuk memberikan hasil seleksi fitur yang dapat membuat nilai akurasi meningkat pada klasifikasi SVM [2].

Hasil yang diharapkan dari tugas akhir ini adalah klasifikasi sentimen opini review film berdasarkan model SVM-PSO yang dibangun sebelumnya melalui pelatihan *data training*. Data sentimen dapat berguna sebagai rekomendasi bagi konsumen film yang belum pernah menonton atau sebagai alat ukur kepuasan konsumen terhadap film yang dibuat.

1.2. Rumusan Masalah

Rumusan masalah yang terdapat pada tugas akhir ini adalah sebagai berikut :

1. Bagaimana mengolah data review film sehingga dapat digunakan untuk memprediksi jenis sentimen?
2. Bagaimana melakukan seleksi fitur menggunakan metode PSO untuk mendapatkan nilai akurasi yang lebih baik pada klasifikasi SVM?
3. Apakah penambahan seleksi fitur menggunakan metode PSO dapat memberikan hasil yang lebih baik dalam pengklasifikasian opini review film?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain :

1. Dataset opini review film yang digunakan berbahasa inggris.
2. Kelas yang diklasifikasi terdiri dari dua jenis, yaitu positif dan negatif.

1.4. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut :

1. Membuat sebuah aplikasi yang dapat memprediksi sentimen opini secara otomatis.
2. Mengimplementasikan algoritma PSO untuk seleksi fitur dalam mengklasifikasikan opini review film.
3. Mengetahui apakah klasifikasi SVM dapat memberikan hasil yang lebih baik dengan penambahan metode seleksi fitur PSO.

1.5. Metodologi

Tahap yang dilakukan untuk menyelesaikan tugas akhir ini adalah sebagai berikut :

1. **Penyusunan Proposal Tugas Akhir**

   Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal-hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah yang ditentukan dalam pengerjaan tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir

mulai dari penyusunan proposal, studi literatur, analisis dan desain perangkat lunak, implementasi perangkat lunak, pengujian dan evaluasi, dan penyusunan buku tugas akhir. Selain itu, jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir juga diberikan.

2. Studi Literatur

Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan aplikasi, yaitu ekstraksi fitur dengan pemrosesan teks menggunakan *case folding*, *tokenization*, *stopwords removal*, dan pembobotan TF-IDF. Selain itu, ada juga referensi lain yang wajib dipelajari, yaitu metode klasifikasi *Support Vector Machine*, metode seleksi fitur *Particle Swarm Optimization*, metode validasi *k-fold cross validation*, dan metode evaluasi *confusion matrix*.

3. Implementasi

Aplikasi ini akan dibangun menggunakan bahasa pemrograman JAVA. Selain itu, akan digunakan *library* LIBSVM untuk melakukan klasifikasi. Aplikasi ini juga akan dibangun dengan menggunakan *Integrated Development Environment* (IDE) Netbeans 8.1 untuk melakukan pembacaan data, melakukan semua tahap ekstraksi fitur dan pemrosesan teks, melakukan klasifikasi, melakukan seleksi fitur, melakukan pengujian, dan melakukan evaluasi. Selain itu, IDE Netbeans 8.1 mempunyai *Java Database Connectivity* (JDBC) yang digunakan untuk melakukan penyimpanan data fitur, dokumen, nilai TF, nilai DF, nilai IDF, nilai TF-IDF, kata *stopword*, dan kata sentimen.

4. Uji Coba dan Evaluasi

Pada tahap ini, dilakukan uji coba aplikasi dan evaluasi terhadap implementasi metode yang digunakan. Pengujian ini mengukur kemampuan aplikasi dalam melakukan

pengelompokan sentimen opini. Sebelum melakukan pengujian, pada dataset dilakukan *cross validation* dengan nilai k sama dengan 5. Pengujian ini meliputi perhitungan nilai akurasi terhadap dataset yang diuji cobakan, dan perhitungan nilai *precision* dan *recall* untuk mengukur kinerja sistem.

5. **Penyusunan Buku Tugas Akhir**

   Tahap ini merupakan tahap dokumentasi dari pembuatan tugas akhir. Buku tugas akhir berisi dasar teori, perancangan, implementasi, hasil uji coba, dan evaluasi dari aplikasi yang dibangun.

1.6. Sistematika Penulisan

Buku tugas akhir ini terdiri atas beberapa bab yang tersusun secara sistematis, yaitu sebagai berikut :

1. Bab I. Pendahuluan

   Bab pendahuluan berisi penjelasan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan pembuatan tugas akhir, metodologi pembuatan tugas akhir, dan sistematika penulisan tugas akhir.

2. Bab II. Tinjauan Pustaka

   Bab tinjauan pustaka berisi penjelasan mengenai dasar teori yang mendukung pengerjaan tugas akhir. Tinjauan pustaka pada tugas akhir ini meliputi pembahasan tentang penggalian opini, pembahasan tentang ekstraksi fitur dengan pemrosesan teks yang memiliki 4 proses, metode klasifikasi *Support Vector Machine* dan penerapannya dalam klasifikasi teks, baik menggunakan metode seleksi fitur *Particle Swarm Optimization* maupun tanpa menggunakan metode seleksi fitur *Particle Swarm Optimization*, metode

validasi menggunakan *k-fold cross validation*, dan metode evaluasi menggunakan *confusion matrix*.

3. Bab III. Analisis dan Perancangan

   Bab analisis dan perancangan berisi penjelasan mengenai dataset yang digunakan untuk pengujian, bentuk data keluaran dari hasil klasifikasi yang akan dilakukan, ekstraksi fitur dengan pemrosesan teks yang meliputi *case folding*, *tokenization*, *stopwords removal*, pembobotan TF-IDF, perancangan sistem klasifikasi *Support Vector Machine*, menggunakan metode seleksi fitur *Particle Swarm Optimization* maupun tanpa menggunakan metode seleksi fitur *Particle Swarm Optimization*, dan perancangan halaman antar muka pengguna.

4. Bab IV. Implementasi

   Bab implementasi berisi pembangunan implementasi klasifikasi opini review film menggunakan metode *Support Vector Machine*, baik menggunakan metode seleksi fitur *Particle Swarm Optimization* maupun tanpa menggunakan metode seleksi fitur *Particle Swarm Optimization* sesuai dengan rumusan dan batasan masalah yang sudah dijelaskan pada bagian pendahuluan. Implementasi berisi lingkungan perangkat implementasi, kode program dari algoritma-algoritma yang digunakan, proses-proses apa saja yang terlibat, keluaran dari masing-masing proses, dan implementasi antar muka pengguna.

5. Bab V. Pengujian dan Evaluasi

   Bab uji coba ddan evaluasi berisi pembahasan mengenai hasil dari uji coba yang dilakukan terhadap aplikasi klasifikasi opini review film menggunakan metode *Support Vector Machine* dengan dan tanpa menggunakan metode seleksi fitur *Particle Swarm Optimization*. Selain pengujian

akurasi, dilakukan evaluasi menggunakan *confusion matrix* untuk mengetahui kinerja sistem.

6. Bab VI. Kesimpulan dan Saran

   Bab kesimpulan dan saran berisi kesimpulan hasil penelitian. Selain itu, bagian ini berisi saran untuk pengerjaan lebih lanjut atau permasalahan yang dialami dalam proses pengerjaan tugas akhir.

[Halaman ini sengaja dikosongkan]

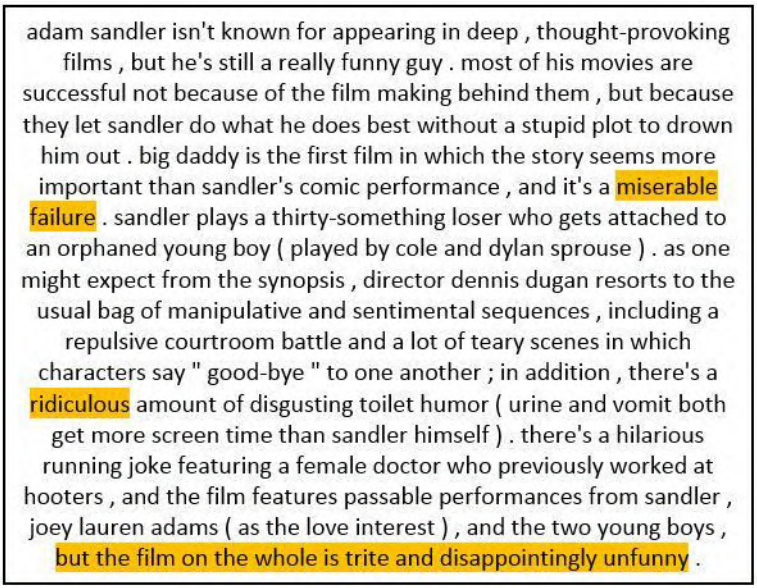
BAB II
TINJAUAN PUSTAKA

Bab tinjauan pustaka berisi mengenai penjelasan teori yang berkaitan dengan implementasi perangkat lunak. Penjelasan tersebut bertujuan untuk memberikan gambaran mengenai sistem yang akan dibangun dan berguna sebagai pendukung dalam pengembangan perangkat lunak. Sub bab pertama membahas tentang penjelasan mengenai penggalian opini. Sub bab kedua membahas tentang ekstraksi fitur yang disertai dengan pemrosesan teks. Terdapat beberapa proses di dalamnya, diantaranya *case folding*, *tokenization*, *stopwords removal*, dan pembobotan TF-IDF. Sub bab ketiga membahas tentang penjelasan algoritma *Support Vector Machine* (SVM) yang sering digunakan dalam klasifikasi teks. Sub bab keempat membahas tentang penjelasan algoritma *Particle Swarm Optimization* (PSO) yang akan digunakan untuk mengoptimalkan kinerja SVM. Sub bab terakhir membahas tentang penjelasan metode validasi dan evaluasi yang digunakan. Metode pengujian menggunakan *k-fold cross validation* dan metode evaluasi menggunakan *confusion matrix*.

2.1. Penggalian Opini

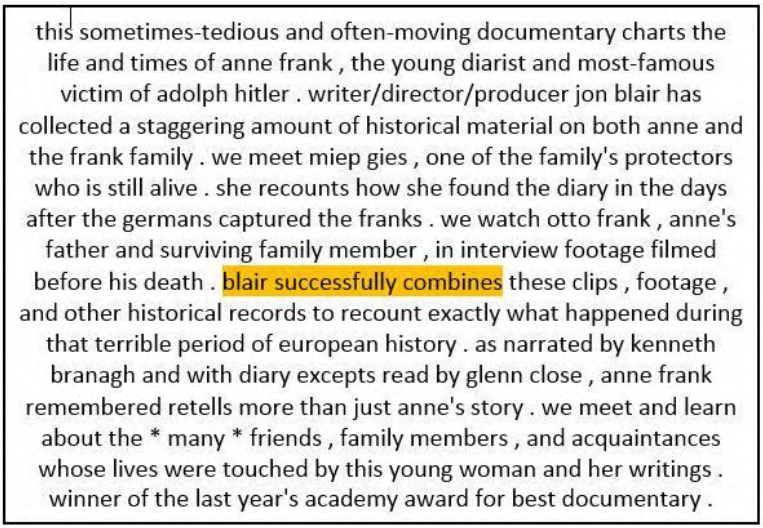
Penggalian opini dapat didefinisikan sebagai sub disiplin komputasi linguistik yang berfokus pada penggalian opini masyarakat melalui website. Penggalian opini merupakan tugas pemrosesan bahasa natural dan ekstraksi informasi yang bertujuan untuk mengetahui perasaan penulis opini yang dinyatakan dalam komentar positif atau negatif dengan menganalisis sejumlah besar dokumen. Secara umum, analisis sentimen bertujuan untuk menentukan sikap pembicara atau penulis sehubungan dengan beberapa topik atau nada suara keseluruhan dokumen. Dalam beberapa tahun ini, terdapat peningkatan eksponensial dalam penggunaan internet dan penyampaian opini publik. Hal ini menjadi faktor pendorong munculnya penggalian opini. Web adalah gudang raksasa data, baik terstruktur maupun tidak

terstruktur sehingga analisis data opini publik menjadi tugas yang menantang [3].


adam sandler isn't known for appearing in deep , thought-provoking films , but he's still a really funny guy . most of his movies are successful not because of the film making behind them , but because they let sandler do what he does best without a stupid plot to drown him out . big daddy is the first film in which the story seems more important than sandler's comic performance , and it's a miserable failure . sandler plays a thirty-something loser who gets attached to an orphaned young boy (played by cole and dylan sprouse) . as one might expect from the synopsis , director dennis dugan resorts to the usual bag of manipulative and sentimental sequences , including a repulsive courtroom battle and a lot of teary scenes in which characters say " good-bye " to one another ; in addition , there's a ridiculous amount of disgusting toilet humor (urine and vomit both get more screen time than sandler himself) . there's a hilarious running joke featuring a female doctor who previously worked at hooters , and the film features passable performances from sandler , joey lauren adams (as the love interest) , and the two young boys , but the film on the whole is trite and disappointingly unfunny .

Gambar 2.1 Contoh Review Film yang tergolong dalam Opini Negatif

Sentimen opini pada tugas akhir ini akan mengklasifikasikan dataset review film menjadi review positif (menunjukkan kebahagiaan atau kepuasan penulis opini) atau review negatif (menunjukkan keadaan sedih, kecewa atau kesal di pihak penulis). Sentimen dapat diberikan skor berdasarkan tingkatannya dari negatif hingga positif. Semakin kecil nilai sentimen, maka sentimen besar kemungkinan termasuk sentimen negatif, begitu pula sebaliknya. Ekspansi web mendorong pengguna internet untuk berkontribusi dan mengekspresikan dirinya sendiri melalui blog, video, situs jejaring sosial. Semua platform ini menyediakan sejumlah besar informasi berharga yang menarik untuk dianalisis [3].


this sometimes-tedious and often-moving documentary charts the life and times of anne frank , the young diarist and most-famous victim of adolph hitler . writer/director/producer jon blair has collected a staggering amount of historical material on both anne and the frank family . we meet miep gies , one of the family's protectors who is still alive . she recounts how she found the diary in the days after the germans captured the franks . we watch otto frank , anne's father and surviving family member , in interview footage filmed before his death . blair successfully combines these clips , footage , and other historical records to recount exactly what happened during that terrible period of european history . as narrated by kenneth branagh and with diary excepts read by glenn close , anne frank remembered retells more than just anne's story . we meet and learn about the * many * friends , family members , and acquaintances whose lives were touched by this young woman and her writings . winner of the last year's academy award for best documentary .

Gambar 2.2 Contoh Review Film yang tergolong dalam Opini Positif

2.2. Ekstraksi Fitur dengan pemrosesan teks

Ekstrasi fitur merupakan proses yang sangat penting dalam penggalian data. Proses ini dilakukan untuk menghilangkan data-data yang tidak diperlukan dalam komputasi. Pemrosesan teks merupakan proses yang dilakukan untuk mendapatkan fitur yang akan digunakan dalam klasifikasi. Ekstraksi fitur dengan pemrosesan teks yang dilakukan pada tugas akhir ini dibagi menjadi empat proses, yaitu proses *case folding*, proses *tokenization*, proses *stopwords removal*, proses pembobotan fitur TF-IDF.

Dataset teks yang dimasukkan umumnya tidak memiliki bentuk teks yang seragam. *Case folding* merubah bentuk setiap huruf abjad dalam teks dari tidak seragam menjadi seragam. Ada 2 metode yang dapat digunakan dalam proses *case folding*, yaitu *lowercase* dan *uppercase*. *Lowercase* adalah metode untuk

mengubah semua huruf abjad dalam teks menjadi huruf kecil sedangkan *uppercase* adalah metode untuk mengubah semua huruf abjad dalam teks menjadi huruf besar. Contoh data teks sebelum dan setelah dilakukan *case folding* dengan metode *lowercase* dapat dilihat pada Gambar 2.3 dan Gambar 2.4.

Tokenization adalah proses memecah aliran teks menjadi kata, frasa, simbol, atau elemen bermakna lain yang disebut token. Aliran teks dipecah menggunakan elemen pemisah. Daftar token

I guess if a film has magic, I don't need it to be fluid or seamless. It can skip background information, go too fast in some places, too slow in others, etc. Magic in this film: the scene in the library.

Gambar 2.3 Contoh teks sebelum *case folding*

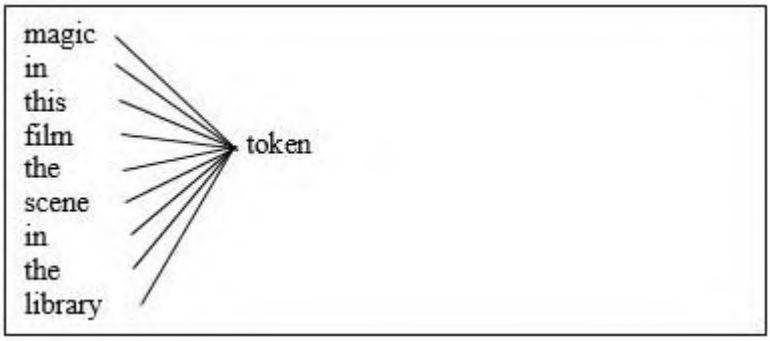
i guess if a film has magic, i don't need it to be fluid or seamless. It can skip background information, go too fast in some places, too slow in others, etc. magic in this film: the scene in the library.

Gambar 2.4 Contoh teks setelah *case folding*

kemudian menjadi masukan untuk diproses lebih lanjut pada proses *stopwords removal*. Contoh data teks sebelum dan setelah dilakukan *tokenization* dengan elemen pemisah berupa spasi (‘ ’) dapat dilihat pada Gambar 2.5 dan Gambar 2.6.

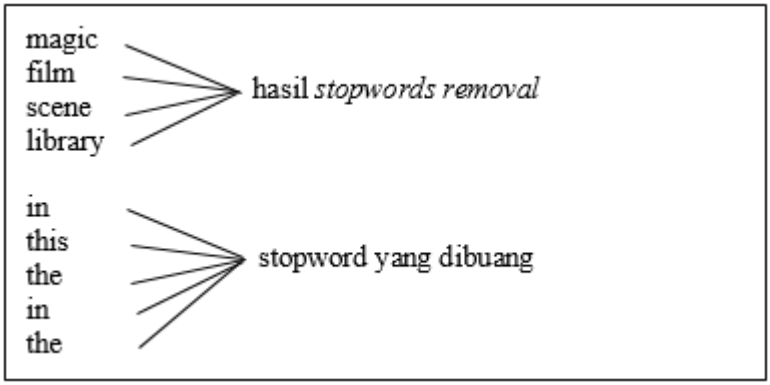
magic in this film: the scene in the library.

Gambar 2.5 Contoh teks sebelum *tokenization*



Gambar 2.6 Contoh hasil *tokenization*

Stopwords Removal adalah proses menghapus kata-kata yang sering muncul pada kumpulan teks sehingga dianggap tidak penting untuk mempercepat proses komputasi. Proses ini juga bertujuan agar komputasi dapat berjalan maksimal. Contoh daftar token sesudah dilakukan *stopwords removal* dapat dilihat pada Gambar 2.7. Contoh daftar kata bahasa inggris yang termasuk *stopword* dapat dilihat pada Tabel 3.1.



Gambar 2.7 Contoh hasil *stopwords removal*

Tabel 2.1 Contoh kata *stopword* berbahasa inggris

about	indeed	que
behind	just	really
came	keep	seen
does	like	tends
eg	maybe	until
former	near	vols
given	okay	what
having	page	zero

Dari pemrosesan teks yang telah dilakukan, dihasilkan fitur-fitur yang akan digunakan untuk klasifikasi. Sebelum masuk tahap klasifikasi, proses yang harus dilakukan terlebih dahulu adalah pembobotan fitur. Salah satu metode pembobotan fitur adalah *Term Frequency-Inverse Document Frequency* (TF-IDF). TF-IDF merupakan metode pembobotan yang banyak digunakan sebagai metode pembanding terhadap metode pembobotan baru. Pada metode ini, perhitungan bobot dilakukan dengan menggunakan persamaan 2.1.

$$W_{dt} = tf_{dt} * idf_t \tag{2.1}$$

W_{dt} merupakan nilai bobot dokumen ke-*d* terhadap kata ke-*t*. tf_{dt} merupakan nilai *term frequency* (TF) dokumen ke-*d* terhadap kata ke-*t*. idf_t merupakan nilai *inverse document frequency* (IDF) terhadap kata ke-*t*. TF merupakan jumlah kemunculan dari suatu kata tersebut pada suatu dokumen. Semakin besar jumlah kemunculan suatu kata dalam kumpulan dokumen, semakin besar pula nilai TF kata tersebut. Semakin besar nilai TF suatu kata, semakin besar pula bobotnya. Contoh hasil perhitungan TF dapat dilihat pada Tabel 2.2.

IDF merupakan nilai invers dari *document frequency* (DF) yang berfungsi mengurangi bobot suatu term jika kemunculannya banyak tersebar di seluruh dokumen. DF merupakan jumlah

dokumen dimana suatu kata muncul. Semakin besar jumlah dokumen yang memunculkan suatu kata, semakin kecil nilai IDF. Semakin kecil nilai IDF, semakin kecil nilai bobot kata tersebut. Perhitungan nilai IDF dapat dilihat pada persamaan 2.2. idf_t merupakan nilai IDF dari kata ke-*t*. df_t merupakan nilai DF dari kata ke-*t*. N merupakan jumlah dokumen yang ada. Contoh hasil perhitungan IDF dapat dilihat pada Tabel 2.3.

Tabel 2.2 Contoh hasil perhitungan TF

Kata	Dokumen	TF
Admire	cv002_17424.txt	2
Admire	cv020_9234.txt	3
Admire	cv245_8938.txt	1
Good	cv002_17424.txt	4
Good	cv020_9234.txt	1
Good	cv245_8938.txt	2
Bad	cv002_17424.txt	0
Bad	cv020_9234.txt	1
Bad	cv245_8938.txt	0
Best	cv002_17424.txt	2
Best	cv020_9234.txt	1
Best	cv245_8938.txt	2

$$idf_t = \log_{10}(\frac{N}{df_t}) \tag{2.2}$$

Tabel 2.3 Contoh hasil perhitungan IDF

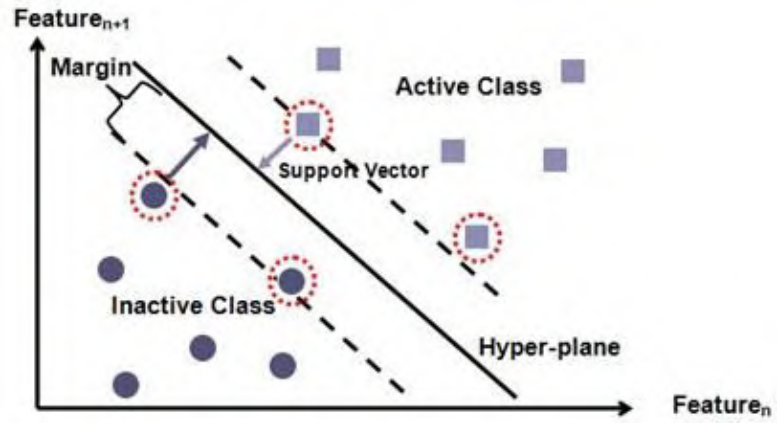
Kata	Nilai DF	Nilai IDF
Adore	7	2.46
Bright	87	1.36
charisma	43	1.67
dependable	4	2.70
Easy	183	1.04

Kata	Nilai DF	Nilai IDF
Genius	68	1.47
Humor	287	0.84
Improve	17	2.07
Overture	1	3.30
Reform	1	3.30

2.3.Suppor Vector Machine

Support Vector Machine (SVM) adalah metode yang dikembangkan oleh Boser, Guyon, Vapnik dan pertama kali diperkenalkan pada tahun 1992. Prinsip dasar SVM adalah *linear classifier* yang kemudian dikembangkan agar dapat bekerja pada *non-linear problem* dengan memasukkan konsep *kernel trick* pada ruang kerja berdimensi tinggi.

Konsep dasar SVM dapat dijelaskan secara sederhana sebagai usaha untuk mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua kelas pada *input space*. SVM menjamin untuk memaksimalkan jarak antara data yang paling dekat dengan *hyperplane*. *Hyperplane* adalah pemisah terbaik antara dua kelas



Gambar 2.8 Contoh alternatif *hyperplane*

atau lebih yang dapat ditemukan dengan mengukur *margin hyperplane* dan mencari titik maksimalnya. Dalam ruang 2 dimensi, *hyperplane* direpresentasikan dengan garis pemotong. Dalam ruang berdimensi lebih dari 2, *hyperplane* direpresentasikan dengan bidang pemotong. *Margin* adalah jarak antara *hyperplane* dengan data terdekat dari masing-masing kelas. Data yang paling dekat dengan *hyperplane* disebut sebagai *support vector*. Data dikatakan sebagai *support vector* apabila data tegak lurus dengan *hyperplane*. Contoh klasifikasi menggunakan metode SVM dapat dilihat pada Gambar 2.8.

Diberikan data masukan $x_i \in \Re^d$ dan masing-masing kelas dinotasikan $y_i \in \{-1, +1\}$ untuk $i = \{1,2,3, \dots, n\}$ dimana n adalah banyaknya data. Fungsi *hyperplane* dibuat dengan persamaan 2.3. Data x_i yang termasuk kelas -1 dapat dirumuskan sebagai data yang memenuhi pertidaksamaan 2.4. Data x_i yang termasuk kelas +1 dirumuskan sebagai data yang memenuhi pertidaksamaan 2.5.

$$w \,.\, x + b = 0 \tag{2.3}$$

$$w \,.\, x_i + b \leq -1 \tag{2.4}$$

$$w \,.\, x_i + b \geq +1 \tag{2.5}$$

Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara *hyperplane* dan titik terdekatnya dengan rumus 2.6. Hal tersebut dapat diselesaikan dengan mencari titik minimal persamaan dengan memperhatikan *constraint* persamaan 2.7 dan persamaan 2.8.

$$\frac{1}{||w||^2} \tag{2.6}$$

$$\min \tau(w) = \frac{1}{2}||w||^2 \tag{2.7}$$

$$y_i(x_i.w + b) - 1 \geq 0, \forall i \tag{2.8}$$

Masalah ini dapat dipecahkan dengan berbagai teknik komputasi di antaranya *Lagrange Multiplier*. α adalah *Lagrange multipliers*, yang bernilai nol atau positif ($\alpha_i \geq 0$). Nilai optimal dari persamaan 2.9 dapat dihitung dengan meminimalkan *L* terhadap *w* dan b, dan memaksimalkan *L* terhadap α_i. Dengan memperhatikan bahwa pada titik optimal *L=0*, persamaan 2.9 dapat dimodifikasi sebagai maksimalisasi *problem* yang hanya mengandung α_i saja, sebagaimana persamaan maksimasi 2.10 dengan *constraint* 2.11. Hasil dari perhitungan ini diperoleh α_i yang kebanyakan bernilai positif. Data yang berkorelasi dengan α_i yang positif inilah yang disebut sebagai *support vector*.

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i\big((x_i.w + b) - 1\big)) \tag{2.9}$$

$$\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.10}$$

$$\alpha_i \geq 0 \ (i = 1,2, \dots, n) \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{2.11}$$

2.4. Particle Swarm Optimization

Particle Swarm Optimization (PSO) adalah teknik optimasi berbasis populasi yang dikembangkan oleh Eberhart dan Kennedy pada tahun 1995, yang terinspirasi oleh perilaku sosial kawanan burung atau ikan [4]. PSO dapat diasumsikan sebagai kelompok burung yang bersama-sama mencari makanan di suatu daerah.

Burung-burung tersebut tidak tahu di mana makanan berada, tetapi mereka tahu sebarapa jauh makanan itu berada. Jadi, strategi terbaik untuk menemukan makanan tersebut adalah dengan mengikuti burung yang terdekat dari makanan tersebut [5].

PSO digunakan untuk memecahkan masalah optimasi. Hampir sama dengan *genetic algorithm* (GA), PSO melakukan pencarian menggunakan populasi (*swarm*) dari individu (partikel) yang akan diperbaharui dari iterasi. PSO memiliki beberapa parameter seperti posisi, kecepatan, kecepatan maksimum, konstanta percepatan, dan berat inersia. PSO memiliki perbandingan lebih atau bahkan pencarian kinerja yang lebih unggul untuk banyak masalah optimasi dengan lebih cepat dan tingkat konvergensi yang lebih stabil [4].

PSO diinialisasi dengan membuat populasi partikel. Setiap partikel diperlakukan sebagai titik dalam ruang *s*-dimensi. Posisi partikel ke-i direpresentasikan dengan persamaan 2.12. Posisi terbaik ($pbest$, posisi yang didapatkan dari nilai fitness terbaik) dari semua partikel dalam satu populasi direpresentasikan dengan persamaan 2.13. Indeks dari partikel terbaik dari keseluruhan populasi direpresentasikan dengan $gbest$. Kecepatan dari partikel ke-i direpresentasikan dengan persamaan 2.14.

$$x_i = (x_{i1}, x_{i2}, x_{i3}, \dots, x_{is}) \tag{2.12}$$

$$p_i = (p_{i1}, p_{i2}, p_{i3}, \dots, p_{is}) \tag{2.13}$$

$$v_i = (v_{i1}, v_{i2}, v_{i3}, \dots, v_{is}) \tag{2.14}$$

Partikel dimanipulasi dengan persamaan 2.15 dan persamaan 2.16 dimana v_{id} merupakan kecepatan partikel ke-i pada dimensi ke-d. w merupakan berat inersia, konstanta percepatan c_1 dan c_2 dalam persamaan (2.15) merupakan pembobotan dari percepatan *stochastic* dari term yang menarik setiap pertikel menuju posisi $pbest$ dan $gbest$. $rand()$ merupakan fungsi yang menentukan

nilai secara acak dalam rentang [0,1]. p_{id} merupakan posisi partikel terbaik ke-i pada dimensi ke-d. $gbest_d$ adalah posisi partikel terbaik dari keseluruhan partikel pada dimensi ke-d. Kecepatan pada setiap dimensi terbatas pada kecepatan maksimum v_{max}.

$$v_{id} = w * v_{id} + c_1 * rand() * (p_{id} - x_{id}) + c_2 * rand() * (gbest_d - x_{id}) \quad (2.15)$$

$$x_{id} = x_{id} + v_{id} \quad (2.16)$$

PSO yang asli pada dasarnya dikembangkan untuk masalah optimasi berkelanjutan. Untuk menampilkan F_s, konsep standar PSO perlu diperpanjang untuk menangani data biner. Secara khusus, pencarian ruang dimensi ke-d mungkin terbatas pada beberapa langkah dan fungsi *fitness* merupakan fungsi diskrit. Beberapa versi fungsi diskrit dan biner PSO banyak diusulkan. Salah satunya ialah fungsi PSO untuk melakukan seleksi fitur. Dengan seleksi fitur, akan didapatkan fitur-fitur yang mempunyai peran penting dalam proses komputasi sehingga akan didapatkan nilai akurasi yang lebih tinggi.

2.5. Validasi dan Evaluasi

Metode validasi yang digunakan pada pembuatan tugas akhir ini adalah *cross validation*. *Cross validation* merupakan salah satu teknik validasi keakuratan sebuah model yang dibangun berdasarkan dataset tertentu. Dataset yang digunakan dalam proses pembuatan model disebut *data training* dan dataset yang digunakan untuk melakukan validasi terhadap model yang telah dibentuk disebut *data testing*.

Salah satu metode *cross validation* yang biasa digunakan adalah *k-fold cross validation*. Metode ini membagi dataset menjadi *k* bagian data yang sama besar. Kemudian dilakukan perhitungan akurasi sebanyak *k* kali dimana satu bagian data

digunakan sebagai *data testing*, sedangkan sisanya digunakan sebagai *data training*. Pada setiap iterasi, *data testing* yang digunakan berbeda. Hal ini mengakibatkan setiap bagian data pernah digunakan sebagai *data testing* sehingga *data testing* yang digunakan di setiap iterasi menjadi berbeda. Nilai akurasi akhir dapat dihitung dengan mengambil nilai rataan dari nilai akurasi yang dihasilkan dari perhitungan pada seluruh iterasi. Rumus perhitungan nilai akurasi akhir (*fa*) dapat dilihat pada persamaan 2.17. Contoh pembagian data pada *k-fold cross validation* dengan nilai *k* sama dengan 5 dapat dilihat pada Tabel 2.4.

$$fa = \frac{akurasi_1 + akurasi_2 + \cdots + akurasi_k}{k} \qquad (2.17)$$

Tabel 2.4 Contoh pembagian data pada *k-fold cross validation* dengan nilai *k* sama dengan 5

Iterasi ke-	Data training	Data testing
1	K2, K3, K4, K5	K1
2	K1, K3, K4, K5	K2
3	K1, K2, K4, K5	K3
4	K1, K2, K3, K5	K4
5	K1, K2, K3, K4	K5

Metode evaluasi yang digunakan untuk menghitung kinerja sistem pada aplikasi tugas akhir ini adalah *confusion matrix*. Setiap kolom dari matriks menunjukkan contoh dalam kelas yang sebenarnya, sementara setiap baris menunjukkan contoh dalam kelas yang diprediksi. Ini membantu untuk mengetahui apakah algoritma klasifikasi umumnya *mislabeling* satu dengan yang lain. Ukuran kinerja ini fokus pada performa hasil luaran berdasarkan kelas yang ada. Pada Tabel 2.5, dapat dilihat *confusion matrix* untuk *classifier* dua kelas dan pada Tabel 2.6, dapat dilihat contoh penyajian *confusion matrix* untuk *classifier* dua kelas dengan

jumlah data positif dan negatif masing-masing 500 dokumen. 4 jenis entri data yang digunakan adalah sebagai berikut :

(a) *True positive* (TP) adalah jumlah dokumen 'positif' yang dikategorikan 'positif'.
(b) *False positive* (FP) adalah jumlah dokumen 'negatif' yang dikategorikan 'positif'.
(c) *False negative* (FN) adalah jumlah dokumen 'positif' yang dikategorikan 'negatif'.
(d) *True negative* (TN) adalah jumlah dokumen 'negatif' yang dikategorikan 'negatif'.

Tabel 2.5 *Confusion matrix* untuk *classifier* dua kelas

	Kelas yang sebenarnya	
	Positif	Negatif
Kelas yang diprediksi		
Positif	TP	FP
Negatif	FN	TN

Tabel 2.6 Contoh penyajian *confusion matrix* untuk *classifier* dua kelas

	Kelas yang sebenarnya	
	Positif	Negatif
Kelas yang diprediksi		
Positif	348	96
Negatif	152	404

Dalam *confusion matrix*, dilakukan perhitungan nilai akurasi, *precision*, dan *recall*. Akurasi adalah rasio jumlah dokumen yang telah diklasifikasikan dengan benar ke jumlah total dokumen saat ini [2]. Akurasi digunakan untuk mengukur performa keseluruhan sistem. Persamaan untuk menghitung nilai akurasi dapat dilihat pada persamaan 2.18.

$$akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.18)$$

Precision adalah rasio *instance* (dokumen) positif yang diprediksi benar positif [2]. *Precision* didapatkan dengan menghitung jumlah data yang dapat diklasifikan dengan benar pada suatu kelas dibagi dengan jumlah data yang diklasifikasikan sebagai kelas tersebut. *Precision* bertujuan untuk mengetahui tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Persamaan untuk menghitung nilai *precision* dapat dilihat pada persamaan 2.19.

$$precision = \frac{TP}{TP + FP} \quad (2.19)$$

Recall adalah rasio *instance* (dokumen) positif yang ditemukan dengan benar [2]. *Recall* didapatkan dengan menghitung jumlah data yang dapat diklasifisikan dengan benar pada suatu kelas dibagi dengan jumlah data yang tergolong pada kelas tersebut. *Recall* bertujuan untuk mengetahui tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Persamaan untuk menghitung nilai *recall* dapat dilihat pada persamaan 2.20.

$$recall = \frac{TP}{TP + FN} \quad (2.20)$$

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Pada bab 3 ini, akan dijelaskan analisis dan perancangan perangkat lunak untuk mencapai tujuan tugas akhir. Terdapat 3 sub bab pada bab ini. Sub bab pertama menjelaskan tentang analisis implementasi metode yang digunakan secara umum. Proses-proses apa saja yang ada dalam aplikasi, mulai dari proses pembacaan data, proses ekstraksi fitur dengan pemrosesan teks, proses klasifikasi SVM, proses klasifikasi SVM dengan menambahkan seleksi fitur PSO, proses validasi, sampai proses evaluasi. Sub bab kedua menjelaskan tentang perancangan dari setiap proses yang ada. Dari inputan yang ada, dihasilkan output yang akan masuk pada proses selanjutnya. Sub bab ketiga menjelaskan tentang perancangan antar muka aplikasi. Antar muka ini akan menghubungkan pengguna dengan sistem pada aplikasi.

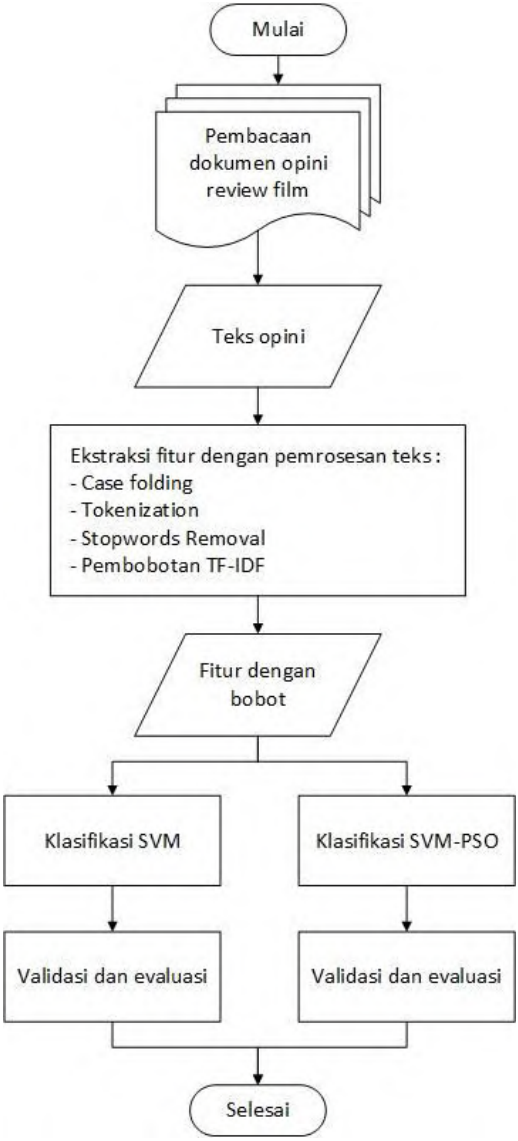
3.1. Analisis Implementasi Metode Secara Umum

Pada tugas akhir ini, akan dibangun sebuah sistem untuk melakukan klasifikasi opini review film. Ada dua metode yang digunakan untuk melakukan klasifikasi, yaitu metode *Support Vector Machine* (SVM) dan metode SVM dengan penambahan metode seleksi fitur *Particle Swarm Optimization* (PSO). Proses-proses yang terlibat di dalam implementasi sistem ini meliputi tahap pembacaan data review, tahap ekstraksi fitur dengan pemrosesan teks, tahap klasifikasi, dan tahap validasi dan evaluasi.

Tahap pertama adalah tahap pembacaan opini review film. Dataset terdiri dari dokumen-dokumen yang berisi satu review di setiap dokumennya. Setelah semua dokumen pada dataset telah dibaca dan dikumpulkan, dilakukan tahap ekstraksi fitur dengan pemrosesan teks. Di tahap kedua ini, kumpulan teks opini akan diolah sehingga sistem akan mendapatkan fitur-fitur disertai dengan nilai bobotnya masing-masing.

Tahap ini dimulai dengan proses *case folding*. Pada tugas akhir ini, setiap huruf abjad pada setiap teks yang bentuknya *uppercase* akan diubah menjadi bentuk *lowercase*. Tahap selanjutnya adalah proses *tokenization*. Teks akan dipecah menjadi kata yang disebut token. Untuk memisahkan teks, sistem menggunakan elemen pemisah berupa elemen selain huruf abjad dan tanda penghubung (-). Hasil dari *tokenization* ini adalah kumpulan token dari setiap teks. Kumpulan token ini kemudian akan masuk pada proses *stopwords removal*. Pada proses ini, setiap token akan dicocokkan dengan kata yang ada pada daftar *stopword*. Jika cocok, token akan dihapus. Proses ini bertujuan untuk membuat proses komputasi berjalan lebih cepat dengan mengurangi fitur-fitur yang tidak penting. Hasil dari proses *stopwords removal* akan digunakan untuk pemilihan fitur. Pada tugas akhir ini, fitur kata yang diambil adalah fitur dari kata sentimen lexicon [6]. Fitur-fitur hasil pemilihan fitur kemudian akan digunakan untuk klasifikasi. Langkah berikutnya adalah perhitungan bobot dari setiap fitur yang telah dipilih menggunakan metode *term frequency-inverse document frequency* (TF-IDF).

Tahap selanjutnya setelah melakukan ekstraksi fitur dengan pemrosesan teks adalah klasifikasi. Ada dua metode yang digunakan dalam klasifikasi, yaitu metode SVM dan metode SVM dengan penambahan metode seleksi fitur PSO. Pada tugas akhir ini, klasifikasi SVM menggunakan *library* LIBSVM. Inputan yang digunakan pada klasifikasi ada empat, yaitu data bobot *data training*, data kelas *data training*, data bobot *data testing*, dan data kelas *data testing*. Inputan berupa *two dimentional array* dimana pada data bobot, fitur digunakan sebagai baris dan dokumen digunakan sebagai kolom. Pada data kelas, kelas digunakan sebagai baris dan dokumen digunakan sebagai kolom. Keluaran dari klasifikasi SVM adalah nilai prediksi *data testing* yang merupakan hasil pelatihan dari *data training*. Selain nilai prediksi, juga akan dihasilkan jumlah dokumen yang diklasifikasin dengan benar dan salah.



Gambar 3.1 Diagram alur proses sistem secara umum

Metode klasifikasi yang kedua adalah SVM dengan penambahan metode seleksi fitur PSO. Sebelum masuk pada klasifikasi SVM, fitur-fitur yang ada diseleksi terlebih dahulu. Pada tugas akhir ini, jumlah fitur yang diambil dan digunakan dalam proses seleksi sebesar 10% dari total fitur yang ada dengan nilai IDF tertinggi. Selanjutnya dilakukan inisialisasi *swarm* dengan jumlah partikel, iterasi, dan fitur yang ditentukan. Pada tugas akhir ini, jumlah partikel diset 100 partikel, jumlah iterasi diset 200 iterasi, dan jumlah fitur diset 200 fitur [7]. PSO mencari *fitness value* terbesar dari semua partikel pada semua iterasi. Dari *fitness value* terbesar ini, akan diketahui fitur-fitur terbaik yang akan dimasukkan ke dalam klasifikasi SVM.

Tahap selanjutnya adalah tahap validasi dan evaluasi. Validasi digunakan untuk mengukur keakuratan sebuah model yang dibangun berdasarkan dataset tertentu. Metode untuk proses validasi ini adalah *k-fold cross validation*. Metode ini membagi dataset menjadi *k* bagian data yang sama. Kemudian satu bagian data digunakan sebagai *data testing* dan sisanya digunakan sebagai *data training*. Nilai akurasi akhir didapatkan dari nilai rataan dari nilai akurasi pada semua iterasi. Tahap selanjutnya adalah proses evaluasi yang menggunakan metode *confusion matrix*. Dari metode ini, kita dapat mengukur kinerja sistem dengan menghitung *precision* dan *recall*. Proses-proses sistem secara umum dapat dilihat pada Gambar 3.1.

3.2. Perancangan Proses

Sub bab ini akan membahas tentang penjelasan perancangan proses yang dilakukan. Penjelasan perancangan proses bertujuan untuk memberikan gambaran secara rinci pada setiap alur implementasi metode pada aplikasi klasifikasi opini review film. Alur tersebut nantinya akan digunakan dalam tahap implementasi.

3.2.1. Tahap pembacaan dokumen

Pada tahap ini, sistem melakukan pembacaan dokumen yang akan diklasifikasikan. Dokumen yang akan digunakan terdiri dari dokumen positif dan dokumen negatif dengan jumlah yang sama. Setiap dokumen berisi teks yang merupakan opini yang diberikan penonton terhadap suatu film tertentu. Hasil dari pembacaan dokumen dapat dilihat pada Gambar 3.2.

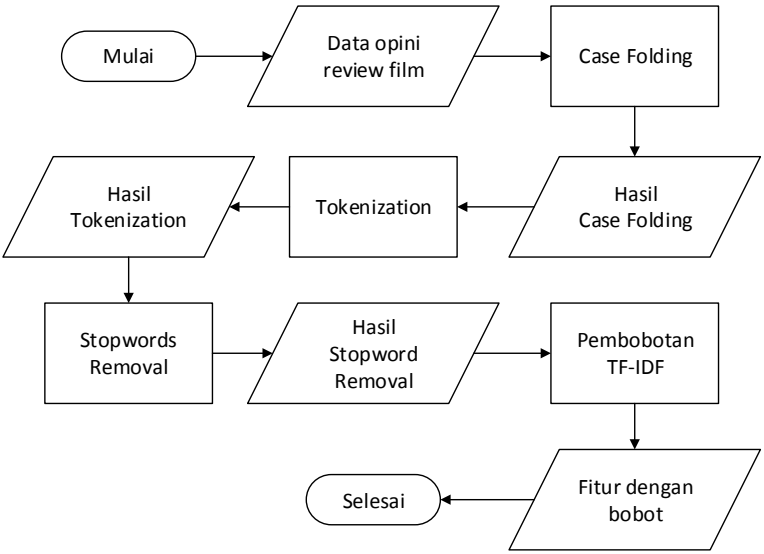
```
1. cv500_10722.txt
you always have to be careful with the first official studio re
2. cv501_12675.txt
synopsis : easily-angered , chainsmoking architect david encoun
3. cv515_18484.txt
when the film features richard lynch in the role of chief villa
4. cv516_12117.txt
what are the warning signs of a * terrible * movie ? making it'
5. cv517_20616.txt
you don't look at a ren ? magritte painting and search for a de
6. cv518_14798.txt
well , as i check my score card for what i've done this holiday
7. cv519_16239.txt
terrence malick made an excellent 90 minute film adaptation of
8. cv520_13297.txt
i cried during _babe_ . i admit it . the special effects , the
```

Gambar 3.2 Hasil pembacaan data

3.2.2. Tahap ekstraksi fitur dengan pemrosesan teks

Pada tahap ini, terdapat beberapa proses. Proses pertama adalah *case folding*. Proses ini mengubah huruf abjad pada teks yang berbentuk *uppercase* akan diubah menjadi bentuk *lowercase*. Proses kedua adalah *tokenization*. Proses ini memecah teks menjadi kata yang disebut sebagai token. Elemen yang digunakan sebagai pemecah teks adalah elemen-elemen selain huruf abjad dan tanda penghubung (-). Proses ketiga adalah *stopwords removal*. Proses ini menghapus kata-kata yang termasuk ke dalam

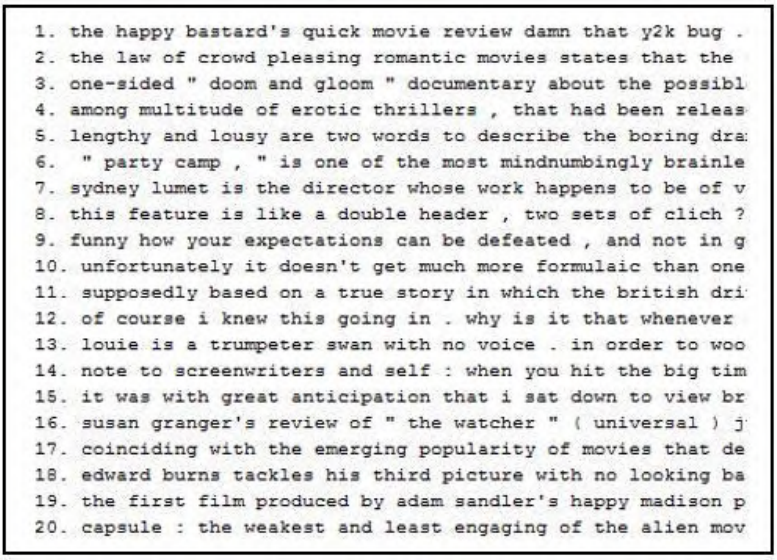
daftar kata stopword. Proses keempat adalah pembobotan TF-IDF (*Term Frequency-Inverse Document Frequency*). Proses ini membuat setiap fitur yang telah dipilih memiliki nilai bobot yang akan digunakan dalam tahap klasifikasi. Proses-proses pada tahap ini dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram alur ekstraksi fitur dengan pemrosesan teks

3.2.2.1. Proses *Case Folding*

Case folding adalah proses pertama dalam tahap ekstraksi fitur dengan pemrosesan teks. *Case folding* mengubah huruf yang berbentuk *uppercase* pada teks menjadi bentuk *lowercase*. Hal ini bertujuan agar pemrosesan teks selanjutnya dapat lebih maksimal. Hasil proses *case folding* dari data teks dapat dilihat pada Gambar 3.4.



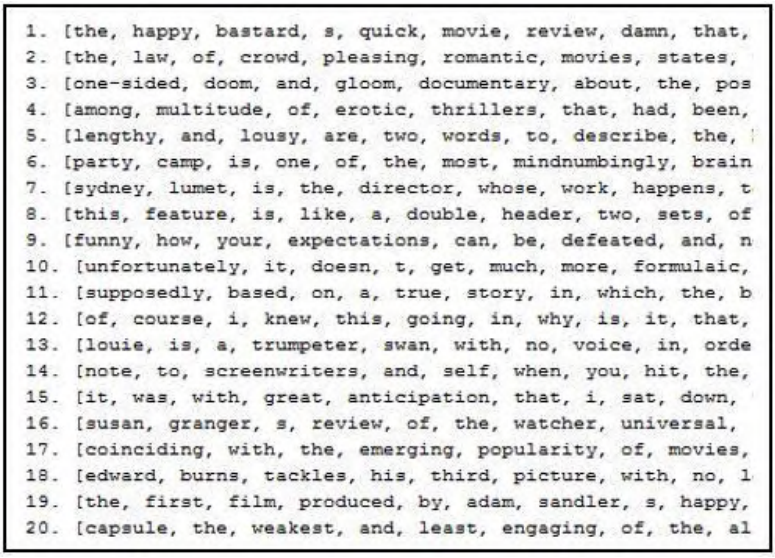
Gambar 3.4 Hasil proses *case folding*

3.2.2.2. Proses *Tokenization*

Tokenization adaah proses kedua setelah *case folding*. Pada proses ini, sistem memecah teks menjadi kata atau frasa. Elemen yang digunakan untuk memecah teks adalah elemen-elemen selain huruf abjad dan tanda penghubung (-). Hasil dari *tokenization* ini adalah setiap dokumen memiliki kumpulan token hasil pemecahan teks. Hasil proses *tokenization* dapat dilihat pada Gambar 3.5.

3.2.2.3. Proses *Stopwords Removal*

Stopword merupakan kata yang sering muncul sehingga dianggap tidak penting untuk mempercepat komputasi pada pemrosesan teks. Kosa kata yang termasuk dalam *stopword* bahasa inggris yang digunakan dalam tugas akhir ini adalah *stopword* yang didefinisikan oleh Rank NL. Daftar lengkap kata yang termasuk dalam *stopwords* akan terlampir pada lampiran A.1.

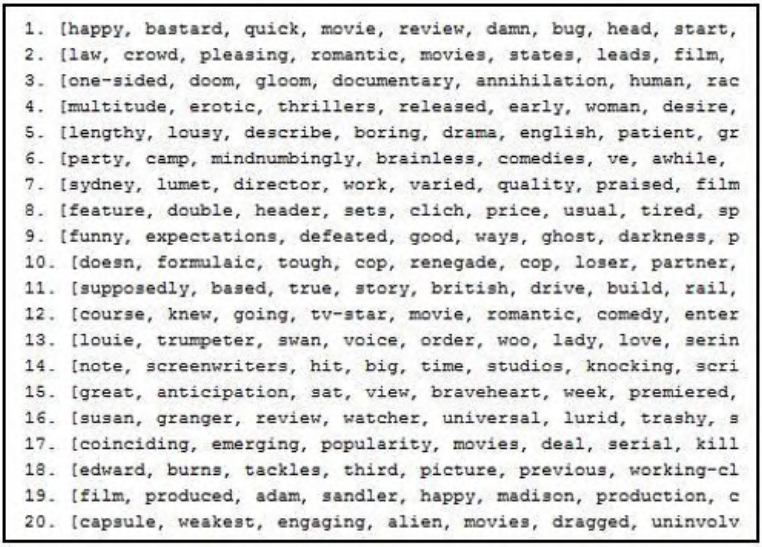


Gambar 3.5 Hasil proses *tokenization*

Proses *stopwords removal* dilakukan setelah melakukan proses *tokenization* pada setiap teks. Setiap token akan dicocokkan dengan kata-kata yang termasuk dalam kata *stopword*. Jika token cocok dengan salah satu kata *stopword*, token akan dihapus. Beberapa kata yang termasuk *stopword* bahasa Inggris yang didefinisikan oleh *Rank NL* dapat dilihat pada Tabel 3.1. Hasil proses *stopwords removal* dapat dilihat pada Gambar 3.6.

Tabel 3.1 Contoh *Stopwords* yang digunakan

across	instead	quite
believe	just	related
contains	known	same
downwards	little	their
enough	meantime	upon
following	neither	various
gotten	out	whatever



Gambar 3.6 Hasil proses *stopwords removal*

3.2.2.4. Proses pembobotan TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) merupakan metode pembobotan yang banyak digunakan. *Term Frequency* (TF) merupakan jumlah kemunculan dari kata tersebut pada setiap dokumen. *Inverse Document Frequency* (IDF) merupakan nilai invers dari *Document Frequency* (DF). IDF berfungsi untuk mengurangi bobot suatu *term* jika kemunculannya banyak tersebar di seluruh dokumen. DF merupakan jumlah dokumen dimana suatu kata muncul. Pada metode ini, perhitungan bobot dilakukan dengan mengalikan TF dengan IDF. Contoh hasil perhitungan DF dan IDF pada *database* dapat dilihat pada Tabel 3.2. Contoh hasil perhitungan TF dan pembobotan TF-IDF pada *database* dapat dilihat pada Tabel 3.3.

Tabel 3.2 Hasil perhitungan DF dan IDF pada *database*

id_term	term	DF	IDF
362	Good	223	0.35
851	Well	189	0.42
1931	Plot	168	0.47
92	Best	153	0.51
958	Bad	131	0.58
488	Love	118	0.63
874	Work	114	0.64
370	Great	112	0.65
1487	Funny	104	0.68
94	Better	93	0.73
336	Fun	71	0.85
1535	Hard	60	0.92
449	Interesting	57	0.94
595	Pretty	56	0.95
1947	Problem	51	0.99

Tabel 3.3 Hasil perhitungan TF dan TF-IDF pada *database*

id_term	id_dokumen	TF	TF-IDF
82	411	9	9.68
362	170	6	2.10
362	233	6	2.10
488	331	6	3.76
526	334	6	13.33
82	449	5	5.38
84	354	5	8.72
362	147	5	1.75
362	324	5	1.75
362	388	5	1.75
362	424	5	1.75
511	253	5	7.22

id_term	id_dokumen	TF	TF-IDF
82	411	9	9.68
535	288	5	5.60
82	285	4	4.30
92	89	4	2.06
92	387	4	2.06
109	82	4	5.01
131	188	4	5.20

3.2.3. Tahap Klasifikasi

Setelah melalui tahap ekstraksi fitur dengan pemrosesan teks, data olahan akan masuk pada tahap klasifikasi. Data olahan akan diklasifikasi menggunakan 2 metode, yaitu metode SVM dan metode SVM dengan penambahan metode seleksi fitur PSO. Hal ini bertujuan untuk mencapai tujuan tugas akhir, yaitu untuk mencari metode mana yang menghasilkan akurasi yang lebih baik.

3.2.3.1. Klasifikasi SVM

Support Vector Machine (SVM) adalah metode learning machine yang bekerja untuk menemukan *hyperplane* terbaik yang memisahkan dua buah kelas pada *input space*. SVM menjamin untuk memaksimalkan jarak antara data yang paling dekat dengan *hyperplane*. Setelah melalui tahap ekstraksi fitur, data yang telah diolah kemudian masuk pada tahap klasifikasi SVM. Proses klasifikasi menggunakan *library* LIBSVM.

Parameter input yang dimasukkan ada empat parameter dalam bentuk *two dimentional array*. Parameter pertama adalah bobot *data training* dengan nama dokumen sebagai kolom dan nama fitur sebagai baris. Parameter kedua adalah kelas *data training* dengan nama dokumen sebagai kolom dan kelas sebagai baris. Parameter ketiga adalah bobot *data testing* dengan nama dokumen sebagai kolom dan nama fitur sebagai baris. Parameter keempat adalah kelas *data testing* dengan nama dokumen sebagai kolom dan kelas sebagai baris.

Ketika input dimasukkan, sistem akan langsung membuat model dari bobot *data training* dan kelas *data training*. Model yang telah dibuat akan digunakan untuk melakukan klasifikasi terhadap *data testing* yang direpresentasikan oleh bobot *data testing*. Kelas yang diklasifikasi ada 2, yaitu kelas negatif (0) dan positif (1). Hasil klasifikasi berupa nilai prediksi dari tiap dokumen. Hasil ini akan dicocokkan dengan kelas *data testing*. Semakin banyak nilai prediksi dokumen yang sama dengan nilai pada kelas *data testing*, maka nilai akurasi akan semakin tinggi.

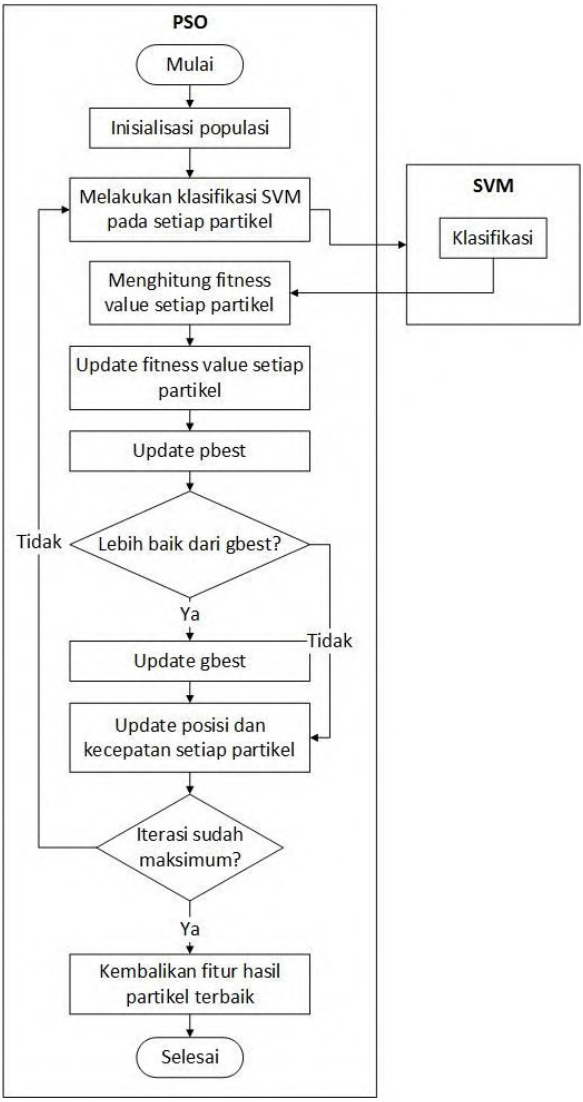
Untuk mendapatkan hasil akurasi yang lebih akurat, digunakan metode *k-fold cross validation*. Metode ini membagi data menjadi *k* bagian. Sistem akan melakukan proses klasifikasi sebanyak *k* kali. Pada setiap proses klasifikasi, *data testing* yang digunakan berbeda. Hal ini menyebabkan semua bagian data pasti pernah menjadi *data testing*. Nilai akurasi akhir dapat dihitung dengan mengambil nilai rataan dari nilai akurasi seluruh iterasi yang dilakukan. Hasil klasifikasi terhadap 500 dokumen wmenggunakan metode SVM dapat dilihat pada Gambar 3.7.

```
==================================================
Klasifikasi SVM BERHASIL.
==================================================
Akurasi Akhir : 50.0
                Relevant        Not Relevant
Retrieved       0               0
Not Retrieved   250             250

Precision : 0.0
Recall    : 0.0
==================================================
```

Gambar 3.7 Hasil klasifikasi menggunakan metode SVM

3.2.3.2. Klasifikasi SVM-PSO



Gambar 3.8 Diagram alur proses seleksi fitur PSO

Particle Swarm Optimization (PSO) adalah teknik optimasi berbasis populasi. PSO melakukan pencarian menggunakan populasi (*swarm*) dari individu (partikel) yang akan diperbaharui dari iterasi. PSO memiliki beberapa parameter seperti posisi, kecepatan, kecepatan maksimum, konstanta percepatan, dan berat inersia. Pada tugas akhir ini, PSO akan digunakan untuk mengoptimasi jumlah fitur yang ada dari 4595 fitur menjadi 200 fitur sehingga dapat menghasilkan nilai akurasi yang lebih baik. Diagram alur proses seleksi fitur PSO dapat dilihat pada Gambar 3.8.

Ada 3 parameter yang perlu diinisialisasi terlebih dahulu sebelum melakukan seleksi fitur PSO, yaitu jumlah partikel (i), jumlah fitur (s), dan jumlah iterasi (n). *Swarm* terdiri dari i partikel (p). Setiap partikel memiliki s fitur. Setiap fitur (f_{is}) memiliki posisi (x_{is}) dan kecepatan (v_{is}). Sistem melakukan inisialisasi awal pada setiap partikel. Fitur diinisialisasi dengan memilih secara acak s fitur dari 10% fitur yang memiliki nilai IDF tertinggi. Nilai awal posisi dari setiap fitur diinisialisasi dengan nilai 0 atau 1. Nilai awal kecepatan juga dinisialisasi secara acak dengan nilai desimal pada rentan antara 0 sampai 1. Contoh di bawah ini adalah seleksi fitur PSO dengan 5 partikel. Setiap partikel memiliki 5 fitur. Inisialisasi fitur, posisi, dan kecepatan pada 5 partikel dapat dilihat berturut-turut pada Gambar 3.9, Gambar 3.10, dan Gambar 3.11.

$$p_1 = (beauty, entertain, heaven, works, loved)$$
$$p_2 = (stunning, critics, charm, talents, solid)$$
$$p_3 = (success, bright, leads, lucky, wild)$$
$$p_4 = (terribly, best, entertain, solid, good)$$
$$p_5 = (shallow, good, beauty, lucky, critics)$$

Gambar 3.9 Inisialisasi fitur

$$x_1 = (0, 1, 1, 1, 1)$$
$$x_2 = (1, 0, 1, 1, 0)$$
$$x_3 = (1, 1, 1, 1, 1)$$
$$x_4 = (0, 1, 1, 0, 0)$$
$$x_5 = (1, 0, 1, 1, 1)$$

Gambar 3.10 Inisialisasi nilai posisi

$$v_1 = (0.34, 0.53, 0.66, 0.71, 0.19)$$
$$v_2 = (0.33, 0.42, 0.61, 0.82, 0.98)$$
$$v_3 = (0.65, 0.61, 0.35, 0.94, 0.59)$$
$$v_4 = (0.05, 0.23, 0.47, 0.64, 0.79)$$
$$v_5 = (0.13, 0.49, 0.03, 0.19, 0.17)$$

Gambar 3.11 Inisialisasi nilai kecepatan

Setelah melakukan inisialisasi fitur beserta nilai kecepatan dan posisinya, sistem melakukan perhitungan *fitness value* awal dengan menggunakan persamaan 3.3. Nilai α dan β merupakan konstanta dengan α ditambah β sama dengan 1. Pada tugas akhir ini, nilai α dan β diset berturut-turut sama dengan 0.85 dan 0.15 [6]. $\gamma(F_i(t))$ merupakan nilai akurasi yang dihasilkan partikel dari fitur-fitur yang memiliki nilai posisi sama dengan 1. Nilai akurasi dihitung dengan menggunakan metode klasifikasi SVM. Berdasarkan Gambar 3.9 dan Gambar 3.10, fitur-fitur yang digunakan untuk klasifikasi pada contoh 5 partikel di atas dapat dilihat pada Gambar 3.13. $|N|$ merupakan jumlah fitur yang diset pada setiap partikel. $|F|$ merupakan jumlah fitur yang memiliki nilai posisinya sama dengan 1 pada partikel. Hasil perhitungan nilai akurasi dapat dilihat pada Gambar 3.12. Hasil perhitungan *fitness value* dapat dilihat pada Gambar 3.14.

$$p_1 = (entertain, heaven, works, loved)$$
$$p_2 = (stunning, charm, talents)$$
$$p_3 = (success, bright, leads, lucky, wild)$$
$$p_4 = (best, entertain)$$
$$p_5 = (shallow, beauty, lucky, critics)$$

Gambar 3.13 Fitur yang digunakan untuk klasifikasi partikel

$$akurasi = (56.35, 53.70, 58.10, 55.40, 51.05)$$

Gambar 3.12 Hasil perhitungan nilai akurasi

$$fitness = \alpha * \gamma\big(F_i(t)\big) + \beta * \frac{|N| - |F|}{|N|} \qquad (3.3)$$

$$fitness = (49.55, 48.75, 51.35, 49.05, 47.95)$$

Gambar 3.14 Hasil perhitungan *fitness value*

Sistem masuk pada iterasi untuk mencari nilai pbest dan gbest. Setiap iterasi terdapat tiga proses. Proses pertama adalah proses untuk memperbarui nilai pbest dan gbest. Pbest adalah *fitness value* terbaik dari suatu partikel pada semua iterasi yang telah dilakukan. Gbest adalah *fitness value* terbaik dari semua partikel pada semua iterasi yang telah dilakukan. Pada iterasi pertama, *fitness value* dari setiap partikel akan dijadikan sebagai pbest setiap partikel. Kemudian nilai pbest tertinggi dari semua partikel akan dijadikan sebagai gbest. Dari Gambar 3.14, nilai gbest awal didapatkan dari partikel nomer 3 dengan *fitness value* sebesar 51.35.

Setelah mendapatkan nilai pbest dan gbest. Proses kedua adalah proses untuk memperbarui nilai kecepatan dan posisi fitur-fitur dari setiap partikel. Sistem melakukan perhitungan nilai kecepatan yang baru dengan menggunakan persamaan 3.5. v_{is} merupakan kecepatan fitur ke-*s* dari partikel ke-*i*. w merupakan bobot inersia dengan nilai yang berbeda pada setiap iterasi. Perhitungan nilai w dilakukan dengan menggunakan persamaan 3.4. c_1 dan c_2 merupakan nilai konstanta. Pada tugas akhir ini, nilai c_1 dan c_2 diset sama dengan 2.0. $rand()$ merupakan fungsi untuk mendapatkan nilai desimal secara acak pada rentan 0 sampai 1. $pbest_i$ merupakan nilai pbest dari partikel ke-*i*. x_{is} merupakan nilai posisi fitur ke-*s* dari partikel ke-*i*. Hasil *update* nilai kecepatan dapat dilihat pada Gambar 3.15.

$$w = 0.5 + \frac{rand()}{2} \tag{3.4}$$

$$v_{is} = w * v_{is} + c_1 * rand() * (pbest_i - x_{is}) + c_2 * rand() * (gbest - x_{is}) \tag{3.5}$$

$$v_1 = (1.89, 0.71, 0.34, 0.18, 1.37)$$
$$v_2 = (0.61, 0.21, 0.17, 0.27, 0.44)$$
$$v_3 = (0.03, 1.92, 0.03, 0.52, 0.18)$$
$$v_4 = (0.44, 1.59, 0.26, 1.23, 1.84)$$
$$v_5 = (0.45, 0.11, 0.58, 0.21, 0.72)$$

Gambar 3.15 Hasil *update* kecepatan

Setelah memperbarui nilai kecepatan, dilakukan perhitungan nilai posisi fitur-fitur yang baru dengan menggunakan persamaan 3.6 dan persamaan 3.7. e merupakan bilangan Eular yang nilainya setara dengan 2.71828183. $rand()$ merupakan fungsi untuk mendapatkan nilai desimal secara acak pada rentan 0 sampai 1. Nilai yang dihasilkan dari fungsi $S(v_{is})$ kemudian akan digunakan

pada persamaan 3.7. Jika nilai $S(v_{is})$ lebih besar dari nilai $rand()$, nilai posisi fitur yang baru sama dengan 1. Jika nilai $S(v_{is})$ lebih kecil dari nilai $rand()$, nilai posisi fitur yang baru sama dengan 0. Hasil *update* nilai posisi dapat dilihat pada Gambar 3.16.

$$S(v_{is}) = \frac{1}{1 + e^{-v_{is}}} \tag{3.6}$$

$$\begin{gathered} if\ \left(rand() < S(v_{is})\right)\ then\ x_{is} = 1 \\ else\ x_{is} = 0 \end{gathered} \tag{3.7}$$

$$\begin{aligned} x_1 &= (1, 1, 0, 1, 1) \\ x_2 &= (1, 0, 1, 1, 1) \\ x_3 &= (1, 1, 1, 0, 1) \\ x_4 &= (1, 1, 0, 1, 0) \\ x_5 &= (0, 1, 1, 0, 1) \end{aligned}$$

Gambar 3.16 Hasil *update* posisi

Setelah proses kedua dilakukan, proses ketiga adalah perhitungan *fitness value*. Dengan nilai posisi fitur yang baru, fitur-fitur yang nilai posisinya bernilai 1 kemudian digunakan untuk menghitung akurasi partikel. Nilai akurasi yang dihasilkan kemudian akan digunakan untuk menghitung *fitness value* partikel dengan menggunakan persamaan 3.3. Proses pada iterasi pertama ini akan dilakukan sampai iterasi ke-*n*. Setiap iterasi, sistem melakukan pengecekan untuk mendapatkan pbest dan gbest terbaik.

Keluaran dari seleksi fitur PSO ini adalah fitur-fitur dari partikel dengan *fitness value* terbaik. Fitur-fitur ini kemudian digunakan untuk melakukan klasifikasi SVM. Seleksi fitur PSO meminimalkan jumlah fitur yang digunakan untuk membuat

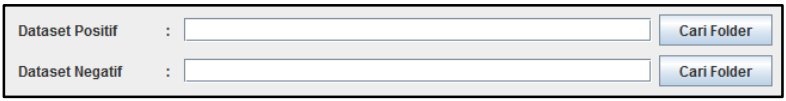
proses komputasi lebih cepat dan hasil klasifikasi yang diberikan lebih optimal.

3.3. Perancangan Antar Muka Perangkat Lunak

Pada sub bab ini, akan dibahas perancangan antar muka perangkat lunak. Ini bertujuan untuk dapat mempermudah interaksi antara sistem dengan pengguna. Antar muka ini hanya memiliki satu halaman yang terdiri dari beberapa sub halaman, yaitu sub halaman untuk melakukan input data, sub halaman untuk menampilkan dokumen dengan nilai prediksi dan nilai aslinya, dan sub halaman untuk menampilkan hasil evaluasi.

3.3.1. Sub halaman Input Data

Sub halaman input data bertujuan untuk mendapatkan alamat folder dataset yang akan diujicobakan. Pada sub halaman ini, terdapat dua *field* yang digunakan untuk input data. *Field* pertama digunakan untuk mendapatkan alamat folder dari dataset opini review film yang bernilai positif. *Field* kedua digunakan untuk mendapatkan alamat folder dari dataset opini review film yang bernilai negatif. Desain antar muka sub halaman input data dapat dilihat pada Gambar 3.17.

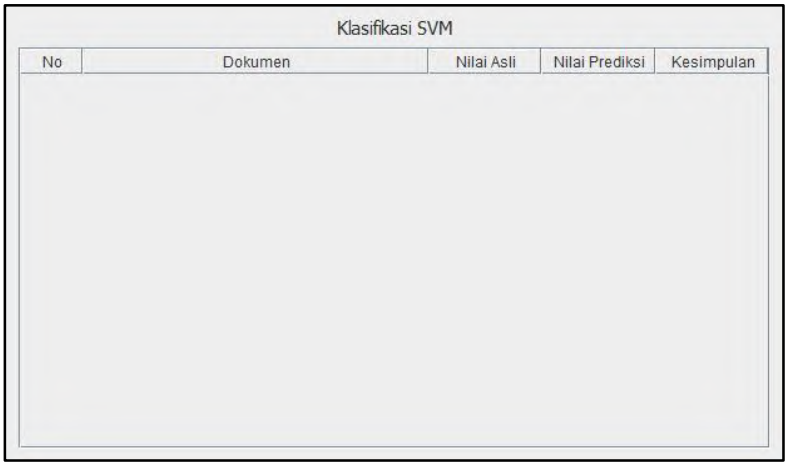


Gambar 3.17 Sub halaman input data

3.3.2. Sub halaman Analisis Dokumen

Sub halaman ini bertujuan untuk menampilkan nilai prediksi dari sistem terhadap dataset. Sub halaman ini terdiri dari nomer dokumen, nama dokumen, nilai prediksi, nilai asli, dan kesimpulan. Pada halaman antar muka pengguna, terdapat dua sub halaman ini, yaitu sub halaman untuk hasil klasifikasi SVM dan

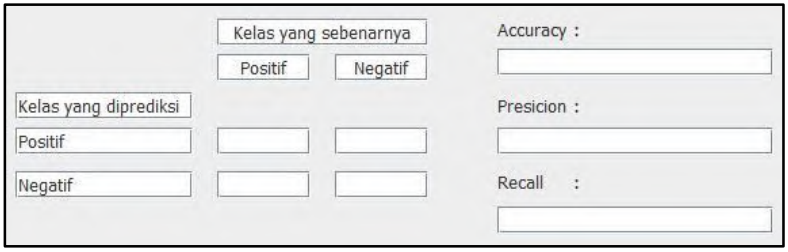
sub halaman untuk hasil klasifikasi SVM-PSO. Desain antar muka sub halaman untuk menampilkan hasil prediksi dokumen dapat dilihat pada Gambar 3.18.



Gambar 3.18 Contoh sub halaman analisis dokumen

3.3.3. Sub halaman Evaluasi

Sub halaman ini digunakan untuk melihat hasil pengujian dan evaluasi yang dilakukan oleh sistem. Pada sub halaman ini, terdapat beberapa *field*. *Field* yang digunakan diantaranya *field* untuk menampilkan dokumen yang bernilai *true-positive* (TP),



Gambar 3.19 Contoh sub halaman evaluasi

field untuk menampilkan dokumen yang bernilai *false-positive* (FP), *field* untuk menampilkan dokumen yang bernilai *false-negative* (FN), *field* untuk menampilkan dokumen yang bernilai *true-negative* (TN), *field* untuk menampilkan nilai *precision*, *field* untuk menampilkan nilai *recall*, dan *field* untuk menampilkan nilai akurasi. Desain antar muka sub halaman evaluasi dapat dilihat pada Gambar 3.19.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem yang telah dibuat. Bahasa pemrograman yang digunakan untuk implementasi sistem adalah bahasa pemrograman JAVA. *Database* yang digunakan adalah *Java Database Connectivity* (JDBC) untuk menyimpan data berupa kata sentimen, kata *stopword*, fitur, dokumen, dan hasil perhitungan lain. Terdapat 3 sub bab pada bab ini. Sub bab pertama menjelaskan tentang kondisi lingkungan dimana sistem diimplementasikan. Sub bab kedua menjelaskan tentang proses-proses apa saja yang diimplementasikan pada sistem, mulai proses pembacaan dokumen sampai proses pengujian dan evaluasi. Pada sub bab ini, juga disertakan kode program yang digunakan untuk dapat menjalankan setiap proses yang dibutuhkan. Sub bab terakhir menjelaskan tentang implementasi dari perancangan antar muka yang telah dibuat.

4.1. Lingkungan Implementasi

Lingkungan implementasi sistem yang digunakan untuk mengembangkan aplikasi tugas akhir ini memiliki spesifikasi perangkat keras dan perangkat lunak seperti yang ditampilkan pada Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Sistem

Perangkat	Spesifikasi
Perangkat keras	Prosesor : Intel® Core™ i5-5200U CPU @ 2.20GHz (4 CPUs) , ~2.2GHz Memori : 8192 MB RAM

Perangkat	Spesifikasi
Perangkat lunak	Sistem Operasi : Microsoft Windows 8.1 Pro 64-bit Perangkat Pengembang : IDE NetBeans 8.1 Perangkat Pembantu : Notepad++ Microsoft Excel 2013 Microsoft Word 2013
Library	LIBSVM

4.2. Implementasi Proses

4.2.1. Implementasi Tahap Pembacaan Dokumen

Pembacaan data dilakukan pada kumpulan dokumen dengan ekstensi txt yang terletak dalam satu folder positif atau folder negatif. Bahasa pemrograman yang digunakan untuk melakukan pembacaan data adalah JAVA. Berikut ini Kode sumber 4.1 untuk menjalankan proses pembacaan data.

```
1   public void readFileOfReview (String dir, int
    kelas) throws IOException {
2       Sistem.out.println("Sedang proses Reading
    Files ...");
3       File directory = new File(dir);
4       this.daftarNamaFile =
    directory.listFiles();
5       String review, namaFile;
6       for (File file : this.daftarNamaFile) {
7           review = "";
8           namaFile = file.getName();
9           this.daftarNFile.add(namaFile);
10          this.kelas.add(kelas);
11          try {
12              FileInputStream fis = new
    FileInputStream(file);
```

```
13              BufferedReader br = new
BufferedReader(new InputStreamReader(fis));
14              String line;
15              while ((line = br.readLine()) !=
null) {
16                  review += line;
17              }
18              this.hasil.add(review);
19              br.close();
20          } catch (IOException x) {
21              Sistem.err.println(x);
22          }
23      }
24      Sistem.out.println("Reading Files
BERHASIL.");
25  }
```

Kode sumber 4.1 Implementasi proses pembacaan dokumen

4.2.2. Implementasi Tahap Ekstraksi Fitur dengan Pemrosesan Teks

Sub bab ini membahas implementasi tahap ekstraksi fitur dengan pemrosesan teks. Implementasi tahap ini menggunakan bahasa pemrograman JAVA. Tahap ini dimulai dengan menjalankan proses *case folding*. Setiap huruf abjad pada setiap teks yang bentuknya *uppercase* akan diubah menjadi bentuk *lowercase*. Berikut ini Kode sumber 4.2 untuk menjalankan proses *case folding*.

```
1   private void
    caseNormalization(ArrayList<String> data) {
2       Sistem.out.println("Sedang proses Case
    Normalization ...");
3       for (String sebelumCN : data) {
4           String setelahCN = "";
5           for (int i = 0 ; i < sebelumCN.length()
    ; i++) {
6               int temp =
    (int)sebelumCN.charAt(i);
```

```
7               if ( temp > 64 && temp < 91) {
8                   temp += 32;
9               }
10              setelahCN += (char)temp;
11          }
12          this.hasil.add(setelahCN);
13      }
14      Sistem.out.println("Case Folding
    BERHASIL.");
15  }
```

Kode sumber 4.2 Implementasi proses *case folding*

Setelah proses *case folding*, proses selanjutnya adalah proses *tokenization*. Teks akan dipecah menjadi kata yang disebut token. Elemen pemisah yang digunakan untuk memecah teks adalah elemen-elemen selain huruf abjad dan tanda penghubung (-). Berikut ini Kode sumber 4.3 untuk menjalankan proses *tokenization*.

```
1   private void tokenization(ArrayList<String>
    data) {
2       Sistem.out.println("Sedang proses
    Tokenization ...");
3       Boolean value;
4       for (String token : data) {
5           ArrayList<String> col = new
    ArrayList<>();
6           this.kata = new char[token.length()];
7           this.kata = token.toCharArray();
8           token = "";
9           value = false;
10          for (int i = 0 ; i < this.kata.length ;
    i++) {
11              int a = (int)this.kata[i];
12              if ((a >= 97 && a <= 122) || a ==
    45)
13                  token += this.kata[i];
14              else {
15                  if ("".equals(token)) {}
```

```
16                  else {
17                      col.add(token);
18                      token = "";
19                  }
20              }
21          }
22          this.hasil.add(col);
23      }
24      Sistem.out.println("Tokenization
    BERHASIL.");
25  }
```

Kode sumber 4.3 Implementasi proses *tokenization*

Hasil dari proses ini adalah kumpulan token. Kumpulan token ini kemudian akan masuk pada proses *stopwords removal*. Pada proses ini, setiap token akan dicocokkan dengan kata yang ada pada daftar *stopwords*. Jika token cocok dengan kata *stopword*, token akan dihapus. Berikut ini Kode sumber 4.4 untuk menjalankan proses *stopwords removal*.

```
1   public void
    stopwordRemoval(ArrayList<ArrayList<String>>
    data, ArrayList<String> stopword) throws
    SQLException {
2       Sistem.out.println("Sedang proses Stopword
    Removal ...");
3       char[] charKata = new char[50];
4       int size1, size2, size3, temp;
5       Boolean value;
6       size1 = data.size();
7       for (int i = 0 ; i < size1 ; i++) {
8           ArrayList<String> col = new
    ArrayList<>();
9           size2 = data.get(i).size();
10          for (int j = 0 ; j < size2 ; j++) {
11              size3 = stopword.size();
12              value = false;
13              for (int k = 0 ; k < size3 ; k++) {
```

```
14                  if
    (data.get(i).get(j).equals(stopword.get(k))) {
15                      value = true;
16                      break;
17                  }
18              }
19              if (value == false) {
20                  col.add(data.get(i).get(j));
21              }
22          }
23          this.hasil.add(col);
24      }
25      Sistem.out.println("Stopword Removal
    BERHASIL.");
26  }
```

Kode sumber 4.4 Implementasi proses *stopwords removal*

Dari hasil proses *stopwords removal*, akan diambil untuk fitur-fitur yang akan digunakan pada tahap klasifikasi. Fitur yang diambil adalah fitur berupa kata sentimen. Kata sentimen yang diambil adalah kata sentimen *lexicon* [6]. Langkah berikutnya adalah menghitung bobot dari setiap fitur yang telah dipilih menggunakan metode *Term Frequency-Inverse Document Frequency* (TF-IDF). Pada metode ini, perhitungan bobot dilakukan dengan mengalikan *term frequency* (TF) dengan *inverse document frequency* (IDF). Berikut ini Kode sumber 4.5 untuk menjalankan proses perhitungan DF dan IDF dan Kode sumber 4.6 untuk menjalankan proses perhitungan TF dan pembobotan TF-IDF.

```
1   public void
    insertToTerms(ArrayList<ArrayList<String>>
    hasilEF, ArrayList<String> term) {
2       Sistem.out.println("Sedang proses insert
    term ke tabel terms ...");
3       try {
4           Statement stmt = con.createStatement();
```

```
5           Boolean value;
6           int size1, size2, size3, df, jumlah =
    0;
7           double idf;
8           size1 = term.size();
9           String sql = "INSERT INTO terms
    VALUES";
10          for (int i = 0; i < size1; i++) {
11              size2 = hasilEF.size();
12              df = 0;
13              for (int j = 0; j < size2; j++) {
14                  size3 = hasilEF.get(j).size();
15                  value = false;
16                  for (int k = 0; k < size3; k++)
    {
17                      if
    (term.get(i).equals(hasilEF.get(j).get(k))) {
18                          value = true;
19                          break;
20                      }
21                  }
22                  //menghitung df
23                  if (value.equals(true)) {
24                      df++;
25                  }
26              }
27              //menghitung idf
28              if (df == 0) {
29                  idf = 0.0;
30              } else {
31                  idf = Math.log10((double) size2
    / (double) df);
32              }
33              sql += "(" + String.valueOf(i+1) +
    ", '" + term.get(i) + "', " +
    String.valueOf(df) + ", " + String.valueOf(idf)
    + ")";
34              jumlah++;
36              if (jumlah % 2000 == 0 || i ==
    size1-1) {
37                  stmt.executeUpdate(sql);
```

```
38                  sql = "INSERT INTO terms
    VALUES";
39                  jumlah = 0;
40              }
41              else {
42                  sql += ", ";
43                  }
44              }
46          Sistem.out.println("INSERT term TO
    terms BERHASIL.");
47      } catch (SQLException err) {
48          Sistem.out.println("Error pada
    insertToTerms.");
49          Sistem.out.println(err.getMessage());
50      }
51  }
```

Kode sumber 4.5 Implementasi proses perhitungan DF dan IDF

```
1   public void
    insertToTfIdf(ArrayList<ArrayList<String>>
    hasilEF, ArrayList<String> term) {
2       Sistem.out.println("Sedang proses insert
    tfidf ke tabel tfidf ...");
3       try {
4           Statement stmt1 =
    con.createStatement();
5           Statement stmt2 =
    con.createStatement();
6           String sql = "SELECT idf FROM terms";
7           ResultSet rs1 =
    stmt1.executeQuery(sql);
8           int size1, size2, size3, jumlah, i = 0,
    jum = 0;
9           double tfidf, idf;
10          sql = "INSERT INTO tfidf VALUES";
11          while (rs1.next()) {
12              size2 = hasilEF.size();
13              for (int j = 0; j < size2; j++) {
14                  size3 = hasilEF.get(j).size();
```

```
15                  //menghitung tf
16                  jumlah = 0;
17                  for (int k = 0; k < size3; k++)
{
18                      if
(term.get(i).equals(hasilEF.get(j).get(k))) {
19                          jumlah++;
20                      }
21                  }
22                  //menghitung tf-idf
23                  if (jumlah > 0) {
24                      idf = rs1.getDouble("idf");
25                      tfidf = (double) jumlah *
idf;
26                      sql += "(" +
String.valueOf(i + 1) + ", " +
String.valueOf(j+1) + ", "
27                                          +
String.valueOf(jumlah) + ", " +
String.valueOf(tfidf) + ")";
28                      jum++;
29                      if (jum % 2000 == 0 || j ==
size2-1) {
30
stmt2.executeUpdate(sql);
31                          sql = "INSERT INTO
tfidf VALUES";
32                          jum = 0;
33                      }
34                      else {
35                          sql += ", ";
36                      }
37                  }
38              }
39              i++;
40          }
41          Sistem.out.println("INSERT nilai_tfidf
TO tfidf BERHASIL.");
42      } catch (SQLException err) {
43          Sistem.out.println("Error pada
insertToTfIdf.");
```

```
44              Sistem.out.println(err.getMessage());
45          }
46      }
```

Kode sumber 4.6 Implementasi proses perhitungan TF dan pembobotan TF-IDF

4.2.3. Implementasi Tahap Klasifikasi SVM

Sub bab ini membahas implementasi tahap klasifikasi SVM. Implementasi tahap ini menggunakan bahasa pemrograman JAVA. Sebelum melakukan tahap klasifikasi SVM, agar akurasi yang diperoleh lebih akurat, sistem menggunakan metode validasi *k-fold cross validation*. Pada tugas akhir ini, variabel *k* diberi nilai 5. Dataset dibagi menjadi 5 bagian data sama besar. Kemudian dilakukan proses klasifikasi SVM sebanyak 5 kali. Pada setiap kali klasifikasi, dataset yang digunakan sebagai *data testing* berbeda. Empat dataset digunakan sebagai *data training* dan sisanya digunakan sebagai *data testing*. Jadi, setiap bagian data pernah menjadi *data testing*. Implementasi proses validasi menggunakan *k-fold cross validation* dapat dilihat pada Kode sumber 4.7.

```
1   private void crossValidation (ArrayList<String>
    file) {
2       Sistem.out.println("Sedang melakukan " +
    this.k + "-fold cross validation ...");
3       int bagi = file.size() / (this.k * 2);
4       int min1 = 0;
5       int max1 = bagi;
6       int min2 = file.size() / 2;
7       int max2 = min2 + bagi;
8       int sisa = file.size() % (this.k * 2);
9       for (int i = 0 ; i < this.k ; i++) {
10          ArrayList<String> train = new
    ArrayList<>();
11          ArrayList<String> test = new
    ArrayList<>();
12          if (i == (this.k - 1)) {
13              max2 += sisa;
```

```
14          }
15          for (int indeks = 0 ; indeks <
    file.size()/2 ; indeks++) {
16              if (indeks >= min1 && indeks <
    max1) {
17                  test.add(file.get(indeks));
18              }
19              else {
20                  train.add(file.get(indeks));
21              }
22          }
23          min1 = max1;
24          max1 += bagi;
25          for (int indeks = file.size()/2 ;
    indeks < file.size() ; indeks++) {
26              if (indeks >= min2 && indeks <
    max2) {
27                  test.add(file.get(indeks));
28              }
29              else {
30                  train.add(file.get(indeks));
31              }
32          }
33          min2 = max2;
34          max2 += bagi;
35
36          this.fileTrain.add(train);
37          this.fileTest.add(test);
38      }
39
40      Sistem.out.println(this.k + "-Fold Cross
    Validation BERHASIL.");
41  }
```

Kode sumber 4.7 Implementasi proses *k-fold cross validation*

Setelah melakukan proses *cross validation*, klasifikasi SVM dilakukan. Sistem akan membuat model terlebih dahulu dari *data training* yang diterima. Model yang terbentuk akan digunakan untuk melakukan klasifikasi pada *data testing*. Proses klasifikasi

menggunakan *library* LIBSVM. Sebelum melakukan klasifikasi, ada beberapa parameter internal metode SVM yang diset terlebih dahulu, yaitu parameter *gamma*, *C*, dan tipe kernel. Parameter *gamma* diset sama dengan 0.5 dan parameter *C* diset sama dengan 100. Sistem menggunakan kernel RBF untuk melakukan klasifikasi SVM. Implementasi proses pembuatan model dapat dilihat pada Kode sumber 4.8 dan implementasi proses klasifikasi dapat dilihat pada Kode sumber 4.9.

Berdasarkan Kode sumber 4.8 dan Kode sumber 4.9, fungsi *training* dan fungsi *testing* membutuhkan dua parameter berupa *two dimensional array* dengan tipe data *double*. Parameter pertama terdiri dari fitur sebagai baris dan dokumen sebagai kolom. Parameter kedua terdiri dari nilai asli sebagai baris dan dokumen sebagai kolom. Implementasi proses pembuatan data yang ada di dalam *database* menjadi data inputan berupa *two dimensional array* pada klasifikasi SVM dapat dilihat pada Kode Sumber 4.10 dan Kode Sumber 4.11**.**

```
1   private svm_model svmTrain(double[][] xtrain,
    double[][] ytrain, String namaModel) throws
    IOException {
2       svm_problem prob = new svm_problem();
3       int recordCount = xtrain.length;
4       int featureCount = xtrain[0].length;
5       Sistem.out.println(featureCount + " " +
    recordCount);
6       prob.y = new double[recordCount];
7       prob.l = recordCount;
8       prob.x = new
    svm_node[recordCount][featureCount];
9       for (int i = 0 ; i < recordCount ; i++) {
10          double[] features = xtrain[i];
11          prob.x[i] = new
    svm_node[features.length];
12          for (int j = 0 ; j < features.length ;
    j++){
13              svm_node node = new svm_node();
14              node.index = j;
```

```
15              node.value = features[j];
16              prob.x[i][j] = node;
17          }
18          prob.y[i] = ytrain[i][0];
19      }
20      svm_parameter param = new svm_parameter();
21      param.probability = 1;
22      param.gamma = 0.5;
23      param.nu = 0.5;
24      param.C = 100;
25      param.svm_type = svm_parameter.C_SVC;
26      param.kernel_type = svm_parameter.RBF;
27      param.cache_size = 2000;
28      param.eps = 0.001;
29
30      svm_model model = svm.svm_train(prob,
    param);
31      svm.svm_save_model(namaModel, model);
32
33      return model;
34  }
```

Kode sumber 4.8 Implementasi proses pembuatan model menggunakan data *training* pada klasifikasi SVM

```
1   private double[] svmPredict(double[][] xtest,
    svm_model model) throws IOException {
2       double[] yPred = new double[xtest.length];
3       for(int k = 0; k < xtest.length; k++){
4           double[] fVector = xtest[k];
5           svm_node[] nodes = new
    svm_node[fVector.length];
6           for (int i = 0; i < fVector.length;
    i++) {
7               svm_node node = new svm_node();
8               node.index = i;
9               node.value = fVector[i];
10              nodes[i] = node;
11          }
12          int totalClasses = 2;
```

```
13          int[] labels = new int[totalClasses];
14          svm.svm_get_labels(model,labels);
15          double[] prob_estimates = new
    double[totalClasses];
16          yPred[k] =
    svm.svm_predict_probability(model, nodes,
    prob_estimates);
17      }
18      return yPred;
19  }
```

Kode sumber 4.9 Implementasi proses klasifikasi *data testing* pada klasifikasi SVM

```
1   public double[][] getX (ArrayList<String>
    files, ArrayList<String> term) throws
    SQLException {
2       double[][] xHasil = new
    double[files.size()][term.size()];
3       Statement stmt =
    con.createStatement();
4       String sql = "SELECT
    id_t,id_f,nilai_tfidf FROM tfidf,documents
    WHERE tfidf.id_f = documents.id_doc AND
    (";
5       for (int i = 0 ; i < files.size() ;
    i++) {
6           sql += "documents.document = '" +
    files.get(i) + "'";
7           if (i < files.size()-1) {
8               //System.out.println(sql);
9               sql += " OR ";
10          }
11          else {
12              sql += ") ORDER BY
    tfidf.id_f,tfidf.id_t";
13          }
14      }
15      ResultSet rs = stmt.executeQuery(sql);
16
```

```
17      int x, y = -1, id = 999999999;
18      double value;
19      while (rs.next()) {
20          if (id != rs.getInt("id_f")) {
21              id = rs.getInt("id_f");
22              y++;
23          }
24          x = rs.getInt("id_t") - 1;
25          value =
rs.getDouble("nilai_tfidf");
26          xHasil[y][x] = value;
27      }
28
29      return xHasil;
30  }
```

Kode Sumber 4.10 Implementasi proses pembuatan inputan berupa bobot fitur dokumen pada klasifikasi SVM

```
1   public double[][] getY(ArrayList<String>
    files) throws SQLException {
2       double[][] yHasil = new
    double[files.size()][1];
3       Statement stmt =
    con.createStatement();
4       String sql = "SELECT kelas FROM
    documents WHERE ";
5       //System.out.println(sql);
6       for (int i = 0 ; i < files.size() ;
    i++) {
7           sql += "document = '" +
    files.get(i) + "'";
8           if (i < files.size()-1) {
9               sql += " OR ";
10          }
11          else {
12              sql += " ORDER BY id_doc";
13          }
14      }
```

```
15      ResultSet rs = stmt.executeQuery(sql);
16      int x = 0;
17      double kelas;
18      while (rs.next()) {
19          kelas =
    (double)rs.getInt("kelas");
20          yHasil[x][0] = kelas;
21          x++;
22      }
23
24      return yHasil;
25  }
```

Kode Sumber 4.11 Implementasi proses pembuatan inputan berupa nilai asli dokumen pada klasifikasi SVM

4.2.4. Implementasi Tahap Klasifikasi SVM-PSO

Sub bab ini membahas implementasi tahap klasifikasi SVM dengan seleksi fitur PSO. Implementasi tahap ini menggunakan bahasa pemrograman JAVA. Sebelum melakukan tahap klasifikasi SVM, semua fitur akan diseleksi terlebih dahulu. Seleksi fitur ini akan meminimalkan jumlah fitur yang digunakan. Terdapat beberapa sub proses yang ada dalam proses seleksi fitur PSO, yaitu inisialisasi partikel pada *swarm*, pencarian nilai pbest dan gbest, *update* partikel, dan perhitungan *fitness value*.

Proses pertama adalah inisialisasi partikel pada *swarm*. *Swarm* terdiri dari 200 partikel. Setiap partikel terdiri dari 3 komponen, yaitu fitur, posisi (*location*), dan kecepatan (*velocity*). Pada sistem, jumlah fitur yang dimiliki oleh setiap partikel sama dengan 200 fitur. Setiap fitur memiliki posisi dan kecepatan. Inisialisasi partikel dilakukan dengan mengambil 200 fitur secara acak yang ada di dalam *database*, mengambil nilai posisi secara acak antara 0 atau 1, dan mengambil nilai desimal untuk kecepatan secara acak pada rentan 0 sampai 1. Implementasi proses inisialisasi partikel pada *swarm* dapat dilihat pada Kode Sumber 4.12.

```
1   private void initializeSwarm() throws
    IOException, SQLException {
2       Sistem.out.println("Sedang melakukan
    inisialisasi Swarm ...");
3       Particle p;
4       for(int i = 0 ; i < SWARM_SIZE ; i++) {
5           p = new Particle();
6
7           String[] fit = new
    String[PROBLEM_DIMENSION];
8           int index;
9           ArrayList<String> termfit = new
    ArrayList<>(this.term);
10          for (int j = 0 ; j < PROBLEM_DIMENSION
    ; j++) {
11              index =
    this.generator.nextInt(termfit.size());
12              fit[j] = termfit.get(index);
13              termfit.remove(index);
14          }
15          Fitur fitur = new Fitur(fit);
16
17          ArrayList<String> termForPSO = new
    ArrayList<>();
18          int[] loc = new int[PROBLEM_DIMENSION];
19          for (int j = 0 ; j < PROBLEM_DIMENSION
    ; j++) {
20              loc[j] = this.generator.nextInt(2);
21              if (loc[j] == 1) {
22                  termForPSO.add(fit[j]);
23              }
24          }
25          Location location = new Location(loc);
26          double accuracy = new
    KlasifikasiSVMwithPSO(this.file,
    termForPSO).getAccuracy();
27
28          double[] vel = new
    double[PROBLEM_DIMENSION];
29          for (int j = 0 ; j < PROBLEM_DIMENSION
    ; j++) {
```

```
30              vel[j] =
    this.generator.nextDouble();
31          }
32          Velocity velocity = new Velocity(vel);
33
34          p.setAccuracy(accuracy);
35          p.setLocation(location);
36          p.setVelocity(velocity);
37          p.setFitur(fitur);
38          Sistem.out.println("Partikel ke-" +
    (i+1) + " telah berhasil ditambahkan.");
39          this.swarm.add(p);
40      }
41  }
```

Kode Sumber 4.12 Implementasi proses inisialisasi partikel pada *swarm*

Proses selanjutnya adalah inisialisasi nilai terbaik (pbest) dari setiap partikel. Nilai awal yang diambil sebagai nilai terbaik adalah nilai posisi dan *fitness value* pertama dari setiap partikel. Nilai posisi dan *fitness value* akan diperbarui jika pada proses iterasi yang dilakukan, ditemukan nilai posisi dan *fitness value* yang lebih baik dari nilai pbest sebelumnya. Implementasi proses inisialisasi awal nilai pbest dapat dilihat pada Kode Sumber 4.13.

```
1   for(int i = 0 ; i < SWARM_SIZE ; i++) {
2       this.pBest[i] = this.fitnessValueList[i];
3
    this.pBestLocation.add(this.swarm.get(i).getLoc
    ation());
4   }
```

Kode Sumber 4.13 Implementasi proses inisialisasi awal nilai pbest

Setelah dilakukan inisialisasi awal nilai pbest, iterasi dilakukan. Setiap iterasi melakukan beberapa sub proses mulai dari *update* pbest, *update* gbest, sampai *update* nilai akurasi, posisi,

kecepatan, dan *fitness value* pada setiap partikel. Pada proses *update* partikel, nilai akurasi selalu berubah di setiap iterasi. Jumlah fitur yang digunakan dalam setiap proses klasifikasi di setiap iterasi bergantung pada nilai posisi setiap fitur. Jika nilai posisi sama dengan 1, fitur akan digunakan untuk proses klasifikasi. Jika nilai posisi sama dengan 0, fitur tidak akan digunakan dalam proses klasifikasi. Implementasi proses-proses di dalam iterasi dapat dilihat pada Kode Sumber 4.14.

```
1  int t = 0;
2  double w;
3
4  while(t < MAX_ITERATION) {
5      Sistem.out.println("PERULANGAN KE-" +
   (t+1));
6      for(int i = 0 ; i < SWARM_SIZE ; i++) {
7          if(this.fitnessValueList[i] >
   this.pBest[i]) {
8              this.pBest[i] =
   this.fitnessValueList[i];
9              this.pBestLocation.set(i,
   this.swarm.get(i).getLocation());
10         }
11     }
12
13     int bestParticleIndex =
   Utility.getMaxPos(this.fitnessValueList);
14     if(t == 0 ||
   this.fitnessValueList[bestParticleIndex] >
   this.gBest) {
15         this.gBest =
   this.fitnessValueList[bestParticleIndex];
16         this.gBestLocation =
   this.swarm.get(bestParticleIndex).getLocation()
   ;
17         this.gBestFitur =
   this.swarm.get(bestParticleIndex).getFitur();
18     }
19     w = 0.5 + (generator.nextDouble() / 2.0);
20     for(int i = 0 ; i < SWARM_SIZE ; i++) {
```

```
21          Sistem.out.println("UPDATE PARTIKEL KE-
" + (i+1) + " : ");
22          double r1 =
this.generator.nextDouble();
23          double r2 =
this.generator.nextDouble();
24          Particle p = this.swarm.get(i);
25          double[] newVel = new
double[PROBLEM_DIMENSION];
26          Sistem.out.print("VELOCITY => ");
27          for (int indeks = 0 ; indeks <
PROBLEM_DIMENSION ; indeks++) {
28              newVel[indeks] = (w *
p.getVelocity().getVel()[indeks]) +
29          (r1 * C1) *
(this.pBestLocation.get(i).getLoc()[indeks] -
p.getLocation().getLoc()[indeks]) +
30          (r2 * C2) *
(this.gBestLocation.getLoc()[indeks] -
p.getLocation().getLoc()[indeks]);
31              Sistem.out.print(newVel[indeks] + "
");
32          }
33          Sistem.out.println();
34          Velocity vel = new Velocity(newVel);
35          p.setVelocity(vel);
36          double r3 =
this.generator.nextDouble();
37          int[] newLoc = new
int[PROBLEM_DIMENSION];
38          ArrayList<String> termForPSO = new
ArrayList<>();
39          double sv;
40          Sistem.out.println("LOCATION => ");
41          for (int indeks = 0 ; indeks <
PROBLEM_DIMENSION ; indeks++) {
42              sv = 1 / (1 + exp(-1 *
newVel[indeks]));
43              if (r3 < sv) {
44                  newLoc[indeks] = 1;
45

termForPSO.add(p.getFitur().getFit()[indeks]);
```

```
46                  }
47                  else {
48                      newLoc[indeks] = 0;
49                  }
50                  Sistem.out.print(newLoc[indeks] + "
");
51              }
52              Sistem.out.println();
53              if (!termForPSO.isEmpty()) {
54                  double accuracy = new
KlasifikasiSVMwithPSO(this.file,
termForPSO).getAccuracy();
55                  p.setAccuracy(accuracy);
56              }
57              Location loc = new Location(newLoc);
58              p.setLocation(loc);
59          }
60          Sistem.out.println();
61          Sistem.out.println("NILAI GBEST (" + (t+1)
+ ") => " + this.gBest);
62          Sistem.out.println();
63          t++;
64          updateFitnessList();
65      }
```

Kode Sumber 4.14 Implementasi proses-proses di dalam iterasi

Di dalam iterasi, terdapat sub proses *update* gbest. Gbest merupakan nilai terbaik yang didapatkan dari nilai pbest terbaik dari semua partikel. Pada setiap iterasi, nilai gbest akan diperbarui setelah diperbaruinya nilai pbest. Gbest terdiri dari fitur dan *fitness value* terbaik. Pencarian nilai gbest dilakukan dengan mencari nilai indeks partikel yang mempunyai *fitness value* terbaik. Setelah partikel ditemukan, fitur yang ada di dalam partikel dianggap sebagai fitur terbaik. Proses *update* gbest berlangsung sampai iterasi selesai. Implementasi sub proses pencarian nilai pbest terbaik (gbest) dapat dilihat pada Kode Sumber 4.15.

```
1   public class Utility {
2       public static int getMaxPos(double[] list)
    {
3           int pos = 0;
4           double maxValue = list[0];
5
6           for(int i = 0 ; i < list.length ; i++)
    {
7               if(list[i] > maxValue) {
8                   pos = i;
9                   maxValue = list[i];
10              }
11          }
12
13          return pos;
14      }
15  }
```

Kode Sumber 4.15 Implementasi proses pencarian nilai gbest

Setelah *update* nilai posisi dan kecepatan, juga dilakukan *update fitness value* pada setiap partikel. Sebelum melakukan *update fitness value*, dilakukan *update* nilai akurasi dengan menggunakan nilai posisi yang telah diperbarui. Nilai akurasi yang telah diperbarui akan digunakan untuk menghitung *fitness value*. Selain menggunakan nilai akurasi, *fitness value* juga dihitung menggunakan fitur yang ada pada setiap partikel. Implementasi proses perhitungan *fitness value* dapat dilihat pada Kode Sumber 4.16.

```
1   public class ProblemSet implements Constant{
2       public static final double ALFA = 0.85;
3       public static final double BETA = 0.15;
4
5       public static double evaluate(Location
    location, double accuracy) throws SQLException,
    IOException {
6           int[] loc = location.getLoc();
7           int jumlah = 0;
```

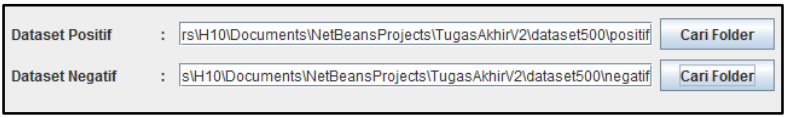
```
8           for (int i = 0 ; i < PROBLEM_DIMENSION
    ; i++) {
9               if (loc[i] == 1) {
10                  jumlah++;
11              }
12          }
13          double result = (ALFA * accuracy) +
    (BETA * (PROBLEM_DIMENSION -
    jumlah)/PROBLEM_DIMENSION);
14
15          return result;
16      }
17  }
```

Kode Sumber 4.16 Implementasi proses perhitungan *fitness value*

4.3. Implementasi Antar Muka

4.3.1. Implementasi Sub Halaman Input Data

Sub halaman ini digunakan untuk mendapatkan alamat folder untuk dataset positif dan dataset negatif. Terdapat 2 *field* yang ada pada sub halaman ini. *Field* pertama adalah *field* yang digunakan untuk mengambil alamat folder dataset positif. *Field* kedua adalah *field* yang digunakan untuk mengambil alamat folder dataset negatif. Implementasi sub halaman input data dapat dilihat pada Gambar 4.1.

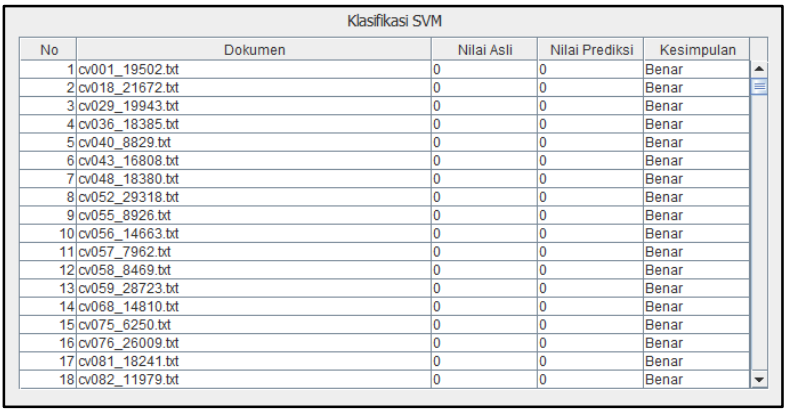


Gambar 4.1 Implementasi sub halaman input data

4.3.2. Implementasi Sub Halaman Analisis Dokumen

Sub halaman ini digunakan untuk menampilkan nilai prediksi dari sistem terhadap dataset dengan menggunakan tabel hasil

klasifikasi. Terdapat 5 kolom pada tabel. Kolom pertama berisi nomer dokumen, kolom kedua berisi nama dokumen, kolom ketiga berisi nilai asli, kolom keempat berisi nilai prediksi, dan kolom kelima berisi kesimpulan. Implementasi sub halaman analisis dokumen dapat dilihat di Gambar 4.2.

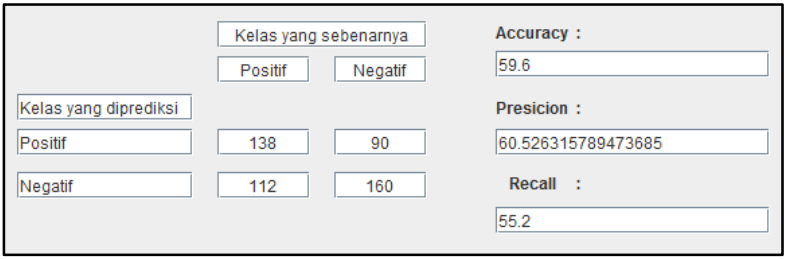

Klasifikasi SVM

No	Dokumen	Nilai Asli	Nilai Prediksi	Kesimpulan
1	cv001_19502.txt	0	0	Benar
2	cv018_21672.txt	0	0	Benar
3	cv029_19943.txt	0	0	Benar
4	cv036_18385.txt	0	0	Benar
5	cv040_8829.txt	0	0	Benar
6	cv043_16808.txt	0	0	Benar
7	cv048_18380.txt	0	0	Benar
8	cv052_29318.txt	0	0	Benar
9	cv055_8926.txt	0	0	Benar
10	cv056_14663.txt	0	0	Benar
11	cv057_7962.txt	0	0	Benar
12	cv058_8469.txt	0	0	Benar
13	cv059_28723.txt	0	0	Benar
14	cv068_14810.txt	0	0	Benar
15	cv075_6250.txt	0	0	Benar
16	cv076_26009.txt	0	0	Benar
17	cv081_18241.txt	0	0	Benar
18	cv082_11979.txt	0	0	Benar

Gambar 4.2 Implementasi sub halaman analisis dokumen

4.3.3. Implementasi Sub Halaman Evaluasi

Sub halaman ini digunakan untuk melihat hasil pengujian dan evaluasi dari klasifikasi yang telah dilakukan oleh sistem. Terdapat 7 *field* pada sub halaman ini. *Field* pertama adalah jumlah dokumen positif yang dikategorikan 'positif' (TP). *Field* kedua adalah jumlah dokumen positif yang dikategorikan 'negatif' (FN). *Field* ketiga adalah jumlah dokumen negatif yang dikategorikan 'positif' (FP). *Field* keempat adalah jumlah dokumen negatif yang dikategorikan 'negatif' (TN). *Field* kelima adalah nilai akurasi dari klasifikasi yang dilakukan. *Field* keenam adalah nilai *precision* dan *field* ketujuh adalah nilai *recall*. Implementasi sub halaman evaluasi dapat dilihat pada Gambar 4.3.



Gambar 4.3 Implementasi sub halaman evaluasi

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas tentang pengujian dan evaluasi terhadap sistem yang telah dikembangkan untuk melakukan klasifikasi opini review film, baik menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Pada pengujian ini, akan ada 4 skenario yang digunakan untuk menguji sistem yang telah dibuat. Skenario pertama adalah perhitungan hasil akurasi, *precision*, *recall* dengan 2 jenis fitur yang berbeda. Skenario kedua adalah perhitungan hasil akurasi, *precision*, *recall* dengan 4 jumlah dokumen yang berbeda. Skenario ketiga adalah perhitungan hasil akurasi, *precision*, *recall* dengan 4 jumlah iterasi pada seleksi fitur PSO yang berbeda. Skenario keempat adalah perhitungan hasil akurasi, *precision*, dan *recall* dengan nilai *k* pada *k-fold cross validation* yang berbeda. Kemudian akan dilakukan analisis terhadap hasil pengujian yang telah didapatkan.

5.1. Lingkungan Uji Coba

Lingkungan uji coba yang digunakan dalam pembuatan tugas akhir ini meliputi perangkat lunak dan perangkat keras yang digunakan untuk melakukan uji coba klasifikasi opini review film. Lingkungan uji coba dapat dilihat pada Tabel 5.1.

Tabel 5.1 Lingkungan Uji Coba Sistem

Perangkat	Spesifikasi
Perangkat keras	Prosesor : Intel® Core™ i3-5200U CPU @ 2.20GHz (4 CPUs) , ~2.2GHz Memori : 8192 MB RAM

Perangkat	Spesifikasi
Perangkat lunak	Sistem Operasi : Microsoft Windows 8.1 Pro 64-bit Perangkat Pengembang: IDE NetBeans 8.1 Microsoft SQL Server 2008 R2 Perangkat Pembantu : Notepad++ Microsoft Excel 2013 Microsoft Word 2013

5.2. Data Uji Coba

Data yang digunakan untuk uji coba implementasi sistem ini adalah opini review film. Data yang diuji cobakan adalah data yang diambil dari dataset opini yang dikumpulkan oleh Pang dan timnya [7]. Setiap opini memiliki panjang yang berbeda-beda. Jumlah data yang digunakan dalam uji coba bergantung pada skenario yang digunakan. Setiap opini memiliki satu label kelas. Contoh data uji yang digunakan dapat dilihat pada Tabel 5.2.

Tabel 5.2 Contoh data masukan uji coba yang digunakan

No	Data Masukan	Kelas
1	**cv280_8267** this three hour movie opens up with a view of singer/guitar player/musician/composer frank zappa rehearsing with his fellow band members . all the rest displays a compilation of footage , mostly from the concert at the palladium in new york city , halloween 1979 . other footage shows backstage foolishness , and amazing clay animation by bruce bickford . the performance of " titties and beer " played in this movie is very entertaining , with drummer terry bozzio supplying the voice of the devil . frank's guitar solos outdo any van halen or	Positif (1)

No	Data Masukan	Kelas
	hendrix i've ever heard . bruce bickford's outlandish clay animation is that beyond belief with zooms , morphings , etc . and actually , it doesn't even look like clay , it looks like meat .	
2	**cv506_17521** this film is extraordinarily horrendous and i'm not going to waste any more words on it .	Negatif (0)
3	**cv857_17527** claire danes , giovanni ribisi , and omar epps make a likable trio of protagonists , but they're just about the only palatable element of the mod squad , a lame-brained big-screen version of the 70s tv show . the story has all the originality of a block of wood (well , it would if you could decipher it) , the characters are all blank slates , and scott silver's perfunctory action sequences are as cliched as they come . by sheer force of talent , the three actors wring marginal enjoyment from the proceedings whenever they're on screen , but the mod squad is just a second-rate action picture with a first-rate cast .	Negatif (0)

Data masukan akan diproses dan diekstraksi sehingga akan dihasilkan fitur-fitur berupa sentimen yang ada pada setiap dokumen. Contoh hasil ekstraksi fitur dapat dilihat pada Tabel 5.3. Dari Tabel 5.3, opini nomer 1 bernilai positif. Ini dapat dilihat dari jumlah kata sentimen positif yang lebih banyak daripada jumlah kata sentimen negatif. Kata sentimen positif dari opini nomer 1 adalah kata ‘foolishness’, ‘amazing’, ‘entertaining’, dan ‘outdo’, sedangkan kata sentimen negatif dari opini nomer 1 adalah kata ‘devil’. Kasus yang sama juga dapat dilihat pada opini nomer 2. Opini nomer 2 bernilai negatif karena jumlah kata sentimen negatif yang lebih banyak daripada jumlah kata sentimen positif. Sentimen negatif pada opini nomer 2 adalah ‘horrendous’ dan ‘waste’, sedangkan kata sentimen positif pada opini nomer 2 adalah

‘extraordinarily’. Namun, ada beberapa opini yang bernilai negatif walaupun jumlah kata sentimen positifnya lebih banyak daripada jumlah kata sentimen negatifnya. Pada opini nomer 3, kata sentimen negatif hanya ada ‘perfunctory’, ‘cliched’, dan ‘marginal’, sedangkan kata sentimen positif ada ‘likable’, ‘originality’, ‘well’, ‘talent’, ‘enjoyment’, dan ‘first-rate’.

Tabel 5.3 Contoh hasil ekstraksi fitur

No	Data Masukan	Kelas
1	**cv280_8267**	Positif (1)
	[foolishness, amazing, entertaining, devil, outdo]	
2	**cv506_17521**	Negatif (0)
	[extraordinarily, horrendous, waste]	
3	**cv857_17527**	Negatif (0)
	[likable, originality, well, perfunctory, cliched, talent, marginal, enjoyment, first-rate]	

5.3.Skenario Uji Coba

Pada sub bab ini, akan dijelaskan skenario uji coba yang akan dilakukan. Skenario uji coba bertujuan untuk mendapatkan hasil klasifikasi yang maksimal dengan memilih metode terbaik dari beberapa metode pada satu tahap. Terdapat beberapa skenario uji coba yang dilakukan, diantaranya yaitu :

1. Perhitungan nilai akurasi, *precision*, dan *recall* dari klasifikasi, baik menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Klasifikasi menggunakan 2 jenis fitur berbeda, yaitu fitur yang berasal dari dataset kata sentimen dan fitur yang berasal dari dataset kata bahasa inggris. Uji coba ini bertujuan untuk mengetahui fitur terbaik yang digunakan untuk tahap klasifikasi.

2. Perhitungan nilai akurasi, *precision*, dan *recall* dari klasifikasi, baik menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Klasifikasi menggunakan 4 jumlah dokumen berbeda, yaitu 500, 1000, 1500, dan 2000 dokumen. Uji coba ini bertujuan untuk melihat perbandingan klasifikasi opini review film, baik menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO dengan jumlah data yang berbeda.
3. Perhitungan nilai akurasi, *precision*, dan *recall* dari klasifikasi menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Klasifikasi menggunakan 4 jumlah iterasi berbeda pada seleksi fitur PSO, yaitu 50, 100, 150, dan 200 iterasi. Uji coba ini bertujuan untuk mengetahui jumlah iterasi terbaik untuk melakukan penyeleksian fitur dengan metode PSO sehingga didapatkan hasil klasifikasi yang lebih baik.
4. Perhitungan nilai akurasi, *precision*, dan *recall* dari klasifikasi menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Klasifikasi menggunakan 2 nilai *k* berbeda pada *k-fold cross validation*, yaitu 2 dan 5. Uji coba ini bertujuan untuk mengetahui nilai *k* terbaik yang digunakan untuk tahap klasifikasi.

5.3.1. Skenario Pengujian 1 : Perhitungan Nilai Akurasi, *Precision*, dan *Recall* dengan 2 Jenis Fitur yang Berbeda

Skenario uji coba pertama adalah perhitungan nilai akurasi, *precision*, dan *recall* dari klasifikasi, baik menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Klasifikasi menggunakan 2 jenis fitur berbeda, yaitu fitur yang berasal dari dataset kata sentimen dan fitur yang berasal dari dataset kata bahasa inggris. Skenario ini bertujuan untuk mengetahui fitur terbaik yang digunakan untuk

tahap klasifikasi. Klasifikasi dilakukan pada 2000 dokumen. Dataset kata sentimen bahasa inggris yang digunakan pada uji coba ini didapatkan dari sentimen *lexicon* milik Professor Bing Liu [8]. Dataset kata bahasa inggris yang digunakan sebagai pembanding dari dataset sentimen pada uji coba ini adalah dataset yang didapatkan dari github dengan akun 'dwyl'. Jumlah fitur, jumlah iterasi dan jumlah partikel pada parameter PSO diset pada nilai berturut-turut 200, 200, 100. Nilai akurasi dihitung dengan jumlah data yang mampu diklasifikasikan dengan benar. Sedangkan *precision* dan *recall* dihitung berdasarkan *confusion matrix*. *Confusion matrix* untuk hasil skenario uji coba ini dapat dilihat pada Tabel 5.4. Nilai akurasi, *precision* dan *recall* pada setiap klasifikasi menggunakan dataset kata sentimen dan dataset kata bahasa inggris dapat dilihat pada Tabel 5.5. *Running time* skenario uji coba pertama dapat dilihat pada Tabel 5.6.

Tabel 5.4 *Confusion matrix* skenario uji coba pertama

Fitur	Kelas yang diprediksi	Kelas yang sebenarnya			
		SVM		SVM-PSO	
		Positif	Negatif	Positif	Negatif
Tanpa Sentimen	Positif	6	0	804	676
	Negatif	994	1000	196	324
Sentimen	Positif	8	0	452	235
	Negatif	992	1000	548	765

Tabel 5.5 Hasil skenario uji coba pertama

Fitur	Akurasi		Precision		Recall	
	SVM	SVM-PSO	SVM	SVM-PSO	SVM	SVM-PSO
Tanpa Sentimen	50.30	56.40	100	54.32	0.60	80.40
Sentimen	50.40	60.85	100	65.79	0.80	45.20

Tabel 5.6 *Running time* skenario uji coba pertama

Metode Klasifikasi	Fitur	Waktu
SVM	**Dictionary**	1 jam 8 menit 33 detik
	Sentimen	54 menit 46 detik
SVM-PSO	**Dictionary**	5 hari 2 jam 13 menit 21 detik
	Sentimen	4 hari 16 jam 43 menit 42 detik

5.3.2. Analisis dan Evaluasi Skenario Pengujian 1

Pada hasil skenario uji coba 1, didapatkan *running time* yang lebih pendek pada klasifikasi menggunakan dataset kata sentimen daripada menggunakan dataset kata bahasa inggris. Baik klasifikasi menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Beberapa hal yang dapat diambil dari hasil skenario uji coba pertama selain waktu yang dibutuhkan adalah sebagai berikut :

1. Pemilihan fitur menggunakan dataset bahasa inggris membuat *running time* program menjadi lebih lama daripada menggunakan dataset kata sentimen. Berdasarkan Tabel 5.6, *running time* program dengan menggunakan dataset kata bahasa inggris lebih lama daripada menggunakan dataset kata sentimen, baik klasifikasi menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Hal ini karena jumlah kata dalam dataset bahasa inggris yang memiliki jumlah kata sebanyak 354934 kata, berbeda jauh dengan jumlah kata dalam dataset kata sentimen yang memiliki jumlah kata sebanyak 6786 kata.
2. Klasifikasi menggunakan dataset kata sentimen menghasilkan nilai akurasi yang lebih baik daripada menggunakan dataset kata bahasa inggris. Berdasarkan Tabel 5.5, nilai akurasi pada klasifikasi SVM

menggunakan dataset kata bahasa inggris adalah 50.30%. Nilai akurasi meningkat menjadi 50.40% saat menggunakan dataset kata sentimen. Peningkatan nilai akurasi juga terjadi pada klasifikasi SVM dengan penambahan seleksi fitur PSO. Dari nilai akurasi klasifikasi menggunakan dataset kata bahasa inggris sebesar 56.40%, menjadi 60.85% saat menggunakan dataset kata sentimen. Hal ini karena beberapa dokumen dapat diklasifikasikan ke dalam katerogi 'positif' setelah ditambahan metode seleksi fitur PSO, baik dokumen positif maupun dokumen negatif. Contoh dan analisis dokumen yang salah terklasifikasikan dengan menggunakan metode SVM, kemudian benar terklasifikasikan dengan menggunakan metode SVM dengan penambahan metode seleksi fitur PSO dapat dilihat pada Tabel 5.7, Tabel 5.8, dan Tabel 5.9.

Tabel 5.7 Analisis Klasifikasi SVM dan SVM-PSO

No	Data Masukan	SVM	SVM-PSO
1	**cv018_20137** [struck, brilliant, convincingly, great, effective, mystery, plot, fun, trouble, work, valiant, work, distraction, valiant, dead, suspect, valiant, clear, good, doom, kill, won, awards, achievement, award, praise, imaginary, hard, work, great, enjoyed, scared, doom, favor] **Hasil Ekstraksi Fitur dengan fitur PSO :** [great, effective, plot, suspect, good, kill, won, hard, great, enjoyed]	Asli : 1 Prediksi : 0	Asli : 1 Prediksi : 1
2	**cv783_14724** **Hasil Ekstraksi Fitur dengan Sentimen :**	Asli : 0 Prediksi : 1	Asli : 0 Prediksi : 0

No	Data Masukan	SVM	SVM-PSO
	[good, good, colorful, dark, depressingly, gloomy, bitter, plot, interesting, ridiculous, fascinating, plot, impossible, senseless, cheap, evil, conspiracy, interesting, evil, stumble, dreadful, bland, fake, poor, continuity, weak, bad, tolerable]		
	Hasil Ekstraksi Fitur dengan fitur PSO : [good, good, colorful, bitter, plot, plot, weak, bad]		

= kata sentimen positif
= kata sentimen negatif

Tabel 5.8 Analisis dokumen positif (cv018_20137)

Fitur	TF	DF	IDF	TF-IDF	Total Bobot
great	2	695	0.46	0.92	4.39
effective	1	166	1.08	1.08	
good	1	1150	0.24	0.24	
won	1	293	0.83	0.83	
enjoyed	1	95	1.32	1.32	
suspect	1	78	1.41	1.41	3.45
kill	1	220	0.96	0.96	
hard	1	386	0.72	0.72	
plot	1	881	0.36	0.36	

Tabel 5.9 Analisis dokumen negatif (cv783_14724)

Fitur	TF	DF	IDF	TF-IDF	Total Bobot
colorful	1	46	1.64	1.64	2.12
good	2	1150	0.24	0.48	
plot	2	881	0.36	0.72	4.11
bitter	1	48	1.62	1.62	
weak	1	90	1.35	1.35	
bad	1	763	0.42	0.42	

Dokumen ‘cv018_20137’ yang bernilai positif diklasifikasikan ‘negatif’ oleh SVM. Namun, dengan penambahan metode seleksi fitur PSO, dokumen dapat diklasifikasikan dengan benar. Hal ini dapat dilihat pada Tabel 5.8. Total bobot kata sentimen positif pada dokumen ‘cv018_20137’ lebih besar dari total bobot kata sentimen negatif. Hal yang sama juga terjadi pada dokumen ‘cv783_14724’ yang bernilai negatif. Dengan SVM, dokumen diklasifikasikan sebagai dokumen ‘positif’. Namun, dengan penambahan metode PSO, dokumen dapat diklasifikasikan dengan benar. Tabel 5.9 menunjukkan total bobot kata sentimen negatif lebih besar dari total bobot kata sentimen positif.

3. Nilai *precision* untuk klasifikasi SVM dengan seleksi fitur PSO mengalami kenaikan dari 54.32% menjadi 65.79%. Berdasarkan Tabel 5.5, nilai *precision* dari klasifikasi menggunakan dataset kata bahasa inggris lebih rendah daripada menggunakan dataset kata sentimen karena banyak dokumen negatif yang dikategorikan ‘positif’ sebesar 676 dokumen dari 1000 dokumen negatif ketika menggunakan dataset kata bahasa inggris. Sementara itu, klasifikasi SVM dengan seleksi fitur PSO menggunakan dataset kata sentimen menghasilkan 235 dokumen yang dikategorikan ‘positif’ dari 1000 dokumen negatif yang diuji cobakan.
4. Nilai *recall* klasifikasi SVM-PSO mengalami penurunan dari 80.40% menjadi 45.20%. Berdasarkan Tabel 5.5, nilai *recall* dari klasifikasi menggunakan dataset kata bahasa inggris lebih tinggi daripada menggunakan dataset kata sentimen karena banyak dokumen positif yang dikategorikan ‘negatif’ sebesar 548 dokumen dari 1000 dokumen positif saat menggunakan dataset kata sentimen (nilai FN lebih tinggi dari nilai TP). Ketika klasifikasi menggunakan dataset kata bahasa inggris, banyak dokumen positif yang dikategorikan ‘positif’ sebesar 804

dokumen dari 1000 dokumen positif (nilai TP lebih tinggi dari nilai FN).

5. Nilai TN lebih tinggi daripada nilai TP pada klasifikasi SVM dengan seleksi fitur PSO menggunakan dataset kata sentimen. Berdasarkan Tabel 5.6, dokumen negatif yang dikategorikan 'negatif' ada sebanyak 765 dari 1000 dokumen negatif saat menggunakan dataset kata sentimen. Namun, dokumen positif yang dikategorikan 'positif' ada sebanyak 452 dokumen dari 1000 dokumen positif. Ini dikarenakan jumlah kata pada dataset kata sentimen negatif (4781 kata) lebih banyak daripada jumlah kata pada dataset kata sentimen positif (2005 kata).

5.3.3. Skenario Pengujian 2: Perhitungan Nilai Akurasi, *Precision*, dan *Recall* dengan 4 Jumlah Dokumen yang Berbeda

Skenario uji coba kedua adalah perhitungan nilai akurasi, *precision*, dan *recall* dari klasifikasi, baik menggunakan metode SVM maupun menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Klasifikasi dilakukan pada 4 jumlah dokumen yang berbeda, yaitu 500, 1000, 1500, dan 2000 dokumen. Skenario ini bertujuan untuk melihat konsistensi klasifikasi opini review film menggunakan metode SVM dengan penambahan metode seleksi fitur PSO lebih baik terhadap metode SVM dengan jumlah data yang berbeda. Jumlah fitur, jumlah iterasi dan jumlah partikel pada parameter PSO diset pada nilai berturut-turut 200, 200, 100. Nilai akurasi dihitung dengan jumlah data yang mampu diklasifikasikan dengan benar. Sedangkan *precision* dan *recall* dihitung berdasarkan *confusion matrix*. *Confusion matrix* untuk hasil skenario uji coba ini dapat dilihat pada Tabel 5.10. Nilai akurasi, *precision* dan *recall* pada klasifikasi dengan jumlah data 500, 1000, 1500, 200 dapat dilihat pada Tabel 5.11. *Running time* uji coba kedua dapat dilihat pada Tabel 5.12.

Tabel 5.10 *Confusion matrix* skenario uji coba kedua

Jumlah Data	Kelas yang diprediksi	Kelas yang sebenarnya			
		SVM		SVM-PSO	
		Positif	Negatif	Positif	Negatif
500	Positif	0	0	143	84
	Negatif	250	250	107	166
1000	Positif	0	0	278	164
	Negatif	500	500	222	336
1500	Positif	3	1	383	250
	Negatif	747	749	367	500
2000	Positif	8	0	452	235
	Negatif	992	1000	548	765

Tabel 5.11 Hasil skenario uji coba kedua

Jumlah Data	Akurasi		Precision		Recall	
	SVM	SVM-PSO	SVM	SVM-PSO	SVM	SVM-PSO
500	50	61.80	0	62.99	0	57.20
1000	50	61.40	0	62.89	0	55.60
1500	50.13	58.87	75	60.51	0.40	51.07
2000	50.40	60.85	100	65.79	0.80	45.20

Tabel 5.12 *Running time* skenario uji coba kedua

Metode Klasifikasi	Jumlah Data	Waktu
SVM	500	13 menit 34 detik
	1000	21 menit 46 detik
	1500	38 menit 10 detik
	2000	54 menit 46 detik
SVM-PSO	500	14 jam 52 menit 11 detik
	1000	1 hari 18 jam 29 menit 15 detik
	1500	2 hari 8 jam 37 menit 45 detik
	2000	4 hari 16 jam 43 menit 42 detik

5.3.4. Analisis dan Evaluasi Skenario Pengujian 2

Pada hasil skenario uji coba 2, didapatkan hasil waktu yang berbanding lurus, yaitu semakin banyak data yang diklasifikasi, semakin lama pula waktu yang dibutuhkan untuk menjalankan program. Beberapa hal yang dapat diambil dari hasil uji coba ketiga selain waktu yang dibutuhkan adalah sebagai berikut :

1. Nilai akurasi, *precision*, dan *recall* terbaik pada klasifikasi menggunakan metode SVM didapatkan ketika jumlah dokumen yang diklasifikasikan sebesar 2000 dokumen.
2. Nilai akurasi dan *recall* terbaik pada klasifikasi menggunakan metode SVM dengan penambahan metode seleksi fitur PSO didapatkan ketika jumlah dokumen yang diklasifikasikan sebesar 500 dokumen. Nilai *precision* terbaik didapatkan ketika jumlah dokumen yang diklasifikasikan sebesar 2000 dokumen.
3. Berdasarkan Tabel 5.11, pada proses klasifikasi dengan menggunakan metode SVM, semakin banyak data yang diklasifikasikan, semakin tinggi nilai akurasi, *precision*, dan *recall* yang dihasilkan. Namun, hal ini berbeda pada proses klasifikasi menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Banyaknya data yang diklasifikasikan tidak terlalu berpengaruh pada nilai akurasi, *precision*, dan *recall* yang dihasilkan. Nilai akurasi dan *recall* cenderung menurun ketika jumlah data yang diklasifikasikan semakin banyak.

5.3.5. Skenario Pengujian 3: Perhitungan Nilai Akurasi, *Precision*, dan *Recall* dengan 4 Jumlah Iterasi pada Seleksi Fitur PSO yang Berbeda

Skenario uji coba ketiga adalah perhitungan nilai akurasi, *precision*, dan *recall* terhadap klasifikasi SVM dengan penambahan seleksi fitur PSO. Jumlah iterasi *swarm* pada seleksi fitur PSO diset sama dengan 50, 100, 150, dan 200 iterasi. Klasifikasi dilakukan pada 2000 dokumen opini review film.

Skenario ini bertujuan untuk mengetahui jumlah iterasi terbaik untuk melakukan penyeleksian fitur dengan metode PSO sehingga didapatkan hasil klasifikasi yang lebih baik. Jumlah fitur dan jumlah partikel pada parameter PSO diset sama dengan 200 dan 100. Nilai akurasi dihitung dengan jumlah data yang mampu diklasifikasikan dengan benar. Sedangkan *precision* dan *recall* dihitung berdasarkan *confusion matrix*. *Confusion matrix* untuk hasil skenario uji coba ini dapat dilihat pada Tabel 5.13. Nilai akurasi, *precision* dan *recall* pada klasifikasi dengan jumlah iterasi *swarm* ang berbeda pada seleksi fitur PSO dapat dilihat pada Tabel 5.14. *Running time* uji coba ketiga dapat dilihat pada Tabel 5.15.

Tabel 5.13 *Confusion matrix* skenario uji coba ketiga

Jumlah Iterasi	Kelas yang diprediksi	Kelas yang sebenarnya	
		Positif	Negatif
50	Positif	400	202
	Negatif	600	798
100	Positif	498	298
	Negatif	502	702
150	Positif	485	304
	Negatif	515	696
200	Positif	452	235
	Negatif	548	765

Tabel 5.14 Hasil skenario uji coba ketiga

Jumlah Iterasi	Akurasi	Precision	Recall
50	59.90	66.45	40
100	60	62.56	49.80
150	59.05	61.47	48.50
200	60.85	65.79	45.20

Tabel 5.15 *Running time* skenario uji coba ketiga

Jumlah Iterasi	Waktu
50	2 hari 3 jam 14 menit 37 detik
100	2 hari 17 jam 44 menit 3 detik
150	3 hari 2 jam 32 menit 53 detik
200	4 hari 16 jam 43 menit 42 detik

5.3.6. Analisis dan Evaluasi Skenario Pengujian 3

Pada hasil skenario uji coba 3, didapatkan hasil waktu yang berbanding lurus, yaitu semakin banyak iterasi yang dilakukan pada tahap seleksi fitur PSO, semakin lama pula waktu yang dibutuhkan untuk menjalankan program. Beberapa hal yang dapat diambil dari hasil uji coba ketiga selain waktu yang dibutuhkan adalah sebagai berikut :

1. Berdasarkan Tabel 5.13, peningkatan jumlah iterasi tidak berpengaruh terhadap hasil klasifikasi yang dilakukan. Dokumen positif cenderung dikategorikan ‘negatif’ dan dokumen negatif cenderung dikategorikan ‘negatif’ oleh sistem. Hal ini dapat dilihat dari nilai TP yang lebih kecil daripada nilai FN dan nilai TN yang lebih besar daripada nilai FP pada setiap percobaan.
2. Berdasarkan Tabel 5.14, peningkatan jumlah iterasi pada seleksi fitur PSO juga tidak berpengaruh pada nilai akurasi, *precision*, dan *recall* yang dihasilkan. Nilai akurasi terbaik didapatkan ketika jumlah iterasi sama dengan 200, yaitu sebesar 60.85%. Nilai *precision* terbaik didapatkan ketika jumlah iterasi sama dengan 50, yaitu sebesar 66.45%. Nilai *recall* terbaik didapatkan ketika jumlah iterasi sama dengan 100, yaitu sebesar 49.80%.

5.3.7. Skenario Pengujian 4 : Perhitungan Nilai Akurasi, *Precision*, dan *Recall* dengan Nilai *k* pada *K-Fold Cross Validation* yang Berbeda

Skenario uji coba keempat adalah perhitungan nilai akurasi, *precision*, dan *recall* dengan nilai *k* pada *k-fold cross validation* yang berbeda. Klasifikasi dilakukan pada 2000 dokumen menggunakan metode SVM dengan penambahan metode seleksi fitur PSO. Skenario ini bertujuan untuk mengetahui nilai *k* terbaik yang digunakan untuk tahap klasifikasi. Nilai *k* pada *k-fold cross validation* yang diuji cobakan ada 2, yaitu *k* sama dengan 2 dan *k* sama dengan 5. Jumlah fitur, jumlah iterasi dan jumlah partikel pada parameter PSO diset pada nilai berturut-turut 200, 200, 100. Nilai akurasi dihitung dengan jumlah data yang mampu diklasifikasikan dengan benar. Sedangkan *precision* dan *recall* dihitung berdasarkan *confusion matrix*. *Confusion matrix* untuk hasil skenario uji coba ini dapat dilihat pada Tabel 5.16. Nilai akurasi, *precision* dan *recall* pada klasifikasi menggunakan metode SVM dengan penambahan metode seleksi fitur PSO dapat dilihat pada Tabel 5.17. *Running time* skenario uji coba keempat dapat dilihat pada Tabel 5.18.

Tabel 5.16 *Confusion matrix* skenario uji coba keempat

Nilai *k*	Kelas yang diprediksi	Kelas yang sebenarnya	
		Positif	Negatif
2	Positif	456	261
	Negatif	544	739
5	Positif	452	235
	Negatif	548	765

Tabel 5.17 Hasil skenario uji coba keempat

Nilai *k*	Akurasi	Precision	Recall
2	59.75	63.60	45.60
5	60.85	65.79	45.20

Tabel 5.18 *Running time* skenario uji coba keempat

Nilai *k*	Waktu
2	13 jam 29 menit 46 detik
5	4 hari 16 jam 43 menit 42 detik

5.3.8. Analisis dan Evaluasi Skenario Pengujian 4

Pada hasil skenario uji coba 4, didapatkan hasil waktu yang berbanding lurus, yaitu semakin tinggi nilai *k* pada tahap klasifikasi, semakin lama pula waktu yang dibutuhkan untuk menjalankan program. Beberapa hal yang dapat diambil dari hasil uji coba keempat selain waktu yang dibutuhkan adalah sebagai berikut :

1. Berdasarkan Tabel 5.16, kenaikan nilai *k* berpengaruh terhadap dokumen yang diklasifikasikan. Jumlah dokumen yang dikategorikan 'negatif' mengalami penurunan dari 717 dokumen menjadi 687 dokumen. Hal ini dapat dilihat dari nilai TP yang lebih kecil daripada nilai FN dan nilai TN yang lebih besar daripada nilai FP pada setiap percobaan.
2. Berdasarkan Tabel 5.17, kenaikan nilai *k* berpengaruh pada nilai akurasi, *precision*, dan *recall* yang dihasilkan. Nilai akurasi mengalami peningkatan dari 59.75% menjadi 60.85%. Nilai *precision* juga mengalami peningkatan dari 63.60% menjadi 65.79%. Namun, nilai *recall* mengalami penurunan dari 45.60% menjadi 45.20%.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Lampiran A. 1 Kode Pengambilan Fitur berupa Sentimen

```
1   private void kumpulkanTermSentimen
    (ArrayList<ArrayList<String>> hasilEF) throws
    SQLException {
2       Sistem.out.println("Sedang proses
    pengumpulan term sentimen ...");
3       int size1, size2, size3;
4       Boolean value;
5       ArrayList<String> sentimen = new
    ArrayList<>(this.efdo.getSentimen());
6       size1 = sentimen.size();
7       size2 = hasilEF.size();
8       for (int i = 0 ; i < size1 ; i++) {
9           value = false;
10          for (int j = 0 ; j < size2 ; j++) {
11              size3 = hasilEF.get(j).size();
12              for (int k = 0 ; k < size3 ; k++) {
13                  if
    (sentimen.get(i).equals(hasilEF.get(j).get(k)))
    {
14
    this.term.add(sentimen.get(i));
15                      this.jumlahTerm++;
16                      value = true;
17                      break;
18                  }
19              }
20              if (value == true)
21                  break;
22          }
23      }
24      Sistem.out.println("Pengumpulan Term
    Sentimen BERHASIL (" + this.jumlahTerm + ").");
25  }
```

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan dari hasil uji coba terhadap sistem yang dibuat sebagai jawaban dari rumusan masalah yang diangkat pada bab pendahuluan dan saran untuk pengembangan sistem pada tugas akhir ini untuk kedepannya.

6.1. Kesimpulan

Kesimpulan yang dapat diberikan setelah proses pengerjaan dari proses perancangan, implementasi dan uji coba yang telah dilakukan terhadap sistem yang dibangun adalah sebagai berikut :

1. Ada beberapa tahap yang dilakukan sebelum data opini review film dapat diprediksi. Tahap pertama adalah ekstraksi fitur dengan pemrosesan teks. Tahap ini meliputi proses pembacaan data opini, *case folding*, *tokenization*, *stopwords removal*, dan pembobotan fitur dengan menggunakan metode *Term Frequency-Inverse Document Frequency* (TF-IDF). Bobot dari fitur inilah yang nantinya akan digunakan untuk memprediksi suatu opini termasuk 'positif' atau 'negatif'.
2. Algoritma *Particle Swarm Optimization* (PSO) digunakan untuk meningkatkan nilai akurasi pada klasifikasi *Support Vector Machine* (SVM). PSO melakukan seleksi dengan menentukan fitur-fitur yang berperan penting dalam proses klasifikasi. Jadi, jumlah fitur yang semula sebesar 4595 fitur dapat dioptimalkan menjadi 200 fitur. Nilai akurasi yang dihasilkan juga semakin meningkat.
3. Dari uji coba yang telah dilakukan, algoritma SVM-PSO terbukti lebih baik dalam melakukan pengklasifikasian data review film daripada menggunakan metode SVM. Dari 2000 dokumen yang diklasifikasikan, algoritma SVM-PSO menghasilkan nilai akurasi sebesar 60.85%, sementara itu algoritma SVM hanya menghasilkan nilai akurasi sebesar

50.40%. Namun, algoritma SVM-PSO membutuhkan *running time* 100 kali lebih lama daripada algoritma SVM.
4. Jumlah iterasi pada proses seleksi fitur dengan menggunakan algoritma PSO tidak berpengaruh pada nilai akurasi yang dihasilkan dalam pengklasifikasian opini review film.
5. Banyaknya data yang diklasifikasikan menggunakan metode SVM dengan penambahan seleksi fitur PSO tidak berpengaruh pada nilai akurasi yang dihasilkan.
6. Semakin besar nilai *k* pada *k-fold cross validation*, klasifikasi SVM dengan penambahan seleksi fitur PSO menghasilkan nilai akurasi yang lebih baik.

6.2.Saran

Saran yang dapat diberikan terhadap pengembangan sistem pada tugas akhir ini untuk kedepannya adalah sebagai berikut :

1. Perlu dilakukan ekstraksi fitur dengan pemrosesan teks lebih lanjut untuk mendapatkan data olahan yang lebih baik.
2. Penyimpanan data dapat menggunakan MySQL, SQL Server, ataupun aplikasi *database* yang lain. Sangat disarankan untuk tidak menggunakan JDBC (*Java Database Connectivity*) karena struktur *database* yang dibuat tidak dapat disalin.
3. Dibutuhkan waktu uji coba yang lebih lama. Hal ini bertujuan agar analisis dapat dilakukan lebih mendalam terhadap topik tugas akhir yang dikerjakan.

DAFTAR PUSTAKA

[1] K. Umamaheswari, S. Rajamohana and G. Aishwaryalakshmi, "Opinion Mining using Hybrid Methods," *International Journal of Computer Applications (0975 – 8887)* , pp. 18-21, 2015.

[2] A. S. H. Basari, B. Hussin, I. G. P. Ananta and J. Zeniarja, "Opinion Mining of Movie Review using Hybrid Method of Support Vector Machine and Particle Swarm Optimization," *Procedia Engineering 53,* pp. 453-462, 2013.

[3] N. M. Shelke, S. Deshpande and V. Thakre, "Survey of Techniques for Opinion Mining," *International Journal of Computer Applications (0975 - 8887) Volume 57 - No. 13,* pp. 30-35, 2012.

[4] T. Park, J. Lee and B. Choi, "Optimization for Artificial Neural Network with Adaptive inertial weight of particle swarm optimization," *2009 8th IEEE International Conference on Cognitive Informatics,* pp. 481-485, 2009.

[5] M. Badrul, "Prediksi Hasil Pemilu Legislatif DKI Jakarta dengan metode Neural Network berbasis Particle Swarm Optimization," 2012.

[6] B. M. Zahran and G. Kanaan, "Text Feature Selection using Particle Swarm Optimization Algorithm," *World Applied Sciences Journal 7 (Special Issue of Computer & IT),* pp. 69-74, 2009.

[7] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Foundations and Trends Information Retrieval,* vol. 2, pp. 1-135, 2008.

[8] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," *Proceedings of the ACM SIGKDD International Conference on Knowledge,* p. 10, 2004.

[9] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," *Evolutionary Programming VII: Proc. EP 98,* pp. 591-600, 1998.

[10] R. R. Saleh, M. M. Valvidia and A. Montejo-Raiz, "Experiments with SVM to classify opinions in different domains," *Expert Systems with Applications 38,* pp. 14799-14804, 2011.

[11] A. L. Maas, R. E. Daly, P. T. Pham and D. Huang, "Learning Word Vectors for Sentiment Analysis," *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011),* 2011.

[12] Y. D. Valle, "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems," *IEEE Transactions On Evolutionary Computation, Vol. 12, No. 2,* vol. 12, pp. 171-196, 2008.

[13] S. Kotsiantis, D. Kanellopoulos and P. Pintelas, "Data Preprocessing for Supervised Learning," *International Journal of Computer Science, vol. 1 N. 2,* pp. 111-117, 2006.

[14] A. S. Nugroho, A. B. Witarto and D. Handoko, "Support Vector Machine-Teori dan Aplikasinya dalam Bioinformatika," *IlmuKomputer.Com,* 2003.

BIODATA PENULIS

Penulis lahir di Surabaya, 25 September 1993. Penulis menyelesaikan sekolah dasarnya di SD Muhammadiyah 10 Surabaya, sekolah menengah pertama di SMP Muhammadiyah 1 Surabaya, sekolah menengah atas di SMA Negeri 6 Surabaya, dan pendidikan sarjana S1 di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Saat menempuh kuliah, penulis aktif di berbagai organisasi. Di tahun kedua, penulis menjadi staf di Keluarga Muslim Informatika (KMI-ITS), Jama'ah Masjid Manarul Ilmi (JMMI-ITS), Badan Eksekutif Mahasiswa (BEM-ITS), dan Himpunan Mahasiswa Teknik Computer-Informatika (HMTC-ITS). Di tahun ketiga, penulis melanjutkan keaktifannya dengan menjadi pengurus inti KMI-ITS. Penulis juga menjadi staf ahli di JMMI-ITS. Di tahun keempat, penulis aktif sebagai ketua Ikatan Mahasiswa Muhammadiyah Sepuluh Nopember (ITS, PENS, PPNS). Selain aktif berorganisasi, penulis juga pernah menjadi asisten dosen pada mata kuliah Matematika Diskrit dan Komputasi Numerik. Penulis juga menjadi administrator laboratorium Komputasi Cerdas dan Visi.

Dalam menyelesaikan pendidikan S1, penulis mengambil rumpun mata kuliah (RMK) Komputasi Cerdas dan Visi serta memiliki ketertarikan di bidang Analisis Media Sosial dan Sistem Temu Kembali Informasi. Untuk keperluan komunikasi, penulis dapat dihubungi melalui email : addien.hanief@gmail.com.