TUGAS AKHIR – KI091391

RANCANG BANGUN *MIDDLEWARE* PENDETEKSI PESAN *SPAM* PADA TWITTER

RADITYA ANDRE NURWITANTYO
NRP 5110 100 014

Dosen Pembimbing I
Ir. Muchammad Husni, M.Kom.

Dosen Pembimbing II
Baskoro Adi Pratomo, S.Kom.,M.Kom.

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2014

UNDERGRADUATE THESIS – KI091391

DESIGN AND DEVELOPMENT MIDDLEWARE FOR SPAM MESSAGE DETECTION ON TWITTER

RADITYA ANDRE NURWITANTO
NRP 5110 100 014

First Supervisor
Ir. Muchammad Husni, M.Kom.

Second Supervisor
Baskoro Adi Pratomo, S.Kom.,M.Kom.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2014

LEMBAR PENGESAHAN

RANCANG BANGUN *MIDDLEWARE* PENDETEKSI PESAN *SPAM* PADA TWITTER

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
RADITYA ANDRE NURWITANYO
NRP: 5110 100 014

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Ir. Muchammad Husni, M.Kom. .....
   NIP: 19600221 198403 1 001 (Pembimbing 1)

2. Baskoro Adi Pratomo, S.Kom., M.Kom. .....
   NIP: 510000003 (Pembimbing 2)

SURABAYA
Juli, 2014

RANCANG BANGUN *MIDDLEWARE* PENDETEKSI PESAN *SPAM* PADA TWITTER

Nama : **Raditya Andre Nurwitantyo**
NRP : **5110 100 014**
Jurusan : **Teknik Informatika – FTIf ITS**
Dosen Pembimbing I : **Ir. Muchammad Husni, M.Kom.**
Dosen Pembimbing II : **Baskoro Adi Pratomo, S.Kom., M.Kom.**

Abstrak

Bertambah tingginya penggunaan sosial media, menjadikan sosial media sebagai salah satu sarana dalam melakukan tindak kejahatan dalam dunia maya.

Dalam Tugas Akhir ini dibagun sebuah aplikasi *middleware* pendeteksi *tweet spam* yang dikirimkan kepada pengguna yang terdiri dari modul komunikasi protokol HTTP untuk mengakses *server* Twitter, komunikasi antara *client* dan *server* dengan protokol TCP, proses analisis kebiasaan pesan *tweet*, analisis kebiasaan pada profil pengirim *tweet*, proses analisis pesan *tweet* yang dilakukan dengan melakukan klustering untuk setiap *tweet* yang akan diperiksa menggunakan metode K-*means*, dan proses pengambilan keputusan menggunakan metode klasifikasi *decision tree*.

Sistem telah berhasil diimplementasikan dan dilakukan serangkaian uji coba fungsionalitas, kasus, dan performa. Uji coba fungsionalistas dilakukan dengan menguji proses dan jalannya suatu fungsi di dalam sistem mulai dari masukan hingga keluaran yang didapat yaitu analisis kebiasaan *tweet* dan analisis kebiasaan pada profil pengirim *tweet*. Pada uji coba kasus dilakukan dengan menguji penggunaan jumlah kata pada tiap *shingle* untuk memeriksa kesamaan teks, penentuan nilai *k* pada metode pengelompokan K-Means, dan penententuan model untuk pengambilan kuputusan menggunakan metode klasifikasi *decision tree.* Sedangkan pada uji coba performa dilakukan dengan

mengetahui waktu yang dibutuhkan untuk proses yang terjadi antara *client* dan *server* dalam mengolah data.

Dari hasil uji coba fungsionalitas dapat dilihat perbandingan akurasi pada model yang diujikan. Hasil dengan akurasi yang paling optimal digunakan untuk membentuk model dan jalannya proses hingga mampu mendapatkan hasil yang sesuai. Selain itu, hasil uji coba fungsionalitas dan performa ikut mendukung dalam menbangun sistem yang mampu memberikan hasil yang cukup baik.

Kata kunci: decision tree, deteksi spam, K-Means, Twitter, W-Shingling Resemblance.

DESIGN AND DEVELOPMENT OF MIDDLEWARE FOR SPAM MESSAGE DETECTION ON TWITTER

Name	**: Raditya Andre Nurwitantyo**
NRP	**: 5110 100 014**
Department	**: Teknik Informatika – FTIf ITS**
Supervisor I	**: Ir. Muchammad Husni, M.Kom.**
Supervisor II	**: Baskoro Adi Pratomo, S.Kom., M.Kom.**

Abstract

Since the number of social media networks users has been highly increasing, it makes social media become one of the target of committing a crime in cyberspace.

A middleware application for detecting spam tweets that sent to users has been built for this final project. This application consists of a communication module that using HTTP to access the Twitter server, the communication between client and server using TCP, analysing process of message behavior, analysing tweet using K-means clustering method, and decision-making process using decision tree classification method.

The system has been successfully implemented and carried out a series of test functionality, system, and performance. Functionality testing is done by giving inputs and see the outputs. Then system testing was done by testing the number of word that used in every shingle to check the text similarity, determine the value of k in K-means, and determine the model to make a decision using one a classification method decision tree. Lastly, the performance testing is done by giving the time required for the process that occured between the client and the server in processing the data.

From the test results we can see the comparison of the accuracies on the tested models. Model with best results was used, thus we could get appropriate result. Moreover, from the results of

functionality and percormance testing, the system could give good results.
Keywords: decision tree, K-Means, spam detector, Twitter, W-Shingling Resemblance

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah Subhanahu Wata'alla, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul "**RANCANG BANGUN *MIDDLEWARE* PENDETEKSI PESAN *SPAM* PADA TWITTER**" dengan baik.

Dalam kesempatan kali ini, penulis ingin mengucapkan terima kasih, memberikan penghormatan, dan apresiasi setinggi-tingginya kepada pihak-pihak yang telah berperan penting bagi penulis dalam menyelesaikan Tugas Akhir ini, maupun dalam proses menempuh pendidikan pada jenjang Strata Satu Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Penulis *haturkan* ucapan kepada:

1. Allah SWT yang selalu melimpahkan rahmat dan hidayah kepada hamba-Nya.
2. Bapak, Ibu, dan adik-adik yang selalu memberikan dukungan dalam berbagai hal agar penulis mampu terus memacu kemampuan dan mendapatkan yang terbaik.
3. Bapak Husni dan Bapak Baskoro selaku dosen pembimbing yang telah banyak membantu penulis dalam menuangkan ide menjadi sebuah Tugas Akhir.
4. Bapak Daniel selaku dosen wali yang telah memberikan nasihat dan masukannya kepada penulis dalam merancang program studi pada tiap semesternya.
5. Bapak/Ibu Dosen dan segenap Staf Karyawan Jurusan Teknik Informatika FTIF-ITS yang telah membantu penulis selama menempuh pendidikan di Jurusan "TC".
6. Happy Ayu Ch. yang telah membantu, menemani, menghibur, dan memberi semangat kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir.
7. Teman dan saudara angkatan 2010 dan C1A, kami mampu dengan segala keterbatasan kami untuk terus berkarya.

8. Teman-teman AJK dan GCL yang telah membantu penulis dalam memberi masukan dan semangat dalam menyelesaikan Tugas Akhir.
9. Pengurus Kabinet Bersatu tempat dimana penulis bergabung belajar dan membesarkan nama HMTC, yang menjadikan penulis sosok yang lebih bijak.
10. Dan semua kerabat, teman, dan saudara yang tidak dapat disebutkan satu per satu.

Permintaan maaf terbesar dari penulis atas segala kekurangan yang masih terdapat dari diri penulis, baik yang terdapat pada buku ini maupun hingga selesainya buku ini. Saran dan kritik yang membangun penulis terima dengan tangan terbuka.

Surabaya, 1 Juli 2014

Raditya Andre N.

DAFTAR ISI

KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xvii
DAFTAR TABEL xix
DAFTAR PERSAMAAN xxi
DAFTAR KODE SUMBER xxiii
BAB I PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Rumusan Permasalahan 2
1.3. Batasan Masalah 3
1.4. Tujuan dan Manfaat 3
1.4.1. Tujuan 3
1.4.2. Manfaat 4
1.5. Metodologi 4
1.6. Sistematika Penulisan 5
BAB II TINJAUAN PUSTAKA 7
2.1. Twitter 7
2.2. *Spam* 9
2.3. K-Means 9
2.4. *Decision Tree* 11
2.5. Weka 13
2.6. W-*Shingling resemblance* 15
BAB III PERANCANGAN PERANGKAT LUNAK 17
3.1. Deskripsi Umum Sistem 17
3.2. Arsitektur Sistem 18
3.3. Perancangan Basis Data 18
3.4. Skenario *Use Case* 19
3.5. *Fitur* Aplikasi 20
3.5.1. *Fitur* pada *Server* 20
3.5.2. *Fitur* pada *Client* 23
3.6. Diagram Alur Aplikasi Sistem 23
3.6.1. *Login* 23

3.6.2. Proses Pengambilan Data *Tweet* 24
3.6.3. Analisis Kebiasaan pada Profil Pengirim Tweet . 25
3.6.4. Analisis Kebiasaan Pesan *Tweet* 26
3.7. Rancangan Antarmuka 28
BAB IV IMPLEMENTASI 33
4.1. Implementasi Perangkat Keras 33
4.2. Implementasi Perangkat Lunak 34
4.2.1. Implementasi pada *Server* 34
4.2.2. Implementasi pada *Client* 37
4.3. Implementasi Antarmuka Perangkat Lunak 39
BAB V UJI COBA DAN EVALUASI 41
5.1. Lingkungan Uji Coba 41
5.1.1. Perangkat Keras 41
5.1.2. Perangkat Lunak 41
5.2. Uji Coba Fungsionalitas 42
5.2.1. *Login* 42
5.2.2. Memasukkan Kode Otorisasi 44
5.2.3. Meminta Data *Timeline* 45
5.2.4. Meminta Data *Mention* 46
5.2.5. Proses Analisis Kebiasaan Pesan *Tweet* 48
5.2.6. Proses Analisis Kebiasaan dan Profil Pengirim *Tweet* 49
5.3. Uji Coba Kasus 51
5.3.1. Uji Coba Jumlah Kata pada Setiap *Shingle* 52
5.3.2. Uji Coba Pengaruh Nilai *k* pada *Cluster* K-*Means* 55
5.3.3. Uji Coba Pemodelan Pengambilan Keputusan Menggunakan Metode Klasifikasi *Decision Tree* 56
5.4. Uji Coba Waktu Respon 57
BAB VI PENUTUP 61
6.1. Kesimpulan 61
6.2. Saran 62
DAFTAR PUSTAKA 63
LAMPIRAN 65

BIODATA PENULIS 85

DAFTAR TABEL

Tabel 3.1 Penjelasan *Use Case* sistem 20
Tabel 5.1 Uji Coba Proses *Login* 42
Tabel 5.2 Uji coba memasukkan kode otorisasi 44
Tabel 5.3 Uji coba meminta data *timeline* 45
Tabel 5.4 Uji coba meminta data *mention* 47
Tabel 5.5 uji coba proses analisis pesan *tweet* 48
Tabel 5.6 uji coba proses analisis kebiasaan dan profil pengirim tweet 49
Tabel 5.7 Tabel hasil uji coba jumlah n-kata pada setiap shingle 54
Tabel 5.8 Tabel untuk menentukan jumlah nilai–*k* pada metode K-*Means* 55
Tabel 5.9 Uji coba penentuan model 56
Tabel 5.10 Uji coba rata-rata akurasi pengunaan model pada pengguna 57
Tabel 5.11 uji coba performa pada 2 *client* 57

DAFTAR GAMBAR

Gambar 2.1 Kluster pertama 10
Gambar 2.2 Klasifikasi objek secara random 10
Gambar 2.3 Kluster berdasarkan *centroid* terdekat..... 11
Gambar 2.4 Hasil kluster akhir 11
Gambar 2.5 *Decision tree* 12
Gambar 2.6 Penggunaan Weka dengan algoritma *decision tree*. 14
Gambar 2.7 Representasi *tree* pada Weka 15
Gambar 3.1 Diagram arsitektur sistem 18
Gambar 3.2 *Conceptual Data Model* sistem..... 19
Gambar 3.3 *Use Case* sistem 19
Gambar 3.4 *Flowchart login* pada *client* 24
Gambar 3.5 *Flowchart login* pada *server* 25
Gambar 3.6 *Flowchart* proses pengambilan data *tweet* 26
Gambar 3.7 *Flowchart* analisis kebiasaan pada profil pengirim *tweet* 27
Gambar 3.8 *Flowchart* pemeriksaan kesamaan teks..... 29
Gambar 3.9 *Flowchart* pemeriksaan kesamaan URL 30
Gambar 3.10 Rancangan antar muka tampilan aplikasi..... 31
Gambar 4.1 *Pseudocode* proses *Login* pada *server* ke-1 35
Gambar 4.2 *Pseudocode* proses *Login* pada *server* ke-2 35
Gambar 4.3 *Pseudocode* proses pengambilan *tweet* 35
Gambar 4.4 *Pseudocode* proses kesamaan teks 36
Gambar 4.5 *Pseudocode* proses kesamaan URL..... 36
Gambar 4.6 Proses pengelompokan data menggunakan K-*Means* 36
Gambar 4.7 *Psudocode* proses analisis akun berdasarkan *tweet*. 37
Gambar 4.8 *Pseudocode* pengambilan keputusan menggunakan *decision tree* 38
Gambar 4.9 *Pseudocode* proses *login* pada *client*..... 38
Gambar 4.10 *Pseudocode* proses melakukan pencarian tempat . 39

Gambar 4.11 Implementasi tampilan aplikasi 40
Gambar 5.1 Tampilan *login* Twitter jika berhasi *login* 43
Gambar 5.2 Tampilan *login* Twitter jika gagal *login* 43
Gambar 5.3 Kode otorisasi diterima..... 45
Gambar 5.4 Kode otorisasi salah..... 46
Gambar 5.5 Data *timeline* berhasil ditampilkan pada layar aplikasi 47
Gambar 5.6 Data *mention* berhasil ditampilkan pada layar aplikasi 48
Gambar 5.7 *Server* mendapatkan hasil kesamaan URL dan teks 49
Gambar 5.8 *Server* mendapatkan hasil ujicoba skenario 1-3 51
Gambar 5.9 *Server* mendapatkan hasil uji coba skenario 4-6 51
Gambar 5.10 Contoh kumpulan *tweet* pada akun *spam* 53
Gambar 5.11 Salah satu *tweet* yang muncul pada *timeline* 53
Gambar 5.12 Kumpulan *tweet* terbaru pada *profile* pengirim 54
Gambar 5.13 Contoh data *tweet* yang dikategorikan sebagai *spam* 56
Gambar 5.14 Persentase akurasi penggunaan model..... 58
Gambar 5.15 Model klasifikasi *decision tree* ke-3..... 59

DAFTAR PERSAMAAN

Persamaan (2.1)……..16

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

Kode Sumber 1 proses *login* pada *server*.....66
Kode Sumber 2 proses *login* pada *client*.....67
Kode Sumber 3 proses pengambilan data *tweet*.....69
Kode Sumber 4 pemeriksaan kesamaan *tweet*.....70
Kode Sumber 5 pemeriksaan kesamaan URL.....72
Kode Sumber 6 analisis kebiasaan pada profil pengirim *tweet*...75
Kode Sumber 7 pengelompokan data menggunakan K-*Means*..76
Kode Sumber 8 pengambilan keputusan menggunakan *decision tree*.....77

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal yang menjadi dasar pada pengerjaan Tugas Akhir, meliputi latar belakang, rumusan masalah, tujuan, dan manfaat pembuatan Tugas Akhir, serta metodologi dan sistematika penulisan buku Tugas Akhir.

1.1. Latar Belakang

Teknologi informasi terus berkembang dengan sangat pesat. Berbagai macam teknologi informasi pun bermunculan, bermula dari televisi, telepon, *short message service* (SMS), hingga berbagai macam *social media*. Keberadaan *social media* sendiri merupakan sarana untuk bertukar informasi, tidak hanya kawula muda sebagai pengguna paling banyak, orang-orang penting seperti presiden negara juga menggunakan *social media* sebagai sarana komunikasi dan bertukar informasi. Salah satu *social media* yang menjadi tren saat ini adalah Twitter.

Twitter merupakan *social media* untuk saling berkomunikasi dan bertukar informasi antar tiap penggunanya. Dengan fungsi yang ditawarkan begitu sederhana dan tidak rumit, Twitter menjadi salah satu *social media* yang banyak digunakan oleh masyarakat. Berbagai macam aplikasi telah dibuat untuk mendukung masyarakat dalam berkomunikasi melalui Twitter, baik berupa aplikasi *desktop* maupun *mobile.* Hal ini mengakibatkan peningkatan pengguna Twitter dalam jangka waktu beberapa tahun terakhir. Dengan semakin bertambah tingginya angka pengguna Twitter, maka memunculkan peluang baru bagi pelaku-pelaku kejahatan di dunia maya salah satunya adalah mengirimkan pesan *spam.*

Spam merupakan suatu pesan yang dikirimkan oleh pengirim atau biasa disebut *spammer* kepada pengguna lain tanpa dikehendaki atau bahkan pengguna yang mendapat pesan tersebut

tidak pernah mengenal atau mengetahui sebelumnya siapa pengirim pesan tersebut. Hal ini jelas sangat merugikan pengguna.

Pada Twitter, *spam* telah menjadi salah satu hal yang sangat mengganggu. Pesan *spam* di Twitter biasanya berisi tentang promosi suatu produk atau layanan yang dikirimkan berupa *link* atau alamat *web* yang merujuk pada *website* tertentu. Namun dari sekian banyak pesan *spam* yang dikirimkan oleh *spammer*, mayoritas berisi *link* atau alamat *web* yang dapat membahayakan pengguna yang mengaksesnya.

Oleh karena itu pada Tugas Akhir ini, dirancang dan diimplementasikan suatu aplikasi *middleware* pendeteksi pesan *spam* pada Twitter. Aplikasi ini memiliki model *client-server* di mana *server* berfungsi untuk menyaring pesan yang masuk pada suatu akun Twitter sebelum sampai pada pemilik akun atau *client*. Dengan adanya aplikasi ini diharapkan dapat mengurangi pesan-pesan *spam* yang masuk sehingga tidak lagi meresahkan dan mengganggu pengguna Twitter.

1.2. Rumusan Permasalahan

Berikut beberapa hal yang menjadi rumusan masalah dalam Tugas Akhir ini:

a) Bagaimana proses pengambilan *tweet* yang muncul untuk dapat dideteksi sebagai *spam* tanpa diterima terlebih dahulu oleh pengguna?
b) Bagaimana cara menampung data *tweet* terlebih dahulu pada *server* tanpa terkirim dahulu pada *client*?
c) Bagaimana cara mengetahui jumlah frekuensi dari pengguna akun Twitter dalam mengirimkan pesan *tweet*?
d) Bagaimana cara mengetahui interval jarak waktu antar pesan *tweet* yang dikirimkan oleh pengguna akun Twitter?
e) Bagaimana cara mengambil kesimpulan akhir untuk menentukan apakah pesan *tweet* yang telah didistribusikan termasuk kategori *spam* atau tidak?
f) Bagaimana cara menampilkan *tweet* yang merupakan *tweet* yang sah?

1.3. Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada Tugas Akhir ini, yaitu:

a) Untuk mendeteksi *spam* atau tidaknya suatu *tweet*, aplikasi ini membutuhkan *history* dari *tweet* akun sebelumnya sehingga aplikasi ini tidak dapat berjalan secara *real time*, tetapi harus mengambil dan mengumpulkan data dari akun pengguna yang akan disimpan sebagai *history*.
b) *Tweet* yang dapat dideteksi sebagai *spam* atau tidak adalah *tweet* yang bersifat publik. *Tweet* yang bersifat privat atau berupa *direct message* tidak dapat dideteksi.
c) Tidak dapat mendeteksi suatu kondisi URL berbahaya yang terkandung pada *tweet*.
d) Tidak dapat melakukan pengambilan data *tweet* secara agresif seperti pengambilan *tweet* dalam jumlah banyak atau interval waktu yang terlalu cepat karena keterbatasan *library*.
e) Metode yang digunakan yaitu metode *clustering* K-*Means* dan metode klasifikasi *decision tree.*
f) Kinerja *clustering* dan klasifikasi yang dihasilkan diuji dengan 50 data *training* yang telah melalui tahap *pre-process*.

1.4. Tujuan dan Manfaat

Pemenuhan setiap *fitur* pada Tugas Akhir ini, untuk menghasilkan tujuan dan manfaat bagi pengguna. Tujuan dan manfaat tersebut diharapkan dapat menghasilkan suatu hal yang lebih baik di kemudian hari.

1.4.1. Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah membangun sebuah sistem *middleware* yang dapat membantu pengguna untuk mengurangi pesan *spam* yang masuk pada akun. Sistem ini juga dapat mencegah terjadinya hal-hal yang tidak diinginkan pengguna ketika mengakses pesan *spam* yang mayoritas mengandung URL dan merupakan URL berbahaya.

1.4.2. Manfaat

Dengan dibangunnya aplikasi ini, diharapkan mampu mengurangi pesan *spam* yang diterima pengguna sehingga dapat dibedakan mana yang merupakan pesan *spam* dengan pesan yang sah, serta diharapkan dapat mengurangi kejahatan pada dunia maya yang menyerang *social media* di mana mayoritas pesan *spam* mengandung unsur URL dan tidak sedikit dari pesan tersebut merupakan URL yang berbahaya.

1.5. Metodologi

Adapun langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir

   Tahap awal untuk memulai pengerjaan Tugas Akhir ini adalah penyusunan proposal Tugas Akhir. Pada proposal ini penulis mengajukan gagasan dan rancang bangun *middleware* pendeteksi suatu *tweet* yang dikategorikan sebagai pesan *spam* dengan menggunakan metode *clustering* dan klasifikasi *decision tree*. Apabila terdeteksi adanya *tweet* yang dikategorikan sebagai *spam* maka *tweet* tersebut akan disembunyikan dan tidak ditampilkan pada *client.*

2. Studi literatur

   Tugas Akhir ini menggunakan literatur *paper* beserta artikel dari internet. *Paper* yang digunakan adalah "*Towards Online Spam Filtering in Social Networks". Paper* ini menjadi acuan utama dan dasar dalam pengerjaan Tugas Akhir ini.

3. Perancangan sistem

   Tahap ini merupakan perancangan sistem dengan menggunakan studi literatur dan mempelajari konsep aplikasi yang akan dibuat. Dengan berbekal teori, metode, dan informasi yang sudah terkumpul pada tahap sebelumnya, diharapkan dapat membantu proses perancangan sistem.

4. Implementasi perangkat lunak

   Tahap ini merupakan implementasi rancangan sistem yang telah dibuat. Tahap ini merealisasikan apa yang terdapat pada

tahapan perancangan yang telah dibuat sebelumnya sehingga menjadi sebuah rancang bangun sistem yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi

Aplikasi akan diuji setelah selesai diimplemetasikan menggunakan skenario yang sudah dipersiapkan. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Pengujian dan evaluasi dimaksudkan untuk mengevaluasi jalannya program, mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

Pada tahap ini disusun laporan Tugas Akhir sebagai dokumentasi pelaksanaan Tugas Akhir, yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.6. Sistematika Penulisan

Buku Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut:

BAB I. PENDAHULUAN

Bab ini berisi latar belakang, permasalahan, tujuan, batasan permasalahan, metodologi, dan sistematika penulisan.

BAB II. TINJAUAN PUSTAKA

Bab ini berisi dasar teori yang mendukung pembahasan Tugas Akhir ini.

BAB III. PERANCANGAN PERANGKAT LUNAK

Bab ini berisi tentang perancangan sistem yang digambarkan dalam diagram aktivitas untuk menggambarkan alur proses yang terjadi, dan perancangan antarmuka yang akan dibuat.

BAB IV. IMPLEMENTASI PERANGKAT LUNAK

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya

BAB V. EVALUASI DAN UJI COBA

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan beberapa jenis pengujian seperti pengujian fungsionalitas, performa maupun dalam sebuah kasus. Pengujian

dilakukan dalam beberapa macam skenario yang berbeda maupun dilakukan dengan contoh data masukan yang berbeda untuk melihat perubahan yang dapat mempengaruhi perilaku sistem.

BAB VI. PENUTUP

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini dijelaskan beberapa hal mengenai teori yang berkaitan dengan sistem yang diimplementasikan. Hal ini ditujukan untuk memberikan gambaran secara umum terhadap sistem yang akan dibuat. Selain itu, hal tersebut berguna untuk menunjang pembuatan sistem sehingga kebutuhannya dapat diketahui.

2.1. Twitter

Twitter adalah layanan jejaring sosial yang memungkinkan penggunanya untuk mengirim dan membaca pesan berbasis teks hingga 140 karakter, yang dikenal dengan sebutan kicauan (*tweet*). Twitter didirikan pada bulan Maret 2006 oleh Jack Dorsey, dan situs jejaring sosialnya diluncurkan pada bulan Juli. Sejak diluncurkan, Twitter telah menjadi salah satu dari sepuluh situs yang paling sering dikunjungi di internet. Di Twitter, pengguna tak terdaftar hanya bisa membaca *tweet*, sedangkan pengguna terdaftar bisa mengirim *tweet* melalui antarmuka situs *web*, pesan singkat (SMS), atau melalui berbagai aplikasi untuk perangkat seluler. Melalui Twitter seorang pengguna atau pemilik akun dapat melakukan *update* status atau berinteraksi dengan sesama pemilik akun. Jika membutuhkan privasi dalam berinteraksi dengan sesama, pengguna dapat mengirimkan pesan pribadi ke pengguna lain yang biasa dikenal dengan sebutan *direct message*. Di dalam Twitter, terdapat istilah *follow* dan *followers* yang di mana *follow* berarti pengguna dapat mengikuti kebiasaan pengguna lain melalui *tweet–tweet* yang dikirimkan pengguna tersebut. Secara otomatis, *tweet* dari penggguna yang diikuti akan muncul di *timeline* pengguna. Sedangkan *followers* merupakan pengguna lain yang mengikuti (*follow*) akun kita.

Twitter mengalami pertumbuhan yang pesat dan dengan cepat meraih popularitas di seluruh dunia. Hingga bulan Januari

2013, terdapat lebih dari 500 juta pengguna terdaftar di Twitter, 200 juta di antaranya adalah pengguna aktif. Pada awal 2013, pengguna Twitter mengirimkan lebih dari 340 juta *tweet* per hari, dan Twitter menangani lebih dari 1,6 miliar permintaan pencarian setiap hari. Hal ini menyebabkan posisi Twitter naik ke peringkat kedua sebagai situs jejaring sosial yang paling sering dikunjungi di dunia, dari yang sebelumnya menempati peringkat dua puluh dua. Seiring dengan bertambah naiknya jumlah pengguna Twitter, angka kejahatan di dunia maya yang memanfaatkan Twitter sebagai media ikut bertambah tinggi [1].

Berikut istilah-istilah yang digunakan pada Twitter [2]:

a. *Timeline*, sekumpulan *tweet* terbaru dari teman kita yang ditampilkan secara *real-time.*
b. *Profile*, halaman pada Twitter yang berisi semua informasi pemilik akun dan menampilkan informasi tentang pemilik akun, serta kumpulan *tweet* yang telah oleh pemilik akun.
c. *Follow*, proses atau tindakan pengguna untuk menjadi teman pengguna lain pada Twitter untuk mengikuti kebiasaan seorang pengguna dalam mengirimkan *tweet* atau *update* status.
d. *Home*, sekumpulan *tweet* dari pemilik akun lain yang di-*follow* oleh pengguna dan ditampilkan secara *real-time.*
e. *Following*, daftar dari akun Twitter yang di-*follow* pemilik akun.
f. *Follower*, dari dari akun Twiitter yang mem-*follow* pemilik akun.
g. *Username*, *nickname* atau alias dari pemilik akun pada Twitter. Setiap *username* bersifat unik dan terdiri dar 15 karakter.
h. *Reply*, sebuat *tweet* yang dikirim untuk membalas pesan *tweet* yang dikirimkan oleh pengguna lain atau pesan *tweet* yang dimulai dengan *username* '@'.

2.2. *Spam*

Spam adalah pesan yang tidak diminta yang dikirim ke banyak orang. Contoh kiriman pesan yang berisi spam: iklan, advertensi, tawaran untuk bergabung ke MLM, undian, informasi palsu, *phishing*, penipuan [3].

Berikut adalah beberapa taktik umum yang sering digunakan *spammer* pada Twitter [4]:

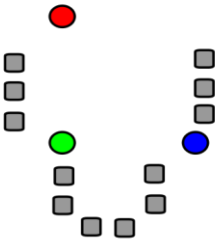
a. Mengirimkan atau memuat pesan *tweet* berulang kali supaya menjadi *trending topics* untuk mencoba menarik perhatian.
b. Mengirimkan atau memuat pesan *tweet* yang sama berulang kali.
c. Mengirimkan atau memuat *tweet* yang berisi *link* berbahaya.
d. Menyalahgunakan fungsi '@' *reply* atau '@' *mention* untuk mengirimkan pesat yang tidak diinginkan kepada pengguna.
e. Mengirimkan atau memuat *link* yang tidak relevan dengan isi *tweet.*
f. Melakukan aktivitas *follow* secara agresif.

2.3. K-Means

K-Means adalah suatu metode penganalisaan data atau metode Data *Mining* yang melakukan proses pemodelan tanpa supervisi (*unsupervised*) dan merupakan salah satu metode yang melakukan pengelompokan data dengan sistem partisi. Metode K-*means* berusaha mengelompokkan data yang ada ke dalam beberapa kelompok, di mana data dalam satu kelompok mempunyai karakteristik yang sama satu sama lainnya dan mempunyai karakteristik yang berbeda dengan data yang ada di dalam kelompok yang lain. Dengan kata lain, metode ini berusaha untuk meminimalkan variasi antar data yang ada di dalam suatu *cluster* dan memaksimalkan variasi dengan data yang ada di *cluster* lainnya [5].

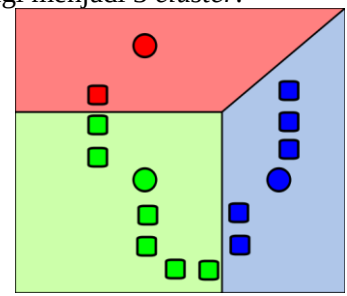
Berikut langkah-langkah K-*means* dalam melakukan proses *clustering* [6]:

- Awalnya terdapat sekelompok objek dengan variabel dan nilai yang berbeda, yang membuat koordinat di bidang XY berbeda satu sama lain. Objek-objek yang ada belum ter-*cluster*. Objek berwarna merah, hijau dan biru merupakan *centroid* yang sudah dibagi pada Gambar 2.1.



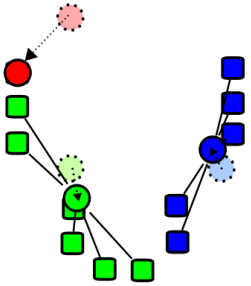
Gambar 2.1 Kluster pertama [6]

- Langkah berikutnya adalah mengklasifikasikan objek-objek tadi ke dalam kategori yang ada secara *random*. Pada Gambar 2.2 direpresentasikan bahwa nilai *k*=3 yang berarti dibagi menjadi 3 *cluster*.



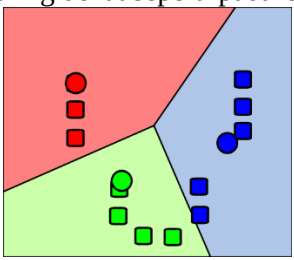
Gambar 2.2 Klasifikasi objek secara random [6]

- Langkah selanjutnya pada Gambar 2.3 yaitu membandingkan objek-objek yang ada dengan seluruh nilai *centroid*. Masing-masing objek mencari nilai *centroid* yang paling dekat dengan mencari selisih koordinat dari objek tersebut dengan *centroid* tujuan.



Gambar 2.3 Kluster berdasarkan *centroid* terdekat [6]

- Langkah 2 dan 3 diulangi terus menerus hingga menemukan hasil *cluster* yang optimal berdasarkan nilai *centroid* yang paling dekat seperti pada Gambar 2.4.

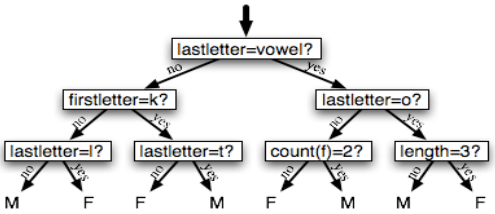


Gambar 2.4 Hasil kluster akhir [6]

2.4. *Decision Tree*

Pohon Keputusan (*Decision Tree*) merupakan metode klasifikasi dan prediksi yang sangat kuat dan terkenal. Metode pohon keputusan merubah fakta yang sangat besar menjadi pohon keputusan yang merepresentasikan aturan. Aturan dapat dengan mudah dipahami dengan bahasa alami. Aturan ini juga dapat diekspresikan dalam bentuk bahasa basis data seperti SQL untuk mencari *record* pada kategori tertentu. Pohon keputusan juga berguna untuk mengeksplorasi data, menemukan hubungan tersembunyi antara sejumlah calon variabel masukan dengan sebuah variabel target. Karena pohon keputusan memadukan antara eksplorasi data dan pemodelan, pohon keputusan ini sangat

bagus sebagai langkah awal dalam proses pemodelan bahkan ketika dijadikan sebagai model akhir dari beberapa teknik lain [7]. Contoh penggunaan *decision tree* pada Gambar 2.5.



Gambar 2.5 ***Decision tree***

Salah satu keuntungan dari *decision tree* adalah sifat transparan dalam pengambilan keputusan. Tidak seperti model pengambilan keputusan lainnya, *decision tree* membuat semua kemungkinan yang ada terlihat eksplisit dan metode ini mampu melacak setiap keputusan yang mungkin dalam satu model, di mana hal ini memudahkan perbandingan setiap keputusan yang akan diambil. Penggunaan *node* dalam menghasilkan suatu keputusan terbentuk secara transparan.

Dalam proses pembentukan *tree* terdapat 2 metode yang diterapkan, yaitu ID3 dan C45. ID3 merupakan algoritma yang digunakan untuk membangun sebuah *decision tree* yang ditemukan oleh J. Ross Quinlan, dengan memanfaatkan Teori Informasi atau *Information Theory* milik Shanon. ID3 sendiri merupakan singkatan dari *Iterative Dichotomiser 3*.

Algoritma ID3 membentuk pohon keputusan dengan metode *divide-and-conquer* data secara rekursif dari atas ke bawah. Strategi pembentukan *decision tree* dengan algoritma ID3 adalah:

- Pohon dimulai sebagai node tunggal (akar/*root*) yang merepresentasikan semua data.
- Sesudah *node root* dibentuk, maka data pada *node root* akan diukur dengan *information gain* untuk dipilih atribut mana yang akan dijadikan atribut pembaginya.

- Sebuah cabang dibentuk dari atribut yang dipilih menjadi pembagi dan data akan didistribusikan ke dalam cabang masing-masing.
- Algoritma ini akan terus menggunakan proses yang sama atau bersifat rekursif untuk dapat membentuk sebuah *decision tree*. Ketika sebuah atribut telah dipilih menjadi *node* pembagi atau cabang, maka atribut tersebut tidak diikutkan lagi dalam penghitungan nilai *information gain*.
- Proses pembagian rekursif akan berhenti jika salah satu dari kondisi dibawah ini terpenuhi:
    1. Semua data dari anak cabang telah termasuk dalam kelas yang sama.
    2. Semua atribut telah dipakai, tetapi masih tersisa data dalam kelas yang berbeda. Dalam kasus ini, diambil data yang mewakili kelas yang terbanyak untuk menjadi label kelas pada *node* daun.
    3. Tidak terdapat data pada anak cabang yang baru. Dalam kasus ini, *node* daun akan dipilih pada cabang sebelumnya dan diambil data yang mewakili kelas terbanyak untuk dijadikan label kelas.

Dari semua sifat yang dimiliki oleh ID3, beberapa kelebihan yang dimiliki oleh C45.

- Mampu menangani atribut dengan tipe diskrit atau kontinu.
- Mampu menangani atribut yang kosong (*missing value*).
- Bisa memangkas cabang.

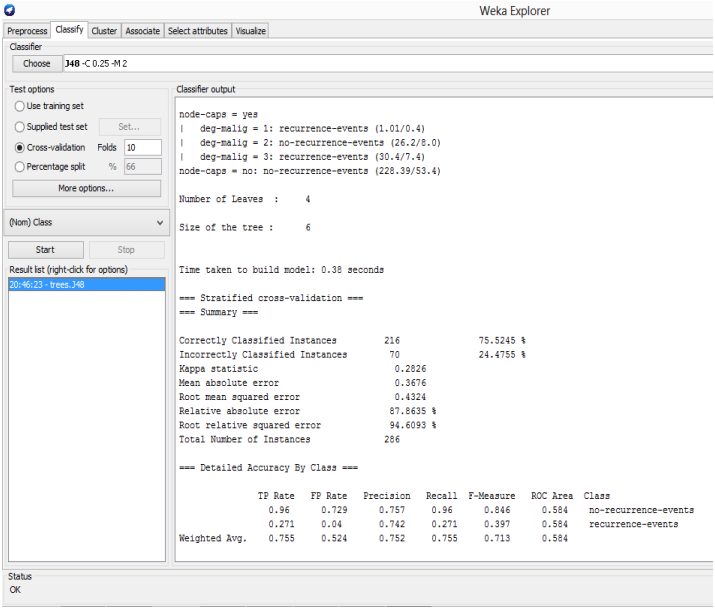
2.5. Weka

Weka merupakan kakas yang populer dalam bidang data mining. Kakas ini dikembangkan oleh Universitas Wakaito (*University of* Waikato), New Zealand. Di dalam weka terdapat *fitur* untuk data *pre-processing*, klasifikasi, regresi, *clustering*, aturan asosiasi, serta visualisasi data.

Weka dikembangkan menggunakan bahasa pemrograman Java. Aplikasi tersedia dalam bentuk GUI (*Graphical User*

Interface) serta dapat digunakan sebagai pustaka pemrograman. Melihat dari kebutuhan sistem yang mengimplemetnasikan bahasa pemrograman C# dalam pengembangan, Weka digunakan untuk mendapatkan hasil akurasi yang paling tinggi pada kebutuhan uji coba dan menghasilkan pemodelan yang paling optimal.

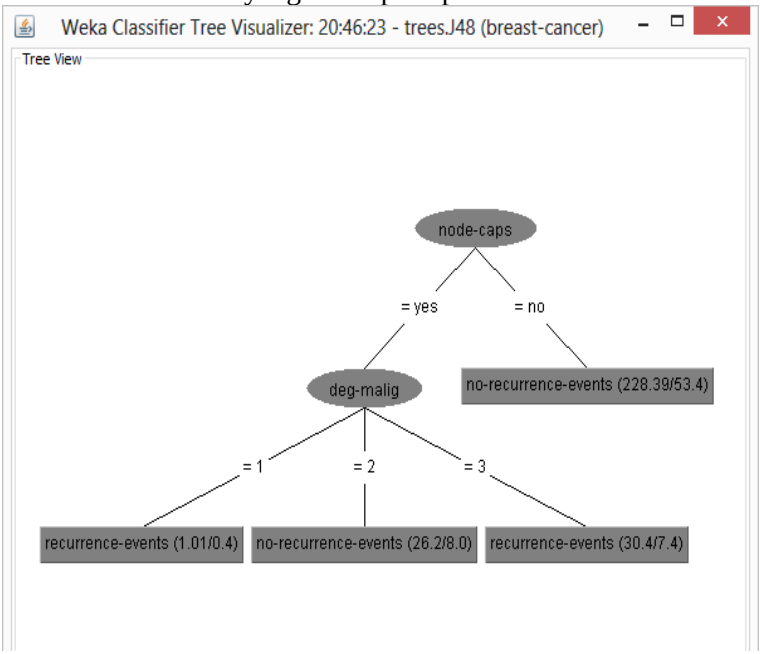
Weka menggunakan dokumen berformat ARFF sebagai masukkan data *training* ataupun sebagai data *set*. Selain menggunakan data *set* untuk pengujian, pustaka pemrograman Weka dapat menggunakan masukkan data uji yang dibuat melalui kode sumber. Dalam pengembangan, digunakan Weka dengan *versi* 3.6.



Gambar 2.6 Penggunaan Weka dengan algoritma *decision tree*

Gambar 2.6 merupakan contoh dari penggunaan Weka untuk melihat akurasi yang dimiliki oleh suatu data *training*. Pada contoh di atas terdapat sekumpulan data *training* mengenai kondisi-kondisi yang mengarah pada ada tidaknya suatu penyakit kanker. Data *training* tersebut akan digunakan untuk membentuk model

yang mampu mendeteksi penyakit kanker pada seseorang. Proses pengambilan keputusan dilakukan dengan menggunakan metode klasifikasi *decision tree* dengan menerapkan *tree* C45 atau yang dikenal sebagai J48. Untuk mengetahui akurasi dari data *training* yang dijadikan model, maka digunakan Weka sebagai alat dalam pengolahan data. Berdasarkan contoh tersebut dapat dilihat, akurasi yang dimiliki dari model adalah 75,52% disertai dengan nilai *precission* dan *recall*. Selain itu, untuk penggunaan algoritma *decision tree*, Weka mampu merepresentasikan *tree* yang terbentuk dari model yang ada seperti pada Gambar 2.7.



Gambar 2.7 Representasi *tree* pada Weka

2.6. W-*Shingling Resemblance*

W-*Shingling* merupakan metode pemeriksaan kesamaan teks. Pemeriksaan kesamaan suatu teks dilakukan dengan cara membagi teks menjadi *unique shingles* [8].

Contoh: “saya makan nasi goreng di depan Sakinah”

- Pertama pecah teks berdasarkan spasi.
  (saya, makan, nasi, goreng, di, depan, Sakinah)
- Kelompokkan kata-kata tersebut per *shingle* yang berisi n-kata. Dimisalkan per *shingle* berisi 4 kata sehingga didapatkan 4 *shingle*.
  - (saya makan nasi goreng), (makan nasi goreng di), (nasi goreng di depan), (goreng di depan sakinah)

Pencocokan nilai kesamaan teks dapat dilakukan dengan formula:

$$\boldsymbol{r(A,B) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}} \quad \textbf{(2.1)}$$

BAB III
PERANCANGAN PERANGKAT LUNAK

Bab ini secara khusus akan menjelaskan perancangan sistem yang dibuat dalam Tugas Akhir ini.

3.1. Deskripsi Umum Sistem

Pada Tugas Akhir ini dibangun suatu sistem *middleware* yang dapat mendeteksi *tweet* berupa pesan *spam* dengan menggunakan metode kluster dan klasifikasi.

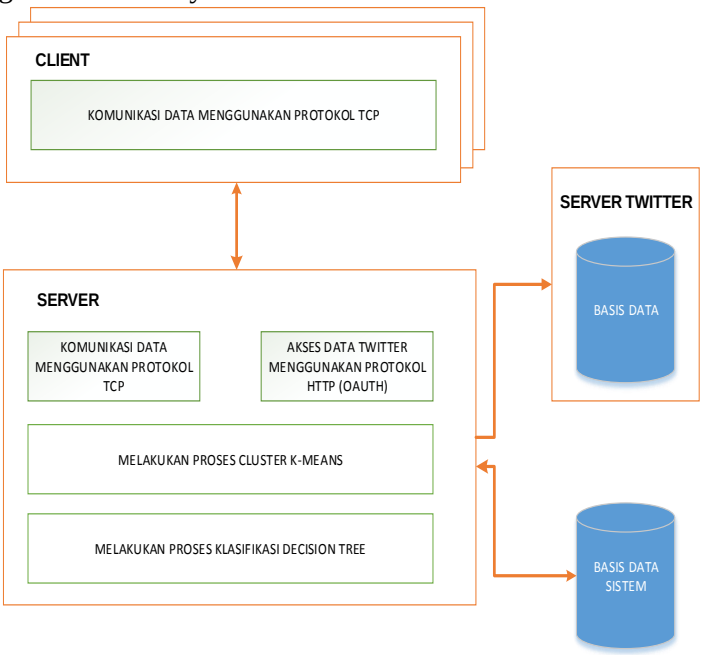
Pada tahap kluster, data *tweet* yang diterima dikelompokkan berdasarkan kemiripannya menggunakan metode K-*Means* dengan dua buah nilai atribut yaitu nilai kesamaan *tweet* dan nilai rata-rata kesamaan URL. Sehingga menghasilkan beberapa *cluster* dan diambil *cluster* yang berukuran paling besar sebagai *cluster* yang terindikasi merupakan *tweet* sah atau bukan *spam*.

Metode klasifikasi di sini bertujuan untuk menyatakan apakah data *tweet* pada *cluster* bukan *spam* merupakan *tweet* yang terbukti sah. Metode klasifikasi yang digunakan adalah pohon keputusan (*decision tree*). Klasifikasi dilakukan menggunakan atribut jumlah *tweet* berisi interaksi atau *reply*, perbandingan *following* dengan *follower*, jumlah *tweet* per hari, interaksi terhadap pemilik akun, rata-rata penggunaan URL pada pengiriman *tweet*, dan rata-rata interval pengiriman *tweet*. Detail analisis penggunaan atribut sebagai dasar untuk metode klasifikasi dapat dilihat pada Lampiran G.

Sistem ini menggunakan arsitektur *client-server* dimana sebuah *server* bertindak sebagai *middleware* untuk pengambilan dan pengolahan data *tweet* sesuai permintaan *client*. Proses komputasi pada *server* berupa pengelompokan *tweet* menggunakan metode *clustering* dan memutuskan apakah *tweet* tersebut berupa pesan sah dengan menggunakan metode klasifikasi. Sehingga data *tweet* yang diterima oleh *client* merupakan pesan yang sah atau bukan *spam*.

3.2. Arsitektur Sistem

Rancangan arsitektur dari sistem yang dibuat dapat dilihat pada Gambar 3.1. Pada sisi *client* maupun *server* memiliki blok modul tersendiri yang bekerja untuk memenuhi kebutuhan sistem. Dalam suatu proses, tiap blok modul bekerja saling berkaitan dengan modul lainnya.



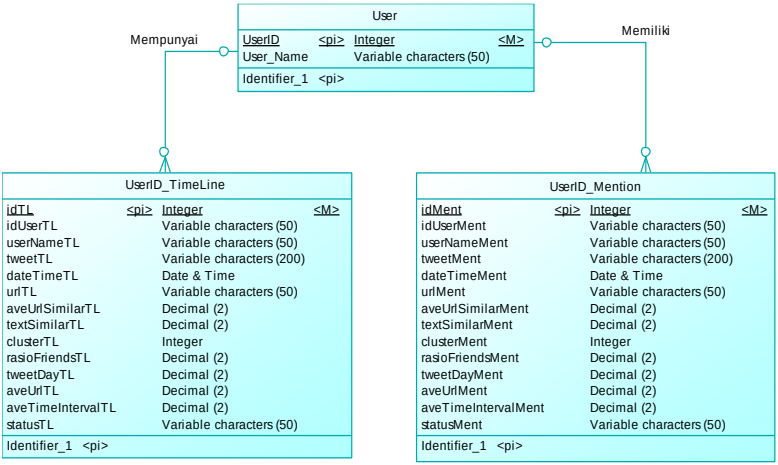
Gambar 3.1 Diagram arsitektur sistem

3.3. Perancangan Basis Data

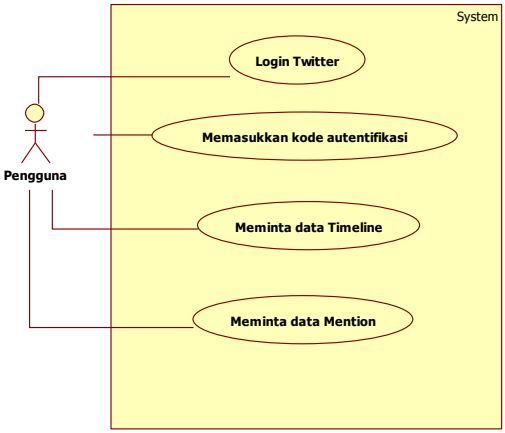
Perancangan basis data menggambarkan struktur tabel yang akan digunakan dalam penyimpanan kebutuhan data sistem pada basis data. Gambar 3.2 menggambarkan *Conceptual Data Model* dari basis data yang digunakan dalam sistem ini. Terdapat empat tabel dengan masing-masing atribut yang digunakan untuk menyimpan kebutuhan sistem.

3.4. Skenario *Use Case*

Skema *use case* pada sistem ini dapat dilihat pada Gambar 3.3. Sedangkan pada Tabel 3.1 berisi penjelasan dari masing-masing *activity* yang terdapat pada *use case*.



Gambar 3.2 *Conceptual Data Model* sistem



Gambar 3.3 *Use Case* sistem

Tabel 3.1 Penjelasan *Use Case* sistem

No	Kode	Nama *Use Case*	Keterangan
1	UC-001	*Login* Twitter	Pengguna dapat masuk ke dalam aplikasi
2	UC-002	Memasukkan kode otorisasi	Pengguna dapat memasukkan kode otorisasi
3	UC-003	Meminta data *timeline*	Pengguna dapat meminta data *timeline*
4	UC-004	Meminta data *mention*	Pengguna dapat meminta data *mention*

3.5. *Fitur* Aplikasi

Sistem ini dikembangkan pada dua sisi, yakni sisi *server* dan sisi *client*, dimana keduanya dapat memiliki beberapa *fitur* yang berbeda sesuai dengan peran dan fungsinya masing-masing. *Fitur-fitur* tersebut akan dijelaskan pada subbab-subbab berikut ini.

3.5.1. *Fitur* pada *Server*

Aplikasi pada *server* dapat melakukan beberapa hal sebagai berikut:

a) Menerima *request* dan mengirim data pada *client* melalui TCP

Sudah menjadi tugas utama *server* dalam memberikan data yang dibutuhkan oleh *client*. Untuk itu *server* harus dapat berkomunikasi melakukan proses pertukaran data dengan *client*. Pada sistem ini digunakan protokol TCP sebagai jalur komunikasi antara *client* dengan *server*.

b) Mengirim *request* dan menerima data dari *server* Twitter melalui HTTP

Server akan meminta *request* data pada *server* Twitter setelah menerima *request* dari *client*. Data yang diminta adalah *request login, profile, timeline,* dan *mention*. Pada sistem ini protokol HTTP digunakan sebagai jalur komunikasi.

c) *Login*

Fitur *login* pada server dibagi menjadi dua proses yaitu:

- *Server* menerima *request* dari *client* untuk *login* pada Twitter. Setelah itu *server* akan meminta *request token* untuk proses otorisasi dan *server* akan mengembalikan *request token* berupa alamat *web* yang akan dikirimkan pada *client* agar *client* dapat melakukan otorisasi pada Twitter untuk menggunakan aplikasi.
- *Server* menerima kode otorisasi yang dikirimkan oleh *client*. Setelah itu *server* akan mengirimkan kode tersebut untuk melakukan proses otorisasi sehingga mendapatkan data berupa akses *token* dan akses *token secret* dari akun *client.*

d) Membaca dan menulis pada basis data

Basis data yang digunakan untuk menampung seluruh kebutuhan data sistem terletak pada *server.* Sehingga jika *client* membutuhkan suatu informasi maka *server* yang akan menyediakannya. Untuk itu aplikasi pada *server* akan melakukan proses baca tulis pada basis data utuk memenuhi kebutuhan sistem.

e) Mengambil data *timeline* atau *mention*

Client akan mengambil 50 *tweet timeline* atau *tweet mention* sesuai permintaan dari *client*.

f) Melakukan proses pemeriksaan kesamaan teks

Proses dilakukan dengan mengolah data *timeline* atau *mention client* satu per satu untuk diperiksa kesamaan teksnya. Proses pemeriksaan kesamaan teks dilakukan dengan cara mengambil profil dari pengirim *tweet* tersebut yang berupa 30 *tweet* terbaru yang akan dicocokkan dengan *tweet* yang diterima oleh *client.*

g) Melakukan proses pemeriksaan kesamaan URL

Proses dilakukuan dengan mengolah data *timeline* atau *mention client* satu per satu untuk diperiksa kesamaan URL nya. Proses pemeriksaan kesamaan URL dilakukan dengan cara mengambil profil dari pengirim *tweet* tersebut yang berupa 30 *tweet* terbaru yang akan dicocokkan dengan *tweet* yang diterima oleh *client.*

h) Melakukan proses penghitungan persentase jumlah *tweet* yang mengandung interkasi

Proses penghitungan dilakukan dengan cara mengambil profil dari pengirim *tweet* yang berupa 30 *tweet* terbaru. *Tweet* tersebut akan diperiksa satu per satu apakah mengandung interaksi.

i) Melakukan proses penghitungan perbandingan *following* dengan *follower*

Proses penghitungan dilakukan dengan cara mengambil profil dari pengirim *tweet* yang berupa jumlah *following* dan *follower*.

j) Melakukan proses penghitungan jumlah *tweet* per hari

Proses penghitungan dilakukan dengan cara mengambil profil dari pengirim *tweet* yang berupa tanggal dibuatnya akun Twitter dan total jumlah *tweet* yang dibuat.

k) Melakukan proses penghitungan interval pengiriman setiap *tweet*

Proses penghitungan dilakukan dengan cara mengambil profil dari pengirim *tweet* yang berupa waktu pengiriman setiap *tweet* dan dihitung intervalnya.

l) Melakukan proses penghitungan interaksi *client* dengan pengirim *tweet*

Proses penghitungan dilakukan dengan cara mengambil *tweet* terbaru yang dikirim *client* dan diperiksa apakah mengandung interaksi dengan pengirim *tweet.*

m) Melakukan proses penghitungan persentase penggunaan URL pada *tweet*

Proses penghitungan dilakukan dengan cara mengambil profil dari pengirim *tweet* yang berupa *tweet* terbaru dan diperiksa satu per satu apakah mengandung URL dan dihitung persentasenya terhadap jumlah *tweet* yang diambil.

n) Melakukan proses pengelompokan *tweet* menggunakan metode K-*Means*

Proses ini dilakukan setiap ada *tweet* baru yang diambil. Pengelompokan bertujuan untuk membedakan *tweet* yang terindikasi merupakan pesan *spam* atau tidak.

o) Melakukan proses pengambilan keputusan menggunakan metode *decision tree*

Proses ini dilakukan untuk melakukan pengambilan keputusan *tweet* merupakan pesan *spam* atau tidak.

3.5.2. *Fitur* pada *Client*

Adapun *fitur-fitur* pada *client* yang tersedia adalah sebagai berikut:

a) *Login*

Fitur login pada *client* dibagi menjadi dua proses yaitu:

- *Client* meminta alamat *web* pada *server* untuk melakukan otorisasi penggunaan aplikasi pada Twitter.
- *Client* mendapatkan kode otorisasi yang akan dikirimkan pada *server*.

b) Meminta dan menampilkan data *timeline*

Client meminta data *timeline* terbaru pada *server* dan menampilkannya.

c) Meminta dan menampilkan data *mention*

Client meminta data *mention* terbaru pada *server* dan menampilkannya.

3.6. Diagram Alur Aplikasi Sistem

Diagram alur yang menggambarkan alur kerja suatu proses yang terdapat pada sistem akan dijelaskan pada *flowchart* yang terdapat pada subbab-subbab berikut.

3.6.1. *Login*

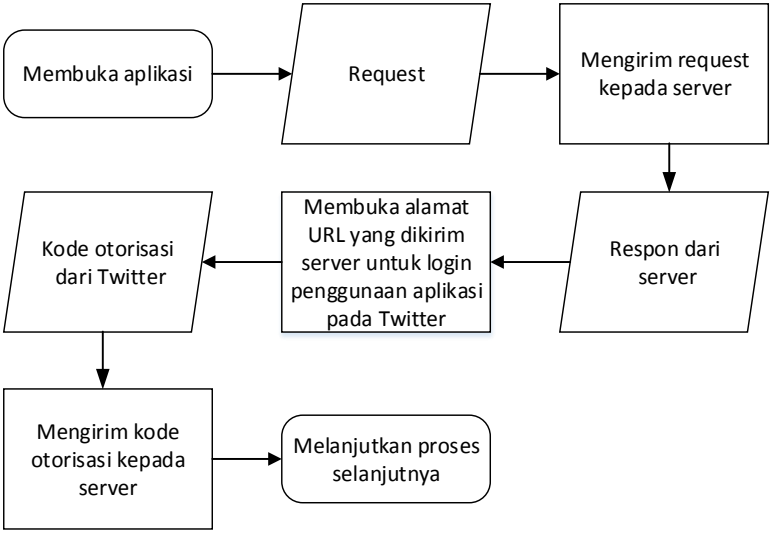
Proses *login* pada sistem dibagi menjadi dua bagian yaitu:

a. *Login* pada *client*

Gambar 3.4 memodelkan alur terjadinya proses *login* pada *client. Client* akan melakukan *request* koneksi pada *server*. *Client* akan menerima balasan *request token* yang berupa alamat URL untuk *login* aplikasi pada Twitter. Setelah *client* melakukan *login*, *client* akan mendapatkan kode otorisasi yang akan dikirimkan pada *server*.

b. *Login* pada *server*

Gambar 3.5 memodelkan alur terjadinya proses *login* pada *server*. *Server* akan memproses *request* permintaan *login* dari *client*. Kemudia *server* melakukan *request* pada *server* Twitter untuk mendapatkan *request token* berupa alamat URL yang akan dikirimkan pada *client. Server* akan mendapatkan balasan berupa kode otorisasi yang akan dikirimkan pada *server* Twitter untuk mendapatkan akses *token* dan akses *token secrete* dari akun *client.*

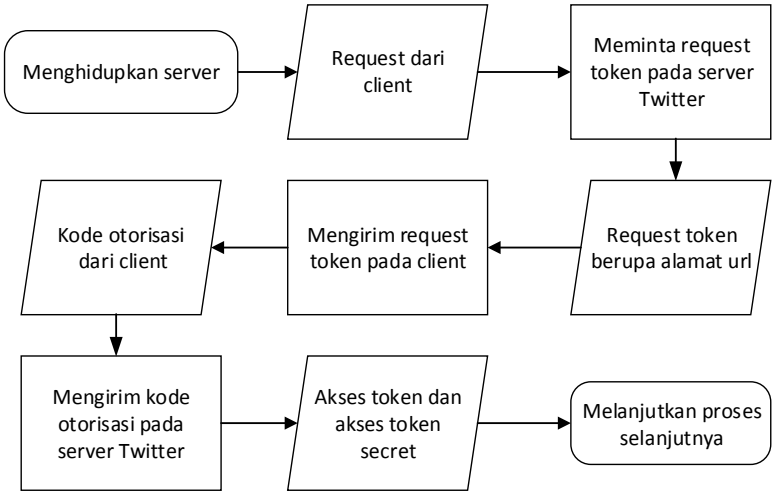


Gambar 3.4 *Flowchart login* pada *client*

3.6.2. Proses Pengambilan Data *Tweet*

Proses pengambilan data *tweet* dilakukan oleh sistem ketika pengguna baru pertama kali menggunakan aplikasi. Ketika aplikasi sudah berjalan, aplikasi mengambil secara otomatis 50 *tweet mention* atau *timeline* yang akan disimpan dalam basis data. Data-data tersebut akan dikelompokan menggunakan metode *clustering* K-*Means*. Hasil dari pengelompokan data tersebut akan

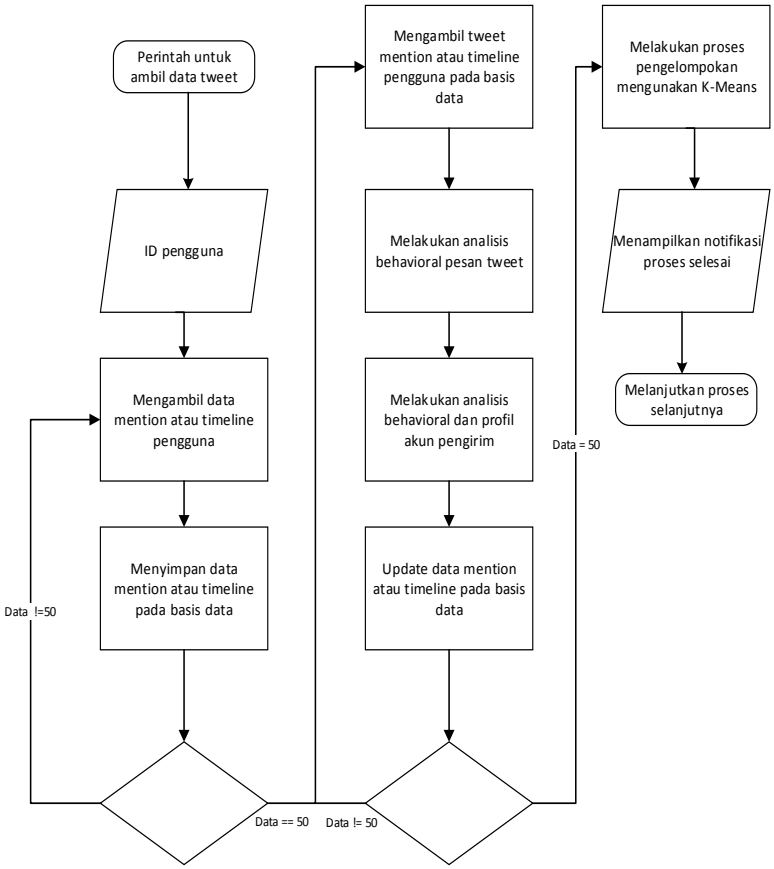
dijadikan data *training* untuk pengambilan keputusan merupakan *tweet spam* atau normal. Proses dapat dilihat pada Gambar 3.6.



Gambar 3.5 *Flowchart login* pada *server*

3.6.3. Analisis Kebiasaan pada Profil Pengirim Tweet

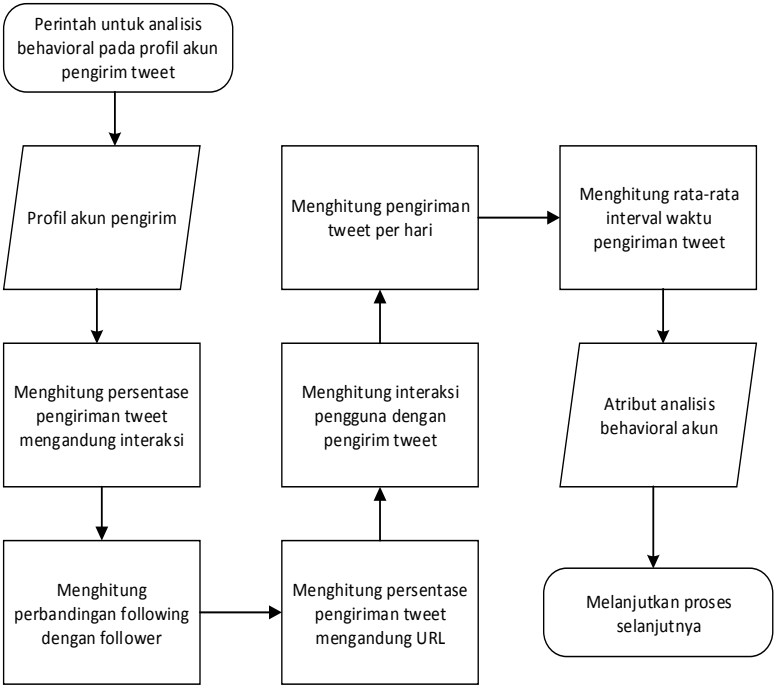
Proses analisis kebiasaan pengirim *tweet* dilakukan untuk mengambil kebiasaan pengirim *tweet* dalam mengirim pesan *tweet.* Dalam proses analis, sistem akan mengevaluasi atribut pada setiap *tweet* yang diterima penguna dengan cara mengambil profil dari akun pengirim *tweet* tersebut. Atribut yang diambil pada akun pengirim yaitu jumlah *tweet* berisi interaksi atau *reply*, perbandingan *following* dengan *follower*, jumlah *tweet* per hari, interaksi terhadap pemilik akun, rata-rata pengunaan URL pada pengiriman *tweet*, dan rata-rata interval waktu pengiriman *tweet*. Setelah semua proses selesai dilakukan dan *tweet* yang pengguna terima memiliki atribut, sistem akan menjalankan proses selanjutnya. Proses dapat dilihat pada Gambar 3.7.



Gambar 3.6 *Flowchart* proses pengambilan data *tweet*

3.6.4. Analisis Kebiasaan Pesan *Tweet*

Prosses analisis kebiasaan pesan *tweet* dilakukan untuk mengambil atribut dari kebiasaan suatu pesan *tweet*. Analisis dibagi 2 bagian yaitu pemeriksaan kesamaa teks dan pemeriksaan kesamaan URL.



Gambar 3.7 *Flowchart* analisis kebiasaan pada profil pengirim *tweet*

3.6.4.1. Pemeriksaan Kesamaan Teks

Gambar 3.8 memodelkan alur proses pemeriksaan kesamaan teks. Sistem mengambil kumpulan data *mention* atau *timeline* untuk diperiksa kesamaan teksnya satu per satu. Data akan dipecah menjadi beberapa *shingle* dan setiap data akan diambil *username*-nya untuk mengambil profil pengirim berupa *tweet* terbaru yang dikirim.

Proses selanjutnya yaitu memecah *tweet-tweet* terbaru dari pengirim menjadi beberapa *shingle* lalu diperiksa kesamaannya dengan *tweet mention* atau *timeline* yang dikirim kepada pengguna aplikasi. Proses ini berlanjut hingga data *mention* atau *timeline* telah selesai diperiksa seluruhnya.

3.6.4.2. Pemeriksaan Kesamaan URL

Gambar 3.9 memodel alur dari proses kesamaan URL. Sistem mengambil kumpulan data *mention* atau *timeline* untuk diperiksa kesamaan URL nya satu per satu. Data akan diperiksa kepemilikan URL nya. Jika data memiliki URL makan akan diambil *username* pengirimnya untuk mengambil profil dari pengirim berupa kumpulan *tweet* terbaru yang dikirim.

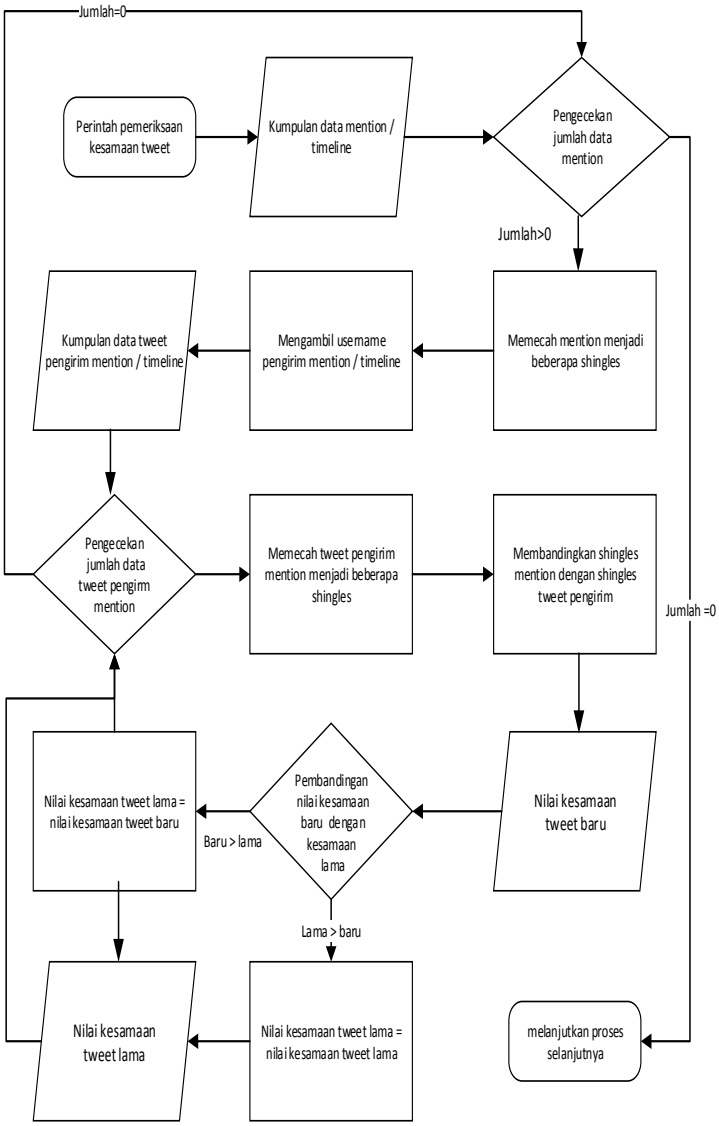
Proses selanjutnya yaitu mencari URL pada kumpulan *tweet* terbaru dari pengirim yang sudah diambil. Jika terdapat URL maka URL tersebut akan diperiksa kesamaannya dengan URL yang dikirim kepada pengguna aplikasi. Proses ini berlanjut hingga data *mention* atau *timeline* telah selesai diperiksa seluruhnya.

3.7. Rancangan Antarmuka

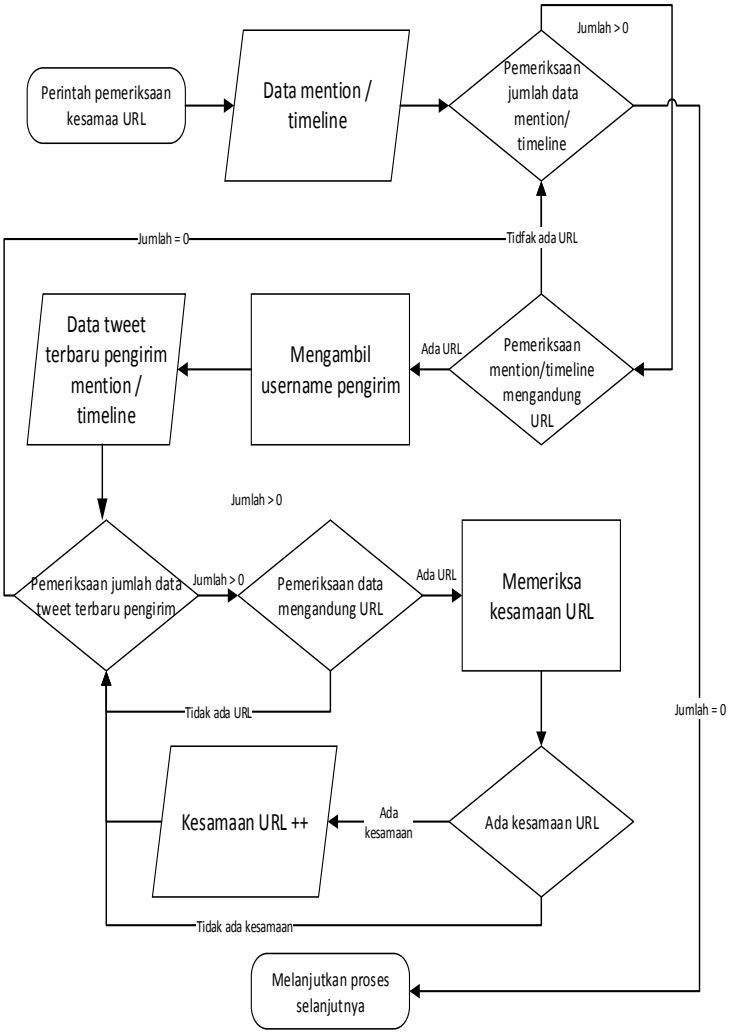
Interaksi antara pengguna dengan sistem dilakukan melalui sebuah perangkat bergerak berbasis *dekstop* dengan konektifitas jaringan untuk dapat terhubung dengah *server*. Segala proses permintaan data dilakukan oleh *client* dan *server* akan memberikan data-data yang diminta.

Dalam sistem kali ini, dibangun sebuah rancangan antarmuka pada aplikasi agar pengguna dapat dengan mudah mendapatkan informasi yang diminta. Perancangan antarmuka meliputi proses menerima masukan dari pengguna dan menampilkan data yang diterima dari *server* agar dapat dipahami oleh pengguna.

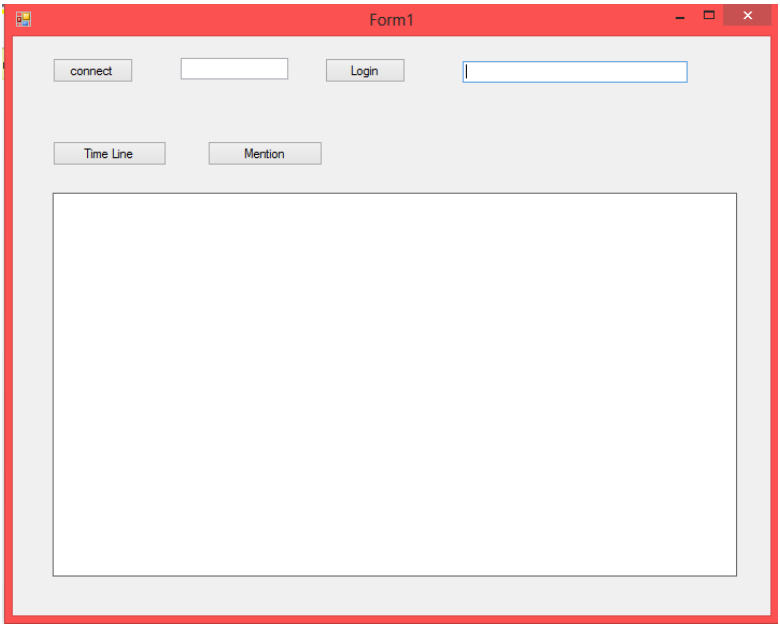
Gambar 3.10 merupakan rancangan antarmuka tampilan awal aplikasi. Terdapat tombol *connect* untuk melakukan koneksi dengan *server* dan tombol *login* untuk melakukan fungsi otentifikasi aplikasi. Pada tombol *timeline* digunakan untuk meminta data *timeline* dari *server* dan tombol *mention* untuk meminta data *mention.*



Gambar 3.8 *Flowchart* pemeriksaan kesamaan teks



Gambar 3.9 *Flowchart* pemeriksaan kesamaan URL



Gambar 3.10 Rancangan antar muka tampilan aplikasi

BAB IV
IMPLEMENTASI

Setelah melalui proses analisis dan perancangan perangkat lunak, dilakukan implementasi sistem. Bab ini membahas tentang proses implementasi pada sistem. Tahapan implementasi meliputi implementasi program dalam *pseudocode* dan antarmuka yang telah dibahas pada BAB III.

4.1. Implementasi Perangkat Keras

Dalam proses implementasi sistem ini, digunakan beberapa perangkat pendukung untuk memenuhi kebutuhan sistem yang terdiri dari perangkat keras dan perangkat lunak.

Dalam proses implementasi, digunakan beberapa perangkat keras yang digunakan untuk memenuhi kebutuhan sistem, yaitu:

- 1 buah laptop Intel® Core™ i7-3517U CPU @1.9GHZ 64-bit dengan 4GB RAM sebagai *server*.
- 1 buah laptop Intel® Core™ i3-3217U CPU @1.80 GHz 64-bit dengan 4GB RAM sebagai *client*

Beberapa perangkat lunak yang digunakan untuk kebutuhan implementasi sistem diantaranya adalah:

- Sistem operasi Windows 8 Single Language 64 bit
- Microsoft Visual Studio 2012 Ultimate untuk membagun aplikasi
- Microsoft Office Word 2013 untuk melakukan dokumentasi program dan menyelesaikan buku
- Weka 3.6.10 untuk melakukan proses uji coba pemodelan data pada sistem
- MySQL Server 5.5.27 sebagai basis data
- MySQL Connector Net 6.8.3 sebagai penghubung basis data dengan aplikasi pengembang Microsoft Visual Studio
- XAMPP sebagai *server* yang berjalan pada *localhost* dan digunakan untuk menjalankan basis data MySQL

- Microsoft Visio Professional 2013 untuk membuat rancangan antarmuka dan *flowchart* diagram alur
- StarUML 5.0.2.1570 untuk membuat *use case*
- PowerDesigner versi 15.0 untuk membuat *Conceptual Data Model* dan *Pyshical Data Model*

Selain perangkat lunak yang telah disebutkan di atas, aplikasi pada *client* juga menggunakan tiga pustaka pendukung untuk meningkatkan fungsionalitas yang sudah ada, yaitu:

- TweetSharp-*Unofficial* versi 2.3.1.2 sebagai penunjang pengembangan aplikasi dalam mengambil informasi pengguna pada *server* Twitter

4.2. Implementasi Perangkat Lunak

Implementasi pada perangkat lunak terbagi menjadi dua bagian, yaitu proses implementasi perangkat lunak pada aplikasi *server* dan implementasi perangkat lunak pada aplikasi *client*.

Berikut ini adalah penjelasan implementasi dari proses utama yang terjadi pada sistem. Proses-proses implementasi akan dijelaskan sesuai dengan urutan berjalannya sistem. Setiap bagian akan dijelaskan lebih lanjut pada masing-masing subbab.

4.2.1. Implementasi pada *Server*

Pada subbab-subbab berikut akan dijelaskan implementasi perangkat lunak pada aplikasi *server*. Penjelasan implementasi ditulis dalam bentuk *pseudocode*.

4.2.1.1. *Login*

Proses *login* merupakan proses awal yang akan dilakukan oleh sistem ketika pengguna akan menggunakannya. Secara garis besar, proses *login* pada *server* dibagi menjadi dua bagian dan berjalan seperti terlihat pada Gambar 4.1 dan Gambar 4.2. Kode sumber dari proses ini terdapat pada Lampiran A yang berisi proses *login* secara lengkap.

```
1  Menerima request login dari client
2  Mengirim request login ke server Twitter
3  If request = true
```

```
4       Mengirim request token kepada client
5   Else
6      Request eror
```

Gambar 4.1 *Pseudocode* proses *Login* pada *server* ke-1

```
1   Menerima kode otorisasi dari client
2   Mengirim kode otorisasi ke server Twitter
3   If request = true
4       Menerima akses token dan akses token secret
5       Menyimpan akses token dan akses token secret
6       Mengirim pesan kepada client bahwa login
7   berhasil
8   Else
9      Login eror
```

Gambar 4.2 *Pseudocode* proses *Login* pada *server* ke-2

4.2.1.2. Proses Pengambilan *Tweet*

Dalam proses pengambilan *tweet*, sistem membedakan proses menjadi 2 kondisi. Kondisi pertama ketika sistem medeteksi bahwa belum terdapat data pada basis data, sedangkan proses kedua ketika sistem mendeteksi sudah terdapat pada basis data seperti terlihat pada Gambar 4.3. Kode sumber dari proses ini terdapat pada Lampiran B yang berisi proses *login* secara lengkap.

```
1   Menerima perintah dari client
2   IF perintah == timeline
3       Deklarasi timeline = 50
4       Foreach (data in timeline)
5           IF(tanggal data > tanggal max
6   useridTimeline
7               Simpan data ke useridTimeline
8   Else
9       Deklarasi mention = 50
10      Foreach (data in mention)
11          IF(tagnggal data > tanggal max
12  useridMention
            Simpan data ke useridMention
```

Gambar 4.3 *Pseudocode* proses pengambilan *tweet*

4.2.1.3. Proses Analisis Kebisasaan Pesan *Tweet*

Proses analisis *tweet* merupakan proses untuk mendapatkan nilai atribut dari setiap *tweet*. Setiap *tweet* diperiksa dari sisi

kesamaan URL pada Gambar 4.5 dan kesamaan teks pada Gambar 4.4. Kode sumber dari proses ini terdapat pada Lampiran C yang berisi proses *login* secara lengkap.

```
1   Mengambil data dari basis data
2   Hitung jumlah data diterimayang diambil
3   WHILE( jumlah data diterima != 0)
4       Memecah data menjadi beberapa shingle
5       Mengambil username data
6       Mengambil  data  tweet  terbaru  berdasarkan
7   username yang didapat
8       WHILE (jumlah data tweet pengirim !=0)
9           Memecah data tweet menjadi beberapa shingle
10          Membandingkan  kesamaan  shingle  data
11     diterima dengan data    tweet pengirim
12          IF (kesamaan teks baru > lama)
13              Kesamaan teks lama = kesamaan teks baru
```

Gambar 4.4 *Pseudocode* proses kesamaan teks

```
1   Mengambil data dari basis data
2   Hitung jumlah data diterimayang diambil
3   WHILE( jumlah data diterima != 0)
4     IF(data diterima mengandung URL)
5        Mengambil username data
6        Mengambil  data  tweet  terbaru  berdasarkan
7        username yang didapat
8        WHILE (jumlah data tweet pengirim !=0)
9          IF(data tweet pengirim mengandung URL)
10          Membandingkan kesamaan URL
11          IF (kesamaan URL== ada)
12              Kesamaan URL++
```

Gambar 4.5 *Pseudocode* proses kesamaan URL

```
1  Mengambil data diterima dari basis data
2  WHILE(jumlah data ditermia!=0)
3      DO pemeriksaan kesamaan teks
4      DO pemeriksaan kesamaan URL
5  DO K-Means
```

Gambar 4.6 Proses pengelompokan data menggunakan K-*Means*

4.2.1.4. Proses Pengelompokan Data Menggunakan K-*Means*

Proses pengelompakan data menggunakan K-*Means* dilakukan ketika atribut kesamaan teks dan kesamaan URL telah

siap untuk diolah. Proses ini bertujuan untuk membedakan kelompok *tweet* yang terindikasikan *spam* dan tidak. Implementasi pada proses ini dapat dilihat pada Gambar 4.6. Kode sumber dari proses ini terdapat pada Lampiran D yang berisi proses *login* secara lengkap.

4.2.1.5. Proses Analisis Kebiasaan pada Profil Pengirim *Tweet*

Proses analis akun berdasarkan *tweet* dilakukan untuk mendapatkan nilai atribut akun dari pengirim *tweet*. Nilai atribut yang diambil yaitu persentase jumlah *tweet* mengandung interaksi, perbandingan *following* dengan *follower*, jumlah *tweet* per hari, dan rata-rata interval pengiriman *tweet*. Implementasi pada proses ini dapat dilihat pada Gambar 4.7. Kode sumber dari proses ini terdapat pada Lampiran E yang berisi proses *login* secara lengkap.

```
1  Mengambil data diterima dari basis data
2  WHILE(jumlah data diterima!=0)
3       DO penghitungan interaksi
4       DO perbandingan following dengan follower
5       DO jumlah tweet per hari
6       DO rata-rata interval pengiriman tweet
```

Gambar 4.7 *Psudocode* proses analisis akun berdasarkan *tweet*

4.2.1.6. Proses Pengambilan Keputusan Menggunakan *Decision Tree*

Proses pengambilan keputusan menggunakan *decision tree* dilakukan untuk mendapatkan status dari *tweet* merupakan pesan *spam* atau tidak. Implementasi pada proses ini dapat dilihat pada Gambar 4.8. Kode sumber dari proses ini terdapat pada Lampiran F yang berisi proses *login* secara lengkap.

4.2.2. Implementasi pada *Client*

Pada subbab-subbab berikut akan dijelaskan implementasi perangkat lunak pada aplikasi *client*. Penjelasan implementasi ditulis dalam bentuk *pseudocode*.

4.2.2.1. *Login*

Proses *login* pada aplikasi *client* dimulai dengan menerima masukan dari pengguna. Implementasi pada proses ini dapat dilihat pada Gambar 4.9.

4.2.2.2. Melakukan Proses Meminta Data *Timeline* atau *Mention*

Proses ini digunakan oleh pengguna untuk meminta data *timeline* kepada *server*.

```
1   Mengambil nilai kluster terbesar
2   Mengambil data diterima
3   WHILE (data diterima !=0)
4     IF( kluster != kluster terbesear)
5       Status = SPAM
6     ELSE IF (aveTime > 14.6)
7           Status = BUKAN SPAM
8         ELSE
9           IF (aveTIime <=14.6)
10            IF(foll <= 1)
11              Status = BUKAN SPAM
12            ELSE IF (foll > 1)
13                    IF (tweet > 34.41)
14                      Status = SPAM
15                    ELSE IF (tweet <= 34.41)
16                          IF( reply <= 73.33)
17                            Status = BUKAN SPAM
18                           ELSE status = SPAM
```

Gambar 4.8 *Pseudocode* pengambilan keputusan menggunakan *decision tree*

```
1  Mengirim request login pada server
2  Menerima request token dari sever
3  Membuka default browser
4  GET kode otorisasi pengguna
5  Mengirim kode otorisasi pada server
6  Menerima pesan dari server login berhasil
```

Gambar 4.9 *Pseudocode* proses *login* pada *client*

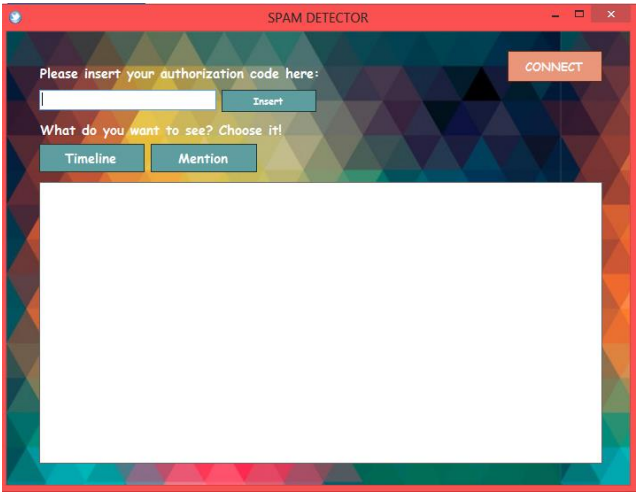
4.3. Implementasi Antarmuka Perangkat Lunak

Implementasi antarmuka perangkat lunak diimplementasikan pada aplikasi *client* dimana pengguna melakukan interaksi dengan sistem sesuai dengan rancangan yang telah dibahas pada BAB III.

```
1  Menekan tombol pada layar
2  IF perintah == timeline
3      Mengirim request timeline pada server
4  Else perintah == mention
5      Mengirim request mention pada server
6  Else perintah tidak ada
7  Menerima balasan dari server
8  Menampilkan data pada listview
```

Gambar 4.10 *Pseudocode* proses melakukan pencarian tempat

Pada tampilan ini, pengguna diminta untuk menekan tombol *connect* untuk meminta halaman *web login* untuk melakukan *login* pada Twitter. Terdapat tombol *enter* digunakan untuk memasukkan kode otorisasi yang didapat setelah *client* login. Tombol *timeline* digunakan untuk meminta data *timeline* untuk meminta data *timeline* yang sudah diolah oleh *server* dan begitu pula untuk tombol *mention.* Tampilan antarmuka dapat dilihat pada Gambar 4.11.



Gambar 4.11 Implementasi tampilan aplikasi

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dibahas mengenai uji coba dari segi fungsionalitas dan performa dari aplikasi. Uji coba fungsionalitas dan performa akan dibagi ke dalam beberapa skenario uji coba.

5.1. Lingkungan Uji Coba

Dalam proses uji coba, digunakan beberapa perangkat keras dan perangkat lunak sebagai penunjang kebutuhan yang akan dijelaskan pada subbab berikut.

5.1.1. Perangkat Keras

Kebutuhan perangkat keras yang digunakan dalam uji coba yaitu:

- 1 buah laptop Intel® Core™ i7-3517U CPU @1.9GHZ 64-bit dengan 4GB RAM sebagai *server*.
- 1 buah laptop Intel® Core™ i3-3217U CPU @1.80 GHz 64-bit dengan 4GB RAM sebagai *client*

5.1.2. Perangkat Lunak

Kebutuhan perangkat lunak yang digunakan dalam uji coba yaitu:

- Sistem operasi Windows 8 Single Language 64 bit
- Microsoft Visual Studio 2012 Ultimate untuk membagun aplikasi
- Weka 3.6.10 untuk melakukan proses uji coba pemodelan data pada sistem
- MySQL Server 5.5.27 sebagai basis data
- MySQL Connector Net 6.8.3 sebagai penghubung basis data dengan aplikasi pengembang Microsoft Visual Studio
- XAMPP sebagai *server* yang berjalan pada *localhost* dan digunakan untuk menjalankan basis data MySQL

5.2. Uji Coba Fungsionalitas

Uji coba fungsionalitas merupakan sebuah pengujian terhadap jalannya fungsi-fungsi utama yang ada pada aplikasi. Uji coba fungsionalitas meliputi semua *use case* yang telah dijelaskan pada BAB III beserta fungsionalitas pada *server*, yaitu:

1. *Login*
2. Memasukkan kode otorisasi
3. Meminta data *timeline*
4. Meminta data *mention*
5. Proses analisis kebiasaan pesan *tweet*
6. Proses analisis kebiasaan pada profil pengirim *tweet*

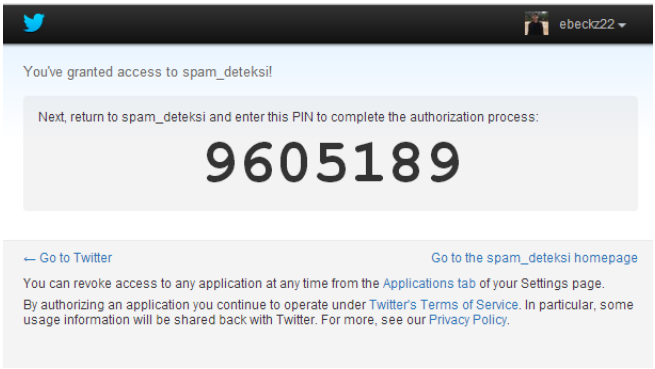
5.2.1. *Login*

Proses pertama yang dilakukan ketika menjalankan aplikasi *client* adalah proses *login* atau validasi pengguna. Tabel 5.1 menunjukkan prosedur uji coba yang dilakukan pada proses *login*.

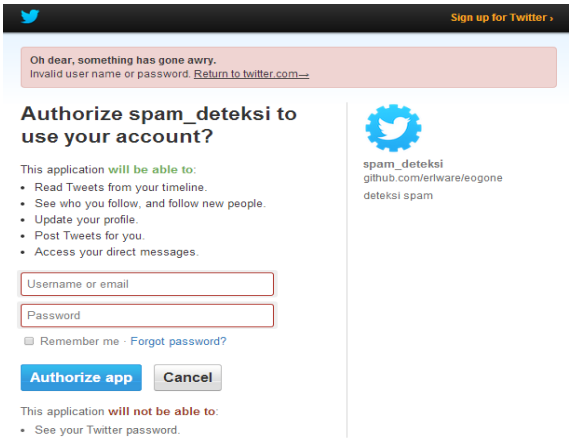
Tabel 5.1 Uji Coba Proses *Login*

ID	**UJ-01**
Referensi *Use Case*	**UC-001**
Nama	Uji Coba Proses *Login* Twitter
Tujuan Uji Coba	Menguji proses pengiriman *login* ke Twitter dan proses mendapatkan kode otorisasi
Kondisi Awal	Aplikasi berjalan
Skenario 1	**Pengguna memasukkan *username* dan *password* akun Twitter yang sesuai dengan benar**
Masukan	*Username* dan *password* yang sesuai
Keluaran yang diharapkan	Kode otorisasi dari Twitter untuk dimasukkan ke dalam sistem oleh pengguna
Hasil Uji Coba	BERHASIL
ID	**UJ-01**
Skenario 2	**Pengguna memasukkan *username* dan *password* yang tidak sesuai**

Masukan	*Username* dan *password* yang tidak sesuai
Keluaran yang diharapkan	Halaman Twitter yang menunjukkan kesalahan memasukkan *username* dan *password*
Hasil Uji Coba	BERHASIL



Gambar 5.1 Tampilan *login* Twitter jika berhasi *login*



Gambar 5.2 Tampilan *login* Twitter jika gagal *login*

Proses ini bertujuan untuk memberikan ijin aplikasi mengakses *tweet*, *mention*, dan *direct message* pengguna pada *server* Twitter. Proses pengiriman kode otorisasi oleh *server*

Twitter. Pada saat pengguna memasukkan *username* dan *password* Twitter dengan benar, maka akan ditampilkan menu seperti yang tampak pada Gambar 5.1. Jika proses *login* gagal, maka aplikasi akan menampilkan pesan kesalahan yang tampak pada Gambar 5.2.

5.2.2. Memasukkan Kode Otorisasi

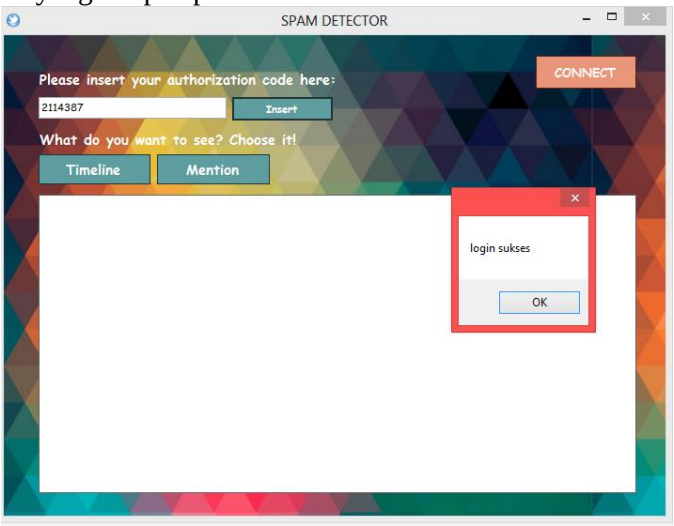
Proses memasukkan kode otorisasi bertujuan untuk memberikan akses kepada *server* aplikasi untuk mengambil data pengguna pada *server* Twitter. Tabel 5.2 menunjukkan prosedur uji coba yang dilakukan pada proses memasukkan kode otorisasi.

Tabel 5.2 Uji coba memasukkan kode otorisasi

ID	**UJ-02**
Referensi *Use Case*	**UC-002**
Nama	Uji coba memasukkan kode otorisasi
Tujuan Uji Coba	Menguji proses validasi kode otorisasi oleh sistem
Kondisi Awal	Aplikasi berjalan
Skenario 1	**Pengguna memasukkan kode otorisasi yang didapat setelah pengguna *login* Twitter**
Masukan	Kode otorisasi
Keluaran yang diharapkan	Sistem menampilkan pesan tanda kode diterima dan mengambil ID pengguna Twitter untuk melakukan proses selanjutnya
Hasil Uji Coba	BERHASIL
Skenario 2	**Pengguna memasukkan kode otorisasi yang didapat setelah pengguna *login* Twitter**
ID	**UJ-02**
Masukan	Kode otorisasi

Keluaran yang diharapkan	Aplikasi menampilkan pesan terjadi kesalahan
Hasil Uji Coba	BERHASIL

Jika proses otorisasi berhasil makan sistem akan menampilkan pesan kode diterima dan data pengguna *login* seperti yang terlihat Gambar 5.3. Jika pengguna salah memasukkan kode otorisasi, maka aplikasi akan menampilkan pesan kesalahan seperti yang tampak pada Gambar 5.4.



Gambar 5.3 Kode otorisasi diterima

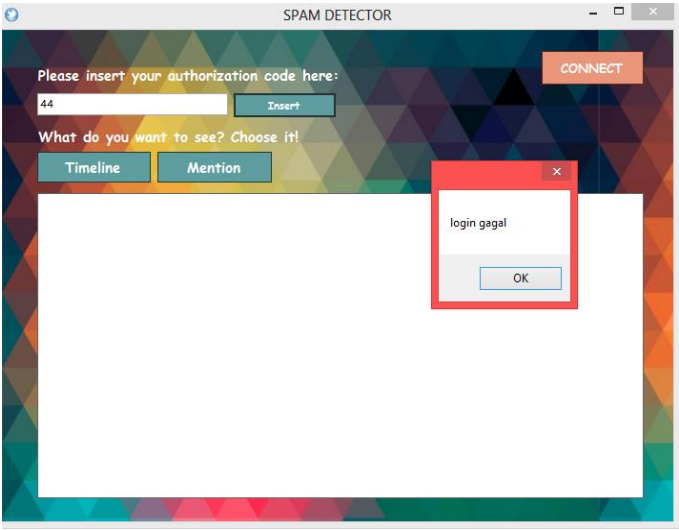
5.2.3. Meminta Data *Timeline*

Pada uji coba ini akan dilakukan proses meminta data *timeline* pada *server*. Tabel 5.3 menunjukkan prosedur uji coba yang dilakukan pada proses minta data *timeline*. Pada Gambar 5.5 ditunjukkan tampilan antarmuka data *timeline* pada *client.*

Tabel 5.3 Uji coba meminta data *timeline*

ID	**UJ-03**
Referensi *Use Case*	**UC-003**

ID	**UJ-03**
Nama	Uji coba meminta data *timeline*
Tujuan Uji Coba	Menguji proses permintaan data *timeline* pada *timeline*
Kondisi Awal	Aplikasi berjalan, pengguna sudah *login*
Skenario 1	**Pengguna menekan tombol *timeline***
Masukan	Perintah ambil data *timeline*
Keluaran yang diharapkan	Aplikasi akan menampilkan data *timeline* terbaru yang telah diolah oleh *server*
Hasil Uji Coba	BERHASIL



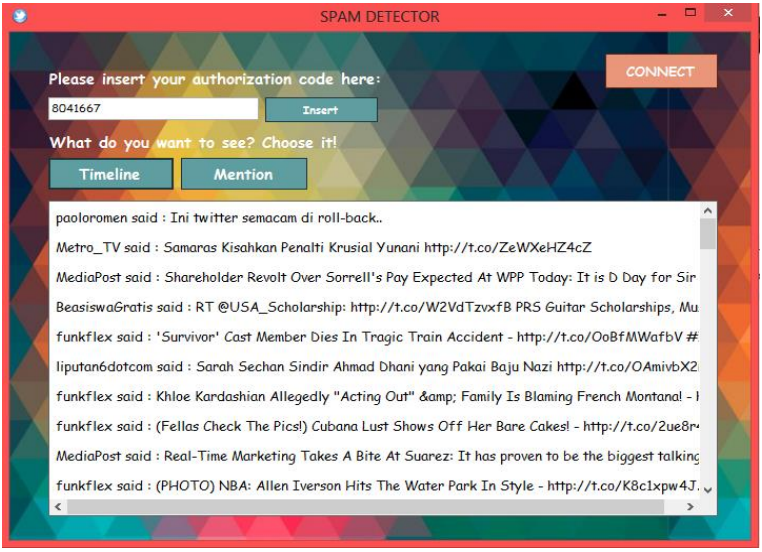
Gambar 5.4 Kode otorisasi salah

5.2.4. Meminta Data *Mention*

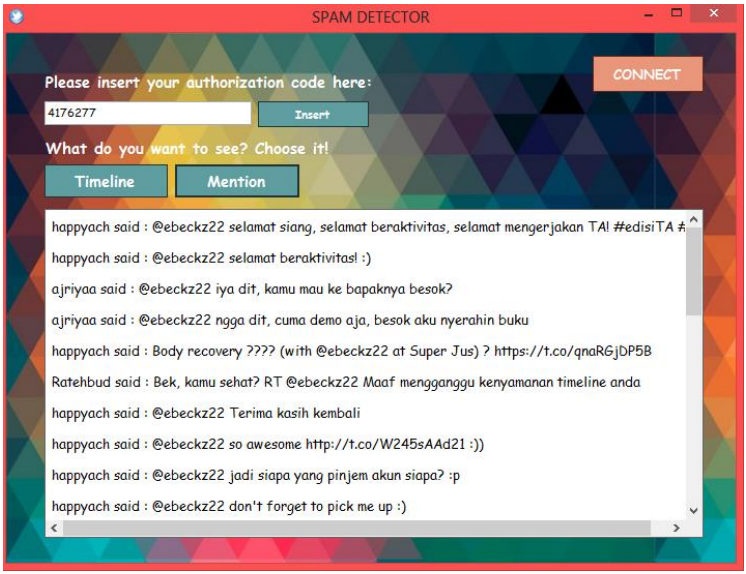
Pada uji coba ini akan dilakukan proses meminta data *mention* pada *server*. Tabel 5.4 menunjukkan prosedur uji coba yang dilakukan pada proses minta data *mention*. Pada Gambar 5.6 ditunjukkan tampilan antarmuka data *mention* pada *client*.

Tabel 5.4 Uji coba meminta data *mention*

ID	**UJ-04**
Referensi *Use Case*	**UC-004**
Nama	Uji coba meminta data *mention*
Tujuan Uji Coba	Menguji proses permintaan data *mention* pada *server*
Kondisi Awal	Aplikasi berjalan, pengguna sudah *login*
Skenario 1	**Pengguna menekan tombol *mention***
Masukan	Perintah ambil data *mention*
Keluaran yang diharapkan	Aplikasi dapat menampilkan data *mention* yang telah diolah *server*
Hasil Uji Coba	BERHASIL



Gambar 5.5 Data *timeline* berhasil ditampilkan pada layar aplikasi



Gambar 5.6 Data *mention* berhasil ditampilkan pada layar aplikasi

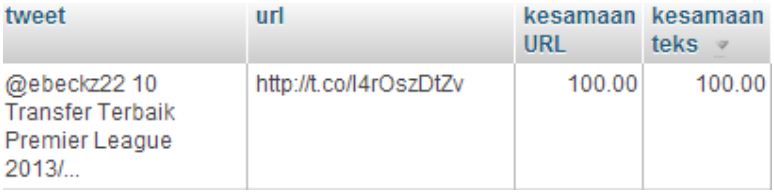
5.2.5. Proses Analisis Kebiasaan Pesan *Tweet*

Pada uji coba ini akan dilakukan proses analisis pesan *tweet.* Pada Tabel 5.5 menunjukkan prosedur uji coba yang dilakukan pada proses analisi pesan *tweet*. Proses analisis pesan *tweet* menggunakan dua skenario yaitu analisis pemeriksaan kesamaan teks dan pemeriksaan kemasaan URL. Pada Gambar 5.7 ditunjukkan server mendapatkan hasil kemiripan URL dan teks.

Tabel 5.5 uji coba proses analisis pesan *tweet*

ID	**UJ-06**
Nama	Uji coba proses analisis kebiasaan pesan *tweet*
Tujuan Uji Coba	Menguji proses analisis kebiasaan pesan *tweet*
Kondisi Awal	Aplikasi berjalan, pengguna sudah *login*
Skenario 1	**Sistem melakukan analisis pemeriksaan kesamaan teks**
Masukan	Perintah pemeriksaan kesamaan teks

ID	**UJ-06**
Keluaran yang diharapkan	*Server* mendapatkan hasil pemeriksaan kesamaan teks
Hasil Uji Coba	BERHASIL
Skenario 2	**Sistem melakukan analisis pemeriksaan kesamaan URL**
Masukan	Perintah pemeriksaan kesamaan URL
Keluarang yang diharapkan	*Server* mendapatkan hasil pemeriksaa kesamaan URL
Hasil Uji Coba	BERHASIL



tweet	url	kesamaan URL	kesamaan teks
@ebeckz22 10 Transfer Terbaik Premier League 2013/...	http://t.co/l4rOszDtZv	100.00	100.00

Gambar 5.7 *Server* mendapatkan hasil kesamaan URL dan teks

5.2.6. Proses Analisis Kebiasaan dan Profil Pengirim *Tweet*

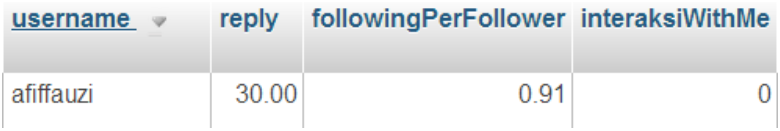
Pada uji coba ini akan dilakukan proses analisis kebiasaan dan profil pengirim *tweet*. Pada Tabel 5.6 menunjukkan prosedur uji coba yang dilakukan pada proses analisis kebiasaan dan profil pengirim *tweet*. Proses analisis pesan *tweet* menggunakan lima skenario yaitu persentase jumlah *tweet* berisi interaksi atau *reply*, perbandingan *following* dengan *follower*, jumlah *tweet* per hari, interaksi terhadap pemilik akun, persentase rata-rata pengunaan URL pada pengiriman *tweet*, dan rata-rata interval pengiriman *tweet*. Pada Gambar 5.8 dan Gambar 5.9 ditunjukkan bahwa server mendapatkan hasil uji coba berdasarkan skenario yang diujikan.

Tabel 5.6 uji coba proses analisis kebiasaan dan profil pengirim tweet

ID	**UJ-07**
Nama	Uji coba proses analisis kebiasaan dan profil pengirim *tweet*

ID	**UJ-07**
Tujuan Uji Coba	Menguji proses analisis kebiasaan dan profil pengirim *tweet*
Kondisi Awal	Aplikasi berjalan, pengguna sudah *login*
Skenario 1	**Sistem melakukan penghitungan persentase pengiriman *tweet* mengandung interaksi atau *reply***
Masukan	Perintah penghitungan persentase pengiriman *tweet* mengandung interaksi
Keluaran yang diharapkan	*Server* mendapatkan hasil penghitungan persentase pengiriman *tweet* mengandung interaksi
Hasil Uji Coba	BERHASIL
Skenario 2	**Sistem melakukan penghitungan perbandingan *following* dengan *follower***
Masukan	Perintah penghitungan perbandingan *following* dengan *follower*
Keluarang yang diharapkan	*Server* mendapatkan hasil penghitungan perbandingan *following* dengan *follower*
Hasil Uji Coba	BERHASIL
Skenario 3	**Sistem melakukan penghitungan jumlah interaksi pengguna dengan pengirim *tweet***
Masukan	Perintah penghitungan jumlah interaksi pengguna dengan pengirim *tweet*
Keluaran yang diharapkan	*Server* mendapatkan hasil penghitungan jumlah interaksi pengguna dengan pengirim *tweet*
Hasil Uji Coba	BERHASIL
Skenario 4	**Sistem melakukan penghitungan pemeriksasan jumlah pengiriman *tweet* per hari**
Masukan	Perintah penghitungan pemeriksasan jumlah pengiriman *tweet* per hari
Keluaran yang diharapkan	*Server* mendapatkan hasil pemeriksaan kesamaan teks

ID	**UJ-07**
Hasil Uji Coba	BERHASIL
Skenario 5	**Sistem melakukan penghitungan pemeriksaan persentase penggunaan URL pada pengiriman *tweet***
Masukan	Perintah penghitungan pemeriksaan persentase penggunaan URL pada pengiriman *tweet*
Keluarang yang diharapkan	*Server* mendapatkan hasil penghitungan pemeriksaan persentase penggunaan URL pada pengiriman *tweet*
Hasil Uji Coba	BERHASIL
Skenario 6	**Sistem melakukan penghitungan pemeriksaan rata-rata interval pengiriman *tweet***
Masukan	Perintah penghitungan pemeriksaan rata-rata interval pengiriman *tweet*
Keluarang yang diharapkan	*Server* mendapatkan hasil penghitungan pemeriksaan rata-rata interval pengiriman *tweet*
Hasil Uji Coba	BERHASIL



username	reply	followingPerFollower	interaksiWithMe
afiffauzi	30.00	0.91	0

Gambar 5.8 *Server* mendapatkan hasil ujicoba skenario 1-3

username	tweetPerHari	averageURL	averageTimeInterval
afiffauzi	5.26	10.00	665.46

Gambar 5.9 *Server* mendapatkan hasil uji coba skenario 4-6

5.3. Uji Coba Kasus

Pada pengujian kasus, akan dijelaskan beberapa kasus yang mungkin akan dihadapi sistem. Uji coba yang akan dilakukan akan dijelaskan pada subbab berikut.

5.3.1. Uji Coba Jumlah Kata pada Setiap *Shingle*

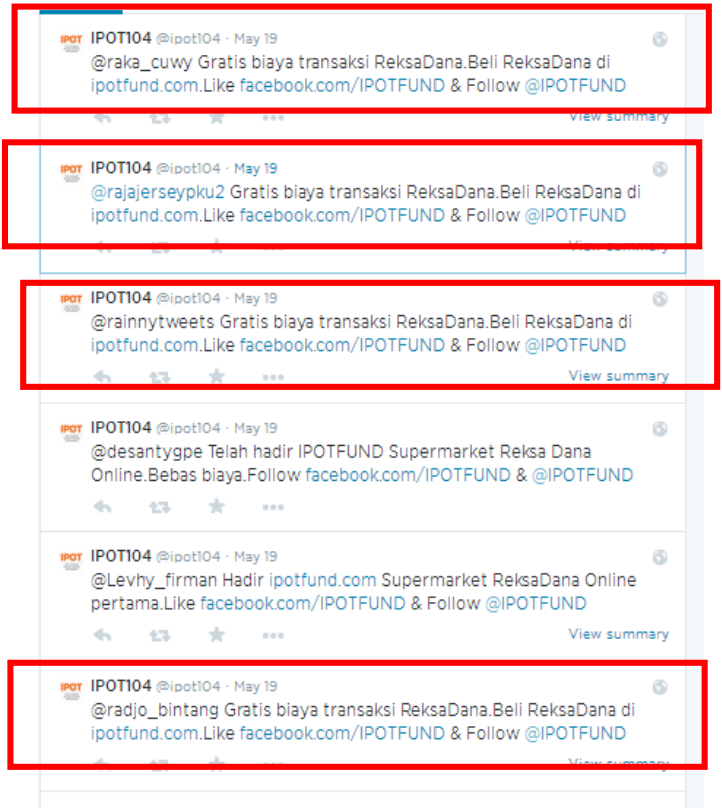
Pada pengujian ini dilakukan uji coba jumlah kata pada setiap *shingle*. Tujuan dari pengujian ini yaitu untuk mencari persentase tertingi dalam akurasi pemeriksaan kesamaan teks.

Pada Gambar 5.10 ditampilkan kumpulan *tweet* pada akun *spam*. Dapat dilihat pada kotak berwarna merah bahwa terdapat *tweet* yang identik satu dengan lainnya dan jeda waktu pengiriman *tweet* berjarak sangat dekat. Sehingga dapat disimpulkan bahawa akun *spam* dalam mengirim pesannya menggunakan pesan yang identik dalam jeda waktu yang berdekatan karena mayoritas akun *spam* menggunakan aplikasi untuk mengirim *tweet*.

Untuk mendapatkan akurasi yang maksimal dalam pemeriksaan kesamaan teks maka digunakan metode W-*Shingling resemblance* seperti yang telah dijelaskan pada Bab 2. Uji coba kali ini bertujuan untuk mendapat nilai *n* untuk jumlah kata pada setiap *shingle*.

Uji coba dilakukan pada mengambil salah satu *tweet* pada *mention* pada Gambar 5.11. Maka untuk memeriksa kesamaan teks diambil profil dari pengirim *tweet* sebagai data pembanding seperti pada Gambar 5.12.

Pada Tabel 5.7 dapat dilihat bahwa persentase kemiripan dari *tweet* yang terdapat pada Gambar 5.11 ketika dibandingkan dengan masing-masing *tweet* yang terdapat pada Gambar 5.12. Berdasarkan Tabel 5.7 dapat dilihat, persentase kemiripan antar *tweet* yang dibandingkan semakin berkurang seiring bertambahnya nilai *n*, di mana ketika persentase menurun hal tersebut menunjukkan tingkat akurasi yang lebih tinggi. Sesuai hasil uji coba dapat dilihat bahwa *tweet mention* yang diperiksa kesamaannya identik dengan *tweet* ke-2 pada data *tweet* terbaru profil pengirim dengan persetase 66,67 % pada nilai $n = 4$.



Gambar 5.10 Contoh kumpulan *tweet* pada akun *spam*

Gambar 5.11 Salah satu *tweet* yang muncul pada *timeline*

Gambar 5.12 Kumpulan *tweet* terbaru pada *profile* pengirim

Tabel 5.7 Tabel hasil uji coba jumlah n-kata pada setiap *shingle*

Tweet ke-	Persentase kemiripan jumlah *n* (%)			
	1	**2**	**3**	**4**
1	45.45	7.69	0	0
2	77.77	75	71.43	66.67
3	25	0	0	0

Tweet ke-	Persentase kemiripan jumlah *n* (%)			
	1	**2**	**3**	**4**
4	45.45	7.69	0	0
5	25	0	0	0
6	45.45	7.69	0	0

5.3.2. Uji Coba Pengaruh Nilai *k* pada *Cluster* K-*Means*

Pada pengujian ini dilakukan uji coba nilai *k* pada proses pengelompokan *tweet.* Nilai-*k* merupakan jumlah *cluster* yang akan digunakan pada proses pengelompokan *tweet.* Dalam pengelompokan *tweet* menggunakan dua atribut sebagai dasar yaitu persentase kesamaan teks dan persentase kesamaan URL.

Pengujian dilakukan untuk melihat pengaruh nilai-*k* pada tingkat akurasi dari *cluster* yang dihasilkan oleh sistem. Dari seluruh kluster yang terbentuk, salah satu *cluster* harus memiliki persentase *spam* sebesar 0%, untuk dapat merepresentasikan *tweet* yang tidak terindikasikan *spam* atau pesan normal. Seperti pada Gambar 5.13 dimana *tweet spam* mengandung nilai atribut kesamaan nilai URL dan teks. Pada Tabel 5.8 dijelaskan bahwa dengan menggunakan nilai-*k* = 3 didapatkan *cluster* yang mempunyai persentase nilai *spam* sebesar 0%.

Tabel 5.8 Tabel untuk menentukan jumlah nilai–*k* pada metode K-*Means*

Jumlah data	Persentase *spam* pada jumlah kluster = 3 (%)		
	1	**2**	**3**
48	100	0	Tidak ada data
100	100	0	Tidak ada data
703	100	100	0
990	100	100	0
1094	100	100	0

5.3.3. Uji Coba Pemodelan Pengambilan Keputusan Menggunakan Metode Klasifikasi *Decision Tree*

Pada pengujian ini dilakukan uji coba pengambilan keputusan menggunakan klasifikasi *decision tree*. Pengambilan keputusan ini dilakukan supaya dapat diambil keputusan terhadap pesan *tweet* merupakan pesan *spam* atau pesan sah.

tweet	datetime	url	averageurlSama	nilaiText
@ebeckz22 Galaxy 11, berkumpul sejumlah nama pesep...	2014-05-14 14:32:56	http://t.co/cvAoLtFQfZ	100	100
@ebeckz22 Naruto chapter 676 terbaru, lebih menega...	2014-05-14 15:02:54	http://t.co/kJ2krKUKqE	100	100
@ebeckz22 10 Transfer Terbaik Premier League 2013/...	2014-05-14 14:34:22	http://t.co/I4rOszDtZv	100	100
@ebeckz22 maaf ya sudah mention :D #pmqs #sorrySpa...	2014-06-11 19:40:20		0.00	100
@ebeckz22 selamat malam semuanya :D #PillIsNPotion...	2014-06-10 20:05:55		0.00	100
@ebeckz22 selamat malam semuanya :D #PillIsNPotion...	2014-06-10 20:05:45		0.00	100

Gambar 5.13 Contoh data *tweet* yang dikategorikan sebagai *spam*

Tabel 5.9 Uji coba penentuan model

Model ke- n	Jumlah Data	Akurasi (%)
1	48	93.75
2	100	98.00
3	703	98.58
4	990	99.50
5	1094	99.36

Setelah dilakukan serangkaian uji coba, didapatkan lima buah model klasifikasi *decision tree* untuk pengambilan keputusan. Dapat dilihat pada Tabel 5.9 merupakan akurasi data yang akan digunakan sebagai uji coba pemodelan pada *client.* Uji coba penggunaan model dilakukan terhadap 7 *client* berbeda.

Berdasarkan hasil uji coba didapatkan rata-rata hasil pencapaian akurasi pada setiap model. Hasil uji coba dapat dilihat pada Tabel 5.10, di mana akurasi tertinggi dimiliki oleh model ke-3 seperti dapat dilihat pada Gambar 5.14. Pada Gambar 5.15 ditampilkan model ke-3 sebagai model dengan akurasi tertinggi dengan nilai persentase sebesar 99,40 %. Detail uji coba dan model dapat dilihat pada Lampiran G.

Tabel 5.10 Uji coba rata-rata akurasi pengunaan model pada pengguna

Model ke-n	Akurasi (%)
1	84.03
2	97.08
3	99.4
4	97.86
5	98.48

5.4. Uji Coba Waktu Respon

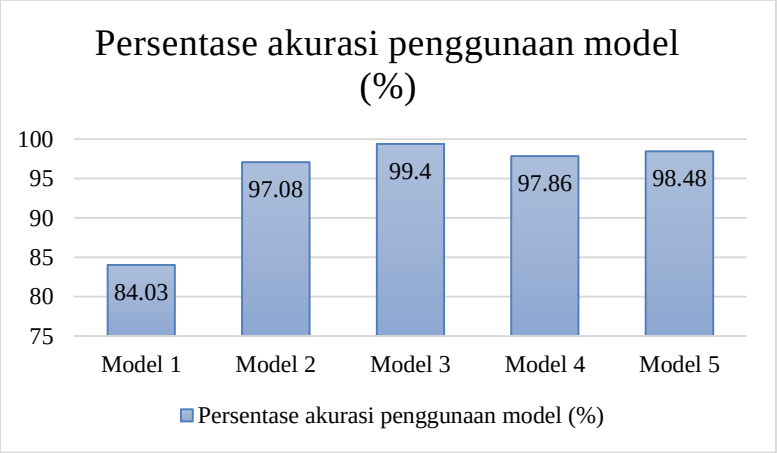
Uji waktu respon dilakukan untuk mengetahui waktu respon yang dibutuhkan dari *client* ke *server* dan kembali lagi ke *client* dalam melakukan satu alur fungsionalitas secara lengkap.

Tabel 5.11 uji coba performa pada 2 *client*

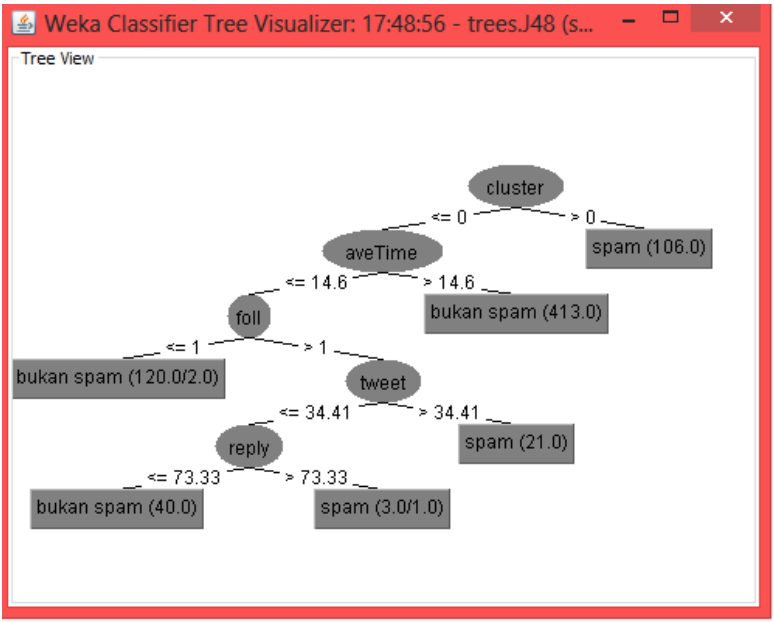
client	*Request* ke *Server* (detik)	GET *login* sukses (detik)	Olah Data (detik)	Proses *clustering* K-*Means* (detik)	Proses klasifikasi *decision tree* (detik)	*Client get result* (detik)
1	0.38	2.4	107.71	0.148	0.065	127.7
2	2.4	1.67	87.27	0.032	0.052	113.4
Rata-rata	1.39	2.035	97.49	0.18	0.0585	120.6

Dalam uji coba kali ini dilakukan serangkaian alur penggunaan sistem oleh 2 *client* sekaligus. Penggunaan *client* berjumlah 2 karena keterbatasan dari *library* yang digunakan. *Library* yang digunakan tidak dapat mengambil data terlalu banyak dalam rentang waktu yang cepat karena akan menyebabkan *limit* pada pengambilan data di *server* Twitter.

Parameter yang digunakan dalam pengujian meliputi waktu respon permintaan otorisasi *client* menggunakan aplikasi di *server* Twitter (*request* ke *server*), pengiriman kode otorisasi untuk dapat menggunakan aplikasi ke *server* (*get login* sukses), pengolahaan data *training*, melakukan proses *clustering* menggunakan metode K-*Means*, dan proses pengambilan keputusan. Hasil uji coba dapat dilihat pada Tabel 5.11. Detail uji coba dapat dilihat pada Lampiran G.



Gambar 5.14 Persentase akurasi penggunaan model



Gambar 5.15 Model klasifikasi *decision tree* ke-3

BAB VI
PENUTUP

Pada bab terakhir ini akan dibahas mengenai kesimpulan yang diperoleh selama pengerjaan Tugas Akhir hingga selesai. Pada bab ini juga akan menjawab pertanyaan yang dijabarkan pada BAB I. Untuk memperbaiki semua kelebihan dan kekurangan dari sistem, akan dijelaskan pada subbab saran.

6.1. Kesimpulan

Berikut adalah beberapa kesimpulan yang diperoleh dari proses pengerjaan Tugas Akhir ini:

1. Sistem dapat menyaring data *tweet* berupa pesan *spam* dan tidak menampilkannya pada *client* sehingga pesan *tweet* yang dikirimkan pada pengguna adalah pesan normal atau tidak terindikasi sebagai pesan *spam.*
2. Penggunaan W-*Shingling resemblance* dapat menjadi pilihan untuk kebutuhan pemeriksaan kesamaan *tweet*. Didapatkan akurasi yang cukup tinggi pada pemeriksaan kesamaan *tweet* dengan menggunakan jumlah kata pada setiap *shingle* berjumlah 4 kata.
3. Pemilihan nilai *k* pada metode pengelompokan *tweet* menggunakan K-*Means* mempengaruhi pengelompokan data. Berdasarkan hasil uji coba didapat nilai *k*=3 dimana pada salah satu kluster hanya berisi data *tweet* normal.
4. Penentuan model menggunakan metode klasifikasi *decision tree* mempengaruhi sistem dalam pengambilan keputusan. Berdasarkan hasil uji coba, didapatkan model *decision tree* dengan tingkat akurasi paling tinggi dengan persetase 99,4 %.

6.2. Saran

Adapun saran dari penulis yang dapat diberikan dari kekurangan sistem yang ada, maupun pengembangan lebih lanjut yang dapat dilakukan yaitu:

1. Pengembangan sumber daya sistem baik dari sisi perangkat lunak maupun perangkat keras akan sangat dibutuhkan ketika jumlah pengguna dan kebutuhan sistem semakin meningkat.
2. Penggunaan API Twitter *premium* sebagai *library* sistem dapat meningkatkan kinerja sistem sehingga tidak ada lagi batasan atau *limit* dalam pengambilan data pada Twitter. Dengan penggunaan API Twitter *premium*, jumlah akses pengguna juga dapat ditingkatkan.
3. Kebutuhan koneksi internet yang stabil merupakan permasalahan yang terjadi dalam sistem. Dengan koneksi internet yang stabil pengolahan data dapat berjalan lebih cepat.

DAFTAR PUSTAKA

[1] Twitter, "Twitter Definiton," [Online]. Available: https://discover.twitter.com. [Diakses 15 Februari 2014].

[2] Twitter, "Rules and best practices in Twitter," [Online]. Available: https://support.twitter.com/groups/56-policies-violations/topics/237-guidelines/articles/69214-rules-and-best-practices. [Diakses 8 Juni 2014].

[3] Techscape, "Spam," [Online]. Available: http://techscape.com/?p=61. [Diakses 15 Februari 2014].

[4] Twitter, "Reporting spam on Twitter," [Online]. Available: https://support.twitter.com/entries/64986. [Diakses 8 Juni 2014].

[5] A. Chaturvedi, K. Foods dan P. E. Green, "K-Modes Clustering," *Journal of Classification,* vol. 18, pp. 35-55, 2001.

[6] M. Inaba, N. Katoh dan H. Imai, "Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering," dalam *Proceedings of 10th ACM Symposium on Computational Geometry*, 1994.

[7] J. Quinlan, "Induction of Decision Trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.

[8] A. Z. Broder, S. C. Glassman dan M. S. Manasse, "Syntactic Clustering of the Web," *Proceedings of WWW6 and Computer Networks* , vol. 29, pp. 8-13, 1997.

LAMPIRAN

Bagian ini merupakan lampiran sebagai dokumen pelengkap dari buku Tugas Akhir dimana akan diberikan potongan kode sumber dari fungsi-fungsi yang digunakan untuk membangun sistem dan table-tabel yang berisikan hasil pengujian.

A. *Login*

1. *Login pada Server*

Pada proses *login*, *server* menerima permintaan *login* dari *client*. Proses *login* pada *server* dibagi menjadi 2 bagian. Proses tersebut diantaranya mendapatkan *request token* dari Twitter dan memasukkan kode otorisasi untuk mendapatkan akses *token* dan akses *token secrete*. Kode dapat lihat pada Kode Sumber 1.

```
public string auth1(string tanda,string verifier)
{
        string url = "";
    if (tanda == "1")
    {
        requestToken = service.GetRequestToken();
        Uri uri =
service.GetAuthorizationUri(requestToken);
        url = uri.ToString();
        return url;
    }
    else
    {
         try
       {
         OAuthAccessToken          akses          =
         service.GetAccessToken(requestToken,
         verifier);
         url = "login sukses##" + akses.Token + "##"
         + akses.TokenSecret;
       }
       catch
       {
```

```
          url= "login gagal##";
          Console.WriteLine("login gagal");
        }
    return url;
          }
}
```

Kode Sumber 1 Proses *login* pada *server*

2. *Login* pada *client*

Proses *login* pada *client* dimulai ketika *client* menekan tombol *connect* pada antarmuka. Ketika tombol *connect* ditekan, *client* akan terhubung dengan *server*. Kemudian *server* akan mengirim alamat *request token* untuk mendapatkan kode otorisasi *client*. Kode dapat dilihat pada Kode Sumber 2.

```
if (tipe == "konek")
{
    string request = "1##"+ '\0';
    MessageBox.Show("sent: " + request);
    stream.Write(Encoding.ASCII.GetBytes(request),
    0, request.Length);

    int i;
    while ((i = stream.ReadByte()) != 0)
    {
        sb.Append((char)i);
    }

    Process.Start(sb.ToString());
    MessageBox.Show("masukkan auth: ");
else if (tipe == "login")
{
     string request = "2##"+textBox1.Text+"##"+
     '\0';
     MessageBox.Show("sent:  "  +  request);  ;
     stream.Write(Encoding.ASCII.GetBytes(request),
     0, request.Length);

     int i;
     while ((i = stream.ReadByte()) != 0)
     {
         sb.Append((char)i);
```

```
      }
      MessageBox.Show(sb.ToString());
}
```

Kode Sumber 2 Proses *login* pada *client*

B. Pengambilan data *tweet*

Proses pengambilan data *tweet* merupakan proses pengambilan data pada *server* Twitter untuk diolah. Kode dapat dilihat pada Kode Sumber 3.

```
public string fungsiTweet(string token, string
tokenSecret, string tipe)
{
   bool tanda = false;
   int idMax = 0, clus = 0;

   service.AuthenticateWith(token, tokenSecret);
   TwitterUser tweetuser =
   service.GetUserProfile(new
   GetUserProfileOptions()
   {
       IncludeEntities = false,
       SkipStatus = false
   });

   string userID = tweetuser.Id.ToString() + tipe;
   db.createUser(userID);
   var dataAmbil =
   service.ListTweetsMentioningMe(new
   ListTweetsMentioningMeOptions
   {
       Count = 50
   });
   if(tipe.Equals("timeline"))
   {
      dataAmbil =
      service.ListTweetsOnHomeTimeline(new
      ListTweetsOnHomeTimelineOptions
     {
       Count = 50
     });
    }
   int jumlah = dataAmbil.Count(
```

```
if (dataAmbil!=null)
{
  string[][] dataMentionAwal = new
  string[jumlah][];
   for (int i = 0; i < jumlah; i++)
   {
  dataMentionAwal[i] = new string[4];
   }
   DateTime[] date = new DateTime[jumlah];
   idMax = db.cekIdTerbesar(userID);
   int penandaMention = 0;
   foreach (var item in dataAmbil)
   {
     string url = "";
     TimeZoneInfo cstZone =
     TimeZoneInfo.FindSystemTimeZoneById("SE Asia
     Standard Time");
     DateTime cstTime =
     TimeZoneInfo.ConvertTimeFromUtc(item.CreatedD
     ate, cstZone);
      if (idMax > 0){
       tanda = db.cekTanggal(userID, cstTime);
      }
     else tanda = true;
     if (tanda == true)
     {
     foreach (var itemUrl in item.Entities.Urls)
     {
      url = itemUrl.Value;
     }
     dataMentionAwal[penandaMention][0] =
     item.User.Id.ToString();
     dataMentionAwal[penandaMention][1] =
     item.User.ScreenName;
     dataMentionAwal[penandaMention][2] =
     item.Text;
     dataMentionAwal[penandaMention][3] = url;
     date[penandaMention] = cstTime;
     penandaMention++;
     }
    }
    Console.WriteLine("jumlah data ambil:
    "+penandaMention);
     for (int i = penandaMention - 1; i > -1; i--)
```

```
      {
        db.insertData(userID,dataMentionAwal[i][0],
        dataMentionAwal[i][1],dataMentionAwal[i][2],
        dataMentionAwal[i][3], date[i]);}
```

Kode Sumber 3 Proses pengambilan data *tweet*

C. Proses Analisis Kebiasaan Pesan *Tweet*

1. Kesamaan Teks

Proses kesamaan teks dilakukan untuk mendapatkan nilai kesamaan dari suatu *tweet*. Kode dapat dilihat pada

```
public double textCompare( string []
dataHomeTersangka, string tweetTerdeteksi)
{
 string[] hasil;
 string[] dataTweet;
 string tweetTerdeksiSpam = tweetTerdeteksi;
 double[] nilai = new
 double[dataHomeTersangka.Length];
 int RT = 0;
 hasil = tweetTerdeksiSpam.Split(' ');
 foreach (var tweet in hasil)
 {
   if(tweet.Equals("RT")) RT++;
 }
 double hasilNilaiAkhir = 0;
 if (hasil.Length > 3 && RT < 1)
 {
   string[] shinglingTweet = new
   string[hasil.Length - 3];
   for (int j = 0; j < hasil.Length - 3; j++)
   {
    shinglingTweet[j] = hasil[j] + " " + hasil[j +
    1] + " " + hasil[j + 2] + " " + hasil[j + 3];
    }
    for (int i = 0; i < dataHomeTersangka.Length;
    i++)
    {
     double hasilNilai = 0;
     dataTweet = dataHomeTersangka[i].Split(' ');
     if (dataTweet.Length > 3)                    {
       string[] dataBanding = new
       string[dataTweet.Length - 3];
```

```
        for (int b = 0; b < dataTweet.Length - 3;
        b++)
        {
         dataBanding[b] = dataTweet[b] + " " +
         dataTweet[b + 1] + " " + dataTweet[b + 2] +
         " " + dataTweet[b + 3];
        }
        double intersect =
        (shinglingTweet.Intersect(dataBanding)).Coun
        t();
        double union =
        (shinglingTweet.Union(dataBanding)).Count();
        hasilNilai = (intersect / union) *100;
       }
       else hasilNilai = 0;
       if (hasilNilai>hasilNilaiAkhir && hasilNilai
       != 100)
       {
         hasilNilaiAkhir = hasilNilai;
       }
     }
    }
    else hasilNilaiAkhir = 0;
    if (hasilNilaiAkhir > 0) hasilNilaiAkhir = 100;
    return hasilNilaiAkhir;
}
```

Kode Sumber 4 Pemeriksaan kesamaan *tweet*

2. Kesamaan URL

Proses pemeriksaan kesamaan URL dilakukan untuk mencari persentase kesaaman URL yang dikirimkan. Kode dapat dilihat pada Kode Sumber 5.

```
public int urlCompare(string[] dataHomeTersangka,
string tweetTerdeteksi, int idFor)
{
  int flag = 0, penandaUrl=0;
  string urlTerdeteksi = "";
  string urlTersangka = "";
  try
  {
   if (tweetTerdeteksi != "")
   {
```

```
ICollection keyUrlterdeteksi = HurlAsli.Keys;
foreach (string a in keyUrlterdeteksi)
{
 if (a.Equals(tweetTerdeteksi))
 {
  urlTerdeteksi = HurlAsli[a].ToString();
  penandaUrl = 1;
  break;
 }
}
if (penandaUrl == 0)
{
 urlTerdeteksi = cUrl.UrlCek(tweetTerdeteksi);
}
for (int i = 0; i < dataHomeTersangka.Length;
i++)
{
  penandaUrl = 0;
  if (dataHomeTersangka[i] != null)
  {
   ICollection keyUrls = HurlAsli.Keys;
   foreach (string a in keyUrls)
   {
   if (a.Equals(dataHomeTersangka[i]))
   {
    urlTersangka = HurlAsli[a].ToString();
    penandaUrl = 1;
    break;
   }
  }
  if (penandaUrl.Equals(0))
  {
   urlTersangka =
   cUrl.UrlCek(dataHomeTersangka[i]);
   HurlAsli.Add(dataHomeTersangka[i],
   urlTersangka);
  }
  if (urlTersangka.Equals(urlTerdeteksi)
  {
   string penanda = i.ToString() +
   idFor.ToString();
   flag++;
   }
  }
```

```
     }
    }
    else flag = 0;
    }
    catch
    {
     flag = 1;
    }
    if (flag == 1) flag = 0;
   return flag;
}
```

Kode Sumber 5 Pemeriksaan kesamaan URL

D. Analisis kebiasaan pada profil pengirim *tweet*

Proses analisis kebiasan pada profil pengirim *tweet* dilakukan untuk mendapatkan atribut-atribut yang diperlukan untuk pengambilan keputusan menggunakan metode klasifikasi *decision tree.* Kode dapat dilihat pada Kode Sumber 6.

```
for (int i = 0; i < dataMention.Length; i++)
{
 string tweetTerdeteksi = dataMention[i][1];
 string urlTerdeteksi = dataMention[i][3];
 Console.WriteLine("mention ke-" + (i + 1) + " dari:
 " + dataMention[i][0]);
 Console.WriteLine();
 double fraksiFoll = -1;
 double tweetPerHari = -1;
 ICollection hari = HtweetHari.Keys;
 foreach (string b in hari)
 {
  if (b.Equals(dataMention[i][0]))
  {
   tweetPerHari = Convert.ToDouble(HtweetHari[b]);
  }
 }
 ICollection tFrak = Hfraksi.Keys;
 foreach (string b in tFrak)
 {
  if (b.Equals(dataMention[i][0]))
  {
   fraksiFoll = Convert.ToDouble(Hfraksi[b]);
   }
```

```
}
if (tweetPerHari == -1 && fraksiFoll == -1)
{
 TwitterUser profAkun =
 service.GetUserProfileFor(new
 GetUserProfileForOptions()
 {
  ScreenName = dataMention[i][0]
 });
 TimeZoneInfo cstZone =
 TimeZoneInfo.FindSystemTimeZoneById("SE Asia
 Standard Time");
 DateTime createDateUser =
 TimeZoneInfo.ConvertTimeFromUtc(profAkun.CreatedD
 ate, cstZone);
 double diffCreate =
 System.DateTime.Now.Subtract(createDateUser).Tota
 lDays;

 int jumlahTweet = profAkun.StatusesCount;
 tweetPerHari = jumlahTweet / diffCreate;
 double following = profAkun.FriendsCount;
 double follower = profAkun.FollowersCount;
 if (follower == 0) follower = 1;      fraksiFoll
 = following / follower;
}
int id = -1;
string screenName = dataMention[i][0];
int lengthH = 0;
double intervalH = 0, replyH = 0, aveUrlH = 0;
ICollection key = HsetID.Keys;
foreach (string q in key)
{
 if (q.Equals(screenName))
 {
  id = Convert.ToInt32(HsetID[q]);
 }
}
if (id != -1)
{
 Console.WriteLine("Data sudah ada");
 lengthH =
 Convert.ToInt32(nilaiSpesificUser[id][4]);
 string[] dataHomeTersangka = new string[lengthH];
```

```
string[] dataUrlTersangka = new string[lengthH];
for (int o = 0; o < lengthH; o++)
{
 dataHomeTersangka[o] = text[id][o];
 dataUrlTersangka[o] = url[id][o];
}
replyH = nilaiSpesificUser[id][1];
intervalH = nilaiSpesificUser[id][2];
aveUrlH = nilaiSpesificUser[id][3];
}
else
{
  HsetID.Add(screenName, penandaHash);
  var tweetHome =
  service.ListTweetsOnUserTimeline(new
  ListTweetsOnUserTimelineOptions
  {
   ScreenName = dataMention[i][0],
   Count = 20
  });
  string[] dataHomeTersangka = new
  string[tweetHome.Count()];
  string[] dataUrlTersangka = new
  string[tweetHome.Count()];
  double interval = 0;
  double aveUrl = 0;
  DateTime tanggalAkhir = System.DateTime.Now;
  double jumUrl = 0;
  int jumlahHome = tweetHome.Count();
  nilaiSpesificUser[penandaHash][4] = jumlahHome;
  double reply = 0;
   int penandatTweetHome = 0;
  foreach (var tweet in tweetHome)
  {
   dataHomeTersangka[penandatTweetHome] =
   tweet.Text;
   text[penandaHash][penandatTweetHome] =
   dataHomeTersangka[penandatTweetHome];
   if (tweet.InReplyToUserId != null)
   {
 reply++;
}
if (penandatTweetHome == 0)
{
```

```
    tanggalAkhir = tweet.CreatedDate;
   }
   else
   {
    TimeSpan time =
    tanggalAkhir.Subtract(tweet.CreatedDate);
    interval += time.TotalMinutes;
    tanggalAkhir = tweet.CreatedDate;
   }
   foreach (var itemUrl in tweet.Entities.Urls)
   {
    dataUrlTersangka[penandatTweetHome] =
    itemUrl.Value;
    url[penandaHash][penandatTweetHome] =
    dataUrlTersangka[penandatTweetHome];
    jumUrl++;
    }
    penandatTweetHome++;
   }
   if (reply == 0)
   {
    foreach (var homeski in tweetHome)
    {
     string[] interak =
     homeski.Text.ToString().Split(' ');
     if (interak[0].IndexOf("@") != -1)
     {
      reply++;
     }
    }
   }
   reply = (reply / tweetHome.Count() * 100);
   nilaiSpesificUser[penandaHash][1] = reply;
   interval /= tweetHome.Count();
   nilaiSpesificUser[penandaHash][2] = interval;
   aveUrl = (jumUrl / tweetHome.Count() * 100);
   nilaiSpesificUser[penandaHash][3] = aveUrl;
  }
```

Kode Sumber 6 analisis kebiasaan pada profil pengirim *tweet*

E. Pengelompokan Data *Tweet* Menggunakan K-*Means*

Pengelompokan data *tweet* K-*Means* dilakukan untuk mendapatkan kelompok data yang terindikasi bukan merupakan pesan *spam*. Kode dapat dilihat pada Kode Sumber 7.

```
public void hitungKmeans(int count,
string[][]dataMention, string id)
{
 double[][] dataCluster = new double[count][];
 for (int i = 0; i < dataCluster.Length; i++)
 {
  dataCluster[i] = new double[9];
 }
 for (int i = 0; i < count; i++)
 {
  dataCluster[i][0] =
  Convert.ToDouble(dataMention[i][1]);
  dataCluster[i][1] =
  Convert.ToDouble(dataMention[i][2
  }
  Accord.Math.Tools.SetupGenerator(0);
  KMeans kmeans = new KMeans(4);
  int[] hasilCluster =
  kmeans.Compute(dataCluster);
  for (int i = 0; i < count; i++)
  {
   db.insertCluster(id, dataMention[i][0],
   hasilCluster[i]);
   }
}
```

Kode Sumber 7 Pengelompokan data menggunakan K-*Means*

F. Pengambilan Keputusan Menggunakan *Decision Tree*

Proses pengambilan keputusan menggunakan *decision tree* dilakukan untuk pengambilan keputusan terhadap suatu pesan *tweet* merupakan pesan *spam* atau pesan normal. Kode dapat dilihat pada Kode Sumber 8.

```
for (int i = 1; i <= count; i++)
{
 string getData = "SELECT
 `cluster`,`reply`,`tweetPerHari`" +
```

```
",`followingPerFollower`,`averageUrl`,`averageTimeI
nterval`" +
" FROM `" + userID + "` WHERE `id`='" + i + "'";
connection.Open();
cmd = connection.CreateCommand();
cmd.CommandText = getData;
MySqlDataReader reader;
reader = cmd.ExecuteReader();
reader.Read();
cluster = Convert.ToDouble(reader.GetValue(0));
if (cluster == clus) cluster = 0;
else cluster = 1;

reply = Convert.ToDouble(reader.GetValue(1));
tweetPerHari =
Convert.ToDouble(reader.GetValue(2));
follow = Convert.ToDouble(reader.GetValue(3));
averageUrl = Convert.ToDouble(reader.GetValue(4));
averageTimeInterval =
Convert.ToDouble(reader.GetValue(5));
if (cluster == 0)
{
 if (averageTimeInterval <= 14.6)
 {
  if (follow > 1)
  {
   if (tweetPerHari <= 34.41)
   {
    if (reply > 73.33)
    {
      hasil = "spam";
    }
    else hasil = "bukan spam";
    }
   else hasil = "spam";
  }
  else hasil = "bukan spam";
 }
 else hasil = "spam";

 connection.Close();
```

Kode Sumber 8 Pengambilan keputusan menggunakan *decision tree*

G. Detail Uji Coba

Analisis pengujian dilakukan melalui serangkaian proses. Tabel G 1. merupakan gambaran analisis atribut pada akun *spam* sedangkan pada

Tabel G 2 merupakan gambaran analisis pada akun normal. Pada Tabel G 3 dan Tabel G 4 merupakan hasil uji coba untuk menentukan nilai *k* pada penggunakan metode pengelompokan K-*Means*. Pada Tabel G 5 dan Tabel G 6 merupakan hasil uji coba akurasi penggunaan model pada 7 akun berbeda.

Tabel G 1. Analisis kebiasaan pada profil akun *spam*

Username	*Reply* (%)	*Following / follower*	*Tweet* Per Hari	*Average* URL (%)	*Average* Interval (menit)
IPOT162	100	2.00	84.04	200.00	0.98
IPOTsatu	100	6.00	80.92	100.00	1.00
ipot161	100	1.15	254.47	200.00	0.73
ipotlimabelas	100	20.00	274.30	100.00	0.76
ipottigabelas	100	24.00	104.18	100.00	0.98
ipot104	100	15.00	598.49	33.33	0.73
ipot105	100	17.00	581.07	33.33	0.73
ipot159	100	1.75	215.80	200.00	0.98
ipot103	100	7.50	570.05	33.33	0.73
ipot135	100	4.00	78.62	200.00	0.98
ipotsebelas	100	8.00	99.59	100.00	0.98
ipot102	100	5.00	564.26	33.33	0.73
ipot101	100	11.00	473.08	33.33	0.73
ipot106	100	6.33	581.10	33.33	0.73
ipot107	100	18.00	540.47	33.33	0.73
Rata-rata	100	9.78	340.03	95.56	0.83

Tabel G 2. Analisis kebiasaan pada akun normal

Username	*Reply* (%)	*Following / follower*	*Tweet* Per Hari	*Average* URL (%)	*Average* Interval (menit)
happyach	46.11	0.67	12.55	19.44	720.13
ebeckz22	36.67	0.87	8.24	10.00	1080.16
momo9	73.89	0.65	15.39	36.11	1362.69
evariaayu	28.49	0.74	10.64	37.43	848.82
idhammardy	14.53	0.65	12.18	63.48	1150.82
r_satyaputra	15.64	1.00	4.18	7.87	1312.90
Liputan9	3.89	0.00	14.26	8.33	167.97
MerryRiana	7.22	0.47	32.65	9.44	17.80
hmtc_its	5.06	0.16	7.02	12.22	54.31
MataNajwa	2.22	0.00	11.93	13.33	71.60
my_supersoccer	53.89	0.00	97.65	90.56	4.50
eventsurabaya	1.67	0.69	40.59	24.44	10.78
karniilyas	5.59	0.00	15.06	2.79	460.84
korantempo	0.00	0.00	81.89	91.62	9.79
KickAndyShow	2.78	0.00	7.14	12.78	212.98
detikcom	0.00	0.00	352.05	100.00	3.45
DavidLuiz_4	2.22	0.00	1.62	55.56	1056.81
acmilan	7.22	0.00	21.18	43.33	102.67
realmadrid	0.00	0.00	18.45	50.00	62.71
Youtube	1.67	0.00	4.35	97.78	259.45
agnezmo	25.14	0.00	7.50	40.78	178.12
twitter	1.12	0.00	18.45	96.09	2908.27
justinbieber	13.89	0.00	14.27	66.11	168.17
facebook	0.00	0.00	0.41	78.89	8071.75
instagram	0.00	0.00	3.66	100.56	485.75
indo_line	6.67	0.00	2.92	12.22	91.27

Username	*Reply* (%)	*Following / follower*	*Tweet* Per Hari	*Average* URL (%)	*Average* Interval (menit)
hariankompas	0.00	0.00	26.96	85.00	88.77
infojember	7.22	0.09	46.16	18.89	48.17
infojakarta	0.00	0.19	68.43	92.22	11.56
infosurabaya	0.56	0.00	66.22	42.78	138.69
radityadika	1.67	0.00	21.19	6.67	121.86
poconggg	20.56	0.00	15.96	22.78	248.11
KeiSavourie	11.11	0.03	36.55	4.44	16.13
TrioMacan2000	0.00	0.01	435.58	45.56	4.77
milanistwit	7.22	0.01	21.55	6.67	967.24
yeahmahasiswa	0.00	0.00	28.04	5.03	143.45
ITS_Surabaya	0.56	0.00	55.51	53.89	24.70
Metro_TV	0.00	0.00	173.80	84.44	8.15
officialRCTI	0.56	0.00	34.47	37.22	31.16
UtdIndonesia	2.22	0.00	16.42	18.89	122.71
ChicaRealMadrid	0.56	0.01	176.75	4.44	9.64
hitmansystem	0.00	0.00	16.17	9.50	40.35
MTLovenHoney	0.00	0.00	21.36	2.78	65.79
raisa6690	14.44	0.00	13.34	12.22	386.23
guardian	0.00	0.00	39.98	100.00	9.13
Troll_Football	0.56	0.00	0.43	26.86	3413.05
9GAG	0.56	0.00	9.44	57.22	79.16
kaskus	0.56	0.00	11.25	83.89	72.05
TravellerKaskus	10.56	0.01	40.59	3.33	47.29
detiksport	0.00	0.00	78.00	94.44	24.47
rata-rata	8.69	0.13	45.41	42.01	539.94

Tabel G 3. Uji coba jumlah kluster = 2

Jumlah data	Persentase *spam* pada jumlah kluster = 2 (%)	
	1	2
48	100	0
100	100	0
703	100	2.5
990	100	8.99
1094	100	14.53

Tabel G 4. Uji coba jumlah kluster= 4

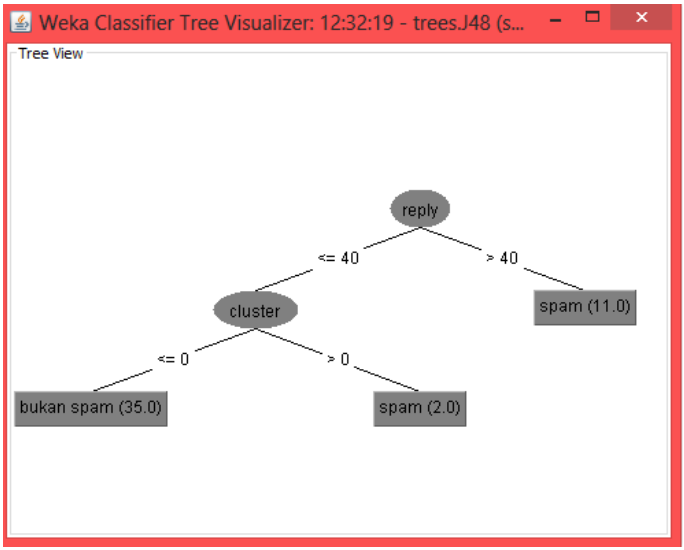
Jumlah data	Persentase *spam* pada jumlah kluster = 4 (%)			
	1	2	3	4
48	100	0	Tidak ada data	Tidak ada data
100	100	0	Tidak ada data	Tidak ada data
703	100	100	0	Tidak ada data
990	100	100	0	100
1094	100	100	0	100

Tabel G 5. Uji coba akurasi model pada 7 data *mention* pengguna

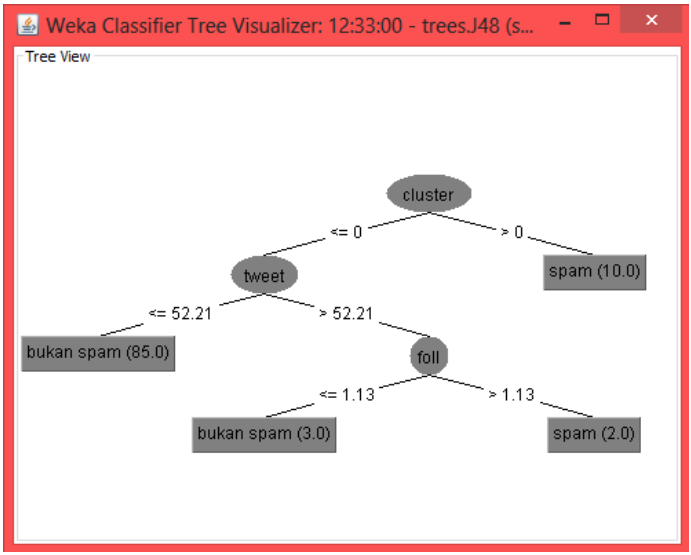
Pengguna ke-n	Akurasi Model ke-n (%) *mention*				
	1	2	3	4	5
1	70.53	95.79	97.89	95.79	97.89
2	84.21	90.53	95.79	90.53	95.79
3	76.00	89.00	99.00	89.00	99.00
4	72.92	97.92	100	97.91	100
5	75	100	100	100	100
6	46.39	97.94	100	97.94	100
7	75	89	100	100	100
Rata-rata	71.44	94.31	98.95	95.88	98.95

Tabel G 6. Uji coba akurasi model pada 7 data *timeline* pengguna

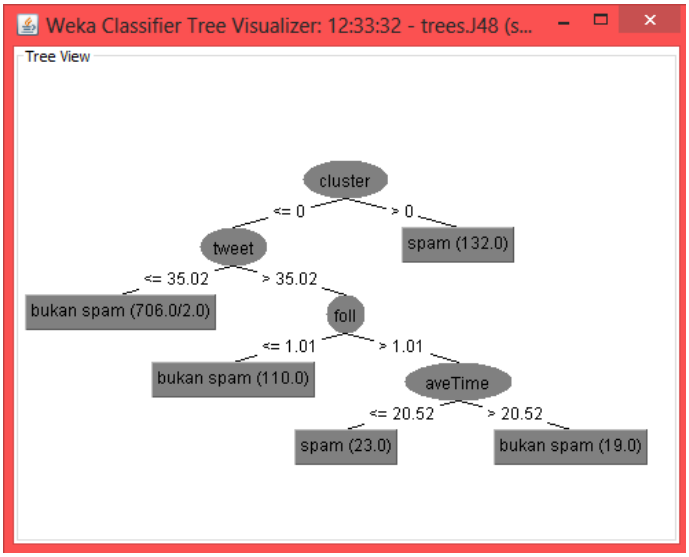
Pengguna ke-n	Akurasi Model ke-n (%) *timeline*				
	1	2	3	4	5
1	98.95	100	100	100	100
2	98.95	100	100	100	100
3	100	100	100	100	99.00
4	98	100	100	100	100
5	94.95	100	100	100	100
6	95.70	98.92	98.92	98.92	87.10
7	89.80	100	100	100	100
Rata-rata	96.62	99.85	99.85	99.85	98.01



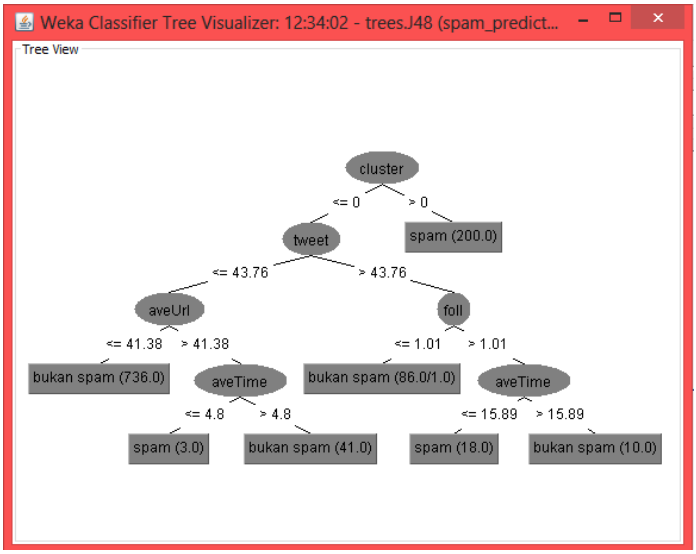
Gambar G 1. Model *decision tree* ke-1



Gambar G 2. Model *decision tree* ke-2



Gambar G 3. Model *decision tree* ke-4



Gambar G 4. Model *decision tree* ke-5

BIODATA PENULIS

Raditya Andre Nurwitantyo, biasa dipanggil Radit atau Ebek, dilahirkan di Jember pada tanggal 22 Mei 1991. Penulis adalah anak pertama dari 3 bersaudara. Penulis telah menempuh pendidikan formal di SD Al-Furqan Jember (1997-2003), SMP Negeri 2 Jember (2003-2006), SMA Negeri 1 Jember (2006-2009). Pada tahun 2009-2010 penulis sempat menempuh kuliah strata satu di jurusan Sistem Informasi, Universitas Jember. Pada tahun 2010 penulis diterima di strata satu Jurusan Teknik Informatika Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya angkatan 2010 yang terdaftar dengan NRP. 5110100014. Di Jurusan Teknik Informatika ini, penulis mengambil bidang minat *Net Centric Computing* (NCC). Selama menempuh kuliah, penulis aktif sebagai staf departemen minat dan bakat pada tahun kedua, dan pada tahun ketiga sebagai kepala departemen kewirausahaan dan minat bakat di Himpunan Mahasiswa Teknik Computer (HMTC). Pada beberapa acara kampus, penulis juga aktif menjadi panitia, baik sebagai anggota maupun koordinator. Selain itu penulis beberapa kali menjadi asisten, diantaranya Asisten Praktikum Jaringan Komputer dan Asisten Praktikum Sistem Operasi. Penulis dapat dihubungi melalui alamat *e-mail* di andre.raditya22@gmail.com.