**TESIS**

**OPINION MINING PADA TWITTER UNTUK BAHASA INDONESIA DENGAN METODE SUPPORT VECTOR MACHINE DAN METODE BERBASIS LEXICON**

**OPINION MINING ON TWITTER FOR INDONESIAN USING SUPPORT VECTOR MACHINE AND LEXICON BASED METHOD**

****

**JAN KRISTANTO WIBISONO**

**10/306141/PPA/03230**

**PROGRAM STUDI S2 ILMU KOMPUTER**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS GADJAH MADA**

**YOGYAKARTA**

**2013**

**HALAMAN 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, 28, Juni 2013

JAN KRISTANTO WIBISONO

**HALAMAN PENGESAHAN**

**tesis**

**OPINION MINING PADA TWITTER UNTUK BAHASA INDONESIA DENGAN METODE SUPPORT VECTOR MACHINE DAN METODE BERBASIS LEXICON**

**oleh**

**Nama Mahasiswa**

**Nomor Pokok Mahasiswa (NPM)**

telah dipertahankan di depan Tim Penguji

pada tanggal ........(lengkapi *tanggal bulan tahun*)

**Susunan Tim Penguji**

**Ketua, Sekretaris,**

**Nama .......... Nama...................**

**NIP. .... NIP. ....**

**Penguji Utama, Penguji Sekretaris,**

**Nama .......... Nama...................**

**NIP. .... NIP. ....**

Diterima dan dinyatakan memenuhi syarat kelulusan pada tanggal .....

**Ketua Jurusan .....**

**Fakultas ....**

**Nama.............**

**NIP.................**

# KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas limpahan kasih, karunia, serta petunjuk-Nya sehingga tugas akhir berupa penyusunan tesis yang berjudul : “Opinion Mining Pada Twitter Untuk Bahasa Indonesia Dengan Metode Support Vector Machine Dan Metode Berbasis Lexicon” telah terselesaikan dengan baik.

Dalam penyusunan tugas akhir ini penulis telah banyak mendapatkan arahan, bantuan, serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapan terima kasih kepada:

* + - 1. Allah Bapa yang di surga atas segala berkat dan kasih-Nya.
      2. Ayah dan Ibu yang selalu memberikan dorongan, motivasi dan fasilitas.
      3. Drs. Edi Winarko, M.Sc., PhD. selaku Dosen Pembimbing yang telah berkenan meluangkan waktu, dan pikiran dalam penyelesaian tugas akhir ini.

1. Para Dosen Penguji yang telah berkenan memberikan saran dan masukan dalam penyelesaian tugas akhir ini.
2. Segenap Dosen dan civitas akademik di lingkungan Program Studi Magister Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada.
3. Pak Sugeng, Mbak Rini, Mas Kuncoro, dan Mas Nanang yang telah bersedia membantu mengurus keperluan penulis.
4. Rekan-rekan seperjuangan S2 Ilmu Komputer UGM atas dukungan dan motivasinya.

Akhirnya, Penulis berharap semoga penelitian ini menjadi sumbangsih yang bermanfaat bagi dunia sains dan teknologi di Indonesia, khususnya disiplin keilmuan yang Penulis dalami.

Yogyakarta, Juni 2013

Penyusun

# DAFTAR ISI

[KATA PENGANTAR v](#_Toc361718142)

[DAFTAR ISI vi](#_Toc361718143)

[DAFTAR TABEL x](#_Toc361718144)

[DAFTAR GAMBAR xi](#_Toc361718145)

[Intisari xiii](#_Toc361718146)

[ABSTRACT xiv](#_Toc361718147)

[BAB I 1](#_Toc361718148)

[PENDAHULUAN 1](#_Toc361718149)

[1.1 Latar Belakang 1](#_Toc361718150)

[1.2 Rumusan Masalah 2](#_Toc361718151)

[1.3 Batasan Masalah 2](#_Toc361718152)

[1.4 Tujuan Penelitian 3](#_Toc361718153)

[1.5 Manfaat Penelitian 3](#_Toc361718154)

[1.6 Metode Penelitian 3](#_Toc361718155)

[1.7 Sistematika Penulisan 5](#_Toc361718156)

[BAB II 7](#_Toc361718157)

[TINJAUAN PUSTAKA 7](#_Toc361718158)

[BAB III 11](#_Toc361718159)

[LANDASAN TEORI 11](#_Toc361718160)

[3.1 Natural Language Processing 11](#_Toc361718161)

[3.2 Part-of-speech 12](#_Toc361718162)

[3.2.1 Part-of-speech Tagging 12](#_Toc361718163)

[3.2.2 Tagset 13](#_Toc361718164)

[3.3 Opinion Mining 13](#_Toc361718165)

[3.4 Support Vector Machine 15](#_Toc361718166)

[3.4.1 Konsep SVM 15](#_Toc361718167)

[3.4.2 SVM linear 16](#_Toc361718168)

[3.4.3 Hyperplane SVM 18](#_Toc361718169)

[3.4.4 SVM Nonlinear 21](#_Toc361718170)

[3.5 TF-IDF 23](#_Toc361718171)

[BAB IV 25](#_Toc361718172)

[ANALISIS DAN RANCANGAN SISTEM 25](#_Toc361718173)

[4.1 Analisis Sistem 25](#_Toc361718174)

[4.1.1 Deskripsi Sistem 25](#_Toc361718175)

[4.2 Analisis Data 27](#_Toc361718176)

[4.3 Preprocessing 27](#_Toc361718177)

[4.3.1 Clear Invalid UTF-8 28](#_Toc361718178)

[4.3.2 Casefolding 28](#_Toc361718179)

[4.3.3 Remove Symbol 28](#_Toc361718180)

[4.3.4 Processing Numbers 28](#_Toc361718181)

[4.3.5 Remove Repeat 29](#_Toc361718182)

[4.3.6 Replace Slang 30](#_Toc361718183)

[4.3.7 Spelling Correction 30](#_Toc361718184)

[4.4 Penentuan Label Data Training 31](#_Toc361718185)

[4.4.1 POS taging 31](#_Toc361718186)

[4.4.2 Opinion Rules 32](#_Toc361718187)

[4.4.3 Check Negation 33](#_Toc361718188)

[4.5 Pelatihan 34](#_Toc361718189)

[4.5.1 Generate Weight 35](#_Toc361718190)

[4.5.2 Traning SVM 36](#_Toc361718191)

[4.6 Penentuan Label dengan SVM 36](#_Toc361718192)

[4.6.1 Pencarian Tweets 37](#_Toc361718193)

[4.6.2 Cek Frase 38](#_Toc361718194)

[4.6.3 Generate bobot 38](#_Toc361718195)

[4.6.4 Penentuan label dengan SVM 38](#_Toc361718196)

[4.6.5 Pemetaan dalam sentimen 38](#_Toc361718197)

[4.7 Arsitektur Sistem 39](#_Toc361718198)

[4.8 Rancangan Sistem 41](#_Toc361718199)

[4.8.1 Class Diagram 41](#_Toc361718200)

[4.8.2 Sequence diagram proses pencarian data twitter 43](#_Toc361718201)

[4.8.3 Sequence diagram preprocessing 43](#_Toc361718202)

[4.8.4 Sequence diagram proses pemberian label dengan metode lexicon 44](#_Toc361718203)

[4.8.5 Sequence diagram proses pemberian label dengan SVM 45](#_Toc361718204)

[BAB V 46](#_Toc361718205)

[IMPLEMENTASI 46](#_Toc361718206)

[5.1 Preprocessing Data 46](#_Toc361718207)

[5.1.1 clearInvalidUTF8 46](#_Toc361718208)

[5.1.2 Casefolding 47](#_Toc361718209)

[5.1.3 removeSymbol 47](#_Toc361718210)

[5.1.4 ProcessNumbers 48](#_Toc361718211)

[5.1.5 RemoveRepeat 49](#_Toc361718212)

[5.1.6 *Replace* *informal* *word* 50](#_Toc361718213)

[5.1.7 *Spelling* *Corrention* 50](#_Toc361718214)

[5.2 POS taging 51](#_Toc361718215)

[5.3 Pelabelan menggunakan metode *lexicon* *based* 52](#_Toc361718216)

[5.4 Pembobotan data 57](#_Toc361718217)

[5.5 Pelatihan 61](#_Toc361718218)

[5.6 Pencarian *Tweet* 61](#_Toc361718219)

[5.7 Pengecekan frase opini 62](#_Toc361718220)

[5.8 Pelabelan dengan metode SVM 63](#_Toc361718221)

[BAB VI 65](#_Toc361718222)

[HASIL PENELITIAN DAN PEMBAHASAN 65](#_Toc361718223)

[6.1 Perbandingan proses pelabelan data pelatihan 65](#_Toc361718224)

[6.1.1 Hasil perbandingan pelabelan data pelatihan 65](#_Toc361718225)

[6.2 Pemberian Label 66](#_Toc361718226)

[6.3 Akurasi Pemberian Label 70](#_Toc361718227)

[BAB VII 71](#_Toc361718228)

[KESIMPULAN DAN SARAN 71](#_Toc361718229)

[7.1 Kesimpulan 71](#_Toc361718230)

[7.2 Saran 71](#_Toc361718231)

[DAFTAR PUSTAKA 73](#_Toc361718232)

[LAMPIRAN A 75](#_Toc361718233)

[CONTOH DATA TRAINING 75](#_Toc361718234)

[LAMPIRAN B 77](#_Toc361718235)

[CONTOH DATA TESTING 77](#_Toc361718236)

[LAMPIRAN C 78](#_Toc361718237)

[CONTOH SVM MODEL 78](#_Toc361718238)

# DAFTAR TABEL

[Tabel 2.1 Perbandingan penelitian lampau dengan sistem yang dibangun 9](#_Toc363558080)

[Tabel 3.1 Contoh Tags untuk bahasa Indonesia 13](#_Toc363558081)

[Tabel 3.2 Fungsi Kernel 23](#_Toc363558082)

[Tabel 4.1 Daftar Konversi 29](#_Toc363558083)

[Tabel 4.2 Contoh Konversi Angka 29](#_Toc363558084)

[Tabel 4.3 Contoh menghapus karakter berualng 30](#_Toc363558085)

[Tabel 4.4 Contoh Pengantian kata tidak baku 30](#_Toc363558086)

[Tabel 4.5 Contoh SpellCorrection 31](#_Toc363558087)

[Tabel 4.6 Opinion Rules 33](#_Toc363558088)

# DAFTAR GAMBAR

[Gambar 3.1 Decision Boundary 16](#_Toc361715974)

[Gambar 3.2 Dimensi Data 22](#_Toc361715975)

[Gambar 4.1 Proses Proses dalam Opinion Mining 26](#_Toc361715976)

[Gambar 4.2 Proses-proses dalam Tahap Pelatihan 35](#_Toc361715977)

[Gambar 4.3 Proses-proses dalam Tahapan Penentuan Label 37](#_Toc361715978)

[Gambar 4.4 Arsitektur Sistem 40](#_Toc361715979)

[Gambar 4.5 Class diagram 42](#_Toc361715980)

[Gambar 4.6 Sequence diagram proses pencarian data twitter 43](#_Toc361715981)

[Gambar 4.7 Sequence diagram preprocessing 44](#_Toc361715982)

[Gambar 4.8 Sequence diagram proses pemberian label dengan metode lexicon 44](#_Toc361715983)

[Gambar 4.9 Sequence diagram proses pemberian label dengan SVM 45](#_Toc361715984)

[Gambar 5.1 Kode method clearInvalidUTF8 47](#_Toc361715985)

[Gambar 5.2 Kode method casefolding 47](#_Toc361715986)

[Gambar 5.3 Kode method removeSymbol 48](#_Toc361715987)

[Gambar 5.4 Kode method processNumbers 49](#_Toc361715988)

[Gambar 5.5 kode dari proses RemoveRepeat 50](#_Toc361715989)

[Gambar 5.6 Kode proses penggantian kata tidak baku 50](#_Toc361715990)

[Gambar 5.7 Kode method correct pada class SpellingCorrection 51](#_Toc361715991)

[Gambar 5.8 Kode proses POS taging 52](#_Toc361715992)

[Gambar 5.9 pencocokan kelas kata dan frase 54](#_Toc361715993)

[Gambar 5.10 kode method analysisTwoWord 55](#_Toc361715994)

[Gambar 5.11 kode method analysisOneWord 55](#_Toc361715995)

[Gambar 5.12 kode method chcekNegation 56](#_Toc361715996)

[Gambar 5.13 Kode method conclusion 57](#_Toc361715997)

[Gambar 5.14 Kode untuk membangun data training 58](#_Toc361715998)

[Gambar 5.15 Kode method getIndex 59](#_Toc361715999)

[Gambar 5.16 Kode method getTfidf 60](#_Toc361716000)

[Gambar 5.17 Kode pemanggil method pelatihan 61](#_Toc361716001)

[Gambar 5.18 Kode untuk mencari data pada twitter 62](#_Toc361716002)

[Gambar 5.19 Kode untuk memeriksa frase opini 63](#_Toc361716003)

[Gambar 5.20 Kode pemanggilan libsvm 64](#_Toc361716004)

[Gambar 5.21 Kode untuk menyimpan hasil pelabelan 64](#_Toc361716005)

[Gambar 6.1 Perbandingan metode emoticon dan pelabelan manual 66](#_Toc361716006)

[Gambar 6.2 Perbandingan metode lexicon based dan pelabelan manual 66](#_Toc361716007)

[Gambar 6.3 Tampilan input kata kunci 67](#_Toc361716008)

[Gambar 6.4 link untuk melakukan preprocessing 67](#_Toc361716009)

[Gambar 6.5 sistem menampilkan data bersih 68](#_Toc361716010)

[Gambar 6.6 Data tweet setelah dilakukan analisis awal 69](#_Toc361716011)

[Gambar 6.7 Pembobotan tweet 69](#_Toc361716012)

[Gambar 6.8 Hasil pelabelan dengan metode SVM 70](#_Toc361716013)

[Gambar 6.9 Akurasi Sistem 70](#_Toc361716014)

# Intisari

**OPINION MINING PADA TWITTER UNTUK BAHASA INDONESIA DENGAN METODE SUPPORT VECTOR MACHINE DAN METODE BERBASIS LEXICON**

Oleh

JAN KRISTANTO WIBISONO

10/306141/PPA/03230

Dalam proses pengambilan keputusan opini atau pendapat orang lain sering dijadikan acuan. Dengan internet dan sosial media yang ada sekarang banyak orang sering berpendapat,mengeluh atau mengkritik tentang sesuatu hal. Sehingga untuk mengetahui pendapat orang lain tentang sesuatu hal kita tidak perlu bertanya kepada mereka satu per satu. Kita dapat mengembangkan metode yang dapat mencari dan menganalisa teks pada sosial media yang menghasilkan output berupa pendapat tentang sesuatu hal.

Metode untuk mendapatkan informasi berupa pendapat yang terdapat dalam suatu teks adalah “Opinion Mining” atau “Sentiment Analysis”. Pada penelitian ini menggunakan metode gabungan Unsupervised yang berbasis pada *lexicon* berbahasa indonesia dan *Supervised* yang berbasis pada *Support* *Vector* *Machine*.

Hasil yang diperoleh berupa polaritas sentimen apakah negatif , positif, ataupun netral. Dari hasil pengujian pada penelitian ini didapatkan akurasi 77,7%.

Kata kunci : *opinion mining, sentiment analysis, lexicon, support vector machine*

# ABSTRACT

**OPINION MINING ON TWITTER FOR INDONESIAN USING SUPPORT VECTOR MACHINE AND LEXICON BASED METHOD**

By

JAN KRISTANTO WIBISONO

10/306141/PPA/03230

Opinions in the decision-making process are often used as a reference. Nowadays, with the internet and social media there are a lot of people often argue, complain or criticize about something. So as to know what others think of something that we do not need to ask them one by one. We can develop a method that can search and analyze text on social media which produces output in the form an opinion about something.

The method to obtain information in the form of opinions contained in the text is an "Opinion Mining" or "Sentiment Analysis". In this research, using a combined Unsupervised method based on Indonesian Lexicon and Supervised based on Support Vector Machine.

Results obtained in the form of sentiment is negative polarity, positive, or neutral. From the test results obtained in this study 77,7% accuracy.

Keyword : *opinion mining, sentiment analysis, lexicon, support vector machine*

# 

# PENDAHULUAN

## Latar Belakang

Opini adalah pendapat pribadi yang tidak obyektif dan tidak melalui proses verifikasi (Quirk et al., 1985). Opini orang lain tentang suatu hal menjadi penting dalam pengambilan keputusan. Misalkan seseorang akan membeli sebuah mobil biasanya dia akan bertanya kepada orang lain bagaimana pendapat mereka tentang mobil tersebut. Atau sering juga ketika kita membeli produk tertentu, kita dipersilakan untuk mengisi kuisoner yang berkaitan dengan produk tersebut. Sekarang ini tidak perlu kita langsung bertanya tentang pendapat orang lain secara langsung. Seiring dengan perkembangan internet dan web, telah banyak orang yang berbagi informasi tentang pendapat dan pengalaman bahkan kritik mereka terhadap suatu produk tertentu.

*Social media* kini telah berkembang dengan pesat, dimulai dari awal Friendster sampai era sekarang ini Facebook, Twitter, Google+ dan Foursquare. Menurut [socialbakers.com](http://socialbakers.com) pengguna Facebook di Indonesia telah mecapai 42.5 juta, yang menjadikan Indonesia menjadi peringkat ke empat pengguna Facebook setelah Amerika, India dan Brazil. Pengguna Twitter di Indonesia juga tidak kalah besar jumlahnya yang mencapai 19.5 juta (Semiocast, 2012) yang menempati peringkat kelima setelah Amerika, Brazil, Japan dan U.K. Perkembangan yang begitu pesat ini membuat orang-orang berkomunikasi dengan mudah. Mereka saling berbagi informasi, baik berbagi tentang aktifitas sehari-hari mereka ataupun memberikan pemberitahuan pada komunitas mereka, bahkan keluhan mereka terhadap suatu hal.

Begitu populernya *social media*, pengguna *social media* sekarang ini bukan hanya kalangan pribadi atau perseorangan saja. Tapi kalangan *corporate* juga memanfaatkan *social media* sebagai media komunikasinya dengan masyarakat. Seperti pengenalan produk baru, penawaran produk, bahkan para legislative juga mempromosikan diri mereka melalui *social media*. Selain yang dianggap lebih murah dari segi biaya, juga lebih tepat pada sasaran.

Berdasarkan pada paparan diatas terdapat potensi besar pada social media untuk dilakukan pengekstrakan opini atau yang disebut *opinion mining,* untuk mengetahui pendapat orang lain tentang suatu hal, yang dapat dimanfaatkan selama proses pengambilan keputusan. Beberapa penelitian yang sudah ada sekarang ini kebanyakan menggunakan *machine learning* baik dengan naïve bayes ( (Alliandu dan Winarko, 2012) , (Pang et al., 2002), (Franky dan Manurung, 2008), (Go et al., 2009), (Pak dan Paroubek, 2010)) , Support Vector Machine ( (Pang et al., 2002) , (Franky dan Manurung, 2008), (Zhang et al., 2008), (Go et al., 2009)) dan Maximum Entropy (Go et al., 2009) ataupun yang berbasiskan lexicon (Komarsilam dan Winarko, 2012). Pada penelitian ini akan dilakukan *opinion mining* dengan mengkombinasikan antara pendekatan berbasis *machine learning* dan berbasiskan lexicon. Metode yang akan digunakan untuk mewakili pendekatan *machine learning* adalah Support Vector Machine (SVM), yang menurut paper (Zhang et al., 2008) sering memberikan akurasi yang terbaik dibanding dengan metode learning yang lainnya. Dengan melakukan kombinasi antara metode Support Vector Machine (SVM) dan berbasiskan lexicon diharapkan dapat meningkatkan akurasi pada proses pengklasifikasian opini.

## Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dipaparkan dapat dirumuskan masalah dalam penelitian ini adalah Bagaimana mengkombinasikan Support Vector Machine (SVM) dengan Lexicon Based untuk menentukan suatu kalimat beropini positif, negative atau netral.

## Batasan Masalah

Adapun yang menjadi batasan masalah pada penelitian ini adalah :

1. Data yang digunakan dalam penelitian ini berasal dari *social media* Twitter berbahasa Indonesia.
2. Untuk menentukan apakah suatu kalimat merupakan kalimat positif, negatif, maupun netral adalah dengan mengkombinasikan antara metode *lexicon-based* dan *learning-based*

## Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian ini adalah mengembangkan metode untuk melakukan *opinion mining* dalam bahasa Indonesia pada *social media* dengan mengkombinasikan metode *lexicon-based* dan *learning-based*.

## Manfaat Penelitian

Penelitian ini diharapkan dapat menghasil metode yang dapat mengekstrak opini dari *social media* yang dapat menjadi pertimbangan dalam proses pengambilan keputusan.

## Metode Penelitian

Metode penelitian yang akan dilakukan terdiri atas beberapa tahap sebagai berikut :

1. Studi Kepustakaan

Melakukan berbagai macam pengumpulan bahan referensi, seperti jurnal penelitian, tesis, buku-buku teori dan sumber lain termasuk informasi yang diperoleh dari internet. Mempelajari litelatur dan teori pendukung penelitian mengenai klasifikasi, khususnya SVM (Support Vector Machine).

1. Analisis Sistem

Melakukan analisa terhadap requirement yang dibutuhkan dalam pengembangan sistem.

1. Perancangan

Pada tahap ini dilakukan perancangan metode kombinasi antara *lexicon based*, dan *Support Vector Machine* (SVM) yang akan diimplementasikan untuk *Opinion mining*. Data-data yang diambil dalam penetian ini diambil dari twitter dengan memanfaatkan Twitter API yang sudah disediakan, selanjutnya data tersebut disimpan dalam basis data. Perancangan sistem meliputi *training* dan *testing*. Training sendiri terdapat beberapa tahap meliputi *preprocessing*, penentuan label dengan metode *lexicon based* dan proses *training* sendiri.

Data *training* pada penelitian ini menggunakan data *training* pada penelitian (Alliandu dan Winarko, 2012). Tahap preprocessing akan melakukan beberapa hal diantaranya :

1. *Clear invalid UTF8* yakni menghapus karakter-karakter yang tidak termasuk dalam UTF8.
2. *Casefolding*  yakni pengubahan setiap karakter huruf menjadi huruf kecil.
3. *removeSymbol* yakni pengeliminasian karakter-karakter ilegal dalam isi *tweet.*
4. *processNumbers* yakni mengganti karakter angka dengan huruf.
5. *removeRepeat* yakni menghapus perulangan karakter yang berurutan.
6. *Replacement* yakni penggantian kata *slang* dengan kata baku yang tersedia pada kamus.
7. *SpellingCorrection* yakni penggantian kata yang tidak terdapat dalam kamus dengan kata yang paling mendekati dalam kamus.

Setelah melewati tahap *preprocessing* maka data *tweet* akan ditentukan labelnya dengan menggunakan metode *lexicon based*. Dimana data *tweet* akan dikelompokan menjadi dalam 3 kelompok, yakni Positif, Negatif dan Netral. Kemudian data *tweet* yang sudah diberi label dihitung bobotnya dan bobot tersebut akan disimpan dalam berkas. Selanjutnya dilakukan proses pelatihan dengan memanfaatkan *library libsvm*. Hasil dari pelatihan ini adalah SVM model yang disimpan dalam berkas.

Pada proses *testing*  akan menggunakan metode SVM yang memanfaatkan *library libsvm.* Pengguna dapat memasukan kata kunci kemudian sistem akan mencari *tweet* yang berkaitan dengan kata kunci menggunakan *twitter API.* Selanjutnya dilakukan preprocessing terhadap data *tweet* yang didapat. Selanjutnya data *tweet* akan di identifikasi mengandung frase opininya. Untuk data *tweet* yang mengandung frase opini akan dilakukan penghitungan bobot dan bobot tersebut disimpan dalam berkas *testing.* Langkah terakhir adalah ditentukan labelnya dengan metode SVM.

1. Implementasi

Pada tahap ini akan dilakukan pembuatan kode program sampai proses pembangunan basis data.

1. Pengujian

Pada tahap ini, dilakukan perhitungan akurasi dari implementasi yang telah dilakukan. Pengujian yang dilakukan adalah perbandingan akurasi antara metode pelabela data *training.*  Metode yang dibandingkan dalam pelabelan data *training* adalah metode *emoticon* dan metode *lexicon* *based*. Selain itu juga dilakukan pengujian akurasi dari metode yang telah di implementasikan pada penelitian ini yaitu metode *lexicon* *based* dan SVM.

## Sistematika Penulisan

BAB I PENDAHULUAN

Bab ini memuat latar belakang masalah, rumusan masalah, batasan masalah,tujuan penelitian, manfaat penelitian, metode penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini memuat penelitian-penelitian terdahulu yang berkaitan dengan penelitian yang dilakukan penulis. Beberapa penelitian terkait Opinion Miningatau *Sentiment analysis* dibahas secara singkat dan dibandingkan dengan penelitian penulis.

BAB III LANDASAN TEORI

Bab ini membahas teori-teori yang berkaitan dengan penelitian. Bagian pertama membahas tentang *Natural* *Language* *Processing*, Postagging, *Opinion* *Mining*, permasalahannya, dan beberapa pendekatan yang digunakan. Bagian kedua membahas tentang *Support* *Vector* *Machine* (SVM), *vector* *space* *model* dan bagaimana model tersebut bekerja.

BAB IV ANALISIS DAN RANCANGAN SISTEM

Bab ini membahas analisis kebutuhan data, analisis model beserta rancangan sistemnya.

BAB V IMPLEMENTASI

Bab ini membahas implementasi opinion mining dengan metode *lexicon* *based* dan *Support* *Vector* *Machine* sesuai rancangan pada bab sebelumnya.

BAB VI HASIL DAN PEMBAHASAN

Bab ini membahas percobaan yang dilakukan pada proses pelatihan dan penentuan kelas beserta uraian mengenai hasil dan perbandingannya.

BAB VII KESIMPULAN DAN SARAN

Bab ini memuat kesimpulan-kesimpulan dari hasil penelitian dan saran-saran yang berguna untuk penelitian selanjutnya.

# 

# TINJAUAN PUSTAKA

Bab ini memuat uraian mengenai penelitian-penelitian yang telah dilakukan sebelumnya yang berkaitan dengan *opinion mining* atau *sentiment analysis* dalam Bahasa Inggris dan Bahasa Indonesia.

(Pang et al., 2002) menggunakan *machine learning* untuk mengklasifikasi *movie reviews*. Penelitian ini melakukan klasifikasi sentimen terhadap review film dan menentukan apakah *review* tersebut memiliki sentimen positif atau negarif. Fitur yang berbeda dari review film tersebut diekstrak dan digunakan algoritma *machine learning* Naïve Bayes dan Support Vector Machine (SVM) untuk menghasilkan klasifikasi model. Mereka memperoleh akurasi antara 78.7% ketika menggunakan Naïve Bayes pada penggunakan *unigram*. Akurasi yang diperoleh ketika menggunakan SVM dengan *unigram* adalah 72.8%.

(Franky dan Manurung, 2008) mencoba mengulangi eksperimen klasifikasi sentimen *movie reviews* oleh (Pang et al., 2002) untuk bahasa Indonesia. Berkaitan dengan ketidaktersediaannya *traning corpora* untuk bahasa Indonesia, maka diaplikasikan *machine translation tools* untuk mentranslasikan corpus Bahasa Inggris yang dibuat (Pang et al., 2002) yang asli ke Bahasa Indonesia dan hasil translasinya digunakan, mulai dari *commercial tool* hingga translasi sederhana kata demi kata dan metoda klasifikasi teks dicoba. Hasil rata-rata yang diperoleh untuk metode Naïve Bayes adalah 74.62% dan 75.62% untuk metode SVM.

*Opinion mining* dalam bahasa china pernah dilakukan oleh (Zhang et al., 2008). Dalam penelitian ini, mereka mempresentasikan hasil kerjanya dengan penekanan *opinion mining* pada *online reviews*. Mereka mengembangkan *opinion mining* tersebut berdasarkan metode *learning*. Menggunakan *dataset* opini yang ada pada AmazonCN. Metode pembelajaran yang digunakan adalah Support Vector Machine dengan menggunakan String kernel.

(Harb et al., 2008) melakukan opinion mining pada media blog. Pendekatan yang dilakukan hanya menghitung jumlah kata sifat positif dan negatif yang terdapat dalam artikel. Kemudian menghitung jumlah perbedaan antara sentimen positif dan negatif. Untuk menambah akurasi sistem, mereka juga melakukan analisa pada kata kerja dan juga kata-kata negasi.

(Go et al., 2009) melakukan *opinion mining* pada *social media* twitter. Memanfaatkan *emoticons* dalam mempermudah pelabelan data pada analisis sentimen dari tweet berbahasa inggris. Penelitian ini mengklasifikasikan sentimen tweet atas 2 kelas yaitu kelas sentiment positif dan negatif. Akurasi yang diperoleh adalah 81,3% dengan menggunakan Naïve Bayes dan 80,5 menggunakan Maximum Entropy serta 82,2% dengan menggunakan SVM.

(Pak dan Paroubek, 2010) menggunakan *emoticons* untuk membangun *corpus* berbahasa inggris dari twitter dengan sentimen positif, negatif dan netral. Untuk kelas netral pak dan paurobek mengambil data *training* dari tweet akun media berbahasa inggris. Metode yang digunakan adalah Naïve Bayes dengan n-grams. Performasi terbaik dihasilkan ketika menggunakan bigram.

(Naradhipa dan Purwarianti, 2011) menggunakan *machine learning* untuk melakukan klasifikasi opini pada *social media*. Yaitu dengan metode SVM dan Maximum entropy akan mengklasifikasikan menjadi kalimat pertanyaan, sentimen positif, sentiment negatif atau netral. Berdasarkan penelitian yang dilakukan klasifikasi terbaik adalah menggunakan metode SVM dengan akurasi 86,66%.

(Komarsilam dan Winarko, 2012) melakukan opinion mining pada review film berbahasa Indonesia dengan metode *unsupervised learning* yang berbasis pada lexicon berbahasa Indonesia dan menggunakan rule untuk menentukan frasa opini dan entitas yang berkaitan dengan hal opini tersebut. Hasil yang diperoleh berupa polaritas sentimen dari suatu teks apakah positif, negatif ataupun netral dan entitas yang menyertai opini yang telah ditentukan, dimana hasil analisa digambarkan dalam sebuah diagram yang menunjukan jumlah persentasi sentimen dan ditampilkan juga label yang merupakan detail dari hasil analisa sentiment. Dari hasil pengujian terhadap sistemnya didapatkan nilai presisi sebesar 0.616 dan nilai *recall* sebesar 0.643.

(Alliandu dan Winarko, 2012) melakukan opinion mining pada twitter berbahasa Indonesia. Metode yang digunakan adalah *supervised learning* yaitu naïve bayes. Fitur pembobotan yang digunakan adalah *term frequency* dengan *laplace smoothing* dan TF-IDR. Dari hasil penelitian diperoleh akurasi 77.45% untuk *term frequency* dengan *laplace smoothing* dan akurasi 75.86% untuk TF-IDF pada test set yang dianotasikan menggunakan *emoticon*.

Perbandingan antara penelitian yang telah dijelaskan diatas dengan sistem yang akan dibangun akan dirangkum dalam Tabel 2.1.

Tabel 2. Perbandingan penelitian lampau dengan sistem yang dibangun

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Penulis & Tahun | Persamaan | Perbedaan |
| 1. | Pang et al .,2002 | Metode *learning* yang digunakan Support Vector Machine (SVM) | Pada sistem yang dibangun tidak hanya menggunakan metode learning SVM tetapi juga menggunakan *lexicon based*. |
| 2. | Harb, et al., 2008 | Metode yang digunakan lexicon based dan pada media online. | Selain *lexicon based* juga menggunakan *machine learning*. |
| 3. | Franky dan Ruli .,2008 | Bahasa yang digunakan bahasa Indonesia. Menggunakan metode *learning* SVM | Bahasa Indonesia yang digunakan pada penelitian ini bukan hasil translate dari bahasa inggris dan diambil dari *social media*. |
| 4. | Zhang et al .,2008 | Metode *learning* menggunakan SVM | Bahasa yang digunakan dalam penelitian ini bahasa indoenesia. Metode yang digunakan tidak hanya SVM saja tapi gabungan dengan *lexicon based*. |
| 5. | Go, et al ., 2009 | Menganalisis dari sumber twitter | Penelitian ini dengan tidak hanya dengan *machine learning* saja tetapi dengan *lexicon based* juga |
| 6. | Pak dan Paroubek .,2010 | Menganalisis dari sumber twitter | Bahasa yang digunakan dalam penetian ini bahasa indonesia.  Metode yang digunakan gabungan antara *lexicon based* dan *machine learning* |
| 7. | Rasyid dan Purwarianti .,2011 | Sumber yang digunakan dari *social media* berbahasa indonesia | Penelitian ini metode yang digunakan gabungan antara *lexicon based* dan *machine learning*. |
| 8. | Komansilan dan Winarko., 2012 | Analisis pada bahasa Indonesia. | Penelitian ini menggunakan *lexicon based* untuk pelabelan data *training*. Pelabelan data *testing* menggunakan metode *support vector machine*.  Pada penelitian ini sumber data yang digunakan dari twiiter. |
| 9. | Aliandu dan Winarko., 2012 | Sumber data yang digunakan dari twiiter.  Analisis pada bahasa Indonesia. | Metode yang digunakan pada penelitian ini gabungan antara *support vector machine* dan *lexicon based*. |

# 

# LANDASAN TEORI

## Natural Language Processing

Natural Language Processing (NLP) adalah salah satu bidang ilmu komputer, kecerdasan buatan, dan bahasa (linguistik) yang berkaitan dengan interaksi antara komputer dan bahasa alami manusia, seperti bahasa Indonesia atau bahasa Inggris. Tujuan utama dari studi NLP adalah membuat mesin yang mampu mengerti dan memahami makna bahasa manusia lalu memberikan respon yang sesuai (Kao dan Poteet, 2006). Secara umum, Jenis aplikasi yang bisa dibuat dalam bidang ilmu NLP terbagi dua, yaitu *text-based application* dan *dialogue-based application*.

*Text-based application* adalah segala macam aplikasi yang melakukan proses terhadap teks tertulis seperti misalnya dokumen, e-mail, buku dan sebagainya. Beberapa jenis aplikasi NLP yang berbasis teks : Program yang mampu mengklasifikasi dan mengambil isi dari suatu dokumen berdasarkan kontennya. Seperti spam filtering (pemfilteran pesan sampah), language identification (identifikasi bahasa), dan Machine Translation,Program yang mampu mentranslasi kalimat baik berupa teks maupun suara dari satu bahasa alami ke bahasa lainnya. Contoh : Google Translate.

*Dialogue-based application* idealnya melibatkan bahasa lisan atau pengenalan suara, akan tetapi bisa juga memasukan interaksi dialog dengan mengetikkan teks pertanyaan melalui *keyboard*. Contoh : Perangkat lunak yang mampu melakukan tugas-tugas dan jasa berdasarkan inputan dari pengguna, lokasi, dan memiliki kemampuan untuk mengakses informasi dari berbagai sumber online (seperti cuaca, keadaan lalu lintas, berita, saham, dll). Contohnya adalah Siri pada produk-produk Apple dan S-Voice pada produk-produk seluler Samsung.

Chatbot adalah program komputer yang didesain untuk mensimulasikan sebuah percakapan cerdas dengan satu atau lebih pengguna manusia melalui inputan suara atau teks, utamanya digunakan untuk percakapan kecil. Contoh : Cleverbot, SimSimi, dan begobet.

## Part-of-speech

*Part-of-speech* menunjukkan kelas atau jenis kata dalam sebuah kalimat. Menurut (Jurafsky dan Martin, 2008), *part-of-speech* dapat dibedakan menjadi dua buah kelas utama, yaitu tipe kelas tertutup dan tipe kelas terbuka. Tipe kelas tertutup terdiri atas *part-of-speech* yang memiliki anggota relatif tetap. Contoh kelas ini adalah preposisi. Tipe kelas terbuka, sebaliknya, terdiri atas kata-kata yang relatif terus mengalami penambahan karena terbentuknya kata baru akibat serapan dari bahasa asing. Ada empat *part-of-speech* tipe kelas terbuka utama yang terdapat dalam hampir semua bahasa di dunia, yaitu kata benda, kata sifat, kata kerja, dan kata keterangan. Kelas kata lainnya merupakan sub bagian dari keempat *part-of-speech* tersebut.

### Part-of-speech Tagging

*Part-of-speech* (POS) *tagging* adalah proses pemberian label pada setiap kata dalam kalimat dengan anotasi *part-of-speech* yang bersesuaian (Manning dan Schuetze, 1999). Anotasi POS menunjukkan jenis kata dalam sebuah kalimat. Masukan dalam pemberian anotasi adalah rangkaian kata beserta sekumpulan anotasi yang mungkin. Keluaran dari proses ini adalah anotasi tunggal terbaik yang sesuai untuk setiap kata dalam rangkaian masukan.

Menurut (Jurafsky dan Martin, 2008), POS *tagging* memiliki beberapa manfaat dalam berbagai penelitian pemrosesan bahasa. POS *tagging* dapat digunakan untuk membuat model bahasa untuk pengenalan wicara (*speech recognition*). Pemberian anotasi juga membantu dalam menentukan pelafalan sebuah kata dalam sistem sintesis wicara. Dalam bidang temu kembali informasi, POS *tagging* dapat membantu proses *stemming* atau pemilihan kata-kata penting tertentu dalam sebuah dokumen.

### Tagset

Hal yang perlu menjadi pertimbangan dalam POS *tagging* adalah *tagset*, yaitu anotasi *part-of-speech* mana saja yang akan digunakan. Ukuran dan isi dari tagset ini perlu diperhitungkan. Menurut (Indurkhya dan Damerau, 2010), penggunaan *tagset* berukuran besar membantu mendapatkan pengetahuan tentang struktur morfologi dan morfosintaksis dari kata, namun dapat mempersulit untuk membedakan kemiripan antar anotasi. Perbedaan anotasi, pada kasus-kasus tertentu, sangatlah kecil sehingga terkadang terjadi perbedaan pendapat dalam penggunaan satu anotasi dengan anotasi lainnya. Contoh *tagset* untuk bahasa Indonesia dapat dilihat pada Tabel 3.1.

Tabel 3. Contoh Tags untuk bahasa Indonesia

|  |  |  |
| --- | --- | --- |
| Anotasi | Definisi | Contoh |
| JJ | Kata Sifat | Baik, Kejam |
| NN | Kata Benda | Mobil, Buku |
| VB | Kata Kerja | Pergi, membeli |
| RB | Kata Keterangan | Sementara, Nanti |
| IN | Kata Depan | Di, Ke, Dari |
| MD | Kata Bantu | Masih, Telah |
| UH | Kata Seru | Ayo |
| CC | Kata Sambung Koordinasi | Atau, Bahwa |
| PR | Kata Ganti | Anda, Dia |
| DT | Kata Tunjuk | Ini, Itu |
| CK | Kata sambung khusus | Dengan, Yang |
| RP | Partikel | Doang, Dong |

## Opinion Mining

*Sentiment analysis* atau *opinion mining* mengacu pada bidang yang luas dari pengolahan bahasa alami, komputasi linguistik dan *text* *mining*. Secara umum, bertujuan untuk menentukan pendapat pembicara atau penulis berkenaan dengan topik tertentu (Liu, 2011).

Tugas dasar dalam analisis sentimen adalah mengelompokkan polaritas dari teks yang ada dalam dokumen, kalimat, atau fitur / tingkat aspek apakah pendapat yang dikemukakan dalam dokumen, kalimat atau fitur entitas / aspek bersifat positif , negatif atau netral. Jika ditinjau lebih dalam lagi sentiment analysis dapat menyatakan emosional sedih, gembira, atau marah.

Sebuah metode yang berbeda untuk menentukan sentimen adalah penggunaan sistem skala dimana kata-kata umumnya terkait memiliki sentimen negatif, netral atau positif dengan mereka diberi nomor pada skala -5 sampai +5 (paling negatif hingga yang paling positif) dan ketika sepotong teks terstruktur dianalisis dengan pemrosesan bahasa alami, konsep selanjutnya dianalisis untuk memahami kata-kata ini dan bagaimana mereka berhubungan dengan konsep.

Setiap konsep kemudian diberi skor berdasarkan bagaimana kata-kata sentimen berhubungan dengan konsep, dan skor yang terkait. Hal ini memungkinkan gerakan untuk pemahaman yang lebih canggih dari sentimen berdasarkan skala 11 titik.

Penelitian lain arah berbeda diantaranya adalah identifikasi subjektivitas / objektivitas. Tugas ini biasanya didefinisikan sebagai menggolongkan suatu teks yang diberikan (biasanya kalimat) ke salah satu dari dua kelas: objektif atau subjektif (Pang dan Lee, 2008). Masalah ini kadang-kadang dapat lebih sulit daripada klasifikasi polaritas. Subjektivitas kata-kata dan frase mungkin tergantung pada konteks dan dokumen objektif mungkin berisi kalimat subjektif (misalnya, sebuah artikel berita mengutip pendapat orang).

Orang sering kali menyatakan lebih dari satu opini "the movie was terrible, but DeNiro's performance was superb, as always", sebuah sarkasme "this is probably the best laptop Dell could come up with", atau menggunakan negasi dan banyak elemen kompleks sehingga sulit untuk diparsing "not that I'm saying this was a bad experience".

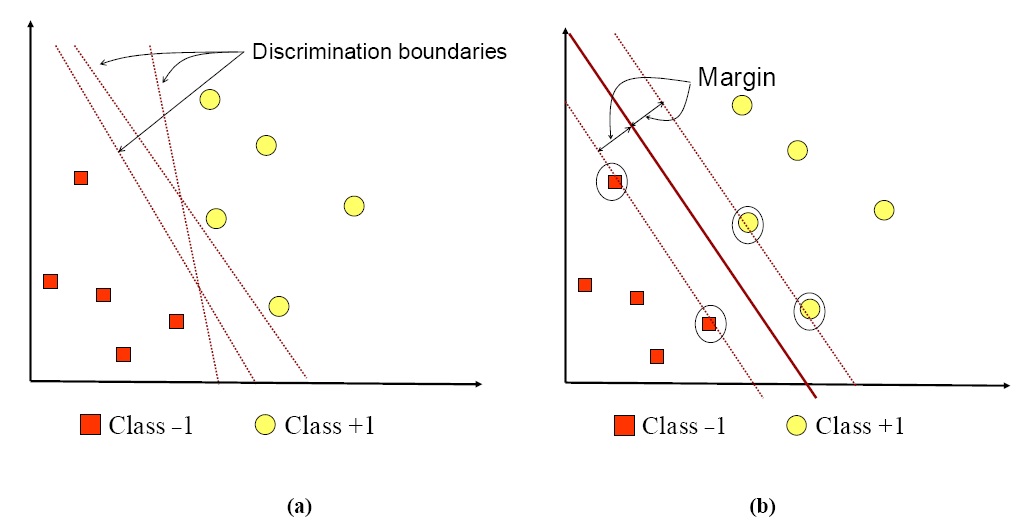
Ekspresi atau sentiment mengacu pada fokus topik tertentu, pernyataan pada satu topik mungkin akan berbeda makna dengan pernyataan yang sama pada subject yang berbeda. Sebagai contoh, adalah hal yang baik untuk mengatakan alur film tidak terprediksi, tapi adalah hal yang tidak baik jika ‘tidak terprediksi’ dinyatakan pada kemudi dari kendaraan. Bahkan pada produk tertentu, kata-kata yang sama dapat menggambarkan makna kebalikan, contoh adalah hal yang buruk untuk waktu start-up pada kamera digital jika dinyatakan “lama”, namun jika” lama” dinyatakan pada usia batere maka akan menjadi hal positif. Oleh karena itu pada beberapa penelitian, terutama pada review produk, pekerjaan didahului dengan menentukan elemen dari sebuah produk yang sedang dibicarakan sebelum memulai proses opinion mining.

## Support Vector Machine

Support Vector Machine (SVM) kini banyak dikembangkan dan diterapkan baik untuk klasifikasi maupun regresi. Metode ini berakar dari teori pembelajaran statistik yang hasilnya sangat menjanjikan untuk memberikan hasil yang lebih baik daripada metode lainnya. SVM juga bekerja dengan baik pada set data berdimensi tinggi.

### Konsep SVM

Ide dasar SVM adalah memaksimalkan batas hyperplane, seperti yang diilustrasikan pada gambar 3.1. pada Gambar 3.1 (a) ada sejumlah pilihan hyperplane yang mungkin untuk data set, dan pada Gambar 3.1 (b) merupakan hyperplane dengan margin yang paling maksimal. Hyperplane yang maksimal akan memberikan generalisasi yang lebih baik pada metode klasifikasi.



Gambar 3. Decision Boundary

Konsep klasifikasi dengan SVM dapat dijelaskan secara sederhana sebagai usaha untuk mencari hyperplane terbaik yang berfungsi sebagai pemisah dua buah kelas data pada ruang input (Nugroho et al., 2003). Gambar 3.1 memperlihatkan beberapa pola yang merupakan anggota dari dua buah kelas data : +1 dan -1. Data yang tergabung dalam kelas -1 disimbolkan dengan bentuk bujur sangkar sedangkan data pada kelas +1 disimbolkan dengan bentuk lingkaran.

Hyperplane pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur margin hyperplane tersebut dan mencari titik maksimalnya. Margin adalah jarak antara hyperplane tersebut dengan data terdekat dari masing-masing kelas. Data yang paling dekat ini disebut *support vector.* Garis solid pada gambar 3.1 (b) menunjukan hyperplane terbaik yaitu yang terleat tepat pada tengah-tengah kedua kelas, sedangkan data lingkaran dan bujur sangkar yang dilewati garis batas margin (garis putus-putus) adalah *support vector*. Usaha untuk mencari lokasi hyperplane ini merupakan inti dari proses pelatihan pada SVM.

### SVM linear

Setiap ada latih dinyatakan oleh ( , ) dimana i = 1,2,...., N , dan = merupakan atribut (fitur) set untuk data latih ke-i € {-1,+1} menyatakan label kelas. Hyperplane klasifikasi linear SVM seperti pada gambar 3.1 (b), dinotasikan dengan

w. + b = 0 ( 3.1 )

w dan b adalah parameter model w. merupakan inner product dalam antara w dan . Data yang masuk kelas -1 adalah data yang memenuhi pertidaksamaan berikut :

w. + b ≤ -1 ( 3.2 )

Sementara yang masuk dalam kelas +1 adalah data yang memenuhi pertidaksamaan berikut:

w. + b ≥ +1 ( 3.3 )

Sesuai pada gambar 3.1 (b), jika ada data dalam kelas -1 misalnya yang bertempat di hyperplane, persamaan 3.1 akan terpenuhi. Untuk data kelas -1 dinotasikan dengan :

w. + b = 0 ( 3.4 )

sementara kelas +1 misal akan memenuhi persamaan

w. + b = 0 ( 3.5 )

Dengan mengurangkan persamaan 3.5 dan 3.4, didapatkan

w.( - ) = 0 ( 3.6 )

- adalah vektor pararel di posisi hyperplane dan diarahkan dari ke . Karena inner product dalam bernilai 0, arah dari 2 harus tegak lurus terhadap hyperplane (Gambar 3.1 (b)).

Dengan memberikan label -1 untuk kelas pertama dan +1 untuk label kelas kedua, prediksi semya data uji akan menggunakan formula

Susuai pada gambar 3.1 (b), hyperplane untuk kelas -1 adalah data pada *support vector* yang memenuhi persamaan

w. + b = -1 ( 3.7 )

Sementara hyperplane kelas +1 memenuhi persamaan

w. + b = +1 ( 3.8 )

dengan demikian margin dapat dihitung dengan mengurangi Persamaan 8 dan 7 didapatkan

w. (- . ) = 2 ( 3.9 )

Margin hyperplane diberikan oleh jarak antara dua hyperplane dari dua kelas tersebut. Notasi di atas diringkas menjadi

||w|| x d = 2 atau d = ( 3.10 )

### Hyperplane SVM

Klasifikasi kelas data pada SVM persamaan 2 dan 3 dapat digabungkan dengan notasi

( w. + b) ≥ 1 , i =1,2.....,N ( 3.11 )

Margin optimal dihitung dengan memaksimalkan jarak antara hyperplane dan data terdekat. Jarak ini dirumuskan dengan persamaan 3.10. Selanjutnya, masalah ini diformulasikan ke dalam quadratic programming (QP) problem, dengan meminimalkan invers Persamaan 3.10, , dengan syarat

Minimalkan :

( 3.12 )

Syarat

( w. + b) ≥ 1 , i =1,2.....,N ( 3.13 )

Optimalisasi ini dapat diselesaikan dengan Lagrange Multiplier

Lp = - ( 3.14 )

adalah Lagrange Multiplier yang berkorespindensi dengan . Nilai adalah nol atau positif. Untuk meminimalkan Lagrangian, Persamaan 3.14 harus diturunkan pada w dan b, dan diset dengan nilai 0 untuk syarat optimalisasi diatas.

Syarat 1 :

= 0 => w = ( 3.15 )

Syarat 2 :

= 0 => = 0 ( 3.16 )

N adalah jumlah data yang menjadi support vector. Karena Lagrage Multiplier () tidak diketahui nilainya, persamaan diatas tidak dapat diselesaikan secara langsung untuk mendapatkan w dan b. Untuk menyelesaikan masalah tersebut, modifikasi persamaan 3.14 menjadi kasus pemaksimalan, syarat untuk dualitasnya menggunakan constraint karush-Kuhn-Tucker (KKT) berikut

Syarat 1 :

( 3.17 )

Syarat 2 :

≥ 0, i=1,2,...,N ( 3.18 )

Dengan menerapakan constraint pada persamaan 3.17 dan 3.18, dipastikan bahwa nilai Lagrange Multiple sama banyaknya dengan data pelatihan, tetapi sebenarnya banyak dari data pelatihan yang Lagrange Multiplier-nya sama dengan nol (Karena hanya beberapa saja yang akan menjadi support vector) ketika menerapkan syarat pertama. Constraint diatas menyatakan bahwa Lagrange Multipioer harus nol, kecuali untuk data latih yang memenuhi persamaan.

( w. + b) = 1 ( 3.19 )

Data pelatihan itu, dengan > 0, terletak pada hyperplane atau dan disebut *support vector.* Data pelatihan yang tidak terletak di hyperplane tersebut memiliki = 0. Persamaan 3.15 dan 3.16 juga menyarankan parameter w dan b, yang mendefinisikan hyperplane, hanya tergantung pada support vector.

Walaupun bisa diselesaikan, masalah optimalisasi diatas masih sulit karena banyak parameter : w , b, ,untuk menyerderhanakannya, persamaan optimalisasi 3.14 harus ditransformasi kedalam fungsi Lagrange Multiplier itu sendiri.

Persamaan lagrange Multiplier 3.14 dapat dijabarkan menjadi

Lp = - – b + ( 3.20 )

Syarat optimal persamaan 3.16 ada dalam suku ketiga ruas kanan dalam persamaan 3.20, dan memaksa suku ini menjadi sama dengan nol. Dengan mengganti w dari syarat 3.15 dan suku = . , persamaan diatas akan berubah menjadi dualitas Lagrange Multiplier yang berupa Ld, didapatkan :

Ld = - ( 3.21 )

Syarat 1 :

= 0 ( 3.22 )

Syarat 2

≥ 0, i = 1,2,.... N ( 3.23 )

Untuk set data yang besar, masalah dualitas optimalisasi tersebut (persamaan 3.21, 3.22, 3.23) dapat diselesaikan dengan metode numerik seperti quadratic programming. Begitu didapatkan, persamaan 3.15 dan 3.17 bisa digunakan untuk mendapakan solusi w dan b. Hyperplane didapat dengan formula

+ b = 0 ( 3.24 )

N adalah jumalah data yang menjadi support vector, merupakan support vector, z merupakan data uji yang akan diprediksi kelasnya dan .z merupakan inner product antara dan z. Untuk nilai b didapat dari persamaan 3.17 pada support vector. Karena dihitung dengan metode nnumerik dan mempunyai error numerik, nilai yang dihitung untuk b bisa jadi tidak sama. Hal ini disebabkan oleh support vector yang digunakan dalam persamaan 3.17, biasanya diambil nilai rata-rata dari b yang didapat untuk menjadi parameter hyperplane. Untuk mendapatkan b, persamaan 3.17 dapat disederhanakan menjadi

b = 1 – (w.) ( 3.25 )

Penjelasa diatas berdasarkan asumsi bahwa kedua kelas kelas dapat terpisah sempurna oleh hyperplane. Akan tetapi pada umumnya kedua kelas tersebut tidak dapat terpisah secara sempurna. Hal ini menyebabkan proses optimalisasi tidak dapat diselesaikan karena tidak ada w dan b yang memuhi persamaan 3.13. Untuk itu, pertidaksamaan tersebut dimodifikasi dengan memasukan variabel slack. Hasilnya adalah

( w. + b) ≥ 1 - ( 3.26 )

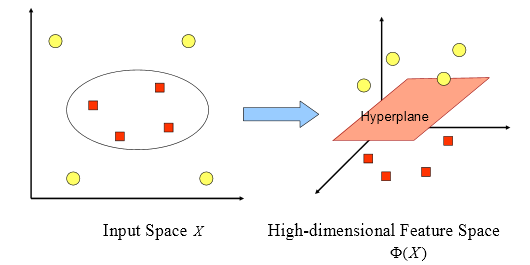
Demikian juga untuk masalah persamaan 3.12

+ C ( 3.27 )

Parameter C berguna untuk mengontrol pertukaran antara margin dan error klasifikasi. Semakin besar nilai C, semakin besar pula pelanggaran yang dikenakan untuk tiap klasifikasi.

### SVM Nonlinear

Dalam SVM ada linear dan nonlinear (kernel trik). SVM sebenarnya adalah hyperplane linear yang hanya bekerja pada data yang dapat dipisahkan secara linear. Untuk data yang distribusi kelasnya tidak linear digunakan pendekatan kernel pada fitur data awal set data. Kernel dapat didefinisikan sebagai suatu fungsi yang memetakan fitur data dari dimensi awal (rendah) ke fitur lain yang berdimensi lebih tinggi.



Gambar 3. Dimensi Data

Ilustrasi kernel yang digunakan untuk memetakan dimensi awal set data ke dimensi baru dapat dilihat pada gambar 3.21. Algoritma memetaan kernel seperti berikut

: →

X → (x)

merupakan fungsi kernel yang digunakan untuk pemetaan, D merupakan data training, q merupakan set fitur dalam satu data yang lama, dan r merupakan set fitur yang baru sebagai hasil pemetaan setiap data training. Sementara x merupakan data training, dimana, ,,......, merupakan fitur-fitur yang dipetakan kefitur berdimensi tinggi r., jadi untuk set data yang digunakan sebagai pelatihan dengan algoritma yang ada dari dimensi fitur yang lama D ke dimensi fitur baru r.

Selanjutnya dilakukan proses pelatihan yang sama sebagaimana pada SVM linear. Proses pemetaan pada fase ini memerlukan perhitungan dot-product dua buah data pada ruang fitur baru. Dot-product kedua buah vektor dan dinotasikan sebagai () . (). Nilai dari dot-product kedua vektor ini dapat dihitung secara tidak langsung, yaitu tanpa mengetahui fungsi transformasi . Teknik komputasi seperti ini kemudian disebut kernel trik, yaitu menghitung dot-product dua vektor diruang dimensi baru dengan memakai komponen kedua buah vektor tersebut diruang dimensi asal, seperti berikut

K (.) = () . () ( 3.28 )

Dan prediksi pada set data dengan dimensi fitur baru diformulasikan dengan

F( (x)) = sign(w. (z) +b) = sign( ) ( 3.29 )

N adalah jumlah data yang menjadi support vecotr, xi adalah support vector dan z adalah data uji yang akan diprediksi. Untuk pilihan fungsi kernel yang umum digunakan digunakan dalam aplikasi dapat dilihat pada tabel 3.2 (Liu, 2011)

Tabel 3. Fungsi Kernel

|  |  |
| --- | --- |
| Nama Kernel | Definisi Fungsi |
| Linear | K (x,y) = x.y |
| Polinomial | K(x,y) = |
| Gaussian RBF | K(x,y) = exp( |

## TF-IDF

Metode *Term Frequency-Inverse Document Frequency* (TF-IDF) adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot, yaitu *Term* *frequency* (TF) merupakan frekuensi kemunculan kata (t) pada kalimat (d). *Document* *frequency* (DF) adalah banyaknya kalimat dimana suatu kata (t) muncul. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata itu di dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut.

Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen. Pada Metode ini pembobotan kata dalam sebuah dokumen dilakukan dengan mengalikan nilai TF dan IDF. Pembobotan diperoleh berdasarkan jumlah kemunculan term dalam kalimat (TF) dan jumlah kemunculan term pada seluruh kalimat dalam dokumen ( IDF). Menurut (Manning et al., 2008) Menghitung bobot (W) masing-masing dokumen dengan persamaan 3.30

= x ( 3.30 )

Dimana :

d : kalimat ke-d

t : kata (term)

: term frequency pada kalimat ke-d

: bobot kata (term) dalam kalimat ke-d

: inverse document frequency suatu term

Sedangkan nilai IDF sebuah term dihitung menggunakan persamaan 3.31

IDF = log ( N / ) ( 3.31 )

Dimana :

N : jumlah kalimat

: jumlah kemunculan kata (term)

# 

# ANALISIS DAN RANCANGAN SISTEM

## Analisis Sistem

*Opinion mining* merupakan sebuah sistem yang dapat digunakan untuk menentukan sentimen atau opini secara otomatis pada sebuah dokumen. Penelitian ini akan mengambil data dari *twitter* yang berbahasa indonesia. Sistem yang dibangun dan dirancang adalah sistem yang memiliki kemampuan untuk menentukan sentimen dari *tweet* hasil *query user* atas *term* pada aplikasi yang terhubung dengan Twitter Search API. Metode yang digunakan untuk menentukan sentimen adalah metode gabungan *unsupersive* yang berbasis pada analisa bahasa dan metode *supervise* yaitu *Support Vector Machine* (SVM).

*Data training* yang digunakan adalah *tweet* yang juga digunakan dalam penelitian (Alliandu dan Winarko, 2012). Data tersebut akan dikategorikan ke dalam kelas sentimen positif, negatif dan netral dengan menggunakan analisis bahasa (*unsupervise*). Hasil pengkategorian *data training* digunakan untuk menentukan sentiment pada *tweet* yang baru dengan menggunakan metode *supervise* yaitu *Supoort Vector Machine* (SVM).

### Deskripsi Sistem

Sistem *Opinion Mining* ini memiliki dua buah proses utama, yaitu pelatihan dan penentuan kelas sentimen pada data baru. Pelatihan digunakan untuk mendapatkan model yang akan digunakan untuk menentuan kelas sentimen pada data baru. Proses-proses yang terjadi dalam sistem ini diperlihatkan pada gambar 4.1.

Pada proses pelatihan dengan menggunakan masukan berupa corpus yang digunakan pada penelitian (Alliandu dan Winarko, 2012) yang telah dilakukan pelabelan dengen metode *lexicon based*. Langkah awal dari pelabelan dengan menggunakan *lexicon based* adalah melakukan *preprocessing* terhadap data. Dalam *preprocessing* terdapat beberapa tahap, secara berurutan adalah membersihkan yang bukan termasuk karater UTF-8, merubah semua karakter menjadi huruf kecil, menghapus simbol-simbol, memproses angka-angka, menghapus perulangan karakter, melakukan penggantian terhadap kata tidak baku, melakukan *spelling correction*.



Gambar 4. Proses dalam Opinion Mining

Setelah selesai tahap *preprocessing*, selanjutnya akan masuk dalam tahan penentuan label. Dalam tahap ini juga terdapat beberapa proses diantaranya *POS taging*, menganalisis tweet dengan rules yang sudah ada, mengecek negasi. Sampai disini telah dihasilkan corpus yang labelnya ditentukan dengan menggunakan *lexicon based* dan siap digunakan untuk proses pelatihan.

Proses berikutnya adalah proses inti pelatihan, dalam penelitian ini terdapat 25000 data pelatihan yang sudah ditentukan labelnya dengan metode *lexicon based,* terdiri dari data 12500 data berlabel positif dan 12500 data berlabel negatif. Proses pelatihan ini nantinya akan menghasilkan SVM model. Yaitu sebuah model yang nantinya digunakan untuk menentukan sentimen dengan metode SVM.

Proses utama yang kedua adalah penentuan label *tweet* baru dengan metode SVM. Langkah pertama pada proses utama ini adalah sistem akan mencari *tweet* yang berhubungan dengan kata kunci yang diinginkan. Selanjutnya akan masuk dalam tahap *preprocessing* yang telah dijelaskan sebelumnya. Kemudian dilakukan pengecekan indikasi keberadaan frase opini pada *tweet* yang telah dipreprocessing. Selanjutnya untuk *tweet* yang terindikasi memiliki frase opini, sentimennya akan ditentukan dengan metode SVM.

## Analisis Data

Data yang dibutuhkan pada penelitian ini terdiri atas dua jenis data, yaitu data yang digunakan untuk pelatihan dan data pengujian pemberian label. Data yang digunakan untuk pelatihan berupa dokumen tweet berbahasa indonesia yang diberi label dengan metode *lexicon based*. Dokumen ini diambil dari penelitian (Alliandu dan Winarko, 2012) yang jumlah 81500 tweet. Namun setelah diproses menggunakan lexicon based data yang digunakan untuk pelatihan hanya 25000.

Berbeda dengan data yang digunakan untuk pelatihan , data yang digunakan pada proses pelabelan menggunakan tweet yang belum memiliki label. Dokumen *tweet* tersebut didapatkan dengan memanfaatkan API twitter yang dapat mencari *tweet* yang ada pada twitter dengan parameter tertentu.

## Preprocessing

*Preprocessing* sangat menentukan dalam proses pelabelan secara *lexicon* *based*. Dimana pelabelan *lexicon* *based* nantinya akan menentukan proses pelabelan yang dilakukan dengan metode SVM. *Preprocessing* juga digunakan dalam kedua proses utama, baik proses pelatihan maupun proses penentuan label dengan SVM. Jadi *preprocessing* ini cukup penting peranannya dalam penelitian ini.

Tahap *preprocessing* sendiri terdiri atas beberapa proses yang akan dibahas satu per satu secara lebih mendetail. Proses-proses tersebut diantaranya adalah pembersihkan karater yang bukan termasuk UTF-8 (*clear* *invalid* *UTF8*), merubah ke huruf kecil (*casefolding*), menghapus simbol-simbol(*remove* *symbol*), memproses angka-angka(*processing* *numbers*), menghapus perulangan karakter (*remove* *repeat*), melakukan penggantian terhadap kata tidak baku(*replace* *slang*), melakukan spelling correction(*spelling* *correction*).

### Clear Invalid UTF-8

Proses ini adalah bagian dari tahap preprocessing. Pada intinya proses ini akan menghilakan karakter-karakter yang tidak termasuk dalam UTF-8. Secara teknik proses ini dilakukan dengan fitur pada bahasa pemrograman yang sudah disediakan.

### Casefolding

Proses ini hanyalah merupan semua karakter yang ada pada tweet menjadi karakter dalam huruf kecil semua. Hal ini dilakukan karena tidak ada kaitan antara huruf kecil dan huruf besar terhadap sentimen, serta mempermudah proses-proses selanjutnya.

### Remove Symbol

Proses ini adalah proses selanjutnya dari tahap preprocessing. Pada proses ini tanda baca seperti titik (.) , koma (,), seru (!), tanya (?) dan lain sebagainya akan dihilangkan. Tanda/simbol ini dapat dihilangkan karena tidak banyak berpengaruh pada penentuan label. Secara teknik ini akan dilakukan dengan bahasa pemrograman dengan mencari simbol-simbol kemudian menghapusnya.

### Processing Numbers

Proses selanjuatnya dalam tahap preprocessing adalah pemrosesan terhadap angka-angka. Kita tahu dalam sosial media, pengguna sering menggunakan angka-angka baik untuk menyingkat kata, atau untuk bergaya atau untuk menyebutkan nilai. Angka-angka yang terdapat dalam sebuah kata akan di convert menurut tabel 4.1.

Tabel Daftar Konversi

|  |  |
| --- | --- |
| **Angka** | **Konversi** |
| 0 | O |
| 1 | i |
| 2 | Copy karakter-karakter sebelumnya sebelumnya |
| 3 | e |
| 4 | a |
| 5 | s |
| 6 | g |
| 7 | T |
| 8 | b |
| 9 | g |

Tabel 4.2 adalah beberapa contoh kata yang akan diubah oleh proses ini bila ditemukan.

Tabel . Contoh Konversi Angka

|  |  |
| --- | --- |
| Masukan | Keluaran |
| ga2l | gagal |
| p3rg1 | pergi |
| Hati2 | Hati-hati |
| Ba9u5 | Bagus |

### Remove Repeat

Proses ini salah satu proses penting dalam tahap prepocessing. Proses ini akan menghapus karakter yang berulang lebih dari dua kali. Seperti diketahui dalam bahasa indonesia perulangan karakter yang berurutan hanya dua karakter, maka jika lebih akan dihapus dan hanya akan disisakan 2 karakter yang sama yang berurut saja. Untuk memperjelas perhatikan tabel 4.3.

Tabel 4.3 Contoh menghapus karakter berualng

|  |  |
| --- | --- |
| **Input** | **Output** |
| Tidaaaaaaaaaak | Tidaak |
| Cemunguuuuudh | Cemunguudh |
| Eaaaaaa | eaa |

### Replace Slang

Proses ini merupakan proses penggantian kata tidak baku menjadi kata baku. Proses ini dilakukan dengan menggunakan bantuan daftar kata-kata informal dan padanannya dalam kata-kata formal. Proses ini akan mengecek kata terdapat dalam daftar kata informal atau tidak. Jika tidak maka kata tidak akan diganti tetapi bila ya maka kata akan diganti dengan padanannya yang terdapat dalam daftar kata baku. Daftar kata baku dan kata tidak baku ini menggunakan daftar yang digunakan pada penelitian (Komarsilam dan Winarko, 2012), yang sudah ditambah secara manual berdasarkan pengamatan dalam proses pengembangan sistem ini. Contoh dari proses ini akan ditunjukan pada tabel 4.4.

Tabel . Contoh Pengantian kata tidak baku

|  |  |
| --- | --- |
| **Kata tidak baku** | **Kata baku** |
| Nggak | Tidak |
| Lu | Kamu |
| Ngadu | Mengadu |

### Spelling Correction

*Spelling* *Correct* adalah proses terakhir pada tahap *preprocessing*. Proses ini akan memperbaiki kesalahan penulisan yang ada data *tweet*. Seperti kita tahu pengguna sosial media sering sengaja menyingkat kata untuk memenuhi 140 karakter (jumlah maksimal karakter pada twitter) atau secara tidak sengaja salah menuliskan kata. Metode yang digunakan akan dengan memanfaatkan daftar kata baku yang sudah ada dan dengan digabungkan dengan fungsi levenshtein. Fungsi levenshtein adalah fungsi yang dapat menghitung kedekatan/kemiripan antar kata. Secara teknis kata yang telah melalui proses sebelumnya dan tidak terdapat dalam daftar kata baku, maka kata ini akan dikenai proses ini. Tabel 4.5 adalah contoh kata-kata yang dapat ditangani oleh proses ini.

Tabel . Contoh SpellCorrection

|  |  |
| --- | --- |
| **Input** | **Output** |
| Bapk | Bapak |
| Menghitumg | menghitung |
| Yng | yang |

Walaupun proses ini memberikan manfaat yang cukup besar. Namun terkadang proses ini memberikan kelemahan. Hal ini terjadi ketika proses ini diberikan input suatu frase atau kata serapan baru yang belum terdapat dalam dafatar kata baku. Misalkan BLBI oleh karena BLBI tidak terdapat dalam daftar kata baku maka BLBI akan digantikan dengan kata yang terdekat dengan kata BLBI. Padahal yang dimaksud dengan kata ini adalah kata serapan baru.

## Penentuan Label Data Training

Tahap selanjutnya setelah preprocessing data adalah penentuan label dengan metode *lexicon* *based*. Tahap ini masih dalam proses utama yang pertama yaitu proses pelatihan. Pada tahap penentuan label ini terdapat beberapa proses di dalamnya. Seperti ditunjukan pada gambar, proses-proses tersebut diantaranya *POS taging*, *Opinion* *Rules* dan *check* *negation*.

### POS taging

Setelah tahap *preprocessing* selesai berarti data training sudah bersih dan telah siap untuk dilakukan proses penentuan label secara *lexicon* *based*. Proses pertama yang dilakukan pada tahap ini adalah melakukan *POS taging* pada data tersebut.

POS taging adalah proses untuk menentukan kelas kata dari setiap kata-kata yang terdapat dalam kalimat. Proses *POS taging* dalam penelitian ini menggunakan metode *dictionary*. Misalnya terdapat satu *tweet* “Kasus penembakan yang dilakukan oleh polisi sangat rumit” maka dengan POS taging akan didapatkan “Kasus/NN penembakan/VB yang/CK dilakukan/VB oleh/IN polisi/NN sangat/RB rumit/JJ”.

Langkah pertama adaalah melakukan tokenisasi, yaitu pross yang memecah-mecah *tweet* menjadi satuan kata. Dimana sistem akan melihat setiap spasi yang ada dalam *tweet* dan berdasarkan spasi tersebut maka kata-kata dapat dipecah.

Langkah kedua yang dilakukan adalah melakukan penyesuaian dengan koleksi kata yang terdapat dalam basisdata. Setiap kata dari hasil tokenisasi tersemua kemudian dicek satu-persatu kelas katanya sesuai dengan koleksi kata dalam basisdata. Jika kata tersebut terdapat dalam basisdata maka akan langsung ditentukan kelas katanya. Namun apabila kata tersebut tidak ditemukan dalam database, maka kata tersebut akan dianggap sebagai kata benda(NN).

Apabila dilihat dari cara kerja *POS taging* ini, ada kata-kata yang tidak bisa dideteksi dengan benar, karena karakteristiknya tidak bisa menangani kata-kata yang memiliki konteks dan maknna ganda. Misalkan pada tweet “Mobil polisi menarik mobil sedang yang kecelakaan.” Kata menarik dalam kali ini akan dideteksi menjadi kata sifat padahal bila dilihat dari konteks nya kata ‘menarik’ disini adalah kata kerja.

### Opinion Rules

Seperti yang telah dijelaskan sebelumnya, dalam penentuan label data training akan menggunakan metode *lexicon* *based*, dimana penentuan labelnya menggunakan rule-rule tertentu. Rule-rule ini mengacu pada penelitian yang dilakukan oleh (Komarsilam dan Winarko, 2012).

Secara teknis langkah yang dilakukan adalah dengan melihat sekuens dari kata-kata hasil *POS taging* sebelumnya yang sudah menyertakan kelas kata dari masing-masing kata. Rule yang dipakai untuk mendeteksi frasa yang terdiri dari dua kata saja dan kata-kata yang berdiri sendiri. Pada tabel 4.6 ditampilkan daftar rule frasa opini yang digunakan dalam sistem ini.

Tabel 4.6 Opinion Rules

|  |  |
| --- | --- |
| **NO** | **Rule** |
| 1 | RB JJ |
| 2 | RB VB |
| 3 | CK VB |
| 4 | NN JJ |
| 5 | NN VB |
| 6 | VB JJ |
| 7 | JJ VB |

Dalam penentuan frase ini terlebih dahulu mendeksi kehadiran dari kata sifat atau kata kerja. Setelah sistem dapat mendeteksi kehadiran dari kata kerja atau kata sifat dalam kalimat, maka sistem akan melihat kata sebelum kata kerja atau kata sifat dan melihat kelas katanya. Sehingga apabila kedua kelas kata dari kedua kata tesebut diurutkan makan akan dicocokan dengan rule yang ada. Jika sama dengan rule yang ada maka frase tersebut bisa dikatakan memiliki opini atau sentimen.

Misal terdapat suatu kalimat “Film itu sangat bagus”, oleh *POS taging* akan ditandai sebagai “Film/NN itu/DT sangat/RB bagus/JJ”. Dari hasil dapat dilihat ada frasa yang sesua dengan rule yaitu ‘sangat bagus’ (RB JJ) karena kata ‘bagus’ memiliki sentimen positif maka kalimat tersebut diberi label positif.

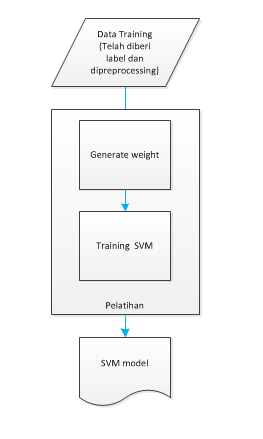
### Check Negation

Proses terakhir pada tahap penentuan rule ini adalah mengecek negasinya. Setalah *tweet* dilakukan *POS taging* dan dicocokan dengan rule-rule yang ada. Apabila terdapat sentimen dari rule-rule dari maka selanjutnya akan dicek apakah terdapat negasi dari hasil rule-rule tadi.

Konsepnya mencari kata negasi sebelum kata sifat atau kata kerja. Dimana daftar kata negasi diambil dari basisdata. Toleransi jarak yang digunakan adalah 5 kata dari kata sifat atau kata benda. Misalkan terdapat tweet “Film itu tidak bagus”. Pertama akan dilakukan POS taging sehingga menghasilkan seperti berikut “Film/NN itu/DT tidak/MD bagus/JJ”. Kemudian akan dilakukan opinion rule yang akan menghasilkan frase positif yang dihasilkan dari kata ‘bagus’. Proses *check* *negation* akan melakukan pengecekan kata-kata sebelum frase tersebut. Jika terdapat kata negasi maka labelnya akan dibalik. Jadi penentuan dari *tweet* tersebut akan menghasilkan label negatif.

## Pelatihan

Pelatihan digunakan untuk membuat model dari data training. Masukan dari proses ini adalah berupa data training yang sudah dilakukan preprocessing dan diberi label dengan menggunakan *lexicon* *based*. Model yang dihasilkan dari proses ini kemudian digunakan untuk proses penentuan label. Pelatihan dilakukan dengan metode SVM dan data training berupa *tweet* diubah kedalam *vector* *space* *model*. Algoritma vector space yang digunakan adalah menggunakan TF-IDF (term frequency inverse document frequency). Tahap pelatihan ini terdiri atas beberapa proses yang diperlihatkan pada gambar 4.2.



Gambar 4. Proses-proses dalam Tahap Pelatihan

### Generate Weight

SVM adalah salah satu metode yang banyak digunakan untuk klasifikasi, tetapi SVM hanya peduli tentang titik dalam ruang, bukan *tweet* atau dokumen. Untuk tujuan ini *vector* *space* *model* digunakan untuk mereprestasikan setiap kata dalam dokumen ke dalam *vector*. *Vector* *space* *model* yang digunakan dalam penelitian ini adalah TF-IDF.

Masalah utama yang dihadapi adalah bagaimana kita menentukan bobot untuk suatu kata dalam *tweet*. Secara sederhana kita dapat menghitung kemunculan suatu kata dalam suatu tweet. Misal kata ‘baik’ yang muncul dalam sebuah tweet sebanyak 2 kali maka kata tweet memiliki bobot 2. Hal ini dikenal dengan istital *term* *frequency* (tf) atau disebut berat lokal. Ini dihitung pada sebuh *tweet* tanpa mengacu pada *tweet* yang lain.

Sedangkan mitra dari berat lokal ini adalah berat global. Yaitu bobot suatu kata pada seluruh *tweet* (dokumen) yang ada. Ada juga yang mengistilahkan dengan berat umum atau *inverse* *document* *frequency*, yang didefinisikan sebagai jumlah dari semua *tweet*(dokumen) dibagi dengan jumlah *tweet*(dokumen) yang mengandung istilah tertentu. Sebagai contoh terdapat 100 dokumen, 9 diantaranya mengandung kata ‘baik’ maka berat global (idf) kata baik adalah 11.1.

Pembobotan ini digabung dengan istilah tf-idf, secara sederhana dengan mengalikannya. Hasil dari pembobotan ini akan disimpan dalam sebuah text dokumen yang nantinya akan digunakan sebagai file input proses pelatihan.

### Traning SVM

Proses pelatihan pada penelitian ini akan dilakukan dengan *library* libsvm. Hasil dari proses pelatihan ini adalah SVM Model yang disimpan dalam bentuk berkas dan nantinya akan digunakan dalam proses penentuan label dengan metode SVM.

## Penentuan Label dengan SVM

Setelah SVM Model berhasil dibuat, tahapan terakhir dalam sistem ini adalah penentuan label dengan menggunakan SVM. Dalam tahap ini terdapat beberapa proses diantaranya, pencarian *tweet*, *preprocessing*, pengecekan frase, generate bobot data testing, penentuan label seperti ditunjukan pada gambar 4.3.



Gambar 4. Proses-proses dalam Tahapan Penentuan Label

### Pencarian Tweets

Pada proses ini sistem akan mencari *tweet* yang terkait dengan kata kunci yang dimasukan oleh pengguna. Proses ini dilakukan dengan bantuan API yang telah disediakan oleh twitter. Berikut adalah format API yang disedikan twitter.

[http://search.twitter.com/search.atom?lang=[bahasa]&rpp=[jumlah]&q=[kata\_kunci](http://search.twitter.com/search.atom?lang=%5bbahasa%5d&rpp=%5bjumlah%5d&q=%5bkata_kunci)]

*Link* tersebut akan mengembalikan kata XML yang berisi dengan *tweet* yang terkait dengan kata kunci. Selanjutnya sistem akan merubah dan menyimpan data XML tersebut ke dalam database.

### Cek Frase

Sebelum Cek Frase sebenarnya dilakukan preprocessing untuk data yang baru saja didapat dari API twitter. Tetapi proses tersebut tidak dibahas disini karena telah dibahas pada subbab sebelumnya. Setelah dipreprocessing data tersebut juga dilakukan *POS taging* yang juga tidak dibahas disini karena prosesnya sama dengan proses *POS taging* pada tahap sebelumnya.

Langkah berikutnya setelah dipreprocessing dan *POS taging* adalah pengecekan apakah terdapat frase yang mengandung opini atau tidak. Jika tidak mengandung opini maka tweet tidak akan ikut diproses pada tahap penentuan sentimen tetapi langsung diberikan label netral. Jika terdapat frase opini maka akan dilakukan proses penentuan label dengan SVM. Proses pengecekan frase dilakukan dengan mencari keberadaan kata sifat atau kata kerja pada tweet.

### Generate bobot

Proses selanjutnya adalah mengenerate bobot untuk *tweet* yang mengadung frase opini. Metode pembobotan yang dilakukan akan dengan menggunakan TF-IDF yang sudah dijelaskan sebelumnya. Proses pembobotan ini akan menghasilkan sebuah berkas yang disebut dengan berkas testing yang berisi bobot dari *tweet* yang memiliki frase opini tadi. Berkas ini selanjutnya akan menjadi masukan bagi proses penentuan label dengan SVM.

### Penentuan label dengan SVM

Pentuan label dengan SVM dilakukan dengan libsvm. Proses ini menggunakan dua buah masukan yaitu SVM Model dan berkas testing. Proses ini akan menghasilkan berkas output yang merupakan hasil pelabelan dari berkas testing.

### Pemetaan dalam sentimen

Seperti diketahui SVM hanya memperdulikan titik pada vector. Output yang dihasilkan juga hanya 1 dan -1. Untuk itu proses ini kan memetakan 1 sebagai sentimen positif dan -1 sebagai sentimen negatif yang kemudian hasil pementaan ini akan disimpan dalam database yang nantinya dapat dianalisa untuk proses pengujian dan perhitungan akurasi sistem.

## Arsitektur Sistem

Arsitektur sistem *Opinion* *Mining* yang dibuat ditunjukan pada gambar 4.4. Alur pada proses pelatihan adalah melakukan *proprocessing* pada data yang sudah ada. Dimana dalam *preprocessing* melibatkan *corpus* kata formal dan kata informal, yang kemudian akan menghasilkan data *tweet* yang sudah di preprocessing atau dengan istilah data bersih. Setelah pelatihan maka akan dilakukan penentuan label pada untuk data *tweet* yang sudah dipreprocessing dengan menggunakan metode *lexicon* *based*. Selanjutnya dilakukan proses pelatihan pada data *tweet* yang sudah diberi label. Pelatihan ini akan menghasilkan SVM model.

Alur proses besar berikutnya adalah proses penentuan sentimen. Pertama sistem akan mencari *tweet* menggunakan tweetAPI yang sesuai dengan kata kunci yang dimasukan oleh pengguna. Proses ini akan menghasilkan data *tweet* dan akan disimpan dalam basisdata. Kemudian dilakukan *preprocessing* pada data *tweet* tersebut. *Prerpocessing* kurang lebih sama seperti yang dilakukan pada alur proses pelatihan. Setelah dilakukan *preprocessing* akan dilakukan pengecekan frase, apakah mengandung frase opini atau tidak. Jika mengandung frase opini maka akan dimasukan dalam proses klasifikasi. Jika tidak maka akan diberi label netral. Selanjutnya adalah klasifikasi dengan metode SVM untuk data *tweet* yang telah teridentifikasi memiliki frase opini. Klasifikasi ini melibatkan SVM model yang sudah dibuat pada proses pelatihan sebelumnya. Kemudian setelah selesai klasifikasi, hasilkan akan disimpan dalam basisdata.



Gambar 4. Arsitektur Sistem

## Rancangan Sistem

Rancangan sistem akan memberikan penjelasan tentang rancangan untuk proses pelatihan dan pemberian label. Rancangan kedua proses tersebut akan digambarkan menggunakan class diagram dan sequence diagram.

### Class Diagram

Pada penelitian ini terdapat beberapa *class* yang digunakan. *Class* *diagram* diperlihatkan pada gambar 4.5. *Class*-*class* pada gambar 4.5 terbagi menjadi beberapa jenis *class*. *Class* *model*, *class* *controller*, *class* *component* dan *class* *view*. Setiap jenis *class* memiliki peranan masing-masing. *Class* *model* berperan melakukan interaksi terhadap basisdata. *Class* *controller* berperan sebagai pengatur *class*-*class* yang lain. *Class* *component* berisi implementasi dari algoritma-algoritma yang dibutuhkan sistem. *Class* *view* berperan untuk menampilkan presentasi kepada pengguna.

*Class*-*class* yang memiliki peran fundamental dalam penelitian ini adalah pada *class* kategori *component*. *Class* MyTwitterComponent yang berfungsi untuk mencari *tweet* pada twitter. *Class* PreprocessingComponent yang memiliki fungsi untuk melakukan *preprocessing*. *Class* JanPOS tagingComponent memiliki peran untuk melakukan *POS taging*. *Class* SpellingCorrectionComponent yang berfungsi untuk melakukan *spelling* *correction* untuk kata-kata yang tidak terdapat dalam corpus. *Class* WeightComponent adalah *class* yang berfungsi untuk menghitung bobot *tweet* agar dalam digunakan dalam SVM. *Class* JanSvmComponent berfungsi algoritma SVM yang didalamnya memanfaatkan library libsvm. *Class* SentimentAnalysisLexiconBasedComponent berperan untuk melakukan analisis sentiment dengan menggunakan metode *lexicon* *based*.



Gambar 4. Class diagram

### Sequence diagram proses pencarian data twitter

Gambar 4.6 menunjukan *sequence* *diagram* proses pencarian *tweet*. User memasukan kata kunci pencarian melalui *class* HuntsView yang kemudian diteruskan pada *class* HuntsController. *Class* HuntsController akan memanggil *method* getAllTweets pada class MyTwitter. Setelah mendapatkan hasilnya *class* HuntsController akan menyimpan hasilnya pada basisdata dengan cara mendelegasikannya pada *Class* Tweet.



Gambar 4. Sequence diagram proses pencarian data twitter

### Sequence diagram preprocessing

Gambar 4.7 menunjukan *sequence* *diagram* *preprocessing* data. Data yang sudah disimpan dalam basisdata akan diambil oleh *class* RepositoriesController dengan mengakses *method* find yang ada pada *class* Repository selanjutnya untuk melakukan *preprocessing,* *class* RepositoriesController akan memanggil method doIt pada *class* Preprocessing. Setelah selesai dilakukan *preprocessing* data akan kembali disimpan dalam basisdata malalui *class* CleanTweet.



Gambar 4. Sequence diagram preprocessing

### Sequence diagram proses pemberian label dengan metode lexicon

Gambar 4.8 menunjukan proses pemberian label dengan metode *lexicon based*. Pertama user akan meminta melakukan analisis dengan metode *lexicon* *based* melalui *method* analisis pada *class* CleanRepositoriesView yang akan diteruskan pada *Class* CleanRepositoriesController. *Class* CleanRepositoriesController akan mengambil data yang sudah dilakukan *preprocessing* dengan bantuan *class* CleanRepository.



Gambar 4. Sequence diagram proses pemberian label dengan metode lexicon

Setelah mendapatkan data yang dimaksud, *class* CleanRepositoriesController akan melakukan *POS taging* pada data tersebut. Proses *POS taging* dilakukan dengan bantuan *class* JanPOS taging dengan cara mengakses *method* posTagDic. Setelah dilakukan *POS taging* maka proses analisis akan dimulai. Proses-proses analisisnya secara berturut turut adalah preliminaryAnalysis, checkNegation dan conclusion yang semuanya dilakukan oleh *class* SentimentAnalysisLexiconBased. Setelah analisis selesai dilakukan data tersebut kembali disimpan kedalam basisdata melalui *class* CleanRepository.

### Sequence diagram proses pemberian label dengan SVM

Gambar 4.9 adalah *sequence* *diagram* dari proses pemberian label dengan motode SVM. Langkah pertama adalah pengambilan data *tweet* yang sudah di *preprocessing* dari basisdata. Proses ini dilakukan oleh CleanTweetsController dengan bantuan *class* CleanTweet. Kemudian akan dilakukan pengecekan apakah *tweet* yang sudah di preprocessing netral atau tidak. Proses ini dilakukan oleh *class* CleanTweetController dengan cara mengakses *method* checkSentiment pada *class* SentimentAnalysisLexiconBased. Kemudian untuk data yang tidak netral akan dilakukan pembobotan yang dilakukan oleh *class* Weight. Selanjutnya Proses pemberian label dengan metode SVM dilakukan oleh *class* JanSvm yang didalamnya memanfaatkan *library* libsvm.



Gambar 4. Sequence diagram proses pemberian label dengan SVM

# 

# IMPLEMENTASI

BAB V digunakan untuk menjelaskan implementasi dari perancangan yang telah dijelaskan pada Bab IV. Implementasi yang akan dibahas adalah proses pelabelan dengan metode lexicon based dan pelabelan dengan SVM serta beberapa proses yang ada didalamnya seperti pembobotan, *preprocessing* dan *POS* *taging*. Bab ini juga digunakan untuk menjelaskan kendala-kendala yang ditemui baik yang sudah dapat diatasi dengan baik maupun yang belum. Implementasi pada bab ini menggunakan Apache *server*, bahasa pemrograman PHP Cakephp *Framework* dan basisdata MySQL serta libsvm.

## Preprocessing Data

*Preprocessing* adalah proses yang digunakan oleh kedua proses buat, baik pelabelan dengan metode *lexicon* *based* maupun pelabelan dengan SVM menggunakan metode *preprocessing* data yang sama. *Preprocessing* sendiri berperan penting dalam penentuan sentimen karena *preprocessing* akan menghasilkan data yang sudah bersih dan siap dilakukan analisis.

*Preprocessing* terdiri dari beberapa subproses seperti cleanInvalidUTF8, casefolding, removeSymbol, processNumbers, removeRepeat, *replace* *informal* *word*, *spelling* *correction* yang masing-masing telah dijelaskan pada IV.

### clearInvalidUTF8

Proses clearInvalid UTF8 adalah proses pembersihan karakter yang tidak termasuk dalam UTF8. Dalam sosial media akan banyak ditemu karakter-karakter aneh yang bukan termasuk UTF8 dan karakter tersebut tidak akan dapat tersimpan dengan baik pada MySQL dan tidak banyak mempengaruhi sentimen maka karakter-karakter tersebut akan dihapus. Implementasi dari proses penghapusan tersebut berada dalam class PreprocessingComponent yang ditunjukan pada Gambar 5.1

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8 | **public** **function** clearInvalidUTF8($str){  $str = iconv("UTF-8","UTF-8//IGNORE",$str);  $str = iconv("UTF-8","ISO-8859-1//IGNORE",$str);  $str = iconv("ISO-8859-1","UTF-8",$str);  $str = (strlen($str) > 255) ? substr($str,0,255) : $str;  **return** $str;  } |

Gambar 5. Kode method clearInvalidUTF8

Pada *method* clearInvalidUTF8 digunakan untuk menghapus karakter yang tidak termasuk dalam UTF8. Untuk melakukan hal ini menggunakan fungsi iconv yang disudah disediakan PHP. Iconv merupakan *method* yang disediakan PHP untuk mengkonfersi string kedalam encoding yang ditentukan.

### Casefolding

Casefolding adalah proses untuk merubah karakter kedalam huruf kecil. Karena dalam penentuan sentimen baik huruf kecil maupun besar tidak akan mempengaruhi sentimen. Tetapi dengan mengkonfersi kedalam huruf kecil akan mudah dalam melakukan pencarian pada *corpus*.

|  |  |
| --- | --- |
| **1**  2  3 | **public** **function** casefolding($tweet){  **return** **strtolower**($tweet);  } |

Gambar 5. Kode method casefolding

Gambar 5.2 merupakan *method* casefolding yang digunakan dalam penelitian ini. *Method* ini terdapat dalam *class* PreprocessingComponent. Karena dalam PHP *method*-*method* untuk pemrosesan *string* sudah cukup lengkap, proses ini dapat dilakukan dengan mudah hanya dengan memanggil *method* strtolower yang sudah disediakan oleh bahasa PHP.

### removeSymbol

RemoveSymbol adalah sebuah proses untuk menghapus symbol-symbol yang tidak terlalu berpengaruh pada penentuan sentimen. Misalkan simbol #, $,^,%,,\* (,). Gambar 5.3 merupakan kode yang digunakan untuk menghapus symbol.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6 | **public** **function** removeSymbol($word){  $word = preg\_replace('/[^\p{L}\p{N}\s]/u',' ', $word);  **if**($word == 'rt'){$word='';}  **return** trim($word);  } |

Gambar 5. Kode method removeSymbol

*Method* ini juga berada pada *class* PreprocessingComponent. Untuk menghapus symbol-symbol menggunakan *method* preg\_replace atau *method* yang berfungsi untuk melakukan penggantian symbol dengan parameter *regular* *expression*.

### ProcessNumbers

ProcessNumbers adalah proses pengkonfersian angka menjadi huruf. Dalam sosial media atau dalam dunia maya sering digunakan baik untuk mempersingkat kata atau hanya sekedar iseng saja. Untuk melakukan pengkonfersian sesuai yang sudah dijelaskan pada bab IV akan dilakukan oleh *method* processNumbers yang berada pada *class* PreprocessingComponent.

Gambar 5.4 merupakan kode dari *method* processNumbers yang intinya akan mengkonfersi angka menjadi karakter seperti dijelaskan pada bab IV. Caranya adalah dengan mengecek satu persatu, bila ditemukan angka dalam karakter maka akan dikonfersi dengan *method* str\_replace.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | **public** **function** processNumbers($word){  $str = "";  **for**($i=0;$i<10;$i++ ){  $str = (**string**)$i;  **if** (**strpos**($word, $str) !== **false**) {  **switch** ($i) {  **case** 0:  $word = str\_replace($str,"o",$word);  **break**;  **case** 1:  $word = str\_replace($str,"i",$word);  **break**;  **case** 2:  $pos = **strpos**($word,$str);  $word = substr($word, 0, $pos).' '.substr($word, 0, $pos );  **break**;  **case** 3:  $word = str\_replace($str,"e",$word);  **break**;  **case** 4:  $word = str\_replace($str,"a",$word);  **break**;  **case** 5:  $word = str\_replace($str,"s",$word);  **break**;  **case** 6:  $word = str\_replace($str,"g",$word);  **break**;  **case** 7:  $word = str\_replace($str,"t",$word);  **break**;  **case** 8:  $word = str\_replace($str,"b",$word);  **break**;  **case** 9:  $word = str\_replace($str,"g",$word);  **break**;  }    }  } |

Gambar 5. Kode method processNumbers

### RemoveRepeat

RemoveRepeat adalah proses untuk menghapus perulangan karakter yang berlebihan. Pada sosial media twitter telah umum digunakan perulangan karakter yang berlebihan. Baik yang tidak sengaja dilakukan karena kesalahan penulisan atau yang sengaja untuk menyangatkan makna. Contohnya adalah misal ingin mengungkapkan setuju terkadang ada yang berlebihan dengan menuliskan “setujuuuuuuuuuuu”. Hal tersebut bila tidak diatasi akan mempengarui penentuan sentimen. Dalam bahasa indonesia perulangan karakter yang berturut-turut hanya dua karakter. Untuk itu proses ini akan melakukan penghapusan karakter berulang berturut-turut yang berlebihan dan hanya disisakan menjadi dua karater yang sama secara berturut-turut.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12 | **public** **function** mytrim($word,$i){  **if**($i+2 < strlen($word)){  **if**(($word[$i] == $word[$i+1]) && ($word[$i+1] == $word[$i+2])){  $new\_word = substr($word, 0,$i);  $word = ($new\_word . substr($word, $i+1));  **return** $this->mytrim($word,$i);  }  **return** $this->mytrim($word,$i+1);  }  **return** $word;  } |

Gambar 5. kode dari proses RemoveRepeat

Gambar 5.5 merupakan kode dari proses RemoveRepeat. *Method* ini merupakan *method* *recursive* yang terdapat dalam *class* PreprocessingComponent. Algoritmanya adalah mengecek apakah karakter setelahnya dan setelahnya lagi adalah karakter yang sama dengan karakter sekarang. Jika benar maka akan dihapus satu karakter yang sama. Hal itu dilakukan sampai karakter terakhir pada suatu kata.

### *Replace* *informal* *word*

*Replace* *informal* *word* adalah proses penggantian kata yang ada dalam kamus kata tidak baku diganti dengan kata yang sesuai yang ada pada kamus kata baku. Gambar 5.6 adalah kode proses penggantian kata ini.

|  |  |
| --- | --- |
| **1**  2  3 | **if**(**isset**($aspal[$word])){  $word = $aspal[$word];  } |

Gambar 5. Kode proses penggantian kata tidak baku

Tidak terdapat algoritma yang rumit dalam algoritma ini hanya mengecek apakah kata terdapat dalam kata tidak baku. Jika ada maka akan langsung diganti dengan kata yang baku.

### *Spelling* *Corrention*

*Spelling* *Correction* adalah proses yang apabila bertemu dengan kata baru yang tidak terdapat dalam kamus maka proses ini akan menggantinya dengan kata yang paling dekat dengan kata yang ada dikamus. Proses ini juga dapat dimanfaatkan untuk mengkoreksi dari kesalahan penulisan walaupun tidak memiliki akurasi 100%. Gambar 5.7 adalah kode dari proses *spelling* *correction* yang terdapat dalam *class* SpellingCorrection.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | **public** **function** correct($word, $dictionary) {  **if**(strlen($word) <255){  $word = **strtolower**($word);  **if**(**isset**($dictionary[$word])) {  **return** $word;  }  $edits1 = $edits2 = **array**();  **foreach**($dictionary **as** $dictWord => $count) {  $dist = levenshtein($word, $dictWord);  **if**($dist == 1) {  $edits1[$dictWord] = $count;  } **else** **if**($dist == 2) {  $edits2[$dictWord] = $count;  }  }  **if**(**count**($edits1)) {  arsort($edits1);  **return** key($edits1);  } **else** **if**(**count**($edits2)) {  arsort($edits2);  **return** key($edits2);  }  }**else**{  $word = '';  }  **return** $word;  } |

Gambar 5. Kode method correct pada class SpellingCorrection

Proses *spelling* *correction* tersebut diimplementasikan menggunakan algoritma levenshtein untuk menghitung kedekatan kata baru dengan kata-kata yang ada dalam kamus. Bahasa PHP sudah mengimplementasikan algoritma levenshtein ini hanya dengan menjalankan *method* levenshtein yang sudah disediakan.

## POS taging

POS taging adalah proses untuk pelabelan kata berdasarkan kelas katanya. Proses ini meminta input sebuah kalimat atau dalam penelitian ini adalah sebuah *tweet* dan memberikan *output* sebuah kalimat yang sudah diberi label kelas katanya. Metode POS taging yang digunakan dalam penelitian ini adalah POS taging dengan menggunakan *dictionary* atau kamus.

Proses POS taging ini akan berfungsi untuk menentukan sentimen pada metode *lexicon* *based* dan akan berfungsi untuk mendeteksi kalimat bersentimen atau tidak pada metode SVM. Implementasi prosess ini dilakukan pada *class* JanPOS tagingComponent. Gambar 5.8 menunjukan kode proses POS taging.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | **public** **function** posTagDic($tweet,$id){  $results = **array**(); $parser = **new** TextParse($tweet);  $toArray = $parser->getWord(1);  $this->FormalWord->recursive = -1;  **foreach**($toArray **as** $index => $word){  **if**($this->FormalWord->find('count',**array**(  'conditions' => **array**('text' => $word)))){  $baku = $this->FormalWord->find('first',**array**(  'conditions' => **array**('text' => $word)));  **array\_push**(  $results,**array**(  'clean\_tweet\_id' => $id,  'urutan' => $index,  'word' => $word,  'jenis' => $baku['FormalWord']['pos'])  );  }**else**{  **array\_push**(  $results,**array**(  'clean\_tweet\_id' => $id,  'urutan' => $index,  'word' => $word,  'jenis' => 'NN')  );  }  }  **return** $results;  } |

Gambar 5. Kode proses POS taging

Langkah pertama dari algoritma dari POS taging menggunakan *dictionary*, adalah memecah kalimat atau *tweet* menjadi kata per kata. Kemudian mencocokan masing-masing kata tersebut ke dalam *dictionary* yang sudah terdapat kelas katanya.

## Pelabelan menggunakan metode *lexicon* *based*

Setelah data berhasil dipreprocessing artinya dapat sudah siap untuk dilakukan analisa. Pada tahap pelabelan data pelatihan, metode yang digunakan untuk pemberian label adalah metode *lexicon* *based*. Proses pelabelan dengan metode *lexicon* *based* ini dilakukan oleh *class* SentimentAnalysisLexiconBased. Pada penelitian ini analisis yang dilakukan hanya memperhatikan sampai dua kata yang berdampingan saja.

Proses analisis untuk melakukan pelabelan ini dibagi menjadi beberapa tahap. Tahap pertama adalah pencocokan kelas kata dengan *rule* yang sudah ada. Pada bab sebelumnya telah dijelaskan bahwa sentimen suatu kalimat ditentukan oleh adanya kata benda (NN) atau kata sifat(JJ). Maka tahap pertama akan mencari kata sifat atau kata benda yang ada. Gambar 5.9 adalah kode untuk mencocokan antara kelas dengan frase yang dapat membentuk opini.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66 | **public** **function** preliminaryAnalysis($sentence){  $sentiments = **array**();  $i=0;  $x = **count**($sentence);  **while**($i< $x){  **if**($i>0){  **if**(($sentence[$i]['jenis'] == 'VB')  &&($sentence[$i-1]['jenis'] == 'RB' || $sentence[$i-1]['jenis'] == 'CK'  ||$sentence[$i-1]['jenis'] == 'NN' || $sentence[$i-1]['jenis'] == 'JJ'  )){  **array\_push**(  $sentiments,  **array**(  'analysis' => $this->analysisTwoWord($sentence[$i-1],$sentence[$i]),  $sentence[$i-1],  $sentence[$i]  )  );  }**else**{  **if**(($sentence[$i]['jenis'] == 'JJ') &&  ($sentence[$i-1]['jenis'] == 'RB' || $sentence[$i-1]['jenis'] == 'NN'  ||$sentence[$i-1]['jenis'] == 'VB'  )){  **array\_push**(  $sentiments,**array**(  'analysis' => $this->analysisTwoWord($sentence[$i-1],$sentence[$i]),  $sentence[$i-1],  $sentence[$i]  ));  }**else**{**if**($sentence[$i]['jenis'] == 'JJ' || $sentence[$i]['jenis'] == 'VB'){  **array\_push**(  $sentiments,  **array**(  'analysis' => $this->analysisOneWord($sentence[$i]),  $sentence[$i]  )  );  }}}  }**else**{**if**($sentence[$i]['jenis'] == 'JJ' || $sentence[$i]['jenis'] == 'VB'){  **array\_push**(  $sentiments,**array**(  'analysis' => $this->analysisOneWord($sentence[$i]),  $sentence[$i]  )  );  }}  $i=$i+1;  }  $senti = **array**();  **foreach**($sentiments **as** $analysis){  **if**(($analysis['analysis'] != **null**) || ($analysis['analysis'] != '') || (!**empty**($analysis['analysis']))){  $senti[] = $analysis;}  }  **return** $senti;  } |

Gambar 5. pencocokan kelas kata dan frase

Pada gambar 5.9 *method* prelimeneryAnalysis akan mencari kata benda atau kata sifat. Bila terdapat kata benda atau kata sifat akan dicek kata selanjutnya apakah ada frase yang bersesuai dengan frase yang membentuk opini atau tidak. Jika terdapat frase yang membentuk opini akan dilakukan analisis dua kata. Gambar 5.10 adalah kode *method* analysisTwoWords. Namun apabila tidak ditemukan frase yang bersesuaian tetapi ditemukan kata benda atau kata sifat maka akan dilakukan analisis satu kata. Gambar 5.11 adalah kode *method* analysisOneWord yaitu *method* yang digunakan untuk menganalisis satu kata. Apabila dalam kalimat atau *tweet* tidak terdapat kata benda ataupun kata sifat maka kalimat atau *tweet* tersebut akan langsung diberi label netral.

Setelah ditentukan setimen awal, tahap berikutnya adalah memeriksa apakah sebelum frase yang menunjukan opini terdapat kata negasi atau tidak.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | **public** **function** analysisTwoWord($one,$two){  $this->FormalWord->recursive = -1;  $sentimenOne = $this->FormalWord->find('first',**array**('conditions' => **array**('text' => $this->streamWord($one['word']))));  $sentimenTwo = $this->FormalWord->find('first',**array**('conditions' => **array**('text' => $this->streamWord($two['word']))));    **if**($sentimenOne['FormalWord']['status'] == $sentimenTwo['FormalWord']['status']){  **return** $sentimenOne['FormalWord']['status'];  }**else**{  **if**(($sentimenOne['FormalWord']['status'] == 'netral' || $sentimenOne == **false**  || $sentimenOne['FormalWord']['status'] == **null**) && $sentimenTwo['FormalWord']['status'] != 'netral'){  **return** $sentimenTwo['FormalWord']['status'];  }**else**{  **if**(  ($sentimenOne['FormalWord']['status'] != 'netral' ) &&  ($sentimenTwo['FormalWord']['status'] == **null** ||  $sentimenTwo['FormalWord']['status']== 'netral' ||  $sentimenTwo == **false**  )){  **return** $sentimenOne['FormalWord']['status'];  }**else**{  **if**(  $sentimenOne['FormalWord']['status'] != 'netral' &&  $sentimenTwo['FormalWord']['status'] != 'netral' &&  ($sentimenOne['FormalWord']['status'] == 'negatif' ||  $sentimenTwo['FormalWord']['status'] == 'negatif'  )  ){  **return** 'negatif';  }**else**{  **return** 'netral';  }  }  }  }  } |

Gambar 5. kode method analysisTwoWord

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7 | **public** **function** analysisOneWord($one){  $this->FormalWord->recursive = -1;  $sentimenOne = $this->FormalWord->find('first',  **array**('conditions' => **array**('text' => $this->streamWord($one['word']))));  **return** $sentimenOne['FormalWord']['status'];  } |

Gambar 5. kode method analysisOneWord

Bila terdapat kata negasi sebelum frase maka sentimen dari kalimat atau *tweet* akan diubah. Jika positif akan menjadi negatif dan bila negatif akan menjadi positif. Jarak kata negasi dengan frase diberi toleransi sebanyak lima kata sebelumnya. Gambar 5.12 adalah kode *method* yang digunakan untuk memeriksa negasi.

Setelah tahap pemeriksaan negasi, tahap berikutnya adalah penarikan kesimpulan. Beberapa hal yang perlu diperhatikan adalah sentimen awal, negasi dan menghitung jumlah frase opini yang ada. Kesimpulan diambil berdasarkan perbandingan jumlah opini yang ada. Bila jumlah frase positif lebih banyak maka kalimat akan diberi label positif, apabila frase negatif lebih banyak maka kalimat akan diberi label negatif. Gambar 5.13 menunjukan kode penarikan kesimpulan yang dilakukan.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | **public** **function** checkNegation($data){  $negation = $this->FormalWord->find('list',**array**('conditions' =>  **array**('status' => 'kontra'), 'fields' => **array**('FormalWord.text','FormalWord.text')));    **for** ($i=0; $i< **count**($data['frase']); $i++){  **if**($data['frase'][$i]['analysis'] != **null**){  $data['frase'][$i]['negation'] = **false**;  **for**($j=0; $j<**count**($data['frase'][$i]) -2;$j++){  **for**($k=1; $k<=5; $k++){  **if**(($data['frase'][$i][$j]['urutan']-$k) >=0 ){  **if**(**isset**($negation[$data[$data['frase'][$i][$j]['urutan']-$k]['word']])){  $data['frase'][$i]['negation'] = **true**;  }  }  }    }  }  }  **return** $data;  } |

Gambar 5. kode method chcekNegation

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | **public** **function** conclusion($frase){  $positif = 0;  $negatif = 0;  **foreach**($frase **as** $f){  **if**($f['analysis'] == 'positif'){// sentiment +  **if**($f['negation']){ // negation true  $negatif++;  }**else**{ // negation false;  $positif++;  }  }**else**{  **if**($f['analysis'] == 'negatif'){//sentiment -  **if**($f['negation']){//negation true  $positif++;  }**else**{ // negation false  $negatif++;  }  }  }  }  $kesimpulan = '';    **if**($positif > $negatif){  $kesimpulan = 'positif';  }**else**{  **if**($positif < $negatif){  $kesimpulan = 'negatif';  }**else**{  $kesimpulan = 'netral';  }  }  **return** $kesimpulan;  } |

Gambar 5. Kode method conclusion

## Pembobotan data

Setelah data berhasil diberi label dengan metode *lexicon* *based*. Data tersebut akan menjadi data training bagi proses pelabelan dengan SVM. Data ini masih berbentuk *tweet* yang tersimpan dalam basisdata sedang SVM hanya mengenali input dan output nya dalam bentuk vector. Untuk itu perlu adanya metode untuk merepresentasikan data ini dalam bentuk vector. Metode tersebut disebut dengan *vector* *space* *model*. Ada beberapa metode *vector* *space* *model*, namun penelitian ini akan menggunakan metode TF-IDF untuk *vector* *space* *model*.

TF-IDF adalah pembobotan yang mengkombinasikan antara berat kata local dan berat global seperti yang sudah dijelaskan pada bab sebelumnya. Metode pembobotan selain digunakan pada proses pembobotan data pelatihan juga digunakan pada proses pembobotan data pengujian. Implementasi dari metode TF-IDF ini dilakukan pada *class* WeightComponent. Gambar 5.14 merupakan kode untuk membangun data pelatihan.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | **public** **function** buildTrainingData(){  file\_put\_contents($this->filetraining->pwd(), "");  $this->CleanRepository->recursive = -1;  $positif = $this->CleanRepository->find('all', **array**(  'conditions' => **array**('CleanRepository.sentiment' => 'positif'),  'limit' => 12500  )  );  $negatif = $this->CleanRepository->find('all', **array**(  'conditions' => **array**('CleanRepository.sentiment' => 'negatif'),  'limit' => 12500  )  );  $all = **array\_merge**($positif, $negatif);  $allDoc = Set::classicExtract($all, '{n}.CleanRepository');  $index = $this->getIndex($allDoc);  **foreach**($allDoc **as** $id => $d){  $this->getTfidf($d,$id,$allDoc,$index,'train');  }  } |

Gambar 5. Kode untuk membangun data training

Dalam *method* buildTrainingData seperti pada gambar 5.14. Program akan mengambil data pelatihan sebanyak 25000 *tweet* dari basisdata, yang terdiri dari 12500 data berlabel positif dan 12500 berlabel negatif. Kemudian masing-masing data tersebut akan di-*index*, yaitu menghitung kemunculannya secara global dan lokal. Gambar 5.15 adalah *method* yang digunakan untuk melakukan *indexing*.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52 | **public** **function** getIndex($collection) {  $dictionary = **array**();  $docCount = **array**();  $result = **array**();  $i =1;  **foreach**($collection **as** $id => $doc) {  $docID = $id;  $terms = **explode**(' ', $doc['content']); // dapat array kata  $docCount[$docID] = **count**($terms); // jumlah kata pada kalimat ke-i  $status = $doc['sentiment'];    **foreach**($terms **as** $term) {  **if**(!**isset**($dictionary[$term])) { // jika menemukan kata baru  // masukan dalam $dictionary  $dictionary[$term] = **array**( 'netral' => 0,'positif' => 0,  'negatif' => 0,'df' => 0, 'postings' => **array**());  $dictionary[$term]['index'] = $i;  $i++;  }  **if**(!**isset**($dictionary[$term]['postings'][$docID])) {  $dictionary[$term]['df']++;  $dictionary[$term]['postings'][$docID] = **array**('tf' => 0);  **if**($status == 'positif'){  $dictionary[$term]['positif']++;  }**else**{  **if**($status == 'negatif'){  $dictionary[$term]['negatif']++;  }**else**{  **if**($status == 'netral'){  $dictionary[$term]['netral']++;  }  }  }  }  $dictionary[$term]['postings'][$docID]['tf']++;  }  }  **unset**($dictionary['']);  **return** **array**('docCount' => $docCount, 'dictionary' => $dictionary);  } |

Gambar 5. Kode method getIndex

Kode pada gambar 5.15 akan menghitung frekuensi kemunculan setiap kata, baik frekuensi kemunculan dalam sebuah dokumen maupun frekuensi kemunculan untuk setiap dokumen. Selanjutnya setelah dilakukan peng-*index*-an akan dilakukan menghitungan nilai TFIDF. Gambar 5.16 merupkan kode untuk menghitung nilai TFIDF.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | **function** getTfidf($doc,$id,$allDoc,$index,$mode) {  $docCount = **count**($index['docCount']);  $terms = **explode**(' ', $doc['content']);  $idDoc = $id;  $status = **array**('netral'=> '0','positif' => '1' ,'negatif' => '-1');  $hasil = **array**();  **if**($doc['sentiment'] != 'netral' ){  **if**($mode == 'test'){  $result = "0";  }**else**{  $result = "{$status[$doc['sentiment']]}";  }  **foreach**($terms **as** $i => $term){  **if**(**isset**($index['dictionary'][$term])){  $urutan = $index['dictionary'][$term]['index'];  $entry = $index['dictionary'][$term];  $score = round($entry['postings'][$idDoc]['tf']  \* log(($docCount+1) / ($entry['df']+1), 2), 5);  $hasil[$urutan] = $score;  }  }  **ksort**($hasil);  **foreach**($hasil **as** $index => $bobot){  $result = $result." {$index}".":"."{$bobot}";    }  $result .= "\n";  **if**($mode == 'test'){  **if**($this->filetesting->exists()){  $this->filetesting->append($result);  }  }**else**{  **if**($this->filetraining->exists()){  $this->filetraining->append($result);  }  }  }  } |

Gambar 5. Kode method getTfidf

Kode pada gambar 5.16 adalah kode untuk menghitung nilai TFIDF seperti yang sudah dijelaskan pada bab sebelumnya. Setelah berhasil dihitung nilai-nilai TFIDF yang dihasilkan akan disimpan dalam berkas pelatihan yang nantinya akan dibaca oleh libsvm.

## Pelatihan

Setelah semua siap maka sekarang proses pelatihan sudah dapat dilakukan. Data pembobotan yang sudah dibuat sebelumnya akan disimpan dalam sebuah berkas. Selanjutnya proses pelatihan ini akan implementasikan memanfaatkan *library* libsvm. Pada penelitian ini proses pelatihan ini di implementasikan pada *class* JanSvmComponent. Gambar 5.17 kode yang digunakan untuk memanggil fungsi pelatihan yang sudah diimplementasikan oleh libsvm.

|  |  |
| --- | --- |
| **1**  2  3  4 | **public** **function** train($filetraining,$filemodel){  chdir('files');  exec('svm-train '.$filetraining.' '.$filemodel);  } |

Gambar 5. Kode pemanggil method pelatihan

Setelah kode pada gambar 5.17 selesai dijalankan maka akan terbentuk SVM model yang disimpan dalam bentuk berkas. Nantinya berkas SVM model tersebut akan dimanfaatkan pada proses *testing*.

## Pencarian *Tweet*

Proses-proses sebelumnya yang sudah dilakukan adalah bagian dari proses utama yaitu proses pelatihan. Pada bagian subbab ini dan seterusnya akan dibahas bagian-bagian dari proses utama yang kedua yaitu *Testing*. Subproses pertama pada proses *testing* ini adalah pengguna akan memasukan kata kunci yang ia akan cari sentimennya. Kemudian sistem akan mencari data *tweet* yang bersesuai dengan kata kunci yang dimasukan oleh pengguna. Proses pencarian ini menggunakan fitur API twitter yang memang sudah disediakan oleh pengembang twitter. Gambar 5.18 adalah kode yang digunakan untuk mencari data pada twitter. *Method* ini terletak pada *class* MyTwitterComponent.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | **public** **function** getTweets($q){  App::uses('Xml', 'Utility');  $output = **array**();  $this->searchURL =  'http://search.twitter.com/search.atom?page='.$this->settings['page'].  '&lang='.$this->settings['lang'].'&rpp='.$this->settings['rpp'].'&q=';  $ch= **curl\_init**($this->searchURL . urlencode($q));  **curl\_setopt**($ch, CURLOPT\_RETURNTRANSFER, 1);  **curl\_setopt**($ch, CURLOPT\_USERAGENT, $\_SERVER['HTTP\_USER\_AGENT']);  $response = **curl\_exec**($ch);  $xmlResult = Xml::toArray(Xml::build($response));    **curl\_close**($ch);  **return** $xmlResult;  } |

Gambar 5. Kode untuk mencari data pada twitter

Pada kode pada gambar 5.18 program akan mencari *tweet* yang bersesuai dengan kata kunci yang dimasukan oleh pengguna. Nilai kembalian dari twitter API dalam bentuk XML akan diubah dalam *array* yang nantinya akan disimpan dalam basisdata sebagai data kotor untuk selanjutnya dilakukan *preprocessing*.

## Pengecekan frase opini

Setelah mendapatkan *tweet* yang bersesuaian dengan kata kunci yang dimasukan dan disimpan dalam basisdata. Tweet-tweet tersebut akan di-*preprocessing*. *Preprocessing* yang dilakukan pada proses *testing* ini sama dengan *preprocessing* yang dilakukan pada proses pelatihan. Kemudian untuk melakukan pengecekan ada atau tidaknya frase yang mengandung opini maka harus dilakukan POS taging pada *tweet*-*tweet* yang sudah di-*preprocessing* tadi. Proses POS taging yang dilakukan pada proses *testing* juga sama dengan POS taging yang dilakukan pada proses pelatihan, jadi tidak akan dijelaskan kembali.

Proses berikutnya adalah proses pengecekan ada atau tidaknya frase opini. Gambar 5.19 merupakan kode untuk melakukan pengecekan ada atau tidaknya frase yang mengandung opini.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | **public** **function** checkSentiment($sentence,$sentimentword){  $status = **false**;  $i=0;  $x = **count**($sentence);  **while**(($i < $x) && (!$status)){  **if**(($sentence[$i]['jenis'] == 'VB') || ($sentence[$i]['jenis'] == 'JJ')){  **if**(**isset**($sentimentword[$sentence[$i]['word']])){  **if**(($sentimentword[$sentence[$i]['word']] == 'negatif')  || ($sentimentword[$sentence[$i]['word']] == 'positif')){  $status = **true**;  }  }  }  $i++;  }  **return** $status;  } |

Gambar 5. Kode untuk memeriksa frase opini

Dalam menentukan ada atau tidaknya frase opini, program akan memeriksa apakah terdapat kata kerja(VB) atau kata sifat(JJ), jika tidak ada maka dianggap tidak memiliki frase opini tetapi apabila terdapat kata kerja atau kata sifat akan dilakukan pengecekan lagi. Apakah kata kerja atau kata sifat yang ditemukan mengandung sentimen atau tidak. Karena tidak semua kata kerja atau kata sifat yang memiliki sentimen. Untuk *tweet* yang tidak memiliki frase opini akan langsung diberi label netral sedangkan untuk *tweet* yang terindikasi memiliki frase opini akan dilanjutkan dalam proses pelabelan dengan menggunakan metode SVM.

## Pelabelan dengan metode SVM

Setelah berhasil dipisahkan data *tweet* yang mengandung frase opini dan yang tidak. Langkah berikutnya adalah mengkonversi *tweet* yang mengandung frase opini menjadi bentuk *vector* yang dapat dikenali oleh SVM. Metode yang digunakan masih sama dengan yang digunakan pada proses pelatihan yaitu menggunakan TFIDF. Implementasi dari TFIDF juga sudah dibahas sebelumnya.

Proses pelabelan dengan metode SVM dilakukan dengan bantuan SVM *library* yang dipanggil melalui *class* JanSvmComponent. Gambar 5.20 adalah kode implementasi pemanggilan libsvm.

|  |  |
| --- | --- |
| **1**  2  3  4  5 | **public** **function** test($filetesting,$filemodel,$fileout){  chdir('files');  exec('svm-predict '.$filetesting.' '.$filemodel.' '.$fileout);  } |

Gambar 5. Kode pemanggilan libsvm

Dengan menjalankan kode pada gambar 5.20 maka akan dilakukan proses *testing* oleh libsvm. Proses ini akan beri label 1 atau -1 untuk setiap *vector*. Langkah terakhir adalah menyimpan dalam basisdata dengan label yang bersesuaian dengan dokumennya. Label 1 diartikan positif dan label -1 diartikan negatif. Gambar 5.21 adalah kode untuk menyimpan label yang sudah dihasilkan oleh libsvm ke dalam basisdata.

|  |  |
| --- | --- |
| **1**  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | $lines=**array**();    $fp=**fopen**(WWW\_ROOT.'files/jan.out', 'r');  **while** (!feof($fp)){  $line=fgets($fp);  $line=trim($line);  $lines[]=$line;  }  fclose($fp);    $hasil['positif'] = 0;  $hasil['negatif'] = 0;  **foreach**($data **as** $i => $d){  **if**($lines[$i] > 0){  $hasil['positif'] += 1;  $this->CleanTweet->id = $d['CleanTweet']['id'];  $this->CleanTweet->saveField('sentiment','positif');  }**else**{  $hasil['negatif'] += 1;  $this->CleanTweet->id = $d['CleanTweet']['id'];  $this->CleanTweet->saveField('sentiment','negatif');  }  } |

Gambar 5. Kode untuk menyimpan hasil pelabelan

Sampai dengan proses ini maka telah selesai proses pelabelan data dengan metode SVM. Data *tweet* yang baru saja cari sudah diberikan label dan siap ditampilkan kepada pengguna.

# 

# HASIL PENELITIAN DAN PEMBAHASAN

Bab VI memuat hasil penelitian yang terdiri dari tiga bagian. Pertama, perbandingan antara proses pelabelan data pelatihan yang dilakukan (Alliandu dan Winarko, 2012) yaitu menggunakan metode emoticon dan pelabelan data pelatihan yang dilakukan dengan metode *lexicon* *based*. Kedua, percobaan pemberian label dengan menggunakan metode gabungan antara *lexicon* *based* dan *support* *vector* *machine*. Ketiga, perbandingan hasil pelabelan metode gabungan dan pelabelan dengan cara manual.

## Perbandingan proses pelabelan data pelatihan

Proses pelabelan data yang dilakukan dalam penelitian ini akan menggunakan metode *lexicon* *based* sedangkan data pelatihannya mengambil dari pelatihan yang dilakukan oleh (Alliandu dan Winarko, 2012). Pada pelatihan yang dilakukan (Alliandu dan Winarko, 2012) metode yang digunakan adalah dengan metode emoticon. Metode emoticon adalah dengan cara mendeteksi emoticon-emoticon tertentu, yang setiap emoticon memiliki korelasi dengan label.

### Hasil perbandingan pelabelan data pelatihan

Percobaan dilakukan menggunakan 300 data yang diambil secara random yang sebelumnya sudah diberi label pada penelitian (Alliandu dan Winarko, 2012). Pada percobaan ini yang dibandingan adalah tingkat akurasi antara metode emoticon yang digunakan pada penelitian (Alliandu dan Winarko, 2012) dan pelabelan manual dibandingkan dengan metode *lexicon* *based* yang dilakukan pada penelitian ini dan pelabelan manual.

Pada pengujian yang dilakukan terhadap metode emoticon dan pelabelan manual memiliki akurasi 0.48666666666667 yang secara detail ditampilkan pada Gambar 6.1.



Gambar 6. Perbandingan metode emoticon dan pelabelan manual

Pada pengujian yang dilakukan terhadap metode *lexicon* *based* dan pelabelan manual memiliki akurasi 0.68 yang secara detail ditampilkan pada Gambar 6.2



Gambar 6. Perbandingan metode lexicon based dan pelabelan manual

Berdasarkan pada kedua percobaan yang sudah dilakukan, pelabelan yang dilakukan dengan metode lexicon based ternyata memiliki akurasi yang lebih baik yaitu 0.68 dibandingkan pelabelan yang dilakukan dengan metode emoticon yaitu hanya 0.48667.

## Pemberian Label

Proses pemberian label dimulai dari pengguna memasukan kata kunci topik yang akan dicari kecenderungan sentimennya. Gambar 6.3 merupakan tampilan dari input kata kunci yang diminta oleh sistem.



Gambar 6. Tampilan input kata kunci

Pada gambar 6.3 menunjukan tampilan input kata kunci. Pengguna sedang akan mencari tahu kecenderungan sentimen dengan kata kunci “telkomsel”. Selanjutnya pengguna harus menekan tombol Search untuk melanjutkan proses. Sistem akan mencari pada twitter, *tweet* yang didalamnya terdapat kata “telkomsel”. Proses pencarian ini menggunakan API yang sudah disediakan oleh pengembang twitter. Setelah ditemukan *tweet* yang mengandung kata “telkomsel” maka tweet yang sudah ditemukan akan disimpan dalam basis data.

Proses berikutnya adalah melakukan preprocessing pada data tweet yang sudah ditemukan. Gambar 6.4 menunjukan pengguna akan meminta sistem untuk melakukan preprocessing. Sistem akan melakukan preprocessing setelah pengguna menekan link “lakukan proprocessing”.



Gambar 6. link untuk melakukan preprocessing

Setelah proprocessing selesai dijalankan. Data *tweet* baru yang sudah dilakukan preprocessing akan kembali simpan sebagai data bersih didalam basis data. Data bersih yang baru saja disimpan akan ditampilkan kepada pengguna. Gambar 6.5 merupakan contoh data bersih yang ditampilkan.



Gambar 6. sistem menampilkan data bersih

Pada gambar 6.5 data yang ditampilkan adalah data yang sudah dilakukan preprocessing dan yang belum dilakukan preprocessing. Jadi sangat mudah untuk melihat perbedaan antara data yang sudah dilakukan preprocessing dan yang belum. Selain menampilkan data *tweet* , pada halaman ini juga terdapat link untuk melakukan proses-proses berikutnya. Misal, “lakukan analisis awal”,”generate test” dan “analysis”

Selanjutnya ada dilakukan proses analisis awal, yaitu memisahkan antara *tweet* yang mengandung frase opini dan yang tidak. *Tweet* yang mengandung frase opini akan dimasukan dalam proses pelabelan dengan SVM sedangkan yang tidak mengandung opini akan langsung diberikan label netral. Gambar 6.6 menunjukan data tweet setelah dilakukan analisis awal.



Gambar 6. Data tweet setelah dilakukan analisis awal

Pada gambar 6.6 terlihat beberapa *tweet*  sudah diberi label netral itu menandakan *tweet* tersebut tidak mengandung frase opini. Sedangkan yang lainnya belum diberi label, hal itu menunjukan dalam *tweet* tersebut mengandung frase opini. Seperti telah dijelaskan sebelumnya *tweet* yang mengandung frase opini akan masukan dalam proses pelabelan dengan SVM. Langkah pertama adalah dengan melakukan perhitungan bobot untuk *tweet* yang mengandung frase opini. Pengguna dapat menekan link “Generate test” untuk melakukan pembobotan, seperti ditunjukan pada gambar 6.7.



Gambar 6. Pembobotan tweet

Pada gambar 6.7 terlihat setelah sistem melakukan pembobotan akan muncul notifikasi bahwa pembobotan data testing sudah selesai dilakukan. Hal ini berarti proses pelabelan dengan SVM sudah siap untuk dilakukan. Pengguna dapat melakukan pelabelan dengan SVM dengan menekan *link* “Analysis”.

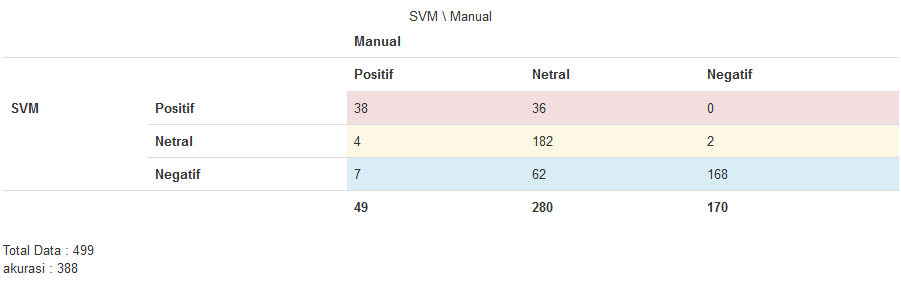


Gambar 6. Hasil pelabelan dengan metode SVM

Gambar 6.8 menunjukan hasil analisis pelabelan *tweet* yang memiliki frase opini dengan metode SVM. Hasil pelabelan ini akan langsung dipisahkan antara *tweet* yang bersentimen positif, negatif dan netral. Hasil dari pelabelan ini juga disimpan dalam basis data.

## Akurasi Pemberian Label

Penghitungan akurasi dari penelitian ini adalah dengan membandingkan hasil pelabelan yang dilakukan dengan metode kombinasi antara *lexicon* *based* dan SVM dibandingkan dengan pelabelan yang dilakukan secara manual. Data yang digunakan untuk perbandingan adalah data dari 3 kali percobaan dengan masukan kata kunci yang berbeda-beda. Berdasarkan 3 percobaan yang dilakukan didapatkan 499 data *tweet.* Gambar 6.9 menunjukan hasil perbandingan yang sudah dilakukan. Pada gambar 6.9 ditunjukan dari 499 *tweet* , jumlah *tweet* yang diberi label oleh sistem dengan benar berjumlah 388 *tweet*. Berdasarkan data tersebut, akurasi dari penelitian ini adalah 0.777



Gambar 6. Akurasi Sistem

# 

# KESIMPULAN DAN SARAN

## Kesimpulan

Penelitian dan percobaan telah dilakukan, menghasilkan kesimpulan sebagai berikut :

1. Pada proses pengumpulan data training metode *lexicon based* memberikan akurasi yang lebih baik dari pada metode *emoticon*. Dalam penelitian ini metode *lexicon based* mencapai akurasi 68% sedangkan metode *emoticon* hanya mencapai akurasi 48,7%. Pengujian ini berdasarkan 300 *tweet* random berbahasa indonesia yang terdapat emoticon pada semua *tweet.*
2. Pada pengujian akhir metode gabungan antara lexicon based dan support vector machine menghasilkan akurasi 77,7%. Pengujian ini berdasarkan 499 *tweet* random berbahasa indonesia yang diambil langsung dari *twitter*.

## Saran

Pada penelitian ini metode postagging yang digunakan adalah menggunakan *dictionary.* Ternyata hasil postagging menggunakan *dictionary* kurang bagus. Padahal postagging digunakan dalam *lexicon based* yang merupakan metode yang digunakan untuk pengumpulan data *training.* Untuk itu diperlukan metode postagging yang memiliki akurasi lebih tinggi. Sehingga hasil dari akurasi pengumpulan data dapat lebih baik lagi.

Pada metode *lexicon based* pencocokan frase dengan data sentimen yang ada pada *dictionary* bersifat tunggal. Artinya satu kata hanya memiliki satu sentimen. Padahal sangat dimungkinkan satu kata dapat memiliki lebih dari satu sentimen. Untuk itu dibutuhkan *dictionary* kata sentimen yang masing-masing katanya dapat memiliki lebih dari satu sentimen.

Pada penelitan ini menggunakan data pelatihan sebanyak 25000. Ternyata data pelatihan belumlah cukup. Penambahan data *training* dapat meningkatkan akurasi dari klasifikasi yang dilakukan dengan SVM.

# DAFTAR PUSTAKA

Alliandu, P. dan Winarko, E., 2012. *Analisis Sentimen Tweet Berbahasa Indonesia di Twitter*. *Tesis*. : Program Pasca Sarjana Universitas Gadjah Mada.Yogyakarta.

Franky dan Manurung, R., 2008. Machine Learning-based Sentiment Analysis of Automatic Indonesian Translations of English Movie Reviews. *In Proceedings of the International Conference on Advanced Computational Intelligence and Its Applications*.

Harb, A. et al., 2008. Web opinion mining: how to extract opinions from blogs? In *Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology*. New York, 2008. Proceeding.

Indurkhya, N. dan Damerau, F., 2010. *Handbook of Natural Language Processing, Second Edition*. Florida: Chapman and Hall/CRC.

Jurafsky, D. dan Martin, J., 2008. *Speech and Language Processing Second Edition*. New Jersey: Prentice Hall.

Kao, A. dan Poteet, S., 2006. *Natural Language Processing and Text Mining*. New York: Springer.

Komarsilam, E. dan Winarko, E., 2012. *Penambangan Opini Pada Situs Review Film Berbahasa Indonesia*. *Tesis*. : Program Pasca Sarjana Universitas Gadjah Mada.Yogyakarta.

Liu, B., 2011. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Berlin: Springer.

Manning, C., Raghavan, P. dan Schütze, H., 2008. *Introduction to Information Retrieval*. Cambridge: Cambridge University Press.

Manning, C. dan Schuetze, H., 1999. *Foundations of Statistical Natural Language Processing*. Cambridge: Cambridge University Press.

Naradhipa, A.R. dan Purwarianti, A., 2011. *Sentiment Classification for Indonesian Messages in Social Media*. : International Conference on Electrical Engineering and Informatics.Bandung.

Pak, A. dan Paroubek, P., 2010. *Twitter as a Corpus for Sentiment Analysis and Opinion Mining*. : European Language Resources Association.Valletta.

Pang, B. dan Lee, L., 2008. Opinion mining and sentiment analysis. In *Foundations and Trends in Information Retrieval 2*., 2008. Now Publisher Inc.

Pang, B., Lee, L. dan Vaithyanathan, S., 2002. Thumbs up?: Sentiment Classification Using Machine Learning Techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*. Stroudsburg, 2002. Proceeding.

Quirk, R., Greenbaum, S., Leech, G. dan Svartvik, J., 1985. *A Comprehensive Grammar of the English Language*. London: Longman.

Semiocast, 2012. *Brazil becomes 2nd country on Twitter, Japan 3rd Netherlands most active country*. [Online] Available at: <http://semiocast.com/en/publications/2012_01_31_Brazil_becomes_2nd_country_on_Twitter_superseds_Japan> [Accessed 30 May 2012].

Zhang, C., Zuo, W., Peng, T. dan He, F., 2008. Sentiment Classification for Chinese Reviews Using Machine Learning Methods Based on String Kernel. In *Proceedings of the 2008 Third International Conference on Convergence and Hybrid Information Technology*. Washington, DC, 2008. Proceeding ICCIT '08.

# LAMPIRAN A

# CONTOH DATA TRAINING

1 1:13.48941 2:10.48941 3:8.22262 4:4.13076 5:10.68205 6:4.76149 7:10.68205 8:8.46704 9:9.73452

1 11:10.40195 12:13.48941 13:12.48941 14:9.44501 15:12.90445 16:10.09709 17:10.02998 18:11.68205 19:3.34348 20:9.40195 21:10.48941 22:2.00157 23:13.48941 24:11.16748 25:10.90445 26:10.68205

1 19:3.34348 22:6.00471 27:12.90445 28:10.31948 29:12.90445 30:2.63376 31:8.16748 32:4.70641 33:9.02998 34:13.48941 35:8.38088 36:11.90445 37:7.74794 38:6.86736 39:9.44501 40:12.90445 41:12.90445 42:10.58252

1 22:2.00157 43:12.48941 44:11.48941 45:6.41795 46:6.49506 47:13.48941 48:9.84555

1 19:3.34348 49:7.01368 50:13.48941 51:13.48941 52:16.72025 53:8.96585 54:12.48941 55:10.58252 56:6.84555

1 22:10.00784 221:7.23202 1846:14.91197 1872:8.31948 2462:11.48941 3058:10.58252 3553:13.48941 3554:13.48941 3555:13.48941 3556:13.48941

1 22:4.00314 170:3.86736 260:4.07578 1187:6.00559 2001:6.06734 2262:11.16748 3557:9.96585

1 6:4.76149 19:3.34348 22:4.00314 125:3.00559 235:8.26059 334:7.05478 344:5.26059 488:5.90821 1497:9.24148 1893:20.71334

-1 1711:8.16748 3094:6.88208

-1 32:4.70641 260:4.07578 272:6.40195 578:7.86005 614:5.69825 816:12.7387 1822:3.59611 2245:8.22262 2567:8.48941 14029:22.05995 20074:13.48941

-1 78:1.86576 198:5.29465 275:3.34986 459:10.48941 769:6.12309 1943:8.68205 2083:6.79592 2453:8.53521 2558:15.87658 3242:10.16748 3324:10.24148 3425:5.80993 14624:23.80889 20075:13.48941

-1 170:3.86736 281:7.8312 1869:7.57055 2909:11.31948 5032:6.92717 9523:11.90445 13109:12.48941 14512:12.90445 20076:13.48941 20077:13.48941 20078:13.48941

-1 308:5.28729 1363:3.48238 1430:25.96956 1849:8.68205 1965:6.34475 2221:5.18563 2446:7.70805 3812:8.58252 7694:11.68205 12317:10.09709 20079:13.48941 20080:13.48941 20081:13.48941 20082:13.48941

-1 78:1.86576 101:2.38809 170:3.86736 210:4.59156 308:5.28729 586:7.34986 1108:4.72454 1156:8.58252 1196:11.48941 1691:4.23674 2012:4.54104 2154:9.20401 2231:5.97371 2660:7.78897 3442:16.97882 8040:10.09709

-1 3861:8.58252 5714:10.58252 12916:12.48941 16591:11.90445

-1 1963:6.88208 3974:8.13186 15167:12.90445 20083:13.48941 20084:13.48941 20085:13.48941

# LAMPIRAN B

# CONTOH DATA TESTING

0 1:0.0904166 2:0.0413477 4:0.0804166 5:0.0404166 6:0.0487173 7:0.042343 8:0.0513477 9:0.062343 10:0.062343 11:0.054567 12:0.064567 13:0.064567 14:0.0804166

0 2:0.0413477 7:0.042343 15:0.064567 16:0.074567 17:0.0704166 18:0.074567 19:0.074567 20:0.074567 21:0.074567 22:0.064567 23:0.0671973 24:0.074567 25:0.0471973 26:0.0375626 27:0.0604166 28:0.0604166 29:0.062343 30:0.074567

0 5:0.0404166 6:0.0487173 7:0.042343 8:0.0513477 15:0.064567 31:0.0704166 32:0.0904166 33:0.0904166 34:0.0904166 35:0.0904166 36:0.0364934 37:0.0904166 38:0.0904166 39:0.0904166 40:0.0904166 41:0.0904166 42:0.0904166 43:0.074567 44:0.0904166

0 5:0.0404166 6:0.0487173 7:0.042343 8:0.0513477 9:0.062343 10:0.062343 11:0.054567 12:0.064567

0 5:0.0404166 6:0.0487173 7:0.042343 8:0.0513477 9:0.062343 10:0.062343 11:0.054567 12:0.064567 45:0.0804166 46:0.064567 47:0.0571973 48:0.024718 49:0.0904166

0 4:0.0804166 5:0.0404166 6:0.0487173 7:0.042343 8:0.0513477 9:0.062343 10:0.062343 11:0.054567 12:0.064567 13:0.064567 14:0.0804166

# LAMPIRAN C

# CONTOH SVM MODEL

svm\_type c\_svc

kernel\_type rbf

gamma 4.92708e-005

nr\_class 2

total\_sv 17595

rho 0.512543

label 1 -1

nr\_sv 8727 8868

SV

1 1:13.48941 2:10.48941 3:8.22262 4:4.13076 5:10.68205 6:4.76149 7:10.68205 8:8.46704 9:9.73452

1 11:10.40195 12:13.48941 13:12.48941 14:9.44501 15:12.90445 16:10.09709 17:10.02998 18:11.68205 19:3.34348 20:9.40195 21:10.48941 22:2.00157 23:13.48941 24:11.16748 25:10.90445 26:10.68205

1 19:3.34348 22:6.00471 27:12.90445 28:10.31948 29:12.90445 30:2.63376 31:8.16748 32:4.70641 33:9.02998 34:13.48941 35:8.38088 36:11.90445 37:7.74794 38:6.86736 39:9.44501 40:12.90445 41:12.90445 42:10.58252