TESIS

RESPONS TWEET OTOMATIS UNTUK *FEEDBACK* PELANGGAN TAKSI MENGGUNAKAN *NAÏVE BAYES CLASSIFIER* DAN *ROCCHIO CLASSIFIER*
(Studi Kasus : Perusahaan Taksi Express Group)

AUTOMATIC TWEET RESPONSE FOR TAXI CUSTOMER FEEDBACK USING NAÏVE BAYES CLASSIFIER AND ROCCHIO CLASSIFIER
(Case Study : Express Group Taxi Company)

IDA MARATUL KHAMIDAH
13/357275/PPA/04468

PROGRAM STUDI S2 ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2017

TESIS

RESPONS TWEET OTOMATIS UNTUK *FEEDBACK* PELANGGAN TAKSI MENGGUNAKAN *NAÏVE BAYES CLASSIFIER* DAN *ROCCHIO CLASSIFIER*
(Studi Kasus : Perusahaan Taksi Express Group)

AUTOMATIC TWEET RESPONSE FOR TAXI CUSTOMER FEEDBACK USING NAÏVE BAYES CLASSIFIER AND ROCCHIO CLASSIFIER
(Case Study : Express Group Taxi Company)

Diajukan untuk memenuhi salah satu syarat memperoleh derajat
Master of Computer Science

IDA MARATUL KHAMIDAH
13/357275/PPA/04468

PROGRAM STUDI S2 ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2017

HALAMAN PENGESAHAN

TESIS

RESPONS TWEET OTOMATIS UNTUK *FEEDBACK* PELANGGAN TAKSI MENGGUNAKAN *NAÏVE BAYES CLASSIFIER* DAN *ROCCHIO CLASSIFIER*
(Studi Kasus : Perusahaaan Taksi Express Group)

IDA MARATUL KHAMIDAH
13/357275/PPA/04468

Telah dipertahankan didepan Dewan Penguji
pada tanggal 19 Juni 2017

Susunan Dewan Penguji

Pembimbing Utama

Dr. Azhari, M.T
NIP. 19620920 1989 03 1 002

Ketua Dewan Penguji

Dr. Tri Kuntoro Priyambodo, M.Sc
NIP. 19591121 1988 03 1 001

Anggota

Dr. Sigit Priyanta, S.Si., M.Kom
NIP. 19770401 2002 12 1 002

Anggota

Dr. tech. Khabib Mustofa, S.Si., M.Kom
NIP. 19720722 1998 03 1 002

Mengetahui
a.n. Dekan FMIPA-UGM
Wakil Dekan Bidang Akademik dan Kemahasiswaan

Dr.rer.nat. Nurul Hidayat Aprilita, M.Si
NIP. 197304071998031002

Tesis ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar *Master of Computer Science*
Tanggal, 26 Juli 2017

Dr. Tri Kuntoro Priyambodo, M.Sc
Pengelola Program Monodisiplin S2 Ilmu Komputer

PERNYATAAN

Dengan ini saya menyatakan bahwa Tesis ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar Master di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 26 Juli 2017

Ida Maratul Khamidah

KATA PENGANTAR

Alhamdulillah, puji syukur penulis panjatkan atas karunia Allah SWT yang diberikan kepada penulis sehingga tesis ini dapat diselesaikan tepat pada waktu yang telah ditentukan. Tesis yang diambil penulis berjudul "**RESPONS TWEET OTOMATIS UNTUK *FEEDBACK* PELANGGAN TAKSI MENGGUNAKAN *NAÏVE BAYES CLASSIFIER* DAN *ROCCHIO CLASSIFIER***". Tujuan penulisan tesis ini adalah sebagai salah satu syarat untuk mendapatkan gelar *Master of Computer Science (M.Cs)* pada Program Pascasarjana Magister Ilmu Komputer Universitas Gadjah Mada. Penulis sangat menyadari bahwa tanpa bimbingan dan dukungan dari semua pihak dalam pembuatan tesis ini, maka penulis tidak dapat menyelesaikan tesis ini tepat pada waktunya. Oleh karena itu ijinkanlah penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Bapak Dr. Azhari S.N., M.T selaku pembimbing tesis yang telah banyak mendukung serta membimbing penulis dalam menyelesaikan tesis ini.
2. Mama Mimi tercinta, kakak dan adik, serta keluarga besar yang selalu dan tak habisnya memberikan dukungan doa, materi dan semangat kepada penulis.
3. Bapak Dr. Tri Kuntoro Priyambodo, M.Sc selaku Pengelola Program Studi Magister Ilmu Komputer Universitas Gadjah Mada.
4. Ibu Prof. Sri Hartati, Ph.D selaku pembimbing akademik selama menempuh perkuliahan.
5. Seluruh Bapak dan Ibu dosen Program Studi S2 Ilmu Komputer Universitas Gadjah Mada yang telah memberikan ilmu dan pelajaran yang sangat berarti bagi penulis selama menempuh masa studi.
6. Bapak Sugeng, Ibu Rini dan Bapak Kuncoro serta seluruh staf dan karyawan Program Studi S2 Ilmu Komputer Universitas Gadjah Mada yang telah sangat membantu dalam pelayanan administrasi selama masa kuliah.

7. Semua pihak yang telah membantu proses penelitian ini baik secara langsung maupun tidak langsung yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa tesis ini masih jauh dari kesempurnaan. Oleh karena itu, dengan segala kerendahan hati penulis mengharapkan masukan berupa kritik dan saran demi kesempurnaan tesis ini. Penulis berharap semoga tesis ini bermanfaat bagi orang lain.

Yogyakarta, 26 Juli 2017

Ida Maratul Khamidah
Penulis

DAFTAR ISI

KATA PENGANTAR i
DAFTAR ISI iii
DAFTAR GAMBAR v
DAFTAR TABEL vi
INTISARI vii
ABSTRACT viii
BAB I PENDAHULUAN 1
1.1 Latar Belakang Masalah 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 4
1.4 Tujuan dan Manfaat Penelitian 4
1.5 Keaslian Penelitian 4
1.6 Metode Penelitian 5
1.7 Sistematika Penulisan 6
BAB II TINJAUAN PUSTAKA 8
BAB III LANDASAN TEORI 15
3.1 Text Mining 15
3.2 Teorema Bayes 16
3.3 Teknik Klasifikasi dan *Naïve Bayes Classifier* (NBC) 16
3.4 TF-IDF 18
3.5 Rocchio Classifier 19
3.6 Evaluasi Model Klasifikasi 19
3.7 Twitter 21
BAB IV ANALISIS DAN PERANCANGAN SISTEM 23
4.1 Analisis Kebutuhan 23
4.2 Data dan Label Data 23
4.3 Deskripsi Umum Sistem 24
4.4 Rancangan *Download* Tweet 26
4.5 Rancangan Preprocessing 27
4.6 Rancangan Klasifikasi Tweet 27
4.6.1 Klasifikasi dengan NBC 27
4.6.2 Klasifikasi dengan *Rocchio Classifier* 31
4.7 Pengiriman Respons 34
4.8 Rancangan Proses 36
4.8.1 Diagram konteks 37
4.8.2 DFD level 1 37
4.9 Rancangan Basisdata 39
4.10 Rancangan Antarmuka Respons Tweet 46
4.11 Rancangan Pengujian 47
BAB V IMPLEMENTASI SISTEM 48
5.1 Implementasi Download Tweet 48

5.2 Implementasi *Preprocessing* 49
5.3 Implementasi Klasifikasi NBC 52
5.4 Implementasi *Rocchio* 55
5.5 Implementasi Pengiriman Respons Tweet 58
BAB VI HASIL DAN PEMBAHASAN 61
6.1 Data 61
6.2 Pengujian *White Box* 62
6.3 Pengujian Akurasi *Naïve Bayes Classifier* 68
6.4 Pengujian Performansi *Rocchio Classifier* 69
6.5 Pengujian Akurasi *Naïve Bayes Classifier* dan *Rocchio Classifier* 70
6.6 Pengujian Respons 71
BAB VII KESIMPULAN DAN SARAN 73
DAFTAR PUSTAKA 74

DAFTAR GAMBAR

Gambar 1.1 Contoh tweet dan respons 2
Gambar 1.2 Contoh tweet 2
Gambar 4.1 Label kelas..... 24
Gambar 4.2 Gambaran model sistem 25
Gambar 4.3 Klasifikasi tweet dengan NBC 28
Gambar 4.4 Rancangan klasifikasi tweet dengan *Rocchio Classifier*..... 32
Gambar 4.5 Rancangan respons tweet 36
Gambar 4.6 Diagram konteks 37
Gambar 4.7 DFD level 1 39
Gambar 4.8 Rancangan ERD 40
Gambar 4. 9 Antarmuka respons *feedback* pelanggan 47
Gambar 5.1 Implementasi download tweet..... 48
Gambar 5.2 Implementasi menghapus URL 49
Gambar 5.3 Implementasi penghapusan tanda baca 50
Gambar 5.4 Implementasi casefolding..... 50
Gambar 5.5 Implementasi tokenization 50
Gambar 5.6 Implementasi replacement..... 51
Gambar 5.7 Implementasi stemming 51
Gambar 5.8 Implementasi stopword removal 52
Gambar 5.9 Implementasi penggabungan negasi..... 52
Gambar 5.10 Implementasi probabilitas bersyarat..... 53
Gambar 5.11 Implementasi probabilitas prior kelas 53
Gambar 5.12 Implementasi *testing* NBC 54
Gambar 5.13 Implementasi term frequency..... 55
Gambar 5.14 Implementasi IDF..... 56
Gambar 5.15 Implementasi TFIDF 56
Gambar 5.16 Implementasi normalisasi TFIDF..... 57
Gambar 5.17 Implementasi perhitungan centroid..... 57
Gambar 5.18 Implementasi perhitungan jarak 58
Gambar 5.19 Implementasi penentuan tweet respons..... 59
Gambar 5.20 Implementasi pengiriman respons..... 59
Gambar 6.1 Nilai probabilitas prior setiap kelas..... 64
Gambar 6.2 Perhitungan probabilitas posterior sistem 65
Gambar 6.3 Centroid kelas..... 67
Gambar 6.4 Perhitungan jarak..... 68
Gambar 6.5 Tampilan tweet dan respons sistem..... 72

DAFTAR TABEL

Tabel 2.1 Tinjauan Pustaka 13
Tabel 4.1 Contoh data tweet bersih 30
Tabel 4.2 Probabilitas bersyarat dari kata t yang terjadi dalam kelas *ci* 30
Tabel 4.3 Probabilitas tweet berada dalam kelas *c* 31
Tabel 4.4 Contoh data tweet 33
Tabel 4.5 Perhitungan tfidf 33
Tabel 4.6 Struktur tabel tweets 42
Tabel 4.7 Struktur tabel respons 42
Tabel 4.8 Struktur tabel nbc_priorclass 43
Tabel 4.9 Struktur tabel nbc_katauniks 43
Tabel 4.10 Struktur tabel vocabs 44
Tabel 4.11 Struktur tabel replace_words 44
Tabel 4.12 Struktur tabel stopwords 44
Tabel 4.13 Struktur tabel operator 45
Tabel 4.14 Struktur tabel rocchio_word 45
Tabel 4.15 Struktur tabel rocchio_centroid 45
Tabel 4.16 Struktur table rocchio_tfidf 46
Tabel 6.1 Data training NBC masing-masing kelas 61
Tabel 6.2 Jumlah data testing NBC masing-masing kelas 62
Tabel 6.3 Jumlah data training Rocchio masing-masing kelas 62
Tabel 6.4 Jumlah data testing Rocchio masing-masing kelas 62
Tabel 6.5 Contoh data testing NBC 63
Tabel 6.6 Hasil preprocessing data testing NBC 63
Tabel 6.7 Probabilitas bersyarat 63
Tabel 6.8 Data tweet 65
Tabel 6.9 Vector dokumen 66
Tabel 6.10 Nilai *precision, recall, f-measure* NBC 68
Tabel 6. 11 Evaluasi Rocchio Classifier 69
Tabel 6.12 Nilai *precision, recall, f-measure Rocchio Classifier* 70
Tabel 6.13 Varian data set 70
Tabel 6.14 Perbandingan dengan metode NBC-NBC 71

INTISARI

RESPONS TWEET OTOMATIS UNTUK *FEEDBACK* PELANGGAN TAKSI MENGGUNAKAN METODE *NAÏVE BAYES CLASSIFIER* DAN *ROCCHIO CLASSIFIER* (Studi Kasus : Perusahaan Taksi Express Group)

Oleh

Ida Maratul Khamidah
13/357275/PPA/04468

Express Group adalah salah satu perusahaan taksi di Indonesia yang menggunakan Twitter untuk menerima *feedback* dari pelanggannya. Pada praktiknya ada *feedback* yang direspons setelah beberapa jam atau sehari setelah tweet diterima, bahkan ada tweet yang tidak direspons sampai akhirnya pelanggan mengirimkan keluhan.

Penelitian ini mengusulkan sistem respons tweet otomatis untuk *feedback* pelanggan taksi. Sistem terlebih dahulu melakukan *download* tweet. Data tweet melalui proses *preprocessing* untuk mendapatkan data bersih. Data bersih diolah untuk menentukan tweet termasuk dalam kelas apa. Oleh karena itu sistem respons tweet otomatis ini mengimplementasikan *classifier algorithm*. *Classifier algorithm* tersebut membangun model klasifikasi agar dapat menetukan kelas tweet. *Classifier* yang digunakan dalam penelitian ini adalah kombinasi antara *Naïve Bayes Classifier* dan *Rocchio Classifier*. Setelah kelas dari tweet diketahui kemudian dilakukan pengiriman respons secara otomatis kepada pelanggan.

Hasil penelitian menunjukkan bahwa respons yang dikirim memiliki akurasi sebesar 86.67%. Model klasifikasi yang dibangun dengan *Naïve Bayes Classifier* memiliki *accuracy* sebesar 83.16%, *precision* sebesar 85.72%, recall sebesar 78.65%, f-measure sebesar 81.84%. Model klasifikasi yang dibangun *Rocchio Classifier* memiliki *accuracy* sebesar 80%, *precision* sebesar 83.40%, recall sebesar 63.28%, f-measure sebesar 63.28%.

Kata kunci : respons otomatis, *Naïve Bayes Classifier*, *Rocchio Classifier*

ABSTRACT

AUTOMATIC TWEET RESPONSE FOR TAXI CUSTOMER FEEDBACK USING NAÏVE BAYES CLASSIFIER AND ROCCHIO CLASSIFIER (Case Study : Express Group Taxi Company)

by

Ida Maratul Khamidah
13/357275/PPA/04468

Express Group is one of the taxi company in Indonesia who uses Twitter to receive customer feedback. Some feedback were responded after a few hours or one day after the tweet feedback received, moreover there were feedback that did not received any respond until the customer sent a complaint.

This research proposes an automatic tweet response system for taxi customer feedback. In the automatic response system there is the process of downloading tweets. Tweets data are passed through the preprocessing process in order to get clean data. Clean data are processed to determine which tweets are included in what class. Therefore the automatic tweet response system implements the classifier algorithm. The classifier algorithm build a classification model in order to determine the tweet class. Classifier used in this study is a combination of Naïve Bayes Classifier and Rocchio Classifier. Once the class of tweet is known then the system sends the response automatically to the customer.

The research shows that the response have an accuracy of 86.67%. Classification model built with Naïve Bayes classifier have an accuracy of 88.23%, a precision of 88.23%, a recall of 88.23%, a f-measure of 88.23% . Classification model built with Rocchio Classifier have an accuracy of 80%, a precision of 83.40%, a recall of 63.28%, a f-measure of 63.28%.

Keyword : automatic response, *Naïve Bayes Classifier, Rocchio Classifier*

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Saat ini media sosial seperti Twitter telah berkembang pesat. Data global menyebut pada akhir Desember 2014 Twitter memiliki 284 juta pengguna aktif. Dick Costolo selaku CEO Twitter menuturkan bahwa jumlah pengguna Twitter di Indonesia mencapai 50 juta orang dan jumlahnya makin bertambah (Azis, 2015).

Express Group sebagai perusahaan penyedia layanan taksi adalah salah satu pengguna Twitter. Express Group menggunakan Twitter sebagai sarana komunikasi dengan pelanggannya. Melalui akun Twitter perusahaan Express Group dapat menerima *feedback* pelanggan taksi berupa penilaian atas layanan yang telah mereka berikan kepada pelanggan, termasuk menerima penilaian terhadap perilaku sopir sebagai *partner* kerja mereka.

Feedback pelanggan (atau tweet) yang diterima oleh Express Group ada beberapa kategori, antara lain keluhan, pujian, *follow* dan *unknown*. Tweet kategori keluhan merupakan tweet yang menginformasikan ketidaknyamanan yang dirasakan oleh pengguna layanan. Tweet kategori pujian merupakan tweet yang menginformasikan kepuasan pengguna atas layanan yang diberikan. Sedangkan kategori *follow* merupakan tweet yang menginformasikan permintaan *follow* akun Twitter milik pengguna layanan. Selain dari ketiga kategori tersebut maka tweet termasuk dalam tweet kategori *unknown*.

Saat ini Express Group merespons tweet dengan cara manual, artinya tweet direspons oleh beberapa operator dengan standar respons yang dimiliki dan biasa digunakan oleh perusahaan Express Group. Jika tweet yang diterima mengandung pujian maka pihak *customer care* akan merespons dengan standar respons “Terima kasih atas apresiasinya. Semoga bisa terus memberikan yang terbaik” dan masih ada beberapa standar responnya lainnya. Contoh tweet pengguna ditunjukkan pada Gambar 1.1.

Gambar 1.1 menunjukkan bahwa tweet mengandung keluhan dan direspons dengan standar respons keluhan. Pada Gambar 1.1 juga menunjukkan bahwa respons yang dikirimkan secara manual terdapat kelemahan, yakni ada beberapa tweet yang direspons setelah beberapa jam tweet diterima, pun ada respons yang diberikan sehari setelah tweet diterima, bahkan ada tweet yang tidak direspons sampai akhirnya pelanggan merasa kesal sehingga mengirimkan tweet seperti pada Gambar 1.2.

Gambar 1.1 Contoh tweet dan respons

Gambar 1. 2 Contoh tweet

Oleh karena adanya kelemahan pada cara manual maka dalam penelitian ini mengusulkan respons tweet otomatis untuk tweet taksi Express Group. Data tweet taksi yang telah diterima oleh pihak Express Group sudah sangat banyak, data

tersebut dapat digunakan untuk data pelatihan yang kemudian dapat digunakan untuk merespons tweet baru dengan beberapa *rule* yang dirancang.

Sistem respons tweet secara otomatis menerapkan kombinasi metode pembelajaran *Naïve Bayes Classifier* (NBC) dan *Rocchio Classifier*. NBC digunakan dalam penelitian ini karena antara tweet yang satu dengan yang lain ada kemungkinan (atau probabilitas) termasuk dalam suatu kelas tertentu. Thakur dan Mann (2014) menyatakan bahwa *Naïve Bayes Classifier* merupakan metode pembelajaran probabilitas. Devi dan Ranjan (2014) menyatakan bahwa NBC dapat digunakan untuk *multiclass classification*, sehingga NBC cocok digunakan dalam penelitian ini yang mengklasifikasikan tweet ke lebih dari dua kelas. Ting dkk. (2011) mengklasifikasikan teks dokumen ke dalam empat kategori, dan menyatakan bahwa *Naives Bayes Classifier* menjadi *classifier* terbaik jika dibandingkan dengan *Support Vector Machine, Nearest Neighbor, Decision Tree*. Hasil uji coba menunjukkan bahwa *Naïve Bayes Classifier* memiliki akurasi tertinggi dengan pencapaian nilai sebesar 96,9% dan membutuhkan waktu yang lebih sedikit jika dibandingkan dengan *classifier* lainnya. Hal yang sama dijelaskan oleh Han dan kamber (2006) bahwa *Naïve Bayes* mempunyai tingkat kecepatan dan akurasi yang tinggi.

Selain menerapkan NBC juga menerapkan *Rocchio Classifier*. *Rocchio Classifier* merupakan salah satu *supervised learning*. *Rocchio classifier* digunakan dalam proses klasifikasi karena menurut Widjojo dkk. (2014) menghasilkan sistem klasifikasi dokumen renungan harian yang baik dengan akurasi yang cukup tinggi. Begitu juga pada penelitian Lumbanraja (2013) yang mengklasifikasi dokumen teks, *Rocchio Classifier* memiliki performansi yang baik dengan akurasi yang cukup tinggi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang dijelaskan terdapat kelemahan pada respons yang dikirimkan secara manual, ada beberapa *feedback* pelanggan yang direspons secara lambat oleh operator, *feedback* pelanggan direspons beberapa jam atau sehari kemudian, bahkan ada *feedback* pelanggan yang tidak direspons sampai

akhirnya pelanggan merasa tidak puas, sehingga dibutuhkan sistem yang dapat merespons tweet pelanggan secara otomatis dengan menggunakan metode klasifikasi *Naïve Bayes Classifier* dan *Rocchio Classifier* untuk mendapatkan kelas daripada *feedback* pelanggan.

1.3 Batasan Masalah

Batasan penelitian adalah :

1. Data tweet yang digunakan adalah tweet bahasa Indonesia yang berasal dari *official account* Twitter perusahaan Express Group, tweet yang mengandung *mention username* Twitter perusahaan Express Group.
2. Data respons diambil dari standar respons yang selama ini digunakan oleh perusahaan Express Group.

1.4 Tujuan dan Manfaat Penelitian

Tujuan penelitian adalah mengembangkan sistem respons tweet otomatis menggunakan *Naïve Bayes Classifier* dan *Rocchio Classifier*. Manfaat penelitian adalah membantu pihak Express Group dalam merespons tweet secara otomatis, khususnya untuk membantu bagian operator yang bertugas merespons tweet.

1.5 Keaslian Penelitian

Penelitian mengenai respons Twitter sudah pernah dilakukan sebelumnya oleh Shin dkk. (2014) dan Sordoni dkk. (2015). Shin dkk. (2014) mengekplorasi respons otomatis yang tergantung pada konteks dengan *Statistic Machine Translation*. Sordoni dkk. (2015) menggunakan arsitektur *Neural Network* untuk *response generation data-driven* dari percakapan media sosial. Penelitian ini berbeda dengan penelitian sebelumnya, penelitian ini mengusulkan respons tweet secara otomatis dengan mengimplementasikan kombinasi algoritma *classifier* yaitu *Naïve Bayes Classifier* dan *Rocchio Classifier*. Kemudian dilanjutkan dengan menggunakan beberapa *rule* dalam merespons *feedback* tweet.

Naïve Bayes Classifier telah digunakan dalam penelitian Rodiansyah (2012), Ling dkk. (2014), Lestari dkk. (2013). *Rocchio Classifier* juga pernah digunakan dalam penelitian Widjojo dkk. (2014) dan Lumbanraja (2013).

Penelitian tersebut tidak membahas klasifikasi teks secara bertingkat. Penelitian yang diajukan akan membahas klasfikasi secara bertingkat, artinya akan dilakukan klasifikasi dalam dua tahapan. Tahap pertama untuk menentukan kelas pujian, keluhan, *follow* atau *unknown* dengan menggunakan NBC. Klasifikasi tahap kedua adalah tahap lanjutan dari kelas keluhan yang dihasilkan oleh NBC. Klasifikasi tahap kedua menentukan kelas keluhan sopir atau keluhan umum dengan menggunkan *Rocchio Classifier*. Setelah itu tweet yang diklasifikasikan akan direspons secara otomatis berdasarkan hasil kelas tweet.

1.6 Metode Penelitian

Penelitian ini dilakukan dengan beberapa tahapan sebagai berikut :

1. Kajian pustaka dan studi literatur

   Kegiatan pada tahap ini adalah melakukan pengumpulan berbagai bahan referensi, seperti jurnal penelitian, tesis/disertasi, buku-buku teori dan sumber-sumber lain termasuk informasi yang diperoleh dari internet sebagai sumber data dan informasi yang terkait dengan penelitian.

2. Pengumpulan Data

   Melakukan pengumpulan data tweet yang mengandung *mention username* @express_group, yaitu *official account* Twitter perusahaan Express Group. Pengumpulan tweet dilakukan dengan memanfaatkan Twitter API yang disediakan oleh Twitter. Adapun data tweet yang diambil adalah id tweet, isi tweet, *username*, tanggal dan waktu tweet. Data tersebut selanjutnya disimpan dalam database.

3. Perancangan Sistem

   Pada tahap ini dilakukan perancangan sistem meliputi perancangan DFD, perancangan database dan perancangan antarmuka pengguna. Proses perancangan ini dilakukan dengan mendeskripsikan secara tekstual dan menggunakan diagram pemodelan.

4. Implementasi

   Sistem yang akan dibangun dalam penelitian ini merupakan sistem berbasis web. Sistem akan diimplementasikan menggunakan bahasa pemrogram PHP dan basis data relasional MySQL.

5. Pengujian Sistem

Pengujian ini betujuan untuk mengetahui efektifitas model klasifikasi yang dihasilkan *classifier algorithm*. Pengujian ini dilakukan dengan menghitung *presicion, recall* dan *f-measusre* dan akurasi.. Pengujian membutuhkan data uji dengan label yang telah diketahui. Selain itu akan dilakukan pengujian respons *feedback* tweet yang dikirimkan oleh sistem apakah respons tersebut berkesinambungan.

1.7 Sistematika Penulisan

Sistematika penulisan laporan penelitian ini terdiri 7(tujuh) bab dengan masing-masing berisi rincian sebagai berikut :

Bab I Pendahuluan

Bab ini berisi latar belakang penelitian, rumasan masalah, batasan masalah, tujuan dan manfaat penelitian, keaslian penelitian, metode penelitian dan sistematika penulisan laporan penelitian.

Bab II Tinjauan Pustaka

Bab ini berisi tinjauan pustaka yang memuat uraian sistematis tentang informasi hasil penelitian yang pernah dilakukan terkait topic yang diteliti. Sejumlah literature dan publikasi ilmiah yang berhasil dikumpulkan mengenai respons otomatis, *text classification*, metode *Naïve Bayes Classifier* dan *Rocchio Classifier*.

Bab III Landasan Teori

Bab ini menguraikan dasar-dasar teori yang digunakan sebagai bahan referensi dalam penelitian. Dasar teori tersebut antara lain, *text mining*, klasifikasi, dan metode yang digunakan yakni *Naïve Bayes Classifier* dan *Rocchio Classifier*.

Bab IV Analisis dan Perancangan Sistem

Bab ini menguraikan tentang analisis dana rancangan sistem. Rancangan sistem yang dibuat adalah rancangan sistem, rancangan DFD, rancangan basis data, rancangan antarmuka sistem.

Bab V Implementasi

Bab ini berisi tentang implementasi yang telah dilakukan berdasarkan rancangan yaitu implementasi metode *Naïve Bayes Classifier* dan *Rocchio Classifier* untuk memeberikan respons tweet secara otomatis.

Bab VI Hasil Penelitian dan Pembahasan

Bab ini menguraikan pengujian dan evaluasi terhadap hasil-hasil penelitian. Bagian-bagian yang diuraikan meliputi pengujian model klasifikasi *Naïve Bayes Classifier* dan *Rocchio Classifier* dan pengujian hasil respons tweet yang dilakukan oleh sistem.

Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil penelitian dan saran-saran yang diajukan untuk pengembangan atau penellitian lanjutan.

BAB II
TINJAUAN PUSTAKA

Penelitian ini menggunakan beberapa referensi sumber pustaka yang berasal dari penelitian yang sudah dilakukan mengenai *automatic response*, klasifikasi teks, *Naïve Bayes Classifier* dan *Rocchio Classifier*. Penelitian mengenai *automatic respons* sudah pernah dilakukan sebelumnya oleh Shin dkk (2014) dan Sordoni dkk. (2015).

Shin dkk. (2014) mengekplorasi respons otomatis yang tergantung pada konteks dengan *Statistic Machine Translation*. Pun mencoba menambahkan *semantic* pada ucapan-ucapan sebelumnya. Hasil ekperimen menunjukkan model *context-dependent* dapat mengungguli baseline model. Akan tetapi jika dibandingkan dengan respons yang sebenarnya kinerjanya lebih buruk.

Sordoni dkk. (2015) mengusulkan arsitektur *Neural Network* untuk sistem *response generation* percakapan media sosial Twitter. Sistem yang dibangun *dikondisikan* pada percakapan sebelumnya yang memberikan informasi kontekstual (*context-sensitive*). Hasil penelitian menunjukkan bahwa Model *context-sensitive* menggungguli *context-independent*.

Dalam penelitian ini juga diteliti mengenai klasifikasi teks. Klasifikasi teks dalam penelitian ini merupakan bagian dari pada sistem yang akan merespons *feedback* tweet. Penelitian mengenai klasifikasi teks pernah dilakukan oleh beberapa peneliti seperti Liliana dkk.(2011), Darujati (2010), Suruliandi dan Selvaperumal (2014), Dilrukshi dkk. (2013), Rodiansyah (2012), Yusuf dan Priambadha (2013), Ling dkk. (2014), Lestari dkk. (2013), Wibowo (2015) dan Adi (2014), Widjojo dkk. (2014) dan Lumbanraja (2013).

Liliana dkk. (2011) mengklasifikasikan berita berdasarkan kategori yang tepat sehingga dapat memudahkan pengguna untuk menemukan berita yang relevan dengan cepat. Penelitian ini menggunakan *Support Vector Machine* (SVM) untuk mengklasifikasikan berita Indonesia. Dokumen yang digunakan adalah dokumen

dari situs berita digital indonesia, yaitu www.kompas.com. Data yang diambil dari web kompas kemudian dikonversi ke dalam file dengan ekstensi (txt). Data yang digunakan terdiri dari 180 berita, dimana data ini akan dibagi menjadi dua bagian. Bagian pertama sebanyak 70% digunakan untuk data *training*. Dan sisanya sebanyak 30% untuk data *testing*. Hasil penelitian menunjukkan hasil yang menjanjikan dengan tingkat rata-rarta akurasi 85%, dimana sistem yang dibuat layak untuk pengklasifikasian berita Indonesia.

Darujati (2010) melakukan penelitian tentang klasifikasi dokumen teks. Dokumen teks yang digunakan adalah kumpulan artikel dari majalah CHIP dan dibagi menurut kelas-kelasnya. Kelas-kelas artikel disesuaikan dengan pengkategorian artikel di dalam majalah CHIP. Adapun kelas dalam majalah CHIP antara lain : *game, hardware software, news dan tip trik.* Penelitian ini mencoba menggunakan stopword removal dan tanpa menggunakan stopword removal dalam *preprocessing*. Algoritma klasifikasi yang digunakan yaitu KNN dan *Naïve Bayes Classifier*. Pada penelitian mencoba mebandingkan penggunaan *stopword removal* dan tanpa penggunaan *stopword removal*. Hasil eksperiman menunjukkan bahwa akurasi *Naïve Bayes Classifier* tanpa *stopword removal* sebesar 87.45%, sedangkan akurasi *Naïve Bayes Classifier* dengan *stopword removal* sebesar 74,2%. Untuk metode *K-Nearest Neighbor* dengan *stopword removal* memiliki akurasi sebesar 41,55%, sedangkan *K-Nearest Neighbor* tanpa *stopword removal* memiliki akurasi sebesar 41,67%. Terlihat bahwa kinerja terbaik diperoleh oleh Algoritma *Naïve Bayes*.

Suruliandi dan Selvaperumal (2014) melakukan penelitian tentang Twitter *topic classification*, yaitu mengklasifikasikan tweet ke dalam beberapa kelas. Kelas tersebut antara lain : *news, sport, politic, entertainment, education*. Mereka mengusulkan metode baru untuk melakukan klasifikasi menggunakan fitur tweet seperti URL, *Retweet* tweet dan *influential user*. Metode tersebut disebut sebagai *classifier using* tweet *feature*. Tidak hanya satu metode saja yang digunakaan dalam penelitian ini, tetapi juga menggunakan metode lain seperti SVM, *Naïve Bayes*, KNN, *Decission Tree*, Rbf *Network*, K *Means*. Kinerja dari seluruh algoritma

tersebut kemudian dibandingkan. Hasil penelitian menunjukkan bahwa metode yang diusulkan lebih unggul daripada metode klasifikasi text konvensional.

Dilrukshi dkk. (2013) melakukan klasifikasi berita yang diambil dari salah satu microblog popular yaitu Twitter. Tweet berita diklasifikasikan ke dalam 12 kelompok yaitu *war-terrorist-crime, economy business, health, sports, development-government, politics, accident, entertain, disaster-climate, education, society and international*. Setiap pesan singkat dalam hal ini tweet diklasifikasikan secara manual, yang kemudian akan digunakan sebagai data training. Teknik yang digunakan untuk melakukan klasifikasi adalah *Support Vector Machine* (SVM). Hasil penelitian menunujukkan bahwa akurasi SVM memberikan hasil terbaik untuk *entertain, health, education.* Sedangkan untuk kategori lain akurasinya mencapai lebvih dari 75%, kecuali untuk kategori *development-government.*

Yusuf dan Priambadha (2013) melakukan penelitian tentang klasifikasi dokumen. Penelitian ini mengusulkan sebuah metode baru untuk klasifikasi dokumen teks berbahasa Inggris dengan terlebih dahulu melakukan pengelompokan menggunakan K-Means Clustering kemudian dokumen diklasifikasikan menggunakan *multi-class Support Vector Machines*. Dengan adanya pengelompokan tersebut, variasi data dalam membentuk model klasifikasi akan lebih seragam. Data yang digunakan sebanyak 540 artikel, 240 artikel untuk data latih dan 300 artikel untuk data uji. Hasil uji coba terhadap judul artikel jurnal ilmiah menunjukkan bahwa metode yang diusulkan mampu meningkatkan akurasi dengan menghasilkan akurasi sebesar 88,1%.

Rodiansyah (2012) menggunakan metode *Naïve Bayes Classifier* untuk mengklasifikasikan *posting* tweet yang digunakan untuk mengetahui kemacetan di kota Bandung. *Posting* tweet diklasifikasikan ke dalam kelas macet dan lancar. Data tweet diambil dari server Twitter dengan memanfaatkan API Twitter. Data tweet yang diambil disimpan di database. Data tweet kemudian melalui tahapan *preprocessing*. Setelah itu dilakukan pembobotan menggunakan *term frequency* dan TF-IDF. Kemudian dari pembobotan tersebut dilakukan klasifikasi menggunakan *Naive Bayes Classifier*. Hasil uji coba menunjukkan bahwa nilai akurasi terkecil 78% dihasilkan pada pengujian dengan sampel sebanyak 100 dan

menghasilkan nilai akurasi tinggi 91,60% pada pengujian dengan sampel sebanyak 13106. Sedangkan hasil pengujian dengan perangkat lunak Rapid Miner 5.1 diperoleh nilai akurasi terkecil 72% dengan sampel sebanyak 100 dan nilai akurasi tertinggi 93,58% dengan sampel 13106 untuk metode *Naive Bayesian Classification*.

Ling dkk. (2014) mengimplementasikan teknik *machine learning* yaitu metode *Naïve Bayes Classifier* untuk melakukan klasifikasi sentimen. Sebelum proses pengklasifikasian terlebih dahulu akan dilakukan *preprocessing*. Tahapan *preprocessing* dalam penelitian ini terdiri dari proses *tokenization, stemming, stopword*. Kemudian dilakukan seleksi fitur dengan metode chi square. Tujuan dilakukannya seleksi fitur untuk mereduksi *noise* dalam klasifikasi oleh NBC. Data yang digunakan dalam penelitian ini merupakan opini berbahasa inggris dari pengguna telepon genggam. Jumlah data yang digunakan terbagi atas 100 buah opini positif dan 100 buah opini negative. Sedangkan data training menggunakan 50 buah opini positif dan 50 buah opini negatif. Sisanya digunakan sebagai data uji. Hasil penelitian menunjukkan bahwa kemunculan frekuensi fitur pada kategori yang diharapkan dan tidak diharapkan memiliki peranan penting dalam seleksi fitur. Untuk analisis sentimen menggunakan metode NBC dalam penelitian ini menghasilkan akurasi sebesar 83%.

Lestari dkk. (2013) mencoba melakukan klasifikasi jenis kepribadian (*sanguine, choleric, melancholic, phlegmatic*) pengguna dengan menggunakan metode *Naïve Bayes Classifier*. Dalam penelitian ini, beberapa data kepribadian pengguna digunakan sebagai dokumen pelatihan dalam proses pelatihan metode *Naïve Bayes*. Data kepribadian seseorang yang diklasifikasi dapat diinputkan secara langsung dalam sistem atau import file (txt). Hasil penelitian menunjukkan bahwa NBC berhasil melakukan klasifikasi jenis kepribadian dengan nilai akurasi sebesar 92,5%.

Wibowo (2015) melakukan penelitian tentang klasifikasi kinerja satpam ke dalam tiga kelas antara lain kinerja baik, cukup, dan buruk. Penentuan kinerja satpam dilihat dari kemampuan, kepribadian dan ketrampilan satpam. Untuk mendapat keseluruhan nilai tersebut dilakukan tes berbasis computer. Data sampel

yang digunakan sebanyak 136 data dan sebanyak 39 data untuk digunakan sebagai data uji. Metode klasifikasi yang digunakan adalah *Naïve Bayes Classifier* dengan menggunakan perhitungan numeric tiga variabel. Hasil penelitian menunjukkan bahwa akurasi NBC sebesar 92,31%.

Adi (2014) menerapkan teknik klasifikasi *Naïve Bayesian* untuk mengklasifikasi calon debitur ke dalam kelas aman atau tidak aman.. Hasil peneletian menunjukkan bahwa akurasi model yang dihasilkan sebesar 80% dengan data sampel sebanyak 100 dan menghasilkan akurasi tertinggi sebesar 98.66% untuk sampel sebanyak 463.

Lumbanraja (2013) menggunakan teknik klasifikasi Rocchio pada domain sistem pencarian data teks. Hasil pengujian sistem klasifikasi yang dibangun pada penelitian ini menunjukkan nilai akurasi pada sistem cukup baik yaitu 76.67%.

Widjojo dkk. (2014) melakukan implementasi klasifikasi Rocchio pada domain renungan harian kristen. Penelitian ini meliputi perhitungan akurasi dengan nilai terbaik 73,33%. Penelitian ini juga menyimpulkan bahwa klasifikasi Rocchio yang diaplikasikan pada sistem menunjukkan performansi yang baik.

Penelitian ini mengusulkan respons tweet otomatis yang dilakukan oleh sistem. Data yang akan digunakan adalah tweet berupa *feedback* pelanggan Express group yang dikirim melalui media sosial Twitter. Data tweet yang diterima oleh Express group berbagai macam kelas. Antar tweet ada kemungkinan termasuk dalam kelas yang sama. Respons yang diberikan untuk setiap tweet berdasarkan pada kelas tweet. Oleh karena itu tweet akan dilakukan klasifikasi terlebih dahulu. Tweet diklasifikasikan ke dalam beberapa kelas antara lain pujian, keluhan, follow atau unknown. Kamudian untuk kelas keluhan akan dilakukan klasifikasi tahap kedua, untuk menentukan apakah *feedback* tweet tersebut termasuk dalam kelas keluhan sopir ataukah keluhan umum. Dalam penelitian ini akan menerapkan kombinasi NBC dan *Rocchio Classifier* sebagai algoritma *classifier*. Setelah didapatkan kelas dari masing-masing tweet kemudian sistem merespons tweet kepada pelanggan yang mengirimkan tweet. Seperti yang telah dijelaskan sebelumnya bahwa respons yang diberikan berdasarkan pada klasifikasi tweet. Sehingga akan dirancang *rule* untuk penentuan respons tweet yang dikirimkan ke

pelanggan. *Rule* yang digunakan terdapat pada bahasan respons *feedback* tweet. Tabel 2.1 merupakan penelitian yang sudah dilakukan sebelumnya yang dijadikan tinjauan pustaka dalam penelitian ini dan penelitian yang akan dilakukan.

Tabel 2.1 Tinjauan pustaka

No.	Penulis	Keterangan	Metode
1.	Darujati (2010)	Klasifikasi dokumen teks dari artikel majalah CHIP kedalam kelas *game, hardware software, news atau* tip trik. Hasil eksperimen menunjukkan bahwa kinerja *Naïve Bayes* lebih baik dari pada KNN.	KNN *dan Naïve Bayes Classifier*
2.	Liliana dkk. (2011)	Melakukan penelitian klasifikasi berita digital dari situs kompas menggunakan *Support Vector Machine*. Data berita dari situs kompas kemudian dikonversi ke dalam file txt. Hasil penelitian menunjukkan bahwa rata-rata akurasi mencapai 85%.	*Support Vector Machine*
3.	Rodiansyah (2012)	.Hasil uji coba menunjukkan bahwa nilai akurasi terkecil 78% dihasilkan pada pengujian dengan sampel sebanyak 100 dan menghasilkan nilai akurasi tinggi 91,60% pada pengujian dengan sampel sebanyak 13106.	*Naïve Bayes Classifier*
4.	Dilrukshi dkk. (2013)	Hasil penelitian menunujukkan bahwa akurasi SVM memberikan hasil terbaik untuk *entertain, health, education.* Sedangkan untuk kategori lain akurasinya mencapai lebih dari 75%, kecuali untuk kategori *development-government.*	*Support Vector machine*
5.	Lestari dkk. (2013)	Hasil penelitian menunjukkan bahwa NBC berhasil melakukan klasifikasi jenis kepribadian dengan nilai akurasi sebesar 92,5%.	*Naïve Bayes Classifier*
6.	Yusuf dan Priambadha (2013)	Dari 240 artikel data latih dan 300 artikel data uji *Multi-Class Support Vector Machines* (SVM) menunjukkan akurasi sebesar 88,1%.	Multi-class SVM yang didukung dengan K-Means Clustering
7.	Lumbanraja (2013)	Melakukan penelitian klasifikasi dokumen teks skripsi. Hasil penelitian menunjukkan bahwa akurasi *Rocchio Classifier* mencapai 76,67%	*Rocchio Classifier*
8.	Adi (2014)	Hasil peneltian menunjukkan bahwa akurasi model yang dihasilkan sebesar 80% dengan data sampel sebanyak 100 dan menghasilkan akurasi tertinggi sebesar 98,66% sebesar 98.66% untuk sampel sebanyak 463.	*Naïve Bayes Classifier*
9.	Shin dkk. (2014)	Hasil ekperimen menunjukkan model *context-dependent* dapat mengungguli baseline model. Akan tetapi jika dibandingkan dengan respons yang sebenarnya kinerjanya lebih buruk.	*Statistic Machine Translation*
10.	Widjojo dkk. (2014)	Klasifikasi renungan harian kristen. Hasil penenlitian menunjukan *Rocchio Classifier* memiliki akurasi sebesar 73.33%.	*Rocchio Classifier*

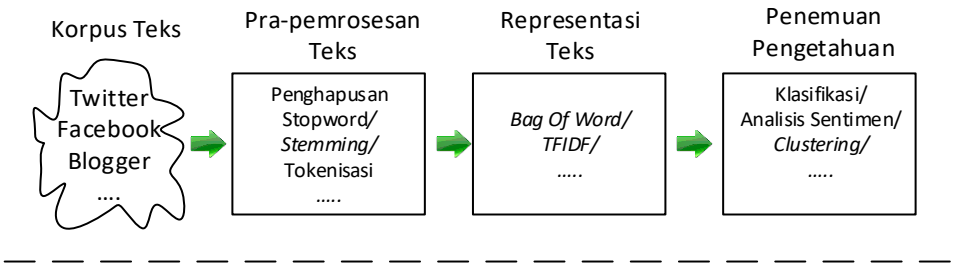
Tabel 2.1. Lanjutan

No.	Penulis	Keterangan	Metode
11.	Suruliandi dan Selvaperum al (2014)	Hasil penelitian menunjukkan bahwa metode yang diusulkan lebih baik. Dalam penelitian ini tidak disertai deteksi tweet *spam*.	*Classfier using* tweet *feature* dibandingkan dengan SVM, *Naïve Bayes*, KNN, *Decission Tree*, Rbf *Network, K Means*
12.	Sordoni dkk. (2015)	Hasil uji coba menunjukkan bahwa respons yang dihasilkan masuk akal, akan tetapi respons cenderung umum. Model *context-sensitive* menggungguli *context-independent*.	*Neural Network*
13.	Wibowo (2015)	Penilaian kinerja satpam dilihat dari test kemampuan, kepribadian, dan ktrampilan yang dimiliki satpam. Hasil penelitian menunjukkan bahwa klasifikasi kinerja satpam menghasilkan akurasi sebesar 92,31%.	*Naïve Bayes Classifier*
14.	Khamidah (2016)	Sistem Respons tweet mengimplimentasikan teknik klasifikasi kombinasi metode NBC dan *Rocchio*. Respons mengacu pada standar respons yang selama ini digunakan oleh Express Group.	*Naïve Bayes Classifier, Rocchio Classifier, Rule Respon*

BAB III
LANDASAN TEORI

3.1 Text Mining

Text mining merupakan proses untuk menemukan pola yang menarik dari koleksi dokumen (Feldman dan Sanger, 2007). Hu dan Liu (2012) menyatakan bahwa *text mining* mengacu pada penemuan pengetahuan yang dapat ditemukan dalam dokumen teks. Pada umumnya *text mining* terdiri dari *text preprocessing*, *text representation*, *knowledge discovery* seperti yang ditunjukkan pada 15Gambar 3.1.



Gambar 3.1 Proses dalam *text mining* (Hu dan Liu, 2012)

Gambar 3.1 menggambarkan proses umum dalam *text mining*. Proses – proses tersebut antara lain :

1. Pra-pemrosesan Teks (*Text preprocessing*)

   Laksana dan purwarianti (2014) menyatakan bahwa *preprocessing* dilakukan untuk menghilangkan *noise* pada teks dan memperbaiki ejaan kata. Hu dan liu (2012) menyatakan bahwa *text preprocessing* bertujuan untuk membuat dokumen input lebih konsisten untuk mempermudah *text representation*, yang diperlukan bagi sebagian besar tugas *text mining*. Metode tradisional *text preprocessing* adalah *stopword removal* dan *stemming*. *Stopword removal* menghapus kata yang termasuk dalam *stopword list*, yang mana kata-kata tersebut dianggap umum dan tak berarti. *Stemming* adalah mereduksi kata berimbuhan ke dalam kata dasar. *Preprocessing* dilakukan untuk menghindari

data yang kurang sempurna, gangguan pada data, dan data-data yang tidak konsisten (Hemalatha dkk, 2012).

2. Representasi Teks *(Text representation)*

   Cara umum yang digunakan untuk merepresentasikan dokumen adalah mentransformasikan dokumen menjadi *vector numeric*. Representasi seperti itu disebut dengan *bag of word*. *Text representation* dalam model *bag of word* yaitu kata direpresentasikan sebagai variabel terpisah yang memiliki bobot kepentingan yang berbeda-beda.

3. Penemuan Pengetahuan *(Knowledge discovery)*

   Ketika berhasil dalam *text representation*, selanjutnya dapat menerapkan *machine learning* atau metode data mining yang ada seperti klasifikasi atau *clustering*.

3.2 Teorema Bayes

Teorema Bayes memiliki bentuk umum pada persamaan (3.1).

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \tag{3.1}$$

Dalam hal ini X merupakan data dengan kelas yang belum diketahui. H adalah hipotesis data X merupakan suatu kelas spesifik. $P(H|X)$ adalah probabilitas hipotesis H berdasar kondisi X (*posteriori probability*). $P(H)$ adalah probabilitas hipotesis H (*prior probability*). $P(X|H)$ adalah probabilitas X berdasar kondisi pada hipotesis H. $P(X)$ adalah probabilitas dari X.

3.3 Teknik Klasifikasi dan *Naïve Bayes Classifier* (NBC)

Klasifikasi merupakan bagian dari *supervised learning* yang digunakan untuk menentukan kelas sebuah objek menggunakan data *training* sebagai acuan untuk proses *learning*, data *test* untuk menguji kemampuan hasil *learning* untuk menentukan prediksi kelas (Learned dan Miller, 2011). Han dan Kamber (2006) menyatakan bahwa klasifikasi data memiliki dua tahap proses. Tahap pertama disebut tahap pembelajaran *(training phase)* dimana tahap ini membangun suatu model atau *classifier*. Algoritma klasifikasi akan membangun sebuah model

klasifikasi atau *classifier* dengan cara menganalisis data *training* yang terdiri dari beberapa *sample* yang disertai dengan label kelas. Karena adanya kelas label dalam setiap *sample* maka disebut sebagai *supervised learning.* Tahap kedua adalah tahap klasifikasi, model yang telah dihasilkan akan digunakan untuk menentukan kelas suatu data.

Salah satu metode klasifikasi yang dapat digunakan adalah metode *Naive Bayes* yang sering disebut sebagai *Naive Bayes Classifier* (NBC). *Naïve Bayes Classifier* merupakan metode pembelajaran probabilitas berdasarkan pada teorema Bayes (Thakur dan Mann, 2014). Ciri utama *Naïve Bayes Classifier* adalah asumsi independensi masing-masing fitur dalam dataset (Rasschka, 2014). Secara sederhana, NBC mengasumsikan bahwa ada atau tidaknya fitur tertentu dari suatu kelas tidak memiliki keterkaitan dengan keberadaan fitur lainnya. Han dan kamber (2006) menyatakan bahwa *Naïve Bayes* mempunyai tingkat kecepatan dan akurasi yang tinggi.

Manning dkk. (2009) menyatakan bahwa probabilitas dokumen d berada dalam kelas c dapat dihitung dengan menggunakan persamaan (3.2).

$$P(c|d) = P(c) \prod_{1 \le k \le n_d} P(t_k|c) \tag{3.2}$$

Dalam persamaan (3.2) $P(t_k|c)$ adalah probabilitas bersyarat kemunculan kata t_k dalam dokumen pada kelas c. Dengan kata lain $P(t_k|c)$ sebagai ukuran seberapa besar kontribusi t_k bahwa c adalah kelas yang benar. $P(c)$ adalah probabilitas *prior* kemunculan dokumen pada kelas c. $\langle t_1, t_2, \dots, t_{n_d} \rangle$ adalah token-token dalam dokumen d dan n_d adalah jumlah token dalam dokumen d. Han dan Kamber (2006) menyatakan bahwa *Naïve Bayes* mengasumsikan token-token $\langle t_1, t_2, \dots, t_{n_d} \rangle$ bersifat independen antara token yang satu dengan yang lain.

Manning dkk. (2009) menyatakan bahwa klasifikasi teks bertujuan untuk menemukan kelas terbaik suatu dokumen. Kelas terbaik dalam *Naïve Bayes Classification* adalah *maximum a posteriori* (MAP) *class* c_{map} yang dituliskan dalam persamaan (3.3).

$$c_{map} = \arg\max_{c \in \mathbb{C}} P(c|d) = \arg\max_{c \in \mathbb{C}} P(c) \prod_{1 \le k \le n_d} P(t_k|c) \quad (3.3)$$

Nilai $P(c)$ dan $P(t_k|c)$ dapat diestimasi berdasarkan data *training*. Untuk mengestimasi probabilitas *prior* menggunakan persamaan (3.4) :

$$P(c) = \frac{N_c}{N} \quad (3.4)$$

dimana N_c adalah jumlah dokumen pada kelas c dan N adalah jumlah total dokumen yang digunakan dalam proses *training*.

Untuk mengestimasi probabilitas kondisional $P(t_k|c)$ menggunakan persamaan (3.5).

$$P(t_k|c) = \frac{T_{ct} + 1}{(\sum_{t' \epsilon V} T_{ct'}) + B} \quad (3.5)$$

T_{ct} adalah jumlah kemunculan kata t_k dalam dokumen *training* pada kelas *c*, termasuk berapa kali kemunculan suatu kata t_k dalam dokumen. $T_{ct'}$ adalah jumlah kemunculan semua kata dalam dokumen *training* yang berada pada kelas c. t' adalah semua kata yang muncul pada kelas c dalam data *training*. B merupakan jumlah semua kata dalam data *training.*

3.4 TF-IDF

Manning dkk. (2009) menyatakan bahwa TF-IDF sebagai skema pembobotan yang memberikan kata t_k suatu bobot dalam dokumen *d*. TF-IDF merupakan kombinasi dari *term frequency* dan *inverse document frequency*. Persamaan (3.6) merupakan persamaan untuk menghitung nilai TF-IDF :

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \quad (3.6)$$

Nilai $tf_{t,d}$ merupakan jumlah kemunculan kata t pada dokumen *d* sedangkan idf_t adalah *inverse document frequency* dari kata t. Persamaan (3.7) adalah persamaan untuk mencari nilai idf_t.

$$idf_{t,d} = \log \frac{N}{df_t} \quad (3.7)$$

Nilai idf_t diperoleh dari hasil logaritma *N* dibagi dengan df_t. *N* merupakan jumlah dokumen keseluruhan, sedangkan df_t adalah banyaknya dokumen yang memuat kata t.

3.5 Rocchio Classifier

Pada algoritma ini *data training* dan *data testing* direpresentasikan sebagai *vector*. Pada *Rocchio Classifier* dihitung batasan kelas dengan menggunakan *centroid* yang diformulasikan pada persamaan (3.8).

$$\vec{\mu}(c) = \frac{1}{|D_c|} \sum_{d \epsilon D_c} \vec{v}\,(d) \tag{3.8}$$

Dimana D_c adalah kumpulan dokumen *training* yang merupakan anggota kelas c. $\vec{v}(d)$ merupakan normalisasi *vector* yang dihitung menggunakan persamaan (3.9).

$$\vec{v}(d) = \frac{\vec{V}(d)}{\left|\vec{V}(d)\right|} \tag{3.9}$$

$\vec{V}(d)$ merupakan *vector* dokumen dengan M komponen $\vec{V}_1(d) \ldots \vec{V}_M(d)$, satu komponen *vector* untuk setiap satu *term* (atau kata). $\left|\vec{V}(d)\right|$ merupakan panjang *vector* yang didefinisikan menjadi $\sqrt{\sum_{i=1}^{M} \vec{V}_i^2(d)}$. *Rocchio Classifier* menentukan kelas dengan jarak minimal centroid dengan *vector* dokumen yang akan diklasifikasi $\min |\vec{\mu}(c) - \vec{v}(d)|$. Manning dkk. (2009) menyatakan bahwa jarak tersebut dapat dihitung menggunakan *euclidean distance*. *Euclidean distance* ditunjukkan pada persamaan (3.10).

$$||\vec{\mu}(c) - \vec{v}(d)|| = \sqrt{\sum_{i=1}^{M} (\vec{\mu}_i(c) - \vec{v}_i(d))^2} \tag{3.10}$$

$||\vec{\mu}(c) - \vec{v}(d)||$ adalah jarak *euclidean* antara centroid kelas dengan *vector* dokumen yang akan diklasifikasi. $\vec{\mu}_i(c)$ merupakan komponen ke-i dari centroid kelas. $\vec{v}_i(d)$ merupakan komponen ke-i dari *vector* dokumen yang akan diklasifikasi. M adalah jumlah komponen pada *vector*.

3.6 Evaluasi Model Klasifikasi

Menurut sebastiani (2002) evaluasi *classifier* biasanya mengukur efektivitas, artinya kemampuan *classifier* untuk mengklasifikasikan dengan tepat. Evaluasi *classifier* dilakukan dengan menghitung *presicion*, *recall* dan *f-measure*.

Untuk menghitung *presisi*, *recall* dan *f-measure* menggunakan komponen pada Tabel 3.1.

Tabel 3. 1 Tabel evaluasi hasil klasifikasi

Actual class	*Classivied class*	
	Positive	*Negative*
Positive	TP (*True Positive*)	FN (*False Negative*)
Negative	FP (*False Positive*)	TN (*True Negative*)

Tabel 3.1 menyajikan evaluasi hasil klasifikasi dengan menggunakan dua kelas yang kemudian dapat dicari nilai *precision*, *recall,* dan *f-measure.*

1. *Precision*

   Precision adalah jumlah data yang *true positive* dibagi dengan jumlah data yang dikenali sebagai positive (Prasetyo, 2014), seperti yang ditunjukkan pada persamaan (3.11).

$$Precision = \frac{TP}{TP+FP} \tag{3.11}$$

2. *Recall*

   Recall berarti jumlah data yang *true positive* dibagi dengan jumlah data yang sebenarnya positive (Prasetyo, 2014), seperti yang ditunjukkan pada persamaan (3.12)

$$Recall = \frac{TP}{TP+FN} \tag{3.12}$$

3. *F-Measure*

   F-measure merupakan salah satu perhitungan evaluasi dalam temu kembali informasi yang mengkombinasikan *recall* dan *precision*. Nilai *recall* dan *precision* pada suatu keadaan dapat memiliki bobot yang berbeda. Ukuran yang menampilkan timbal balik antara *recall* dan *precision* adalah *F-measure* yang merupakan bobot *harmonic mean* dari *recall* dan *precision*.

$$F - measure = \frac{2\ x\ recall\ x\ presicion}{recall+presicion} \tag{3.13}$$

Menurut Widjojo (2014) komponen evaluasi dari klasifikasi didefinisikan sebagai berikut :

a. *True Positive* (TP) adalah jumlah *positive class* yang terklasifikasikan dengan benar oleh sistem klasifikasi .
b. *False Positive* (FP) adalah jumlah *negative class* yang terklasifikasikan sebagai *positive class* oleh sistem klasifikasi.
c. *False Negative* (FN) adalah jumlah *positive class* yang terklasifikasikan sebagai *negative class* oleh sistem.
d. *True Negeative* (TN) adalah jumlah *negative class* yang terklasifikasikan dengan benar oleh sistem klasifikasi.

Pengukuran efektivitas yang lain adalah akurasi. Akurasi *classifier* adalah presentase dari data uji yang diklasifikasikan benar oleh *classifier*. Akurasi *classifier* menurut Liu (2011) diformulasikan pada persamaan (3.14).

$$Accuracy = \frac{Jumlah\ klasifikasi\ benar}{Jumlah\ total\ data\ uji} \tag{3.14}$$

Klasifikasi benar berati bahwa model yang telah dilatih menentukan kelas yang sama dengan kelas asli pada data uji.

3.7 Twitter

Togias & Kameas (2012) menyatakan bahwa Twitter adalah *social network service* dan *microblogging sevice* yang memungkinkan *user* untuk mengirim dan membaca postingan berbasis teks sepanjang 140 karakter, atau yang biasa disebut dengan tweet. Hodeghatta (2013) menyatakan bahwa Twitter menghasilkan lebih dari 65 juta tweet setiap harinya, tweet-tweet tersebut disimpan pada Twitter corpus dan dapat diakses oleh public menggunakan *Application Programming Interface* (APIs). API merepresentasikan aspek Twitter, dan memungkinkan *developer* untuk membuat aplikasi menggunakan API. Twitter APIs secara konstan mengembangkan, dan membangun aplikasi pada platform Twitter bukanlah hal yang dilakukan sekali saja.

Togias & Kameas (2012) Twitter REST API mengikuti desain RESTful API, dalam arti aplikasi menggunakan metode standar HTTP untuk mengambil dan memanipulasi Twitter *resources*. Twitter menggunakan OAuth 2.0 untuk mengizinkan aplikasi yang terotorisasi untuk mengakses data *user*.. API

menyediakan metode HTTP GET dan POST untuk membaca, membuat, *update*, atau menghapus data.

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1 Analisis Kebutuhan

Berdasarkan permasalahan yang telah dirumuskan pada BAB I dan pemahaman terhadap teori sebagai dasar pemikiran yang dituliskan pada BAB III, maka pada bab ini merupakan konsep penyelesaian masalah dengan melakukan analisis dan perancangan sistem. Bab ini menyajikan data dan label data, tahapan analisis dan perancangan sistem. Sistem dibangun berdasarkan pada rancangan yang dihasilkan.

Sistem yang dirancang merupakan sistem yang dapat melakukan respons tweet secara otomatis. Tweet yang digunakan adalah *feedback* yang diterima oleh perusahaan taksi Express Group. Sistem terlebih dahulu melakukan *download* tweet. Data tweet kemudian melalui proses *preprocessing* untuk mendapatkan data bersih. Data bersih kemudian digunakan untuk menentukan tweet termasuk dalam kelas apa, oleh karena itu sistem respons tweet otomatis ini mengimplementasikan *classifier algorithm*. *Classifier algorithm* tersebut membangun model klasifikasi agar dapat menetukan kelas tweet. *Classifier* yang digunakan dalam penelitian ini adalah kombinasi antara *Naïve Bayes Classifier* dan *Rocchio Classifier*. Setelah kelas dari tweet diketahui kemudian dilakukan pengiriman respons secara otomatis kepada pengguna layanan.

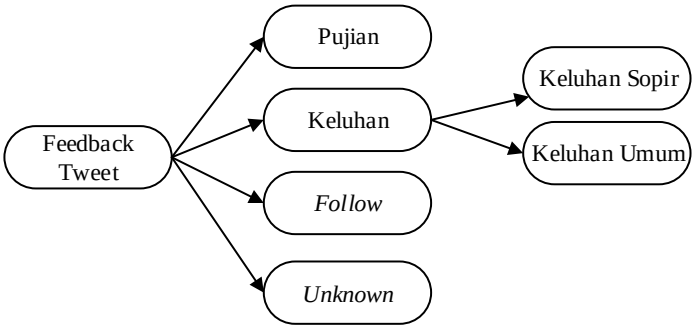
4.2 Data dan Label Data

Data yang digunakan dalam penelitian ini adalah data tweet taksi perusahaan Express Group. Setiap 6 menit sistem mengambil data tweet yang mengandung *mention username* @expressgroup. Adapun data yang diambil sebagai berikut.

1. *User*, pengguna Twitter yang mem-*posting* tweet.
2. Tweet, yakni informasi (atau *feedback*) yang dituliskan pengguna tweet mengenai layanan Express Group.

3. Tanggal dan waktu tweet, yakni menunjukkan kapan *feedback* diposting.

Dalam sistem ini data *feedback* terbagi menjadi 4 label kelas, yakni pujian, keluhan, *follow* dan *unknown*. Karena kelas keluhan memiliki 2 respons khusus, maka kelas keluhan secara khusus diklasifikasikan lagi ke dalam 2 kelas, yakni keluhan sopir dan keluhan umum. Gambar 4.1 menggambarkan label kelas yang akan digunakan dalam penelitian ini.

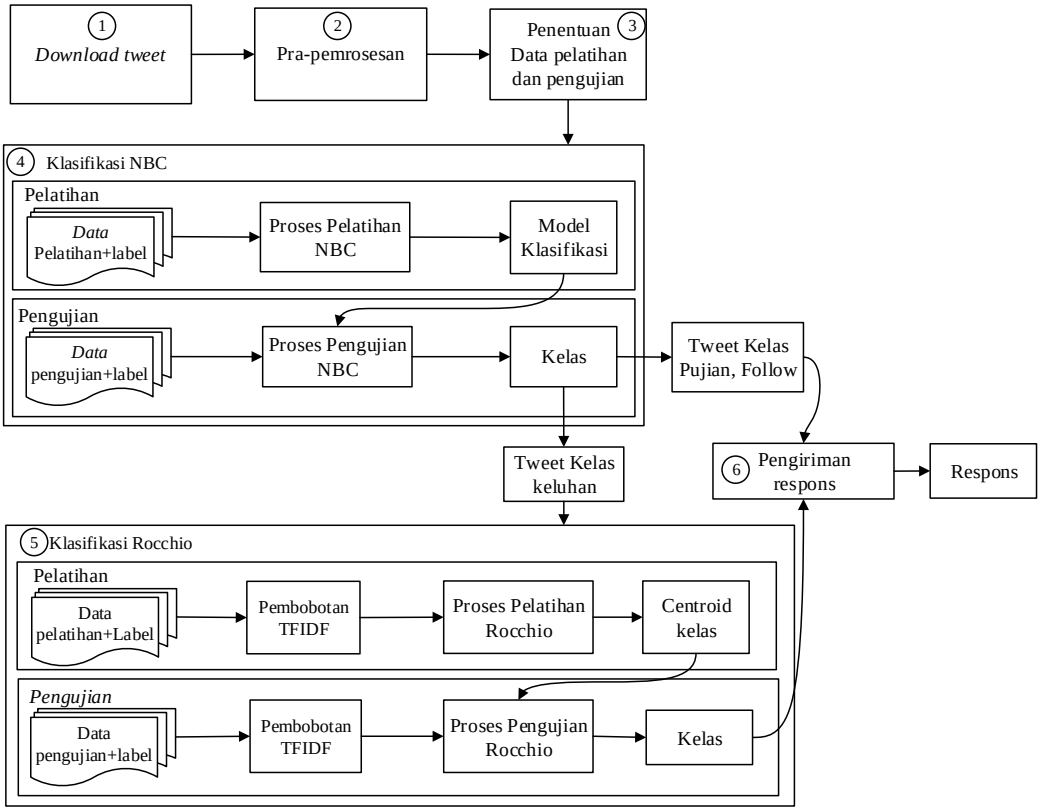


Gambar 4.1 Label kelas

4.3 Deskripsi Umum Sistem

Secara garis besar sistem yang akan dibangun bertujuan untuk merespons tweet yang diterima oleh Express Group. Respons yang diberikan merujuk pada standar respons yang selama ini digunakan oleh Express Group. Pemberian respons tergantung pada kelas dari masing-masing tweet. Antar tweet ada kemungkinan kesamaan masuk ke dalam kelas yang sama. Jadi, tweet akan diklasifikasikan terlebih dahulu, termasuk dalam kelas apakah *feedback* tweet tersebut, kelas pujian, keluhan, *follow*, atau *unknown*. Untuk mendapatkan kelas dari masing-masing tweet membutuhkan suatu algoritma *classifier*. Dalam penelitian ini akan mengimplementasikan kombinasi metode *Naïve Bayes Classifier* dan *Rocchio Classifier*. Jadi, penelitian ini dilakukan klasifikasi secara bertingkat. Klasifikasi pertama untuk mendapatkan apakah termasuk ke dalam kelas pujian, keluhan, *follow*, atau *unknown* dengan menggunakan NBC. Klasifikasi kedua untuk mendapatkan kelas keluhan secara lebih detail, apakah keluhan tersebut termasuk ke dalam kelas keluhan sopir atau keluhan umum dengan menggunakan *Rocchio Classifier*. Setelah didapatkan kelas dari *feedback* selanjutnya dilakukan pengiriman tweet respons *feedback* yang dikirim secara

otomatis oleh sistem. Gambaran Model sistem dalam penelitian ini ditunjukkan pada Gambar 4.2.



Gambar 4.2 Gambaran model sistem

Gambar 4.2 merupakan gambaran model sistem respons tweet yang dibangun pada penelitian ini. Sistem terdiri dari beberapa proses antara lain :

1. Proses *download* tweet adalah proses mengambil data *feedback* pelanggan dari Twitter.
2. Proses pra-pemrosesan (atau *preprocessing*) adalah proses untuk menghilangkan *noise* pada tweet.
3. Proses penentuan data pelatihan (atau *training*) dan data pengujian (atau *testing*) adalah proses menentukan *tweet* mana yang digunakan untuk proses *training* dan tweet mana yang digunakan untuk proses *testing*.
4. Proses klasifikasi NBC adalah proses untuk mengetahui tweet termasuk dalam kelas pujian, keluhan, follow atau unknown. Klasifikasi NBC terdiri dari 2 tahapan, yakni pelatihan NBC dan pengujian NBC.

a. Pada tahapan pelatihan NBC, data pelatihan yang disertai label melalui proses pelatihan menggunakan NBC. Proses pelatihan NBC menghasilkan model klasifikasi atau model probabilitas.

b. Pada tahapan pengujian, data pengujian (atau data testing) yang disetai dengan label manual kemudian dilakukan proses pengujian. Proses pengujian ini menghasilkan keluaran berupa kelas. Tweet yang dikenali sebagai kelas pujian dan follow selanjutnya melalui proses pengiriman respons. Sedangkan tweet yang dikenali sebagai kelas keluhan melalui proses klasifikasi lagi menggunakan metode *Rocchio Classifier*.

5. Proses klasifikasi *Rocchio* adalah proses untuk mengetahui tweet termasuk dalam kelas keluhan umum atau keluhan sopir. Klasifikasi *Rocchio* terdiri dari dua tahapan, yakni pelatihan *Rocchio* dan pengujian *Rocchio*

a. Pada tahapan pelatihan *Rocchio*, data pelatihan yang sudah disertai dengan label manual melalui proses pembobotan TFIDF dan proses pelatihan menggunakan *Rocchio*. Proses pelatihan *Rocchio* menghasilkan centroid kelas.

b. Pada tahapan pengujian Rocchio, data testing (atau pengujian) yang disertai label melalui proses pembobotan TFIDF dan proses pengujian *Rocchio*. Proses pengujian ini menghasilkan keluaran berupa kelas keluhan umum atau keluhan sopir.

6. Pengiriman tweet respons adalah proses pengiriman respons tweet pelanggan. Pengiriman tweet respons dilakukan setelah kelas tweet didapatkan. Kelas tweet menentukan tweet respons mana yang akan dikirimkan.

4.4 Rancangan *Download* Tweet

Data yang digunakan adalah tweet yang diterima oleh Express Group melalui *media social* Twitter. Sistem akan mengumpulkan data tweet yang mengandung *mention* @Express_Group. Proses *download* tweet memanfaatkan API Twitter. Hasil dari proses *download* tweet kemudian disimpan dalam database. Data tweet selanjutnya diolah pada proses *preprocessing.*

4.5 Rancangan Preprocessing

Preprocessing dilakukan untuk menghilangkan *noise* pada teks dan memperbaiki ejaan kata. Pada *preprocessing* terdapat beberapa tahapan yang dilakukan, tahapan *preprocessing* antara lain :

1. Menghapus URL yang terdapat dalam tweet.
2. Menghapus tanda baca dalam tweet, contohnya adalah titik, koma, tanda petik, tanda tanya, tanda seru, dll.
3. *Casefolding* adalah pengubahan semua huruf menjadi huruf kecil.
4. *Tokenization* adalah pemotongan string input berdasarkan tiap kata yang menyusunnya.
5. *Replacement* adalah penggantian kata tidak baku (*slang*) dan singkatan kata menjadi kata baku.
6. *Stopword removal* adalah menghapus kata yang merupakan *stopword list* (seperti “dan”, “yang”, “sebagai”, dll), yang mana kata-kata tersebut dianggap umum dan tak berarti.
7. *Stemming* adalah mereduksi kata imbuhan menjadi kata dasar atau akar kata. Dalam penelitian ini menggunakan stemmer sastrawi. Sastrawi adalah library PHP yang menyediakan stemming kata bahasa Indonesia.
8. Menggabungkan kata negasi, jika terdapat kata negasi maka kata negasi akan digabungkan dengan kata selanjutnya. Misal jika pada tweet terdapat kata “tidak sopan” maka kedua kata tersebut akan dianggap sebagai satu kata menjadi “tidaksopan”.

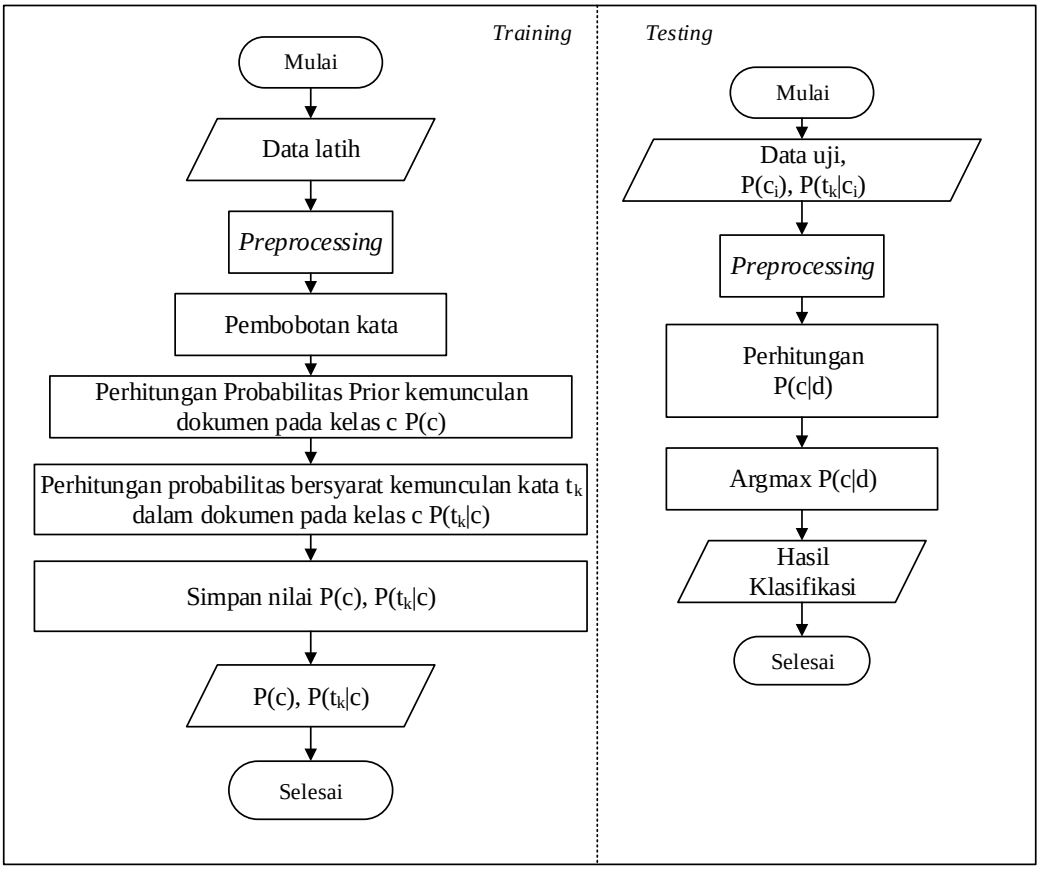
4.6 Rancangan Klasifikasi Tweet

Pada penelitian ini dilakukan dua tahapan klasifikasi. Klasifikasi tahap pertama menggunakan metode *Naïve Bayes Classifier* (NBC). Klasifikasi tahap kedua menggunakan metode *Rocchio Classifier*.

4.6.1 Klasifikasi dengan NBC

Klasifikasi menggunakan NBC dalam penelitian ini untuk mengenali apakah tweet termasuk dalam kelas pujian, keluhan, *follow* atau *unknown*.

Klasifikasi tweet menggunakan NBC ada tua tahapan yang akan dilakukan, yang pertama adalah *training* dan *testing*, seperti yang ditunjukkan pada Gambar 4.3.



Gambar 4.3 Klasifikasi tweet dengan NBC

Training NBC

Tahapan *training* NBC merupakan proses untuk membangun *classifier* (atau model probabilitas). Proses *training* dimulai dengan menginputkan data latih. Data latih yang digunakan disertai dengan label kelas pujian, keluhan, *follow* atau *unknown*. Data latih melalui proses *preprocessing*. Selanjutnya dilakukan pembobotan kata dengan menghitung kemunculan kata pada setiap kelas. Kemudian pada proses *training* dihitung nilai $P(c)$ probabilitas prior kemunculan tweet pada kelas c menggunakan persamaan (3.4) dan probabilitas bersyarat kemunculan kata t_k dalam tweet pada kelas c dengan menggunakan persamaan (3.5). Output dari tahapan *training* berupa nilai $P(c)$ dan $(t_k|c)$.

Testing NBC

Testing NBC adalah proses menentukan kelas dari suatu tweet. Proses *testing* NBC dimulai dengan menginputkan data uji, selain itu *input* yang digunakan adalah nilai $P(c)$ probabilitas prior kemunculan tweet pada kelas c dan nilai $P(t_k|c)$ probabilitas bersyarat kemunculan kata t_k dalam tweet pada kelas c. Data uji disertai dengan label kelas secara manual. Selanjutnya data uji melalui proses *preprocessing* untuk mendapatkan data bersih. Setelah *preprocessing* kemudian dilakukan perhitungan $P(c|d)$ probabilitas tweet berada pada kelas c yang dihitung menggunakan persamaan (3.2). Proses perhitungan $P(c|d)$ menggunakan *output* dari tahapan *training* yaitu $P(c)$ dan $P(t_k|c)$. Kemudian dilakukan proses pencarian argmax $P(c|d)$. *Output* dari proses ini berupa kelas yang memiliki *maximum a posteriori class*. Kelas yang dihasilkan oleh *classifier* kemudian dibandingkan dengan label (atau kelas) manual yang terdapat pada data uji. Jika kedua kelas tersebut sama, maka *classifier* menentukan kelas dengan benar.

Contoh perhitungan $P(c)$, $P(t_k|c)$ dan $P(c|d)$

Contoh perhitungan $P(c)$, $P(t_k|c)$ dan $P(c|d)$ menggunakan data pada Tabel 4.1. Sesuai dengan persamaan (3.4) $P(c)$ probabilitas prior kemunculan dokumen pada kelas $c$didapatkan dari perhitungan jumlah dokumen (atau tweet) dalam kelas c yang dinotasikan dengan N_c dibagi dengan jumlah tweet secara keseluruhan N. Berdasarkan data pada Tabel 4.1 dapat diketahui probabilitas prior setiap kelas sebagai berikut :

$$P(pujian) = \frac{Npujian}{N} = \frac{2}{5} = 0.4$$

$$P(keluhan) = \frac{N_{keluhan}}{N} = \frac{1}{5} = 0.2$$

$$P(follow) = \frac{N_{follow}}{N} = \frac{1}{5} = 0.2$$

$$P(unknown) = \frac{N_{unknown}}{N} = \frac{1}{5} = 0.2$$

Tabel 4.1 Contoh data tweet bersih

No	Data Set	Tweet Bersih	Label kelas
1.	Data training	sopir tidaksopan	keluhan
2.		banyak taksi tunggu	*unknown*
3.		sopir sopan	pujian
4.		nyaman sopir ramah	pujian
5.		admin tolong follow	*follow*
7	Data baru	sopir sopan nyaman	?

Tabel 4.2 Probabilitas bersyarat dari kata t yang terjadi dalam kelas c_i

Kata t	$P(t_k\|c_i)$			
	pujian	keluhan	*follow*	*unknown*
admin	0.06	0.08	0.14	0.07
banyak	0.06	0.08	0.07	0.14
follow	0.06	0.08	0.14	0.07
nyaman	0.13	0.08	0.07	0.07
ramah	0.13	0.08	0.07	0.07
sopan	0.13	0.08	0.07	0.07
sopir	0.19	0.15	0.07	0.07
taksi	0.06	0.08	0.07	0.14
tidaksopan	0.06	0.15	0.07	0.07
tolong	0.06	0.08	0.14	0.07
tunggu	0.06	0.08	0.07	0.14

Perhitungan $P(t_k|c)$ menggunakan persamaan (3.5). Nilai probabilitas bersyarat kemunculan kata t_k dalam tweet pada kelas c ditunjukkan pada Tabel 4.2. Contoh perhitungan probabilitas bersyarat dari kata ramah yang terjadi dalam setiap kelas sebagai berikut .

$$P(t_k|c) = \frac{T_{ct}+1}{(\sum_{t' \epsilon V} T_{ct'})+B}$$

$$P(ramah \mid pujian) = \frac{1+1}{5+11} = 0.13$$

$$P(ramah \mid keluhan) = \frac{0+1}{2+11} = 0.08$$

$$P(ramah \mid follow) = \frac{0+1}{3+11} = 0.07$$

$$P(ramah \mid unknwon) = \frac{0+1}{3+11} = 0.07$$

Perhitungan nilai $P(c|d)$ probabilitas tweet berada dalam kelas c membutuhkan nilai probabilitas prior kelas c dan probabilitas bersyarat kemunculan kata t_k dalam tweet pada kelas c. Nilai-nilai tersebut diestimasi pada proses *training*. Contoh perhitungan probabilitas tweet "sopir sopan nyaman" berada dalam kelas pujian.

$$P(pujian|d) = P(pujian) \prod_{1 \le k \le n_d} P(t_k|pujian)$$
$$= 0.4 \cdot (0.19 \cdot 0.13 \cdot 0.13)$$
$$= 0.4 \cdot 0.000910332 = 0.0011718750$$

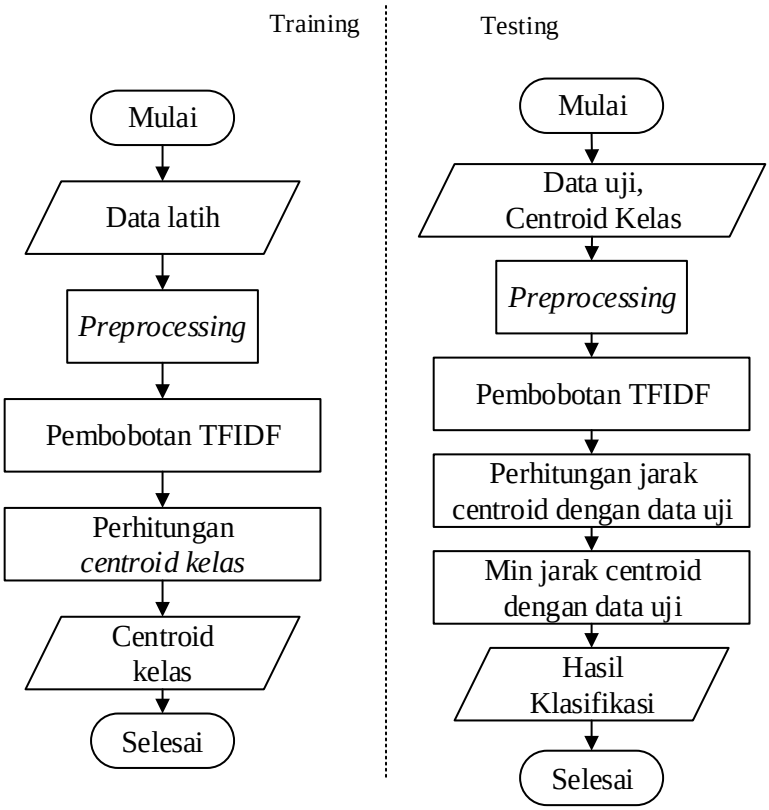
Tabel 4.3 Probabilitas tweet berada dalam kelas *c*

Kata t	$P(c)$	$P(t_k\|c)$			$P(c\|d)$
		sopir	sopan	nyaman	
Pujian	0.5	0.19	0.13	0.13	0.0011718750
keluhan	0.16	0.15	0.08	0.08	0.0001820665
follow	0.16	0.07	0.07	0.07	0.0000728863
unknown	0.16	0.07	0.07	0.07	0.0000728863

Probabilitas tweet "sopir sopan nyaman" berada dalam kelas c ditunjukkan pada Tabel 4.3. Berdasarkan Tabel 4.3, yang memiliki nilai argmax adalah probabilitas posterior kelas pujian. Oleh karena itu dapat ditarik kesimpulan bahwa data "sopir sopan nyaman" termasuk kedalam kelas pujian.

4.6.2 Klasifikasi dengan *Rocchio Classifier*

Klasifikasi dengan *Rocchio Classifier* digunakan untuk menentukan kelas tweet apakah termasuk dalam kelas keluhan sopir atau keluhan umum. Klasifikasi dengan *Rocchio Classifier* hanya dilakukan untuk kelas keluhan yang dikenali oleh NBC. Seperti halnya NBC, klasifikasi tweet menggunakan *Rocchio Classifier* pun terdiri dari dua tahapan, yaitu tahapan *training* dan *testing*. Gambar 4.4 menggambarkan rancangan klasifikasi tweet menggunakan *Rocchio Classifier*.



Gambar 4.4 Rancangan klasifikasi tweet dengan *Rocchio Classifier*

Tahapan *training* dan tahapan *testing* menggunakan *input* yang berbeda, pada tahap *training* data yang digunakan sebagai *input* adalah data latih, sedangkan pada tahap *testing* data *input* berupa data uji. Kemudian data-data tersebut melalui proses *preprocessing* terlebih dahulu sebelum dilakukannya proses pembobotan kata menggunakan TFIDF. Proses pembobotan, tahapan *training* dan tahapan *testing* menggunakaan *Rocchio* akan dijelaskan lebih lanjut pada bagian selanjutnya.

Pembobotan TFIDF

Proses TFIDF sebagai skema pembobotan yang memberikan kata t suatu bobot dalam dokumen d. TFIDF merupakan kombinasi dari *term frequency* dan *inverse document frequency*. Bobot kata yang diperoleh kemudian diolah dalam proses *training* metode Rocchio sebagai *vector* dokumen. Contoh perhitungan TFIDF pada Tabel 4.5 menggunakan data pada Tabel 4.4. Contoh tfidf menggunakan formula TFIDF pada persamaan (3.6).

Tabel 4.4 Contoh data tweet

Data set	No. dokumen (atau tweet)	Kata dalam tweet	Kelas keluhan
Training set	d1	sopir bicara tidaksopan balap	sopir
	d2	sopir balap tidaknyaman	sopir
	d3	sopir balap marah marah lempar dompet tidaksopan	sopir
	d4	AC tidakrasa	umum
	d5	taksi tidakrawat	umum
Data baru	d6	balap tidaknyaman sopir tidaksopan	?

Tabel 4.5 Perhitungan tfidf

Kata	tf						Idf	tfidf					
	Kel.sopir			kel.umum		?		Kel.sopir			Kel.umum		?
	d1	d2	d3	d4	d5	d6		d1	d2	d3	d4	d5	d6
ac				1			0.7	0	0	0	0.7	0	0
balap	1	1	1			1	0.2	0.2	0.2	0.2	0	0	0.2
bicara	1						0.7	0.7	0	0	0	0	0
dompet			1				0.7	0	0	0.7	0	0	0
lempar			1				0.7	0	0	0.7	0	0	0
marah			2				0.7	0	0	1.4	0	0	0
sopir	1	1	1			1	0.2	0.2	0.2	0.2	0	0	0.2
taksi					1		0.7	0	0	0	0	0.7	0
tidaknyaman		1				1	0.7	0	0.7	0	0	0	0.7
tidakrasa				1			0.7	0	0	0	0.7	0	0
tidakrawat					1		0.7	0	0	0	0	0.7	0
tidaksopan	1		1			1	0.4	0.4	0	0.4	0	0	0.4

Contoh perhitungan TFIDF

Contoh perhitungan tfidf untuk kata ac pada dokumen satu (d4) dan tfidf kata balap pada dokumen dua (d3) sebagai berikut :

1. TFIDF kata ac pada dokumen 4 (d4)

$$tf - idf_{ac,d_4} = tf_{ac,d_4} \times idf_{ac}$$
$$tf - idf_{ac,d_4} = 1 \times 0.7$$
$$tf - idf_{ac,d_4} = 0.7$$

2. TFIDF kata balap pada dokumen 3 (d3)

$$tf - idf_{balap,d_3} = tf_{balap,d_3} \times idf_{balap}$$
$$tf - idf_{balap,d_3} = 1 \times 0.2$$
$$tf - idf_{balap,d_3} = 0.2$$

Training Rocchio Classifier

Proses training *Rocchio Classifier* menggunakan *training set* (data latih) sebagai *input*. *Training set* disertai dengan label kelas keluhan sopir atau keluhan umum. *Training set* berlabel melalui proses *preprocessing* terlebih dahulu sebelum dilakukan proses pembobotan TFIDF menggunakan persamaan (3.6). Pada proses *training* menggunakan *Rocchio Classifier* dihitung *centroid* untuk setiap kelas. *Centroid* kelas dihitung menggunakan persamaan (3.8). *Centroid* kelas ini yang menjadi model klasifikasi *Rocchio Classifier* yang kemudian disimpan pada database.

Perhitungan *centroid* kelas membutuhkan variabel jumlah dokumen data training pada setiap kelas $|D_c|$ dan normalisasi *vector* dokumen $\vec{v}(d)$. Jika dilakukan perhitungan *centroid* terhadap data pada Tabel 4.4 maka jumlah dokumen pada kelas keluhan sopir sebanyak 3 dan jumlah dokumen pada kelas keluhan umum sebanyak 2. *Vector* dokumen dapat dilihat pada Tabel 4.5.

Testing Rocchio Classifier

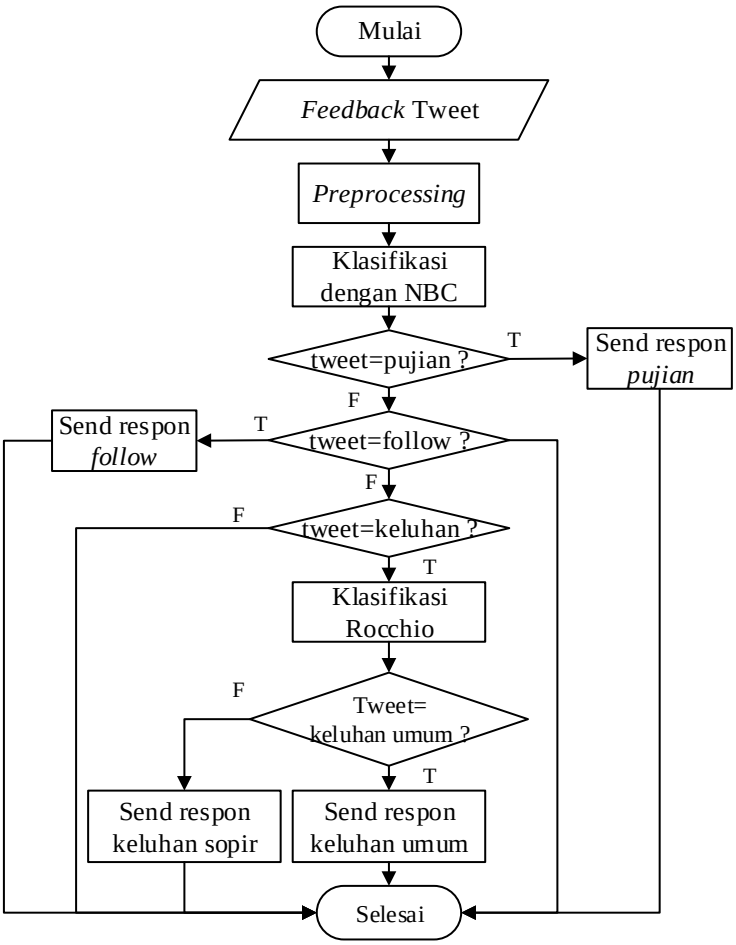
Input proses *testing* berupa data *testing* dan *centroid* kelas. Data *testing* disertai label kelas, selanjutnya melalui proses *preprocessing*. Data uji dilakukan pembobotan kata menggunakan TFIDF. *Centroid* kelas yang dihasilkan pada tahap *training* digunakan untuk menghitung jarak *centroid* setiap kelas dengan data *testing*. Kemudian dicari jarak yang paling minimal antar *centroid* kelas dengan data *testing*. Penentuan kelas pada *rochhio classifier* adalah kelas yang memiliki jarak terpendek (atau jarak minimal) antara *vector* data *testing* dengan *centroid* kelas. Kelas yang dihasilkan oleh *classifier* kemudian dibandingkan dengan label kelas yang ada pada data uji. Jika kedua kelas tersebut sama maka *classifier* menentukan kelas dengan benar.

4.7 Pengiriman Respons

Proses pengiriman respons adalah proses untuk memberikan respons terhadap tweet yang diterima oleh Express Group. Dalam mengirimkan respons *tweet* digunakan *rule* sebagai berikut :

1. Jika *feedback* tweet termasuk dalam kelas pujian, maka respons yang akan dikirimkan adalah "Terima kasih atas apresiasinya. Semoga bisa terus memberikan yang terbaik".
2. Jika *feedback* tweet termasuk dalam kelas *follow*, maka respons yang akan dikirimkan adalah "Akun Anda telah kami follow. Terima kasih."
3. Jika *feedback* tweet termasuk dalam kelas keluhan sopir, maka respons yang akan dikirimkan adalah "Mohon maaf atas salah satu sikap pengemudi kami. Mohon infokan no telp yg bisa dihubungi via DM untuk kami follow up. Terima kasih".
4. Jika *feedback* tweet termasuk dalam kelas keluhan umum, maka respons yang akan dikirimkan adalah "Mohon maaf atas ketidaknyamanannya. Mohon infokan no telp yg bisa dihubungi via DM untuk kami follow up. Terima kasih."

Proses yang dilakukan pertama kali untuk mengirim respons tweet baru secara otomatis adalah *preprocessing* tweet baru. Setelah itu dilakukan klasifikasi dengan NBC untuk mendapatkan kelas tweet. Hasil dari klasifikasi ini yang menentukan respons mana yang dikirimkan. Ketika hasil klasifikasi diketahui maka *rule* respons dibutuhkan untuk memberikan respons tweet. Khusus untuk tweet yang dikenali sebagai kelas keluhan terdapat dua pilihan respons yaitu respons keluhan sopir dan respons keluhan umum. Oleh karena itu tweet tersebut diklasifikasikan lagi menggunakan metode *Rocchio Classifier*, termasuk dalam kelas apakah tweet tersebut, kelas keluhan sopir atau kelas keluhan umum. Setelah kelas dikenali sistem mengirimkan respons berdasarkan kelas tersebut dengan memanfaatkan API Twitter. Rancangan pengiriman respons ditunjukkan pada Gambar 4.5.



Gambar 4.5 Rancangan respons tweet

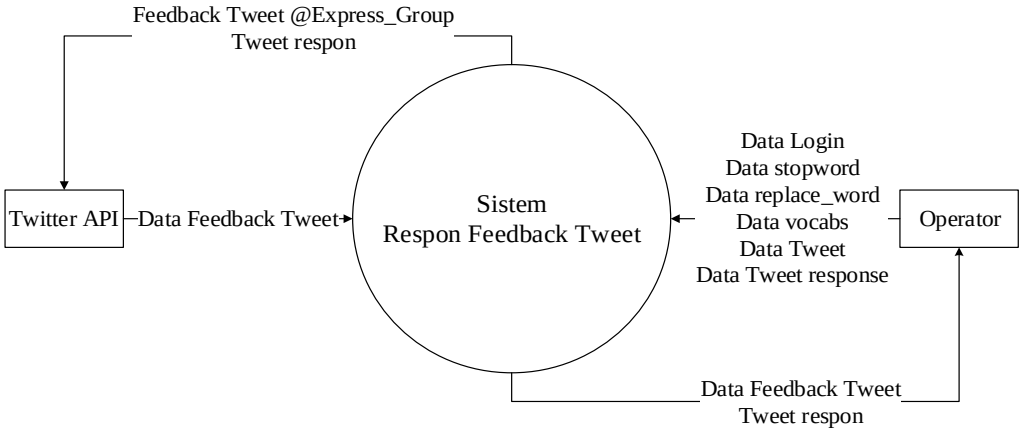
Jika rancangan respons pada Gambar 4.5 diimplentasikan maka tweet baru pada Tabel 4.1 ”sopir sopan nyaman” yang termasuk kedalam kelas pujian akan diberi respons “Terima kasih atas apresiasinya. Semoga bisa terus memberikan yang terbaik”. Tweet baru pada Tabel 4.4 “balap tidaknyaman sopir tidaksopan” yang termasuk kedalam kelas keluhan sopir akan diberi respons “Mohon maaf atas salah satu sikap pengemudi kami. Mohon infokan no telp yg bisa dihubungi via DM untuk kami follow up. Terima kasih”.

4.8 Rancangan Proses

Rancangan proses bisnis dalam penelitian ini digambarkan dengan *Data Flow Diagram* (DFD).

4.8.1 Diagram konteks

Diagram konteks menunjukkan gambaran sistem yang dibangun secara umum. Ada dua entitas yang terkait dalam sistem ini yaitu Twitter API dan operator. Dengan memanfaatkan Twitter API, sistem mendapatkan data tweet yang mengandung mention @Express_Group dan dapat mengirimkan respons tweet. Operator adalah *user* yang menggunakan sistem respons tweet. Operator terlebih dahulu memasukan data *login* agar dapat mengakses sistem. Operator juga dapat mengetahui data tweet beserta respons tweet. Operator dapat menginputkan data *stopwords*, *replace word*, *vocab*, *feedback* tweet (atau tweet), tweet respons. Diagram konteks dalam penelitian ini ditunjukkan pada Gambar 4.6.

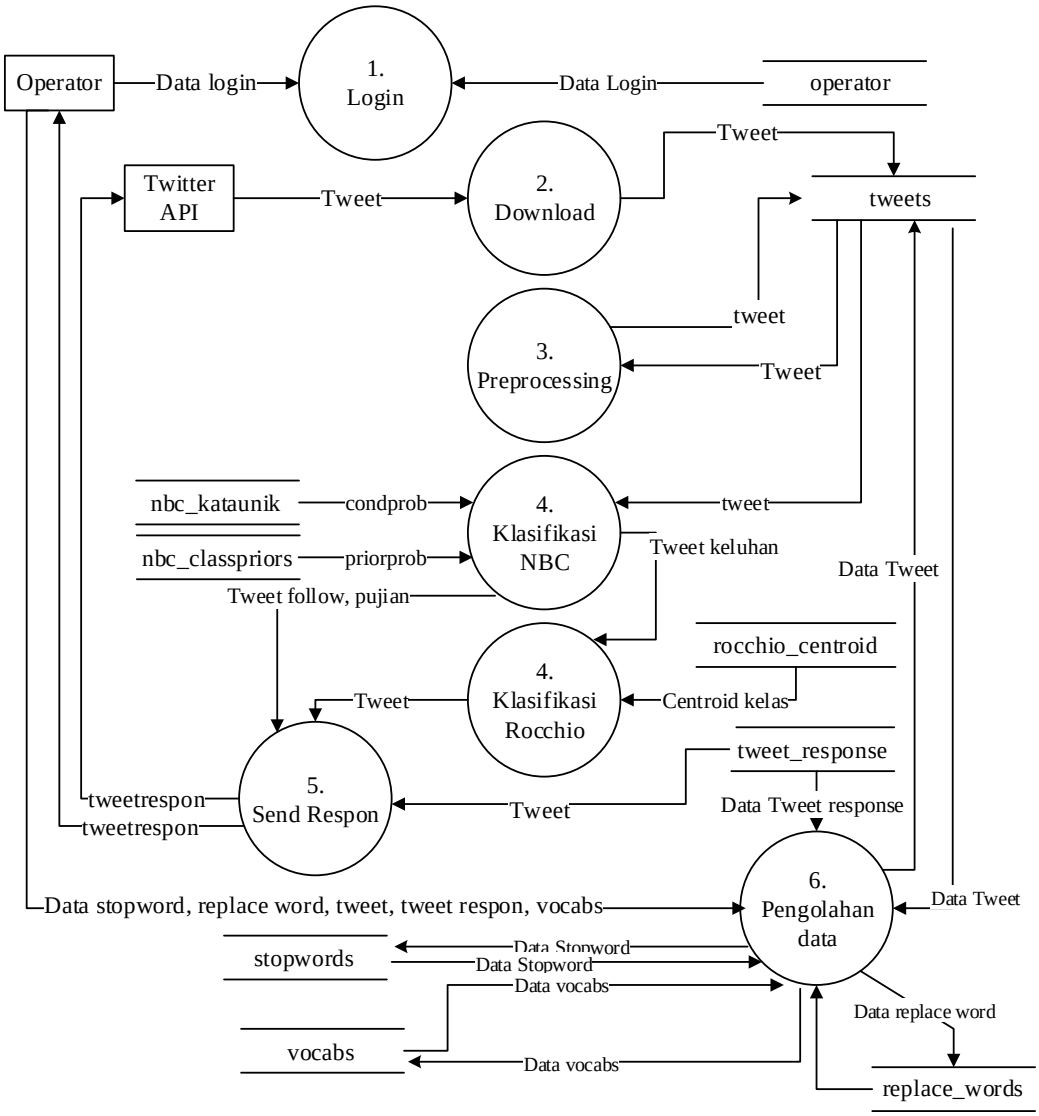


Gambar 4.6 Diagram konteks

4.8.2 DFD level 1

DFD level 1 merupakan pengembangan dari DFD level 0 yang menggambarkan proses-proses dalam sistem. DFD level 1 terdapat beberapa proses antara lain *login*, *download* tweet, *preprocessing*, klasifikasi NBC, klasifikasi Rocchio dan *send* respon (atau pengiriman respons). Proses *login* merupakan proses verifikasi yang dilakukan oleh operator agar dapat mengakses sistem. Dalam proses *login* operator memasukan data *login*. Proses *download* merupakan proses mengambil atau mengekstrak data tweet dengan menggunakan Twitter API. Kemudian tweet hasil *download* disimpan pada tabel tweets. Proses selanjutnya adalah *preprocessing. Preprocessing* ini dilakukan untuk membuang

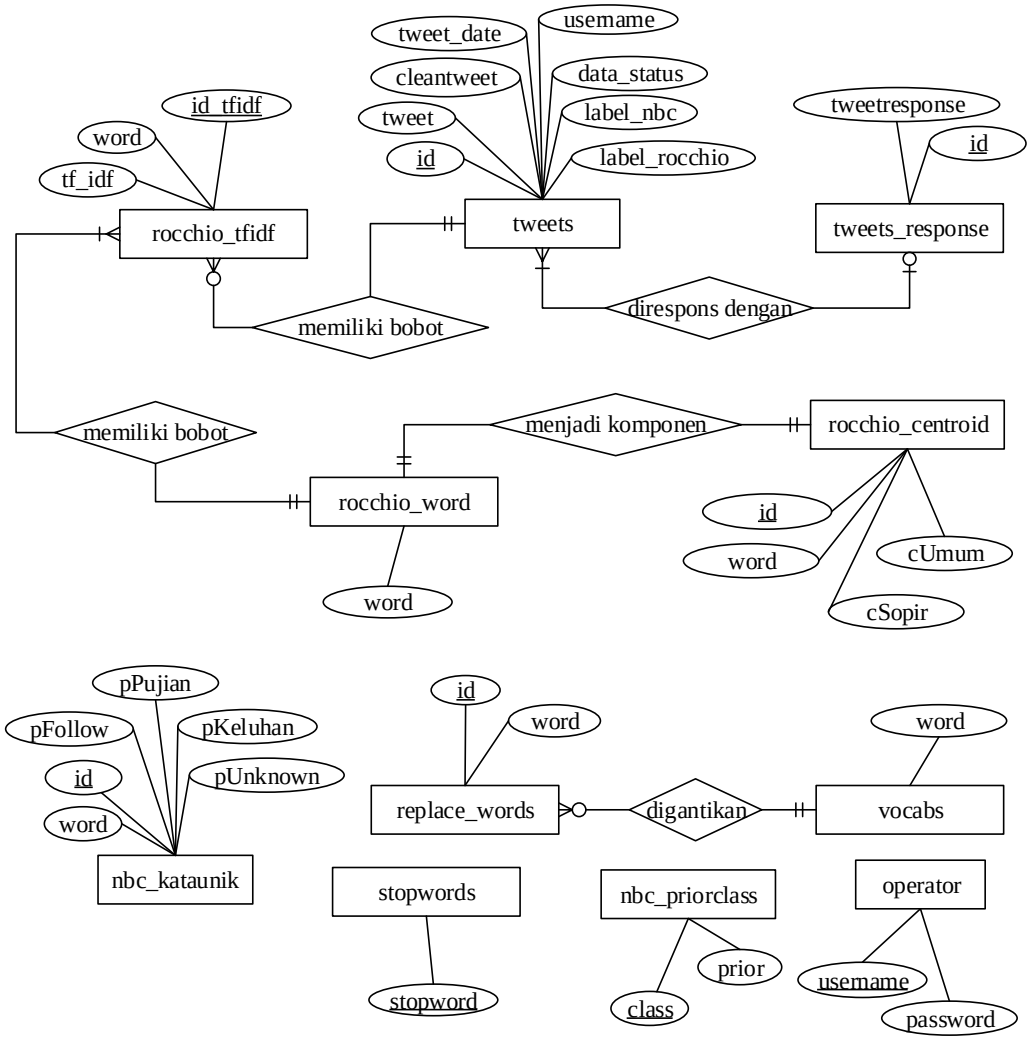
noise pada data tweet hasil proses *download*. Hasil *preprocessing* disimpan dalam tabel tweets. Tweet hasil *presprocessing* selanjutnya melalui proses klasifikasi NBC untuk mengetahui apakah tweet termasuk dalam kelas pujian, keluhan, *follow* atau *unknown*. Dalam proses klasifikasi NBC data *input* berupa kondisional probabilitas dan probabilitas prior kelas yang masing-masing diambil dari tabel nbc_kataunik dan tabel nbc_classpior. Tweet pujian dan keluhan selanjutnya melalui proses pengiriman respons, sedangkan tweet kelas keluhan melalui proses klasifikasi *Rocchio* guna mengetahui apakah tweet termasuk dalam kelas keluhan sopir atau kelas keluhan umum. Proses klasifikasi *Rocchio* membutuhkan data *input* berupa *centroid* kelas. Setelah diketahui kelas dari suatu tweet, proses selanjutnya adalah pengiriman respons tweet. Pada proses pengiriman *respon*, data tweet respons pada tabel tweet_response digunakan sebagai data input. Proses mengirim respons melibatkan Twitter API. Dalam sistem respons *feedback* tweet ini pun terdapat proses pengolahan data. Pengolahan data ini diolah operator. Operator dapat melakukan olah data stopword, replace_words, tweets, tweet_response dan vocabs. DFD level 1 ditunjukkan pada Gambar 4.7.



Gambar 4.7 DFD level 1

4.9 Rancangan Basisdata

Data yang digunakan dalam sistem disimpan dalam basisdata relasional. Pemodelan basis data digambarkan dalam bentuk ERD (Entity Relationship Diagram). Rancangan ERD ditunjukkan pada Gambar 4.8.



Gambar 4.8 Rancangan ERD

Berdasarkan Gambar 4.8 terdapat 11 entitas yakni `tweets`, `tweet_response`, `nbc_kataunik`, `rocchio_tfidf`, `stopwords`, `vocabs`, `replace_word`, `operator`, `nbc_priorclass`, `rocchio_word` dan `rocchio_centroid`. `Tweets` adalah entitas yang memuat data tweet. `tweet_response` adalah entitas yang memuat data tweet untuk merespons feedback. `nbc_kataunik` adalah entitas yang memuat kata dari tweet yang digunakan dalam klasifikasi nbc. `rocchio_tfidf` tabel untuk menyimpan bobot kata. Stopwords adalah entitas yang memuat daftar stopword. `replace_words` adalah entitas ang memuat kata tidak baku dan singkatan. `vocabs` adalah entitas

yang memuat kata dasar. `Operator` adalah entitas yang memuat data untuk keperluan login sistem. `Nbc_priorclass` adalah entitas yang memuat nilai probabilitas prior kelas. `Rocchio_word` adalah entias untuk menyimpan semua kata yang terdapat pada tweet yang digunakan untuk klasifikasi rocchio. `Rocchio_centroid` adalah entitas yang memuat komponen centroid kelas.

Berdasarkan Gambar 4.8 selain entitas juga terdapat relasi yang menghubungkan antar entitas. Antara entitas `tweets` dengan entitas `tweet_response` memiliki relasi di respons dengan, maknanya bahwa setiap `tweets` direspons dengan satu `tweet_response` atau setiap `tweets` boleh tidak direspons dengan `tweet_respons`. Setiap `tweet_response` digunakan untuk merespons satu atau lebih `tweets`. Relasi digantikan merupakan relasi yang menghubungkan antara entitas `replace_word` dengan entitas `vocabs`, maknanya setiap `replace_words` digantikan dengan satu dan hanya satu `vocabs`. Setiap `vocabs` menggantikan satu atau boleh tidak menggantikan `replace_words`. Relasi menjadi komponen adalah relasi antara entitas `rocchio_word` dan `rocchio_centroid`, maknanya adalah setiap `rocchio_word` menjadi satu komponen `rocchio_centroid`. Satu komponen `rocchio_centroid` dijadikan dari satu `rocchio_word`. Antara `rocchio_word` dan `rocchio_tfidf` memiliki relasi memiliki bobot, maknanya adalah setiap satu `rocchio_word` memiliki minimal minimal satu bobot `rocchio_tfidf` dan satu `rocchio_tfidf` dimiliki oleh satu dan hanya satu `rocchio_word`. Antara entitas `tweets` dan `rocchio_tfidf` terdapat relasi memiliki bobot, maknanya setiap `tweets` boleh tidak memiliki bobot `rocchio_tfidf` dan setiap satu bobot `rocchio_tfidf` dimiliki oleh satu dan hanya satu `tweet`.

Struktur Tabel

Rancangan ERD kemudian dijadikan acuan untuk membuat data base sistem. Untuk struktur lengkap rancangan tabel dijabarkan sebagai berikut:

1. Tabel Tweets

Tabel `tweets` berisi data tweet yang di *download* menggunakan API Twitter. Tabel tweets terdiri dari 9 field antara lain `id`, `tweet`, `username`, `tweet_date`, `clean_tweet`, `label_nbc`, `label_rocchio`, `data_status` dan `id_tweetresponse`. Struktur tabel `tweets` ditunjukkan pada Tabel 4.6.

Tabel 4.6 Struktur tabel tweets

Field	Type	Constraint	Keterangan
id	bigint	*primary key*	id tweet
tweet	varchar(200)		tweet pelanggan
username	varchar(50)		username pelanggan
tweet_date	datetime		tanggal dan waktu tweet
clean_tweet	Varchar(200)		tweet hasil preprocessing
label_nbc	int		label manual nbc
label_rocchio	int		label manual rocchio
data_status	enum (‘training’,’testing’)		keterangan sebagai data training atau data testing
id_tweetresponse	int	*foreign key*	berisi id yang merujuk pada tabel tweetresponse

2. Tabel tweet_response

Tabel `tweet_response` berisi daftar respons yang dikirimkan kepada pelanggan. Tabel `tweet_respons` digunakan pada proses pengiriman respons *feedback*. Tabel `tweet_respons` terdiri dari 2 field yaitu `id`, `tweetresponse`. Field `id` berisi nomor pengenal tweet respons. Field `tweetresponse` berisi standard tweet respons Express Group yang biasa digunakan untuk merespons *feedback* pelanggan. Struktur tabel `tweet_response` ditunjukkan pada Tabel 4.7.

Tabel 4.7 Struktur tabel respons

Field	Type	Constraint	Keterangan
id	int	*primary key*	Nomor id tweet respons
tweetresponse	varchar(140)		Tweet respons untuk merespons tweet pelanggan

3. Tabel nbc_priorclass

Tabel `nbc_ priorclass` merupakan tabel yang digunakan untuk menyimpan nilai hasil perhitungan probabilitas prior kelas pada proses *training* NBC. Tabel `nbc_classprior` terdiri dari 2 field yaitu `class dan prior`.

Field `class` berisi label kelas dan menjadi *primary key* pada pada tabel `nbc_priorclass`. Field `prior` untuk menampung nilai probabilitas prior kelas. Struktur tabel `classpriors` ditunjukkan pada Tabel 4.8.

Tabel 4.8 Struktur tabel nbc_priorclass

Field	Type	Constraint	Keterangan
class	int	*primary key*	Label kelas
prior	decimal(15,15)		Nilai probabilitas prior kelas

4. Tabel nbc_kataunik

Tabel `nbc_kataunik` merupakan tabel untuk menyimpan semua kata dari tweet yang digunakan sebagai data *training* pada proses *training* NBC. Tabel `nbc_kataunik` juga digunakan untuk penyimpanan nilai probabilitas bersyarat kata terhadap kelas. Tabel `nbc_kataunik` terdiri dari 6 field antara lain `id`, `word`, `pPujian`, `pKeluhan`, `pFollow` dan `pUnknown`. Field `id` merupakan *primary key* tabel nbc_kataunik yang berisi nomor pengenal kataunik. Field `word` berisi kata dari semua data training NBC. Nilai probabilitas bersyarat kata terhadap kelas pujian, keluhan, *follow* dan *unknown* secara berturut disimpan pada field `pPujian`, `pKeluhan`, `pFollow`, `pUnknown`. Struktur tabel `nbc_katauniks` ditunjukkan pada Tabel 4.9.

Tabel 4.9 Struktur tabel nbc_katauniks

Field	Type	Constraint	Keterangan
id	int	Primary *Key*	Nomor id kataunik
word	varchar(50)		Berisi kata dari tweet_nbc
pPujian	decimal(15,15)		Nilai probabiliras bersyarat dari kata terhadap kelas pujian
pKeluhan	decimal(15,15)		Nilai probabiliras bersyarat dari kata terhadap kelas keluhan
pFollow	decimal(15,15)		Nilai probabiliras bersyarat dari kata terhadap kelas follow
pUnknown	decimal(15,15)		Nilai probabiliras bersyarat dari kata terhadap kelas unknown

5. Tabel vocabs

Tabel `vocabs` merupakan tabel yang merepresentasikan kamus bahasa Indonesia. Dalam tabel ini berisi daftar kata baku bahasa Indonesia. Tabel `vocabs` digunakan pada tahap *preprocessing*, khususnya pada tahap *replacement*. Tabel vocabs memiliki field `word`. Struktur tabel `vocabs` ditunjukkan pada Tabel 4.10.

Tabel 4.10 Struktur tabel vocabs

Field	Type	Constraint	Keterangan
word	varchar(70)		kata dasar

6. Tabel replace_words

Tabel `replace_words` berisi daftar kata tidak baku dan singkatan. Tabel `replace_words` digunakan pada tahap *preprocessing*, khususnya pada tahap *replacement*. Tabel `replace_words` terdiri dari 3 fields antara lain `id`, `word` dan `stem`. Field `id` merupakan *primary key* pada tabel `replace_word`. Field `Word` berisi kata yang akan di-*replace*. Field `stem` berisi kata dasar yang merujuk pada field `word` tabel `vocabs`. Struktur tabel `replace_words` ditunjukkan pada Tabel 4.11.

Tabel 4.11 Struktur tabel replace_words

Field	Type	Constraint	Keterangan
id	int	*primary key*	Nomor pengenal kata
word	varchar(20)		Kata tidak baku dan singkatan
stem	int	*foreign key*	kata dasar pada tabel vocabs

7. Tabel stopwords

Tabel `stopwords` berisi data *stopword*. Tabel ini digunakan pada proses *preprocessing* khususnya proses *stopword removal*. Pada tabel `stopwords` terdapat field `stopword`. Field `Stopword` berisi kata yang merupakan *`stopword`*. Struktur tabel `stopwords` ditunjukkan pada Tabel 4.12.

Tabel 4.12 Struktur tabel stopwords

Field	Type	Constraint	Keterangan
stopword	varchar(20)	*primary key*	Kata stopword

8. Tabel operator

Tabel `operator` berisi data operator. Data operator digunakan pada proses *login* agar dapat megakses sistem respons *feedback* otomatis. Pada tabel `operator` terdapat field `username` dan `password`. Field `username` berisi

username dari operator dan menjadi primary key pada tabel operato. Field `Password` berisi password login operator. Struktur tabel `operator` ditunjukkan pada Tabel 4.13.

Tabel 4.13 Struktur tabel operator

Field	Type	Constraint	Keterangan
username	varchar(50)	*primary key*	Username operator
password	varchar(50)		Password operator

9. Tabel rocchio-word

Tabel `rocchio_word` adalah tabel untuk menyimpan kata yang terdapat pada tweet yang digunakan untuk klasifikasi rocchio. Tabel `rocchio_word` memiliki dua field yaitu `id` dan `word`. Struktur tabel `rocchio_word` ditunjukkan pada Tabel 4.14.

Tabel 4.14 Struktur tabel rocchio_word

Field	Type	Constraint	Keterangan
word	varchar(50)		Berisi kata

10. Tabel rocchio_centroid

Tabel `rocchio_centroid` merupakan tabel yang berisi nilai komponen *centroid* kelas keluhan umum dan komponen *centroid* kelas keluhan sopir hasil proses *training Rocchio*. Tabel `rocchio_centroid` memiliki empat field yaitu `id`, `id_word`, `cSopir`, `cUmum`. Field `id` sebagai *primary key* tabel `rocchio_centroid`. Field `word` ini berisi semua kata yang terdapat pada data *training Rocchio*. `cUmum` berisi nilai komponen *centroid* pada kelas keluhan umum. `cSopir` berisi nilai komponen *centroid* pada kelas keluhan sopir. Struktur tabel `rocchio_centroid` ditunjukkan pada Tabel 4.15.

Tabel 4.15 Struktur tabel rocchio_centroid

Field	Type	Constraint	Keterangan
id	int	*primary key*	Nomor pengenal komponen centroid
word	varchar(50)	*foreign key*	Berisi kata
cSopir	decimal(15,15)		Nilai komponen centroid keluhan sopir
cUmum	decimal(15,15)		Nilai komponen centroid keluhan umum

11. Tabel rocchio_tfidf

Tabel `rocchio_tfidf` merupakan tabel berisi bobot tfidf. Tabel `rocchio_tfidf` terdiri dari 4 field antara lain `id_tfidf, word, kelas, d, tf, idf, tfidf`. Struktur tabel `rocchio_tfidf` ditunjukkan pada Tabel 4.16.

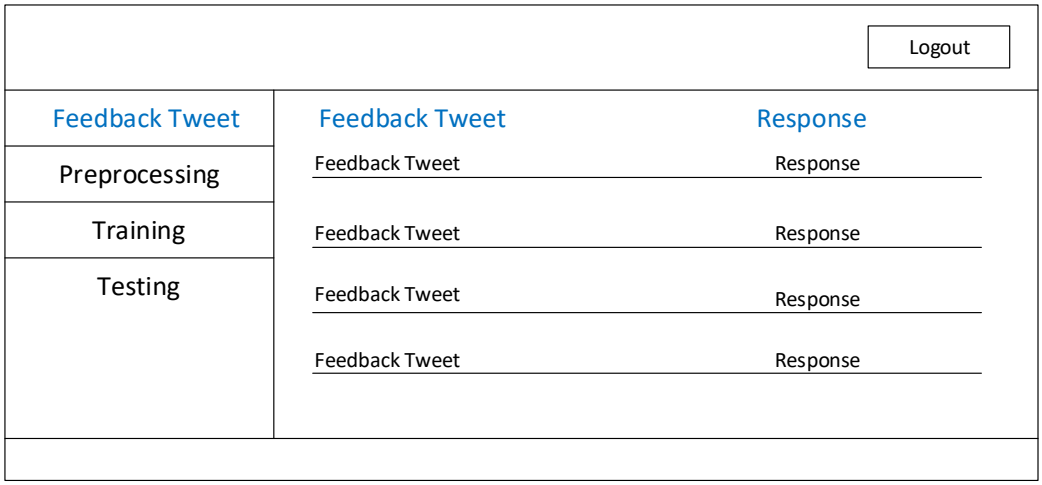
Tabel 4.16 Struktur table rocchio_tfidf

Field	Type	Constraint	Keterangan
id_tfidf	int	*primary key*	Nomor pengenal tfidf
word	varchar(50)	*foreign key*	Berisi kata
tfidf	float		Nilai tfidf
id_tweet	bigint		Berisi nomor tweet rocchio

4.10 Rancangan Antarmuka Respons Tweet

Rancangan antarmuka menggambarkan rancangan antarmuka sistem yang dibangun dalam penelitian ini. Sistem respons *feedback* otomatis hanya digunakan untuk operator Express_Group. Operator harus login terlebih dahulu untuk mengakses sistem respons *tweet* otomatis.

Rancangan antarmuka respons tweet merupakan rancangan *page* yang berisi *feedback, preprocessing, training* dan *testing*. Pada *page feedback* terdapat tampilan tweet dan respons tweet yang direspons secara otomatis. Page *preprocessing* adalah antarmuka proses *preprocessing feedback* tweet. Page *training* adalah antarmuka proses *training* NBC dan Rocchio. Page *testing* adalah antarmuka *testing* NBC dan *Rocchio*. Rancangan antarmuka respons tweet ditunjukkan pada Gambar 4. 9.



Gambar 4. 9 Antarmuka respons *feedback* pelanggan

4.11 Rancangan Pengujian

Rancangan pengujian terbagi menjadi dua yaitu pengujian white box dan pengujian evaluasi *classifier*. Pengujian *white box* digunakan untuk mendeteksi kesalahan logis dalam kode program. Pengujian *white box* dilakukan dengan mencocokan hasil perhitungan secara manual dengan hasil perhitungan yang didapatkan oleh sistem. Pengujian akurasi *classifier* dilakukan dengan mengukur efektivitas atau kemampuan *classifier* mengklasifikasikan dengan tepat. Evaluasi *classifier* dilakukan dengan menghitung *presicion*, *recall* dan *f-measusre* dan akurasi. Perhitungan *presicion* menggunakan persamaan (3.11), perhitungan *recall* menggunakan persamaan (3.12), perhitungan *f-measure* menggunakan persamaan (3.13), dan perhitungan akurasi menggunakan persamaan (3,14).

BAB V
IMPLEMENTASI SISTEM

Sistem yang dibangun dalam penelitian ini diimplementasikan berdasarkan pada rancangan sistem pada bab IV. Sistem yang dibangun berbasis web dan diimplementasikan dengan menggunakan PHP, basis data MySQL dan menggunakan sistem operasi windows 10.

5.1 Implementasi Download Tweet

Download tweet dilakukan dengan memanfaatkan Twitter REST API. Gambar 5.1 merupakan implementasi *download* tweet yang disertai dengan *insert* data tweet ke database.

```
$settings = array(
 'oauth_access_token' => "",
 'oauth_access_token_secret' => "",
 'consumer_key' => "",
 'consumer_secret' => ""
);

$url = 'https://api.twitter.com/1.1/search/tweets.json';
$requestMethod = 'GET';
$getfield='?q=%40express_group&lang=id';
$twitter = new TwitterAPIExchange($settings);
$text_tweet_mention = $twitter->setGetfield($getfield)
   ->buildOauth($url, $requestMethod)
   ->performRequest();
$text_tweet_mention_json_decode=
   json_decode($text_tweet_mention);

$tweet_baru = 0;
foreach($text_tweet_mention_json_decode->statuses as
   $tweet_mention){
   $date=date_create($tweet_mention->created_at);
   $tweet_date = date_format($date,"Y/m/d H:i:s");
   $sql_cari = mysql_query("select tweet_id from tweets
     where tweet_id='".$tweet_mention->id_str."' ");
   $count_cari = mysql_num_rows($sql_cari);
   if($count_cari < 1){
     $tweet_baru += 1;
     mysql_query("insert into tweets
     (id,tweet,username,tweet_date)
     values ('".$tweet_mention->id_str."','".$tweet_mention
     ->text."','".$tweet_mention->user->screen_name."',
     '$tweet_date') ");
   }
}
```

Gambar 5.1 Implementasi *download* tweet

Berdasarkan pada Gambar 5.1 tweet yang di-*download* adalah tweet yang mengandung *mention* @express_group seperti yang dikodekan pada baris 14. Data yang dibutuhkan berupa id_tweet, screen_name, isi tweet, tanggal dan waktu tweet. Selanjutnya data-data tersebut disimpan dalam tabel tweets. Kode baris 29-31 merupakan kode untuk menyimpan tweet ke tabel tweets.

5.2 Implementasi *Preprocessing*

Data hasil proses *download* tweet kemudian melalui proses *preprocessing*. Seperti yang dijelaskan pada rancangan *preprocessing* bahwa *preprocessing* terdiri dari beberapa proses antara lain menghapus URL, menghapus tanda baca, *casefolding*, *tokenization*, *replacement*, *stopword removal*, *stemming* dan menggabungkan kata negasi.

Menghapus URL

URL dalam tweet dihapus pada proses *preprocessing*. Penghapusan URL memanfaatkan fungsi `preg_replace` yang disediakan oleh PHP seperti yang ditunjukkan pada baris 14. Fungsi `preg_replace` digunakan untuk menghapus URL yang disimbolkan pada variabel `$regex`. Implementasi menghapus URL ditunjukkan pada Gambar 5.2.

```
13. $regex = "@(https?://([-\w\.]+[-
\w])+(:\d+)?(/([\w/_\.#]*(\?\S+)?[^\.\s])?)?)@";
14. $text = preg_replace($regex, '', $text);
```

Gambar 5.2 Implementasi menghapus URL

Menghapus Tanda Baca

Tanda baca dalam tweet dihilangkan dengan menggunakan fungsi `omitCharacters`. `$replacelist` pada baris 2 adalah array yang menampung tanda baca yang akan dihilangkan. Kode baris 110 terdapat fungsi php `str_replace` yang berfungsi untuk menghapus tanda baca pada variabel `$replacelist`. Implementasi penghapusan tanda baca ditunjukkan pada Gambar 5.3.

```
106. function omitCharacters($text=array(){
107.  $replacelist = array(
108.  '.',',',"\'",':',';','=','*','%',"\"",'?','!','#','-
   ','_','+','=','$',"\/","\\",'(',')');
109.  foreach($replacelist as $list){
110.  $text = str_replace($list, ' ', $text);
111.  }
112.  return $text;
113. }
```

Gambar 5.3 Implementasi penghapusan tanda baca

Casefolding

Casefolding merupakan proses mengubah seluruh tweet menjadi *lowercase*. Proses *casefolding* menggunakan fungsi yang sudah disediakan oleh PHP yaitu fungsi `strtolower`. Implementasi *casefolding* ditunjukkan pada Gambar 5.4.

```
24. $text = strtolower($text);
```

Gambar 5.4 Implementasi *casefolding*

Tokenization

Tokenization adalah pemotongan string input berdasarkan tiap kata yang menyusunnya. Proses *tokenization* menggunakan fungsi yang sudah disediakan oleh PHP yaitu fungsi `explode`. *Tokenization* dilakukan berdasarkan *delimiter* spasi. Implementasi *tokenization* ditunjukkan pada Gambar 5.5.

```
27. $text = explode(" ", $text);
```

Gambar 5.5 Implementasi *tokenization*

Replacement

Proses *replacement* adalah mengganti kata tidak baku menjadi kata baku. Proses *replacement* dilakukan menggunakan fungsi `wordReplacement`. Implementasi *replacement* word ditunjukkan pada Gambar 5.6.

```
130. function wordReplacement($text=array()){
131.   foreach ($text as $key=>$kata) {
132.      $kata =  mysql_escape_string($kata);
133.      $sql = mysql_query("select rw.word, rw.vocab_id,
     v.word katadasar
134.             from replace_words rw join vocabs v on v.id
     = rw.vocab_id
135.             where rw.word='$kata'");
136.      $d = mysql_fetch_array($sql);
137.      if(mysql_num_rows($sql)>0) $text[$key] =
     $d['katadasar'];
138.   }
139.   return $text;
140. }
```

Gambar 5.6 Implementasi *replacement*

Berdasarkan Gambar 5.6 baris 131-139 dilakukan perulangan untuk setiap kata pada array `$text`. Setiap perulangan dilakukan pengecekatan kata, apakah termasuk kata dalam daftar tabel replace_word. Pada kode baris 137 jika kata tersebut merupakan kata replace_word maka diganti menggunakan kata dasar yang terdapat pada tabel vocabs.

Stemming

Pada tweet terdapat kata berimbuhan. Setiap kata berimbuhan diubah menjadi kata dasar, yang disebut dengan *stemming*. Proses *stemming* dalam *preprocessing* ini memanfaatkan *library* sastrawi. Implementasi *stemming* ditunjukkan pada Gambar 5.7.

```
34. $text = $stemmer->stem($text);
```

Gambar 5.7 Implementasi *stemming*

Menghapus *stopword*

Stopword dianggap tidak berperan penting dalam menentukan kelas suatu tweet. Oleh karena itu dilakukan proses penghapusan *stopword*. Proses hapus *stopword* menggunakan fungsi stopWordRemoval. Pada fungsi `stopWordRemoval` kode baris 127 terdapat fungsi `preg_replace` untuk menghapus stopword yang ditampung dalam variabel `$regex`. Implementasi *stopword removal* ditunjukkan pada Gambar 5.8.

```
119. function stopWordRemoval($text=””){
120.  $sql = mysql_query("select id,stopword from
     stopwords");
121.  $kata = "";
122.  while($stopword=mysql_fetch_array($sql)){
123.    $kata .= $stopword['stopword']."|";
124.  }
125.
126.  $regex = "/\b(".$kata.")\b/";
127.  return preg_replace($regex, '', $text);
128. }
```

Gambar 5.8 Implementasi *stopword removal*

Menggabungkan kata “tidak” dengan kata berikutnya.

Jika ada kata tidak dalam tweet maka akan digabungkan dengan kata selanjutnya. Penggabungan tidak menggunakan fungsi `clusterTidak`. Pada kode baris 96 jika berisi kata tidak maka digabungkan dengan kata selanjutnya seperti yang tercantum pada kode baris 98. Implementasi penggabungan kata tidak ditunjukkan pada Gambar 5.9.

```
93. function clusterTidak($text=array()){
94.  $cluster=array();
95.  for($i=0;$i<=count($text)-1;$i++){
96.   if($text[$i]=="tidak"){
97.     if(isset($text[$i]) && isset($text[$i+1]))
98.        $cluster[]=$text[$i]."".$text[$i+1];
99.     else $cluster[] = $text[$i];
100.       $i++;
101.  }else $cluster[] = $text[$i];
102.  }
103.  return $cluster;
104. }
```

Gambar 5.9 Implementasi penggabungan negasi

5.3 Implementasi Klasifikasi NBC

NBC dalam penelitian ini digunakan untuk melakukan klasifikasi tweet. Apakah tweet masuk dalam kelas pujian, keluhan, *follow*, atau *unknown*. Sebelum dilakukan penetuan kelas atau klasifikasi, NBC akan membangun model klasifikasi atau *classifier* pada proses *training*.

Training NBC

Proses *training* NBC merupakan proses membangun model klasifikasi. Model klasifikasi dalam *training* NBC berupa probabilitas bersyarat kemunculan

kata dalam dokumen pada suatu kelas. Perhitungan probabilitas bersyarat menggunakan persamaan (3.5). Implementasi perhitungan probabilitas bersyarat ditunjukkan pada Gambar 5.10.

```
294. function condprob($Tct,$c,$kata,$B){
295.  $condprob_tc = 0;
296.  if(array_key_exists($c, $Tct)){
297.    $Tct_c_v =0;
298.    if (isset($Tct[$c][$kata])) $Tct_c_v =$Tct[$c][$kata];
299.    $tc_=0;
300.    foreach ($Tct[$c] as $valueTct) {
301.      $tc_ += $valueTct;
302.    }
303.    $condprob_tc=($Tct_c_v + 1)/($tc_+$B);
304.  }
305.  return $condprob_tc;
306. }
```

Gambar 5.10 Implementasi probabilitas bersyarat

Perhitungan probabilitas bersyarat menggunakan fungsi `condprop`. Pada Kode baris 303 adalah kode untuk perhitungan probabilitas bersyarat, variabel `$Tct_c_v` adalah jumlah kemunculan kata pada data *training* kelas c. Variabel `$tc_` adalah jumlah kemunculan seluruh kata pada data *training* kelas c dan variabel `$B` adalah jumlah keseluruhan kata pada data training.

Pada proses *training* NBC juga dilakukan proses perhitungan probabilitas prior menggunakan persamaan (3.4). Implementasi perhitungan probabilitas prior ditunjukkan pada Gambar 5.11.

```
280. function prior ($Nc, $N){
281.   mysql_query("TRUNCATE nbc_priorclass");
282.   foreach ($Nc as $c => $c_val) {
283.     $prior[$c]=$Nc[$c]/$N;
284.     mysql_query("insert into nbc_priorclass values
   ('$c','".number_format($prior[$c],14)."')");
285.   }
286. }
```

Gambar 5.11 Implementasi probabilitas prior kelas

Berdasarkan pada Gambar 5.11 terdapat fungsi prior yang digunakan untuk menghitung probabilitas prior. Dalam fungsi prior terdapat parameter `$Nc[$c]` dan `$N`. `$Nc[$c]` yaitu variabel yang menampung jumlah dokumen kelas c dan `$N` yaitu jumlah keseluruhan dokumen. Pada baris kode 283 adalah kode untuk menghitung probabilitas prior setiap kelas.

Testing NBC

Pada proses *testing* NBC terdapat proses perhitungan probabilitas dokumen *d* berada dalam kelas *c* dengan menggunakan persamaan (3.2). Perhitungan probabilitas posterior kelas membutuhkan model klasifikasi yang telah didapatkan pada proses *training* dan mebutuhkan nilai probabilitas prior kelas. Kode baris 148-151 merupakan kode untuk menghitung probabilitas dokumen *d* berada dalam kelas *c*. Implementasi probabilitas dokumen *d* berada dalam kelas *c* ditunjukkan pada Gambar 5.12.

```
137. foreach($alldata_testing as $i=>$data_testing) {
138.  $score[$i]=array(1=>$prior[1], // pujian
139.                   2=>$prior[2], // keluhan
140.                         3=>$prior[3], // follow
141.                         4=>$prior[4]); // unknown
142. $score[$i]['text'] = $data_testing['text'];
143.  echo "Data test : ";
144.  foreach($data_testing ['text'] as $datatest){
145.  echo "$datatest : ";
146.   foreach ($kataunik as $key=>$kata) {
147.    if($kata['kata']== $datatest){
148.        $score[$i][1] = $score[$i][1] * $kata['pPujian'];
149.        $score[$i][2] = $score[$i][2] * $kata['pKeluhan'];
150.        $score[$i][3] = $score[$i][3] * $kata['pFollow'];
151.        $score[$i][4] = $score[$i][4] * $kata['pUnknown'];
152.    }
153.   }
154.  }
165. $max = $score[$i][1];
166. $kelas = 1;//Pujian
167. if($score[$i][2] > $max){
168.  $max=$score[$i][2];
169.  $kelas = 2;//Keluhan
170. }
171. if($score[$i][3] > $max) {
172.  $max=$score[$i][3];
173.  $kelas = 3;//Follow
174. }
175. if($score[$i][4] > $max) {
176.  $max=$score[$i][4];
177.  $kelas = 4;//Unknown
178. }
```

Gambar 5.12 Implementasi *testing* NBC

Naïve Bayes Classifier menentukan kelas tweet berdasarkan kelas yang memiliki nilai probabilitas posterior kelas maksimum atau *maximum a posteriori class*. Pada kode baris 165 terdapat variabel `$max` yang menampung nilai

probabilitas posterior kelas [1] atau kelas pujian. Kode baris 166 terdapat variabel `$kelas` yang memiliki nilai 1 atau kelas pujian. Kode baris 167 jika probabilitas posterior kelas [2] (atau keluhan) memiliki nilai yang lebih besar dari pada `$max`, maka probabilitas posterior kelas [2] (atau keluhan) menjadi nilai `$max` dan `$kelas` = 2 (atau keluhan). Kode baris 171 jika probabilitas posterior kelas [3] (atau follow) memiliki nilai yang lebih besar dari pada `$max`, maka probabilitas posterior kelas [3] (atau follow) menjadi nilai `$max` dan `$kelas` = 3 (atau follow). Kode baris 175 jika probabilitas posterior kelas [4] (atau unknown) memiliki nilai yang lebih besar dari pada `$max`, maka probabilitas posterior kelas [4] (atau keluhan) menjadi nilai `$max` dan `$kelas` = 4 (atau unknown).

5.4 Implementasi *Rocchio*

Metode *Rocchio* dalam penelitian ini digunakan untuk menentukan tweet kedalam kelas keluhan sopir atau kelas keluhan umum. Seperti halnya NBC, implementasi *Rocchio* pun terdiri dari dua proses, yaitu proses *training* dan *testing*. Berdasarkan rancangan *Rocchio* pada Gambar 4.4, pada proses *training* dan *testing* Rocchio terdapat proses pembobotan TFIDF. Untuk mendapatkan nilai TFIDF maka harus diketahui terlebih dahulu nilai *term frequency* tf dan *invers documen frequency* idf. Tf adalah jumlah kemunculan kata pada suatu dokumen. Implementasi tf ditunjukkan pada Gambar 5.13. Kode baris 151 nilai TF diset bernilai 0, kemudian dilakukan perulangan untuk setiap kata. Jika ditemukan kata yang sama maka selanjutnya niali Tf ditambahkan 1 seperti yang ditunjukkan pada kode baris 154.

```
147. $TF = array();
148. foreach($semua_kata as $key=>$val){
149.  $i=1; // untuk dokumen
150.  foreach ($kata_d as $value1) {
151.  $TF[$key][$value1['kelas']][$i]=0;
152.  foreach($value1['word'] as $k){
153.        if($key==$k){
154.        $TF[$key][$value1['kelas']][$i] ++;
155.        }
156.  }
157.  $i++;
158.  }
159. }
```

Gambar 5.13 Implementasi *term frequency*

Selanjutnya perhitungan *invers document frequency* idf menggunakan persamaan (3.7). Implementasi perhitungan idf ditunjukkan pada Gambar 5.14. Kode baris 179 adalah kode untuk menghitung nilai idf. Dimana `$N` adalah jumlah total dokumen dan `$df` adalah banyaknya dokumen yang mengandung kata t.

```
174. foreach($TF as $term=>$kelasdoc){
175.   $df=0;//df @ jml doc yg mengandung kata t
176.   foreach($kelasdoc as $doc){
177.     foreach($doc as $ftterm){ if($ftterm >0 ) $df++; }
178.    }
179.      $IDF[$term] = log10($N/$df) ;
180.  }
```

Gambar 5.14 Implementasi IDF

Jika tf dan idf sudah diketahui maka selanjutnya dilakukan perhitungan tfidf. Nilai tfidf dihasilkan dengan mengalikan nilai tf dan idf. Implementasi tfidf berdasarkan pada persamaan (3.6). Kode baris 195 adalah kode perhitungan berdasarkan persamaan (3.6). Implementasi tfidf ditunjukkan pada Gambar 5.15.

```
192. foreach($semua_kata as $term=>$val){
193.  $i=1;
194.  foreach ($kata_d as $value1) {
195.  $TFIDF[$term][$value1['kelas']][$i]=
            $TF[$term][$value1['kelas']][$i] * $IDF[$term];
196.  if(isset($sigmatfidfkuadrat [$i])){
197.         $sigmatfidfkuadrat [$i] +=
    ($TFIDF[$term][$value1['kelas']][$i]*$TFIDF[$term][$value1['
   kelas']][$i]);
198.  }else{
199.         $sigmatfidfkuadrat[$i] =
    ($TFIDF[$term][$value1['kelas']][$i]*$TFIDF[$term][$value1['
   kelas']][$i]);
200.  }
201.  $i++;
202.  }
203. }
```

Gambar 5.15 Implementasi TFIDF

Dalam penelitian ini tfidf yang digunakan adalah normalisasi tfidf agar nilai tfidf dalam range [0,1]. Normalisasi TFIDF diimplementasikan berdasarkan pada persamaan (3.9). Perhitungan normalisasi TFIDF adalah membagikan nilai tfidf dengan panjang vector seperti yang dikodekan pada baris 214. Implementasi normalisasi TFIDF ditunjukkan pada Gambar 5.16.

```
210. $Normalisasi_TFIDF = array();
211. foreach($semua_kata as $term=>$val){
212.  $i=1;
213.  foreach ($kata_d as $value1) {
214.   $Normalisasi_TFIDF[$term][$value1['kelas']][$i]
215.   = $TFIDF[$term][$value1['kelas']][$i] /
   $Panjang_Vektor[$i];
216.  $i++;
217. }
```

Gambar 5.16 Implementasi normalisasi TFIDF

Training Rocchio

Berdasarkan pada rancangan klasifikasi *Rocchio* Gambar 4.4 dalam proses *training Rocchio* terdapat proses perhitungan *centroid* kelas. Implementasi perhitungan *centroid* bedasarkan pada persamaan (3.8). Implementasi perhitungan *centroid* ditunjukkan pada Gambar 5.17.

```
220. $Centroid = array();
221. $Kelas = array(1,2);
222. foreach($Kelas as $c){
223.  foreach ($Normalisasi_TFIDF as $kata => $val) {
224.   If (isset($val[$c])){
225.    foreach ($val[$c] as $key => $valuenormtfidf) {
226.      If (!isset($sigma_normtfidf[$kata][$c]))
           $sigma_normtfidf[$kata]=array($c=null);
227.       if(isset($Sigma_normtfidf [$kata]))
            $Sigma_normtfidf [$kata][$c]  +=
               $valuenormtfidf;
228.       else $Sigma_normtfidf [$kata][$c] =
               $valuenormtfidf;
229.       }
230.       $Dc= $jml_document_c[$c];
231.       $Centroid[$c][$kata] =
232.       (1/ abs($Dc) ) * $Sigma_normtfidf [$kata][$c];
233.    }
234. }
```

Gambar 5.17 Implementasi perhitungan *centroid*

Kode baris 231 merupakan kode perhitungan *centroid* sesuai dengan persamaan (3.8). Perhitungan *centroid* melibatkan jumlah dokumen kelas (`$Dc`) dan sigma vector normalisasi tfidf.

Testing Rocchio

Proses *testing Rocchio* telah dirancang pada Gambar 4.4. Berdasarkan pada Gambar 4.4 proses *testing Rocchio* terdiri dari proses perhitungan jarak antara data uji dengan *centroid* kelas seperti yang dikodekan pada baris 306. `$v` adalah

sigma hasil kuadrat dari pengurangan komponen centroid kelas dengan komponen vector data uji, kelas tweet ditentukan berdasarkan pada jarak minimal antara tweet data uji dengan *centroid* kelas seperti yang dikodekan pada baris 309-312. Jika jarak data uji dengan centroid kelas keluhan umum `($CUuji[1])` lebih kecil daripada jarak data uji dengan centroid kelas keluhan sopir (`$CUuji[2])` maka kembalikan 1 yaitu kelas keluhan umum, jika tidak kembalikan nilai 2 yaitu kelas keluhan sopir. Implementasi *testing Rocchio* di tunjukan pada Gambar 5.18.

```
297. $sql_centroid =
     mysql_query("select a.cUmum, a.cSopir, b.word
                  from rocchio_centroid a, rocchio_word b
                  WHERE a.word = b.id");
298. $CUuji = array(1=>0,0);
299. while($c=mysql_fetch_array($sql_centroid)){
300.  if(array_key_exists($c['word'], $Normalisasi_TFIDF )){
301.   $CUuji[1] +=
            pow($c['cUmum'] -
   $Normalisasi_TFIDF[$c['word']]['kelas_uji'][$i] , 2);
302.   $CUuji[2] +=
             pow($c['cSopir'] -
   $Normalisasi_TFIDF[$c['word']]['kelas_uji'][$i] , 2);
303.         }
304.       }
305. foreach($CUuji as $k=>$v):
306.  $CUuji[$k] = sqrt($v);
307. endforeach;
308.
309. if($CUuji[1] < $CUuji[2]){
310.  return 1;
311. }
312. return 2;
```

Gambar 5.18 Implementasi perhitungan jarak

5.5 Implementasi Pengiriman Respons Tweet

Implementasi pengiriman respons tweet mengacu pada rancangan respons tweet Gambar 4.5. Respons tweet yang dikirimkan berdasarkan kelas dari *feedback* tweet. Jika kelas *feedback* tweet sudah diketahui maka dapat ditentukan tweet respons yang akan dikirimkan. Implementasi penentuan tweet respons ditunjukkan pada Gambar 5.19.

```
126. $id_tweet_respon=1;
127. if($kelas==1){
128.  $id_tweet_respon=1; //pujian
129. }elseif($kelas==3){
130.  $id_tweet_respon=2;//follow
131. }
132. if($kelas==2){
133. $kata_d['word'] = explode(' ',strtolower($tweets_baru));
134. $cek = klasifikasi_rocchio($kata_d);
135. if($cek==1){ //umum
      $id_tweet_respon=4;
136. }else{
      $id_tweet_respon=3;
137. }
138. }
```

Gambar 5.19 Implementasi penentuan tweet respons

Berdasarkan Gambar 5.19 kode baris 127-128 jika kelas tweet (`$kelas`) bernilai 1 (atau pujian) maka respons dengan id_tweet_respon = 1. Kode baris 129-130 jika kelas tweet bernilai 3 (atau *follow*) maka responnya yang dikirimkan adalah respons yang memiliki id_tweet_respon = 2. Untuk kelas yang bernilai 2 (atau keluhan) diklasifikasikan lagi menggunakan metode *Rocchio Classifier*. Variabel `$cek` pada kode baris 137 akan mengembalikan nilai kelas tweet berupa 1 atau 2. Jika `$cek` (atau kelas tweet) bernilai 1 (atau keluhan umum) maka respons yang dikirimkan adalah respons yang memiliki id_tweet_respon=4. Jika `$cek` bernilai 2 (atau keluhan sopir) maka respons yang digunakan adalah respons yang memiliki id_tweet_respon = 3. Proses pengiriman respons tweet membutuhkan Twitter API. Implementasi pengiriman *respons* dengan memanfaatkan Twitter API ditunjukkan pada Gambar 5.20.

```
126. $url = 'https://api.twitter.com/1.1/statuses/update.json';
127. $requestMethod = 'POST';
128. $postfields = array(
     "status" => "@".$screen_name." ".$d['tweetresponse'],
     "trim_user"=>true,
     "in_reply_to_status_id"=> "%40".$screen_name
129. );
130. $twitter = new TwitterAPIExchange($settings);
131. $text_tweet_mention = $twitter->buildOauth($url,
     $requestMethod)->setPostfields($postfields)->
     performRequest();
132. $text_tweet_mention_json_decode =
     json_decode($text_tweet_mention);
```

Gambar 5.20 Implementasi pengiriman respons

Berdasarkan Gambar 5.20 tweet respons yang dikirimkan terdapat pada kode baris 128 berupa status yang berisi *username* pengirim *feedback* dan `$d['tweetresponse']` sebagai tweet respons yang tersimpan dalam database.

BAB VI
HASIL DAN PEMBAHASAN

Bagian ini membahas mengenai proses pengujian sistem untuk mengetahui perfomansi sistem yang telah dibangun. Proses pengujian dilakukan dengan pengujian *white box* dan evaluasi *classifier*. Evaluasi ini dilakukan dengan menghitung akurasi dari *Naïve Bayes Classifier* dan *Rocchio Classifier* dalam membangun *classifier* (atau model klasifikasi). Selain itu dilakukan pengujian mengenai responsyang dikirimkan oleh sistem. Pengujian ini dilakukan dengan mem-*posting* tweet yang mengandung mention akun Twitter Express_Group.

6.1 Data

Data tweet yang digunakan adalah tweet yang mengandung *mention* @Express_Group. Jumlah data yang berhasil dikumpulkan sebanyak 1291 tweet. Sebanyak 1291 tweet kemudian melalui proses *preprocessing*. Hasil atau *output* proses *preprocessing* sebanyak 981 data, kemudian disimpan dalam tabel clean_tweet. Data clean_tweet digunakan untuk proses *training* dan *testing*.

Data untuk klasifikasi dengan NBC

Dari 981 yang ada, 785 data dijadikan data *training* NBC dan 196 data dijadikan sebagai data *testing*, selebihnya tidak diolah karena data tersebut mengandung *feedback* yang sama, dengan kata lain *feedback* tersebut merupakan *retweet* dari tweet yang pernah di-*posting* sebelumnya. Jumlah data *training* yang digunakan dalam masing-masing kelas ditunjukkan pada Tabel 6.1.

Tabel 6.1 Data *training* NBC masing-masing kelas

Kelas	Jumlah Tweet
Pujian	113
Keluhan	356
Follow	29
Unknown	287

Adapun data yang digunakan untuk proses *testing* sebanyak 196 data *tweet*. Jumlah data *testing* NBC untuk masing-masing kelas ditunjukkan pada Tabel 6.2.

Tabel 6.2 Jumlah data *testing* NBC masing-masing kelas

Kelas	Jumlah Tweet
Pujian	43
Keluhan	95
Follow	10
Unknown	48

Data untuk klasifikasi *Rocchio*

Data yang digunakan dalam klasifikasi *Rocchio* adalah data *training* pada metode NBC yang memiliki label keluhan. Data *training Rocchio* sebanyak 356 data. Jumlah data *training* untuk masing-masing kelas ditunjukkan pada Tabel 6.3. Data *testing* untuk klasifikasi *Rocchio* pun menggunakan data *testing* untuk metode NBC yang memiliki label keluhan. Data *testing Rocchio* sebanyak 95 data. Jumlah data *testing* untuk masing-masing kelas ditunjukkan pada Tabel 6.4.

Tabel 6.3 Jumlah data *training Rocchio* masing-masing kelas

Kelas	Jumlah Tweet
Keluhan umum	126
Keluhan sopir	230

Tabel 6.4 Jumlah data *testing Rocchio* masing-masing kelas

Kelas	Jumlah Tweet
Keluhan umum	25
Keluhan sopir	70

6.2 Pengujian *White Box*

Pengujian *white box* digunakan untuk mendeteksi kesalahan logis dalam kode program. Pengujian *white box* dilakukan dengan mencocokan hasil perhitungan secara manual dengan hasil perhitungan yang didapatkan oleh sistem.

Pengujian NBC

Jumlah data *training* setelah melalui tahap *preprocessing* adalah sebanyak 798 *tweet*. Keseluruhan data digunakan untuk proses *training* dan *testing*. pada

proses *testing* NBC digunakan data sebanyak 174 data tweet. Contoh tweet yang digunakan dalam proses *testing* ditunjukkan pada Tabel 6.5.

Tabel 6.5 Contoh data *testing* NBC

No.	*Feedback*
1	ramahnya sopir @Express_Group, nyaman dah perjalanan gue...

Data *testing* selanjutnya diolah pada proses *preprocessing*, dan menghasilkan data pada Tabel 6.4.

Tabel 6.6 Hasil *preprocessing* data *testing* NBC

No.	Hasil *Preprocessing*
1.	Ramah nyaman

Penentuan kelas dari suatu tweet membutuhkan model klasifikasi. Model klasifikasi (probabilitas bersyarat) untuk masing-masing kata pada contoh data testing ditunjukkan pada Tabel 6.5.

Tabel 6.7 Probabilitas bersyarat

Kata	Probabilitas Bersyarat	
	Pujian	Keluhan
ramah	0.011648223645894	0.001205182283820
nyaman	0.016889924286546	0.000602591141910
Kata	Probabilitas Bersyarat	
	Follow	Unknown
sopir	0.000819000819001	0.000828843762951
pintu	0.000602591141910	0.000414421881475

Proses selanjutnya adalah perhitungan manual nilai probabilitas prior setiap kelas berdasarkan persamaan (3.4). Jumlah tweet masing-masing kelas ditunjukkan pada Tabel 6.1.

$$P(pujian) = \frac{N_{pujian}}{N} = \frac{113}{785} = 0.14394904458599$$

$$P(keluhan) = \frac{N_{keluhan}}{N} = \frac{356}{785} = 0.45350318471338$$

$$P(follow) = \frac{N_{follow}}{N} = \frac{29}{785} = 0.03694267515924$$

$$P(unknown) = \frac{N_{unknown}}{N} = \frac{287}{785} = 0.36560509554140$$

Hasil perhitungan manual probabilitas prior setiap kelas kemudian dibandingkan dengan nilai probabilitas prior setiap kelas yang dihasilkan oleh sistem. Hasil probailitas prior setiap kelas perhitungan sistem ditunjukkan pada Gambar 6.1. Berdasarkan pada Gambar 6.1 dapat diketahui bahwa nilai probabilitas setiap kelas memiliki nilai yang sama dengan hasil perhitungan manual probabilitas prior setiap kelas.

› Probabilitas Prior

No	Kelas	Prior
1	Pujian	0.14394904458599
2	Keluhan	0.45350318471338
3	Follow	0.03694267515924
4	Unknown	0.36560509554140

Gambar 6.1 Nilai probabilitas prior setiap kelas

Proses berikutnya menghitung probabilitas posterior kelas sesuai dengan persamaan (3.4). Perhitungan probabilitas posterior kelas melibatkan nilai probabilitas bersayarat. Nilai probabilitas bersyarat kata terhadap kelas ditunjukkan pada Tabel 6.5.

Perhitungan probabilitas posterior *tweet* pada Tabel 6.3.

$$P(pujian|d) = P(pujian) \prod_{1 \le k \le n_d} P(t_k|pujian)$$

= (0.14394904458599) x ((0.011648223645894) x (0.016889924286546))

= 0.00002832019178

$$P(keluhan|d) = P(keluhan) \prod_{1 \le k \le n_d} P(t_k|pujian)$$

= (0.45350318471338) x ((0.001205182283820) x (0.000602591141910))

= 0.00000032934860

$$P(follow|d) = P(follow) \prod_{1 \le k \le n_d} P(t_k|pujian)$$

= (0.03694267515924) x ((0.000819000819001) x (0.000819000819001))

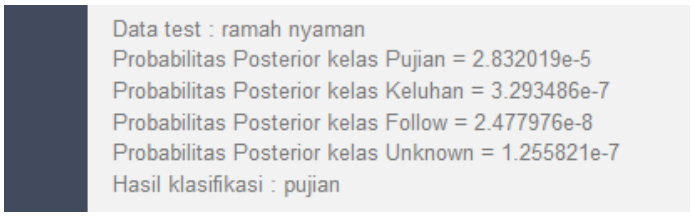
= 0.00000002477976

$$P(unknown|d) = P(follow) \prod_{1 \le k \le n_d} P(t_k|pujian)$$

= (0.36560509554140) x ((0.000828843762951) x (0.000414421881475))

= 0.00000012558206

Probabilitas posterior kelas yang memilki nilai maksimum atau c_{map} kemudian dicari dari nilai probabilitas posterior kelas yang ada, kelas yang memiliki nilai probabilitas posterior kelas maksimum adalah kelas pujian. Maka *feedback tweet* pada Tabel 6.3 termasuk dalam kelas pujian. Nilai-nilai probabilitas posterior yang dihasilkan sama dengan perhitungan sistem. Hasil perhitungan probabilitas posterior sistem ditunjukkan pada Gambar 6.2.



Gambar 6.2 Perhitungan probabilitas posterior sistem

Rocchio Classifier

Pengujian *white box* klasifikasi dengan *Rocchio Classifier* menggunakan data yang ditunjukkan pada Tabel 6.8. Dalam proses *training Rocchio* terdapat proses perhitungan *centroid* kelas. Perhitungan *centroid* kelas berdasarkan pada persamaan (3.8). Perhitungan *centroid* melibatkan vector dokumen dari data *training*. *Vector* dokumen menggunakan pembobotan tfidf. Vector setiap dokumen ditunjukkan pada Tabel 6.9.

Tabel 6.8 Data tweet

Data set	No. dokumen (atau tweet)	Kata dalam tweet	Kelas keluhan
Training set	d1	sopir bicara tidaksopan balap	sopir
	d2	sopir balap tidaknyaman	sopir
	d3	sopir balap marah marah lempar dompet tidaksopan	sopir
	d4	AC tidakrasa	umum
	d5	taksi tidakrawat	umum
Data uji	d6	balap tidaknyaman sopir tidaksopan	?

Tabel 6.9 Vector dokumen

Kata	tfidf					
	Kel.sopir			Kel.umum		?
	d1	d2	d3	d4	d5	d6
taksi	0	0	0	0	0.70	0
tidakrawat	0	0	0	0	0.70	0
ac	0	0	0	0.70	0	0
tidakrasa	0	0	0	0.70	0	0
sopir	0.25	0.28	0.12	0	0	0.25
balap	0.25	0.28	0.12	0	0	0.25
marah	0	0	0.78	0	0	0
lempar	0	0	0.39	0	0	0
dompet	0	0	0.39	0	0	0
tidaksopan	0.46	0	0.22	0	0	0.46
tidaknyaman	0	0.91	0	0	0	0.80
bicara	0.81	0	0	0	0	0

Selanjutnya dilakukan perhitungan *centroid* berdasarkan pada persamaan (3.8).

$$\vec{\mu}(c) = \frac{1}{|D_c|}\sum_{d \epsilon D_c} \vec{v}\,(d)$$

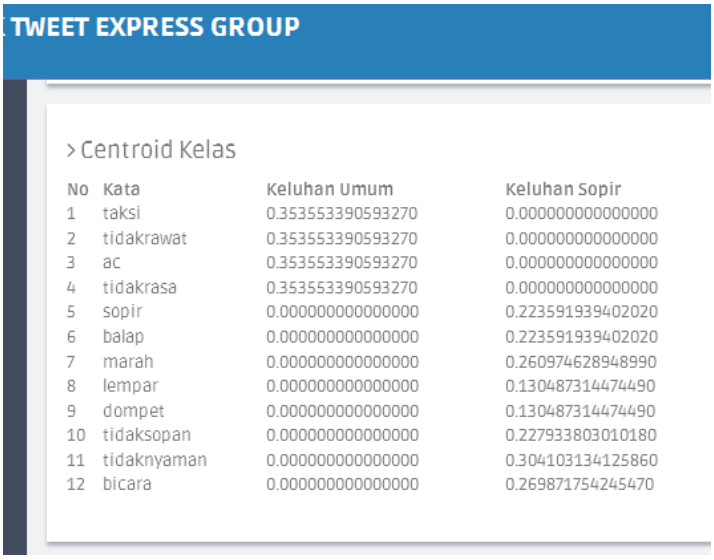
$$\vec{\mu}(keluhan\ sopir) = \frac{1}{3}\,[0\ \ 0\ \ 0\ \ 0\ \ 0{,}6\ \ 0{,}6\ \ 0{,}78\ \ 0{,}39\ \ 0{,}39\ \ 0{,}68\ \ 0{,}93\ \ 0{,}82]$$

$$= [0\ \ 0\ \ 0\ \ 0\ \ 0{,}22\ \ 0{,}22\ \ 0{,}26\ \ 0{,}13\ \ 0{,}13\ \ 0{,}22\ \ 0{,}30\ \ 0{,}27]$$

$$\vec{\mu}(keluhan\ umum) = \frac{1}{2}\,[0{,}7\ \ 0{,}7\ 0{,}7\ \ 0{,}7\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$

$$= [0{,}35\ \ 0{,}35\ \ 0{,}35\ \ 0{,}35\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]$$

Nilai centroid kelas yang dihasilkan dalam perhitungan manual sama dengan nilai centroid kelas yang dihasilkan oleh sistem. Nilai *centroid* perhitungan sistem ditunjukkan pada Gambar 6.3.



No	Kata	Keluhan Umum	Keluhan Sopir
1	taksi	0.353553390593270	0.000000000000000
2	tidakrawat	0.353553390593270	0.000000000000000
3	ac	0.353553390593270	0.000000000000000
4	tidakrasa	0.353553390593270	0.000000000000000
5	sopir	0.000000000000000	0.223591939402020
6	balap	0.000000000000000	0.223591939402020
7	marah	0.000000000000000	0.260974628948990
8	lempar	0.000000000000000	0.130487314474490
9	dompet	0.000000000000000	0.130487314474490
10	tidaksopan	0.000000000000000	0.227933803010180
11	tidaknyaman	0.000000000000000	0.304103134125860
12	bicara	0.000000000000000	0.269871754245470

Gambar 6.3 *Centroid* kelas

Proses selanjutnya adalah mengklasifikasi kelas data uji. Data uji yang digunakan ditunjukkan pada Tabel 6.8. Data uji direpresentasikan dengan vector. Vector data uji ditunjukkan pada Tabel 6.9. Penentuan kelas menggunakan *Rocchio* adalah kelas yang memiliki jarak terpendek dengan vetor data uji. Berikut ini akan dihitung jarak antara *vector* tweet "balap tidaknyaman sopir tidaksopan" dengan masing-masing *centroid* setiap kelas, dengan asumsi kelas keluhan sopir menggunakan variabel *ks* dan kelas keluhan umum menggunakan variable *ku*.

$$|\vec{\mu}_{ks} - \vec{d}_6|$$

$$= \sqrt{\begin{matrix} |0-0|^2 + |0-0|^2 + |0-0|^2 + |0-0|^2 + |0{,}22-0.25|^2 + |0{,}22-0.25|^2 + \\ |0{,}26-0|^2 + |0.13-0|^2 + |0{,}13-0|^2 + |0{,}22-0{,}46|^2 + |0.30-0.80|^2 + |0.26-0|^2 \end{matrix}}$$

$$= \sqrt{0{,}487} = 0.69$$

$$|\vec{\mu}_{ku} - \vec{d}_6|$$

$$= \sqrt{\begin{matrix} |0{,}35-0|^2 + |0.35-0|^2 + |0{,}35-0|^2 + |0{,}35-0|^2 + |0-0{,}25|^2 + \\ |0-0.25|^2 + |0-0|^2 + |0-0|^2 + |0-0|^2 + |0-0.46|^2 + |0-0.80|^2 + |0-0|^2 \end{matrix}}$$

$$= \sqrt{1.5} = 1.22$$

Berdasarkan perhitungan, jarak minimal adalah jarak *centroid* kelas keluhan sopir dengan dokumen baru, oleh karena itu dokumen baru "balap tidaknyaman sopir tidaksopan" termasuk kedalam kelas keluhan sopir. Jarak yang dihasilkan

dalam perhitungan manual kemudian dibandingkan dengan nilai hasil perhitungan jarak yang dilakukan oleh sistem. Jarak antara data uji dengan *centroid* kelas perhitungan sistem ditunjukkan pada Gambar 6.4. Perhitungan jarak secara manual dan perhitungan jarak sistem memiliki hasil yang sama.

Gambar 6.4 Perhitungan jarak

6.3 Pengujian Akurasi *Naïve Bayes Classifier*

Pengujian akurasi model klasifikasi bertujuan untuk mengukur seberapa besar efektifitas model klasifikasi dalam menentukan kelas tweet. Efektifitas model klasifikasi diukur dengan menghitung *precision, recall, f-measure*. Perhitungan *precision, recall, f-measure* menggunakan data uji sebanyak 196 *feedback* tweet. Nilai *precision, recall, f-measure* ditunjukkan pada Tabel 6.10.

Tabel 6.10 Nilai *precision, recall, f-measure* NBC

Kelas	*Precision*	*Recall*	*F-Measure*
Pujian	85.00%	79.07%	81.93%
Keluhan	84.31%	90.53%	87.31%
Follow	100.00%	70.00%	82.35%
Unknown	76.60%	75.00%	75.79%
Rata-rata	85.72%	78.65%	81.84%

Berdasarkan pada Tabel 6.10 dapat diketahui bahwa kelas *follow* memiliki *precision* tertinggi dengan nilai *precision* mencapai 100% dan kelas *unknown* memiliki *precision* terendah dengan nilai *precision* mencapai 76.60%. Kelas *follow*

memiliki *recall* terendah dengan nilai *recall* mencapai 70% dan kelas keluhan memiliki *recall* tertinggi dengan nilai *recall* mencapai 90.53%. Kelas *unknown* memiliki *f-measure* terendah dengan nilai *f-measure* mencapai 75.79% dan kelas keluhan memiliki *f-measure* tertinggi dengan nilai *f-measure* mencapai 87.31%.

Selain menggunakan *precision, recall* dan *f-measure*, efektifitas model klasifikasi juga diukur dengan akurasi. Akurasi model klasifikasi dihitung menggunakan persamaan (3.14). Nilai akurasi model klasifikasi dengan menggunakan data uji sebanyak 196 tweet sebesar 83.16%.

6.4 Pengujian Performansi *Rocchio Classifier*

Pengujian akurasi *Rocchio Classifier* bertujuan untuk mengukur seberapa besar efektifitas model klasifikasi menentukan kelas tweet. Efektifitas model klasifikasi diukur dengan menghitung *precision, recall, f-measure*. Perhitungan *precision, recall, f-measure* menggunakan data uji *Naïve Bayes Classifier* dengan label keluhan sebanyak 95 tweet. Nilai *presicion, recall, f-measure* ditunjukkan pada Tabel 6.11.

Tabel 6. 11 Evaluasi *Rocchio Classifier*

Kelas	*Precision*	*Recall*	*F-Measure*
Keluhan Umum	87.50%	28.00%	42.42%
Keluhan Sopir	79.31%	98.57%	87.90%
Rata-rata	83.40%	63.28%	65.16%

Berdasarkan pada Tabel 6.11 dapat diketahui bahwa kelas keluhan umum memiliki *precision* lebih besar daripada kelas keluhan sopir. *Recall* keluhan sopir lebih besar daripada *recall* keluhan sopir. *F-measure* kelas keluhan sopir lebih tinggi daripada kelas keluhan umum.

Selain menggunakan *precision, recall* dan *f-measure*, efektifitas model klasifikasi juga diukur dengan akurasi. Akurasi model klasifikasi dapat dihitung menggunakan persamaan (3.14). Nilai akurasi model klasifikasi *Rocchio Classifier* dengan menggunakan data uji sebanyak 95 *feedback* tweet sebesar 80%.

6.5 Pengujian Akurasi *Naïve Bayes Classifier* dan *Rocchio Classifier*

Selain dilakukan pengujian masing-masing *classifier* juga dilakukan pengujian akurasi *Naïve Bayes Classifier* dan *Rocchio Classifier* dengan pengujian *precision, recall, f-measure* yang ditunjukkan pada Tabel 6.12.

Tabel 6.12 Nilai *precision, recall, f-measure*

Kelas	*Precision*	*Recall*	*F-Measure*
Pujian	85.00%	79.07%	81.93%
Keluhan Umum	66.67%	24.00%	35.29%
Keluhan Sopir	69.15%	92.86%	79,27%
Follow	100.00%	70.00%	82.35%
Unknown	76.09%	72.92%	74.47%
Rata-rata	79.38%	67.77%	70.66%

Berdasarkan pada Tabel 6.12 dapat diketahui bahwa kelas *follow* memiliki *precision* tertinggi dan kelas keluhan umum memiliki *precision* terendah. Kelas keluhan umum memiliki *recall* terendah dan kelas keluhan sopir memiliki *recall* tertinggi. Kelas keluhan umum memiliki *f-measure* terendah dan kelas follow memiliki *f-measure* tertinggi.

Selain menggunakan *precision, recall* dan *f-measure,* efektifitas model klasifikasi juga diukur dengan akurasi. Akurasi model klasifikasi dihitung menggunakan persamaan (3.14). Nilai akurasi model klasifikasi dengan menggunakan data uji sebanyak 196 tweet sebesar 75%. Sebagai perbandingan dilakukan percobaan dengan berbagai jumlah data set. Akurasi dari berbagai jumlah data set ditunjukkan pada Tabel 6.13.

Tabel 6.13 Varian data set

Data Set	Data training	Data testing	Waktu	Akurasi	Akurasi Rata-rata
981	785	196	5.2215s	75%	70.67%
500	400	100	4.8763s	77 %	
250	200	50	2.4511s	60%	

Berdasarkan pada Tabel 6.13 mencapai akurasi tertinggi ketika menggunakan data set sebanyak 500 tweet, nilai akurasi mencapai 77% dan akurasi terendah ketika menggunakan data set sebanyak 250 tweet dengan nilai akurasi mencapai 60%. Akurasi rata-rata mencapai 70.67%.

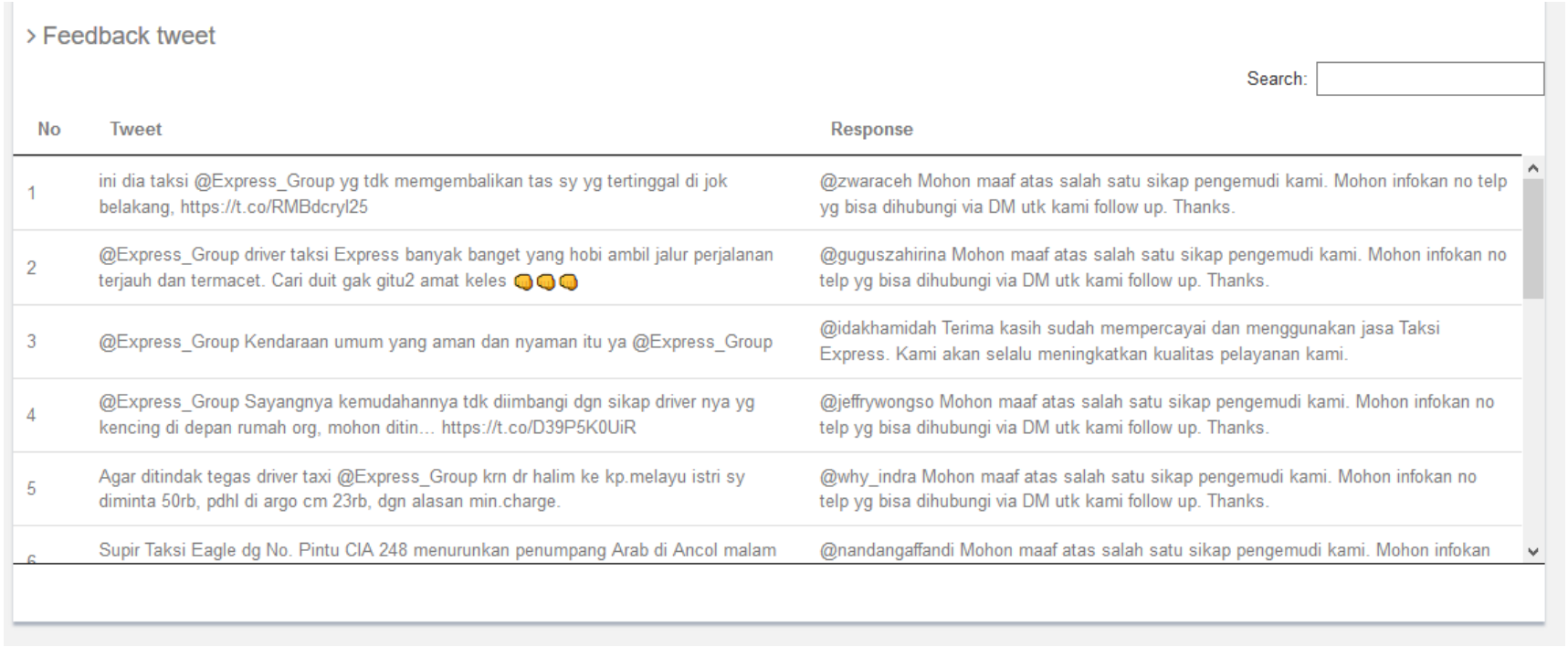
Tabel 6.14 Perbandingan dengan metode NBC-NBC

Percobaan	Metode	Data Set	Data training	Data testing	Akurasi	Akurasi Rata-rata
I	NBC - Rocchio	981 500 250	785 400 200	196 100 50	75% 77 % 60%	70.67%
II	NBC - NBC	981 500 250	785 400 200	196 100 50	76.00% 75.00% 58%	69.67%

Sebagai perbandingan dilakukan percobaan dengan menggunakan metode NBC-NBC. Berdasarkan pada Tabel 6.14 percobaan I memiliki akurasi rata-rata sebesar 70.67% dan percobaan II memiliki akurasi rata-rata sebesar 69.67%. Percobaain I memiliki akurasi lebih tinggi daripada percobaan II.

6.6 Pengujian Respons

Pengujian respons adalah pengujian terhadap respons yang dikirimkan. Pengujian ini dilakukan untuk mengetahui apakah respons yang dikirimkan sistem berkesinambungan dengan tweet yang diterima oleh pihak Espress Group. Sebanyak 15 tweet digunakan untuk menguji sistem dalam pengiriman respons. Tampilan tweet dan respons pada aplikasi di tunjukan pada Gambar 6.5. Sejumlah 15 (lima belas) tweet yang ada tidak semua tweet di respons dengan benar oleh sistem, artinya ada tweet yang responsnya tidak berkesinambungan dengan tweet. Ada 2 tweet yang responnya tidak berkesinambungan. Dua (2) tweet tersebut merupakan kelas keluhan yang dikenali oleh NBC, kemudian *feedback* tweet tersebut diolah lagi menggunakan *Rocchio*, tetapi *Rocchio* tidak mengklasifikasi *feedback* secara benar, sehingga respons yang dikirimkan oleh sistem tidak berkesinambungan. Tweet yang direspons secara benar oleh sistem sebanyak 13 tweet, jadi akurasi sistem dalam memberikan respons sebesar 86.67%.



Gambar 6.5 Tampilan tweet dan respons sistem

BAB VII
KESIMPULAN DAN SARAN

Kesimpulan

Berdasarkan pada hasil penelitian dan pembahasan yang dilakukan maka diperoleh kesimpulan sebagai berikut :

1. Sistem respons tweet otomatis menggunakan metode *Naïve Bayes Classifier* dan *Rocchio Classifier* dengan studi kasus Express Group dapat dibangun. Sistem secara otomatis dapat merespons tweet yang diterima oleh Express Group.
2. Respons yang dikirim secara otomatis memiliki akurasi sebesar 86.67%.
3. Evaluasi kombinasi *Naïve Bayes Classifier* dan *Rocchio Classifier* mencapai rata-rata akurasi sebesar 70.67%, sedangkan evaluasi metode *Naïve Bayes Classfier* dan *Naïve Bayes Classifier* mencapai rata-rata akurasi sebesar 69,67%.

Saran

Pengembangan penelitian lebih lanjut diberikan saran sebagai berikut :

1. Sistem respons tweet otomatis hanya dapat merespons tweet berbahasa Indonesia. Diharapkan penelitian selanjutnya agar dapat merespons tweet bahasa asing.
2. Sistem respons tweet otomatis fitur yang digunakan adalah unigram, untuk penelitian selanjutnya bisa menggunakan n-gram yang diharapkan dapat menambah akurasi model klasifikasi.
3. Sistem respons tweet otomatis belum menggunakan POS Tagger, untuk penelitian selanjutnya dapat menerapkan POS tagger yang diharapkan dapat menambah akurasi model klasifikasi.

DAFTAR PUSTAKA

Adi, S., 2014, Klasifikasi Data Nap (Nota Analisis Pembiayaan) Untuk Prediksi Tingkat Keamanan Pemberian Kredit, *Tesis*, Jurusan JIKE FMIPA UGM, Yogyakarta.

Azis, I., 2015, Jumlah Pengguna Twitter di Indonesia 50 Juta Orang, *http://sidomi.com/368824/jumlah-pengguna-twitter-di-indonesia-50-juta-orang/*, diakses tanggal 15 Mei 2015.

Darujati, C., 2010, Perbandingan Klasifikasi Dokumen Teks Menggunakan Metode Naïve Bayes dengan K-Nearest Neighbor, *Jurnal Link*, 1, 13, 1-8.

Devi, P. dan Ranjan, R.P, 2014, Classification with K-means clustering and Decision Tree, *Internasional Journal of Enginerring Sciences & Research Technology*, Juli 2014, 775-779.

Dilrukshi, I., Zoysa, K., dan Caldera, A., 2013, Twitter News Classification Using SVM. *The 8th International Conference on Computer Science & Education (ICCSE 2013)* (hal. 287-291). Colombo, IEEE Conference Publications, Sri Lanka.

Feldman, R. dan Sanger, J., 2007, *The Text Mining Handbook : Advanced Approach in Analyzing Unstructured Data*, Cambridge University Press, New York.

Han, J., dan Kamber, M., 2006, *Data Mining : Concepts and Techniques (2nd edition),* Morgan Kaufmann, USA.

Hodeghatta, U. R., 2013, Sentiment analysis of Hollywood movies on Twitter, *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*, 1401–1404.

Hu, X. dan Liu, H., 2012, *Mining Text Data.* Phoenix, Springer US, Amherst.

Laksana, J. dan Purwarianti, A., 2014, Indonesian Twitter Text Authority Classification For Government in Bandung, *International Conference of Advanced Informatics: Concept and Application*, 129-134.

Learned, E. G. dan Miller, 2011, *Supervised learning and bayesian Classification,* Department of Computer Science, Amherst.

Lestari, N. M. A., Putra, I. K. G. D. dan Cahyawan, A. A. K., 2013, Personality Types Classification for Indonesian Text in Partners Searching Website Using Naïve Bayes Methods, *IJCSI International Journal of Computer Science*, 1-8.

Liliana, D. Y., Hardianto, A. dan Ridok, M., 2011, Indonesian News Classification using Support Vector Machine, *International Scholarly and Scientific Research & Innovation*, 621–624.

Ling, J., Kencana, I. E. dan Oka, T. B., 2014, Analisis Sentimen Menggunakan Metode Naive Bayes Classifier dengan Seleksi Fitur Chi Square, *E-Jurnal Matematika Vol.3*, 92-99.

Liu, B., 2011, *Web Data Mining in Exploring Hyperlinks, Contents, and Usage Data*, second edition, Springer, Verlag Berlin Heidelberg.

Lumbanraja, F.R., 2013, Sistem Pencarian Data Teks Dengan Menggunakan Metode Klasifikasi Rocchio (Studi Kasus : Dokumen Teks Skripsi). *Kumpulan Makalah Seminar Semirata*, Lampung.

Manning, C. D., Raghavan, P. dan Schutze, H, 2009, *An Introduction to Information Retrieval*, Cambridge University Press, Cambridge.

Natalius, S., 2010, Metoda *Naive* Bayes Classifier dan Penggunaannya pada Klasifikasi Dokumen, *Makalah II2092 Probabilitas dan Statistik – Sem. ITahun 2010/2011*, Program Studi Sistem dan Teknologi Informasi, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Prasetyo, E., 2014, *Data Mining*, Andi Offset, Yogyakarta.

Rascha, S., 2014, Naïve Bayes and Text Classification I - Introduction and Teory, *http://arxiv.org/abs/1410.5329*, 16 Oktober 2014, diakses pada 29 April 2015

Rodiyansyah, S. F., 2012, *Klasifikasi Posting Twitter Kemacetan Lalu Lintas Kota Bandung Menggun*akan Naive Bayesian Classification, *Tesis*, Jurusan JIKE FMIPA UGM, Yogyakarta.

Schapire R. E., Yoram S. dan Amit S., 1998, Boosting and Rocchio Applied to Text Filtering, *Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval*.

Sebastiani, F., 2002. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 1, 34, 1-47.

Shin A. Y., Sasano R., Takamura H. dan Manabu O., 2014, Context-Dependent Automatic Response Generation using Statistical Machine Translation Technique, *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, Juni 2015, 1345-1350.

Sordoni, A., Galley, M., Auli, M., Brocket,C., Ji, Y., Mitchell, M., Nie, J-Y., Gao, J. dan Dollan, B., 2015, A Neural Network Approach to Context-Sensitive Generation of Conversational Responses, *The 2015 Annual Conference of the North American Chapter of the ACL*, 31 Mei 2015 - 5 Juni 2015, 196-205.

Suruliandi, A., dan Selvaperumal, P., 2014, A Short Message Classification, *International Conference on Recent Trends in Information Technology,* IEEE Conference Pubication, Boston.

Thakur, B. dan Mann M., 2014, Data mining with Big Data using C4.5 and Bayesian Classifier. *International journal of Advanced Research in Computer Science and Sofware Engineering*, Agustus 2014, 959-962.

Togias, K. dan Kameas, A., 2012, An Ontology-based Representation of the Twitter REST API, *International Conference on Tools with Artifiacl Intellegent*, 998-1003.

Wibowo, A. P., 2015, Klasifikasi Kinerja Satpam Menggunakan Naïve Bayes Classifier, *Tesis*, Jurusan JIKE FMIPA UGM, Yogyakarta.

Widjojo, E.A., Rachmat, A.C., Santoso, G., 2014, Implementasi Rocchio's Classification dalam Mengkategorikan Renungan Harian Kristen, *ULTIMATICS*, 1, VI, 1-8.

Yusuf, A. dan Priambadha, T., 2013, Support Vector Machine yang Didukung K-Means Clustering dalam Klasifikasi Dokumen, Jurnal Ilmiah Teknologi Informasi, 1, 11, 13-16.