BAB III
ANALISIS DAN PERANCANGAN

Dalam bab ini akan dijabarkan analisa, yang meliputi analisa masalah dan gambaran umum masalah yang sedang dibahas, perancangan sistem serta desain antarmuka *(user interface)* dari aplikasi.

3.1 Analisa Masalah dan Gambaran Umum

Klasifikasi merupakan *prediction modelling* (model prediski) yang berkaitan dengan pembuatan model yang dapat melakukan pemetaan terhadap setiap himpunan variable ke setiap targetnya, kemudian dari model yang diperoleh tersebut dapat memberikan nilai target pada himpunan yang baru. Jadi, klasifikasi merupakan proses pengelompokan data berdasarkan pemetaan data fitur yang akan menghasilkan suatu model untuk pengelompokan pada suatu kelas tertentu.

Dalam pembangunan aplikasi, data yang akan diklasifikasi oleh sistem adalah data komentar. Data tersebut diambil dari www.viva.co.id, www.kompas.com, dan beberapa website lainnya dengan kategori politik. Data yang diambil dari situs tersebut antara lain adalah data *blog post* yang terdiri dari judul, tanggal ketika mempublikasikan *blog post,* dan isi atau *content* dari *blog post*, kemudian data komentar yang terdiri dari nama *author* (nama orang yang memberikan komentar), tanggal ketika mempublikasikan komentar, dan isi dari komentar itu sendiri.

Sebelum dilakukan proses klasifikasi, data akan melalui proses preprosesing data terlebih dahulu. Proses preprosesing yang dilakukan antara lain casefolding, tokenizing, *stopword removal,* dan stemming. Setelah proses preprosesing, akan dilakukan seleksi fitur dan pembobotan. Seleksi fitur yang dilakukan adalah mendeteksi komentar yang berisi *link* aktif, mendeteksi pengisian nama *author* komentar dengan anonim *author*, mendeteksi seberapa besar perbedaan waktu komentar dan *posting,* mendeteksi hubungan antara *blog post* dan komentar, mendeteksi ratio duplikasi kata, mendeteksi ratio dari *stopwords*, dan mendeteksi kalimat promosi atau ajakan.

Proses selanjutnya adalah proses klasifikasi. Pada proses ini, metode yang digunakan adalah metode *Support Vector Machine* (SVM). SVM adalah konsep

klasifikasi yang dilakukan dengan cara mencari hyperplane (batas keputusan) terbaik sebagai fungsi pemisah dua buah kelas data pada ruang imput [4]. Hyperplane pemisah terbaik antara dua kelas dapat di temukan dengan mengukur margin hyperplane tersebut dan mencari titik maksimalnya. Margin adalah jarak antara hyperplane dengan data terdekat dari masing-masing kelas. Dan dari data yang diklasifikasi nantinya akan diklasifikasikan berdasarkan 2 kelas yaitu *spam* dan *non-spam.*

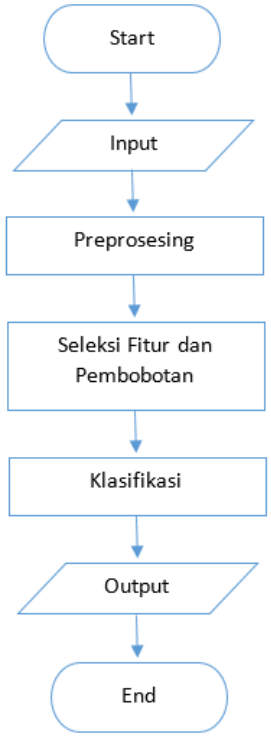
Pada terakhir yaitu tahap pengujian, akan dilakukan berdasarkan hasil klasifikasi komentar spam yang dihasilkan oleh aplikasi yang akan dibandingkan dengan hasil klasifikasi yang dilakukan secara manual. Untuk evaluasi dilakukan dengan melakukan penghitungan *precision*, *recall*, dan *f-measure.*

3.2 Perancangan Sistem

Perancangan sistem adalah merancang, membangun, dan mendesain sebuah sistem yang baik. Perancangan sistem berupa langkah-langkah operasi dalam pengolahan data dan prosedur yang mendukung dalam pengoperasian sebuah sistem.

3.2.1 Flowchart sistem

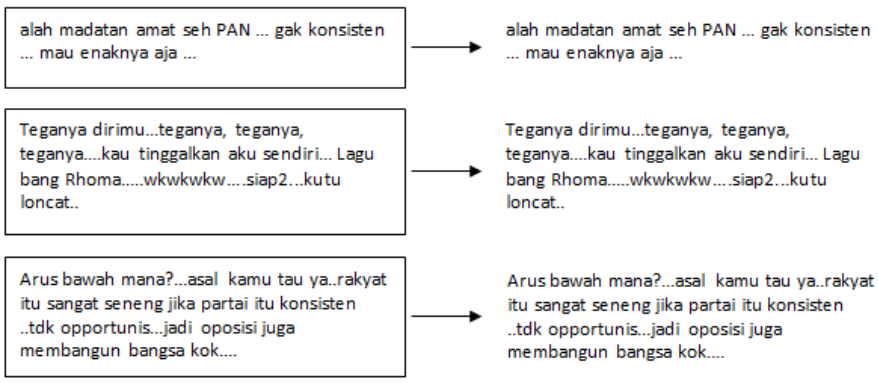
Pada sistem klasifikasi komentar *spam* dan *non-spam* memiliki beberapa tahapan. Tahap pertama adalah tahap preprosesing data dengan menggunakan metode text mining. Tahap kedua adalah tahap seleksi fitur dan pembobotan. Tahapan yang ketiga adalah tahap klasifikasi komentar dengan menggunakan algoritma *Support Vector Machine* (SVM). Komentar tersebut dibagi menjadi 2 kelas yaitu komentar *spam* dan komentar *non-spam.*



Gambar 3. 1 Flowchart Sistem

3.2.2 Tahap Preprosesing dan Seleksi Fitur

Pada penelitian ini, dilakukan tahap preprosesing dan seleksi fitur dengan menggunakan 3 contoh kasus. Berikut adalah dokumen yang dijadikan sebagai contoh kasus :



Gambar 3. 2 Contoh kasus

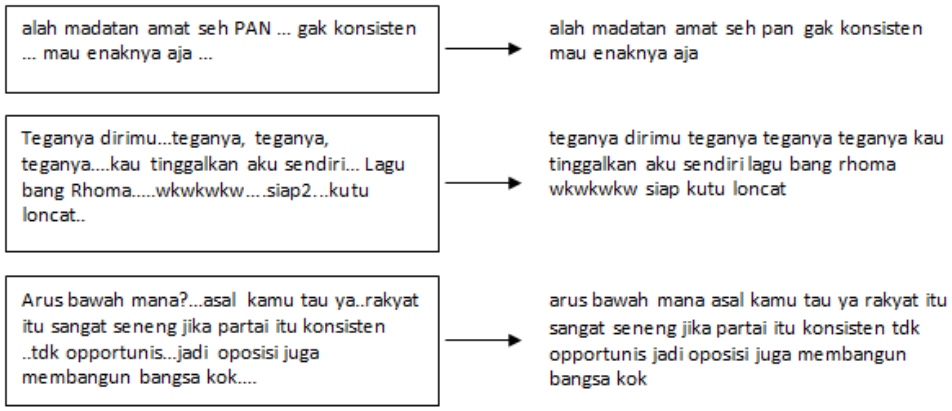
Untuk menyelesaikan percobaan diatas, tahap pertama yang harus dilakukan yaitu dengan melakukan text mining. Text mining meliputi casefolding, filtering, tokenizing, dan stimming. Tahap kedua yaitu dengan melakukan klasifikasi dengan menggunakan algoritma SVM.

a. Text Mining

(1) Casefolding

Casefolding biasanya digunakan untuk menormalisasi teks untuk tujuan perbandingan. Biasanya case folding digunakan untuk mengatasi permasalahan perbandingan case sensitif. Casefolding digunakan untuk mengubah teks dalam dokumen menjadi huruf kecil. Selain itu data teks diolah sedemikian rupa sehingga hanya huruf 'a' hingga 'z' saja yang diterima. Jadi dalam proses casefolding tanda baca dan angka akan dihilangkan.

Dari contoh kasus pada gambar 3.2, akan dilakukan proses casefolding dengan mengganti semua huruf menjadi kecil, kemudian akan dihilangkan tanda bacanya.



Gambar 3. 3 Hasil dari proses casefolding

(2) Tokenizing

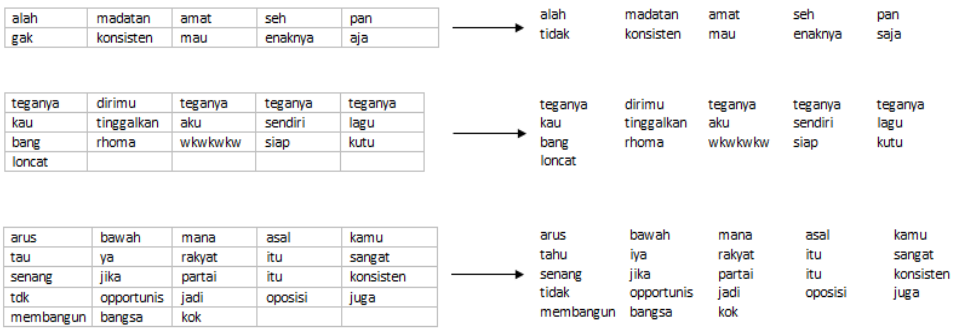
Tahap selanjutnya adalah Tokenizing. Tokenizing merupakan proses parsing kalimat menjadi kata-kata atau frase-frase. Kalimat akan diparse berdasarkan berdasarkan spasi atau berdasarkan tanda baca. Hasil dari tokenizer terdiri dari kata-kata penting dan kata-kata tidak penting.

Gambar 3. 4 Hasil dari proses tokenizing

Dari hasil proses casefolding pada gambar 3.3 akan dilakukan proses tokenizing dengan melakukan parsing menjadi kata-kata. Karena tanda baca pada proses casefolding sudah dihilangkan, makan parsing dilakukan berdasarkan spasi.

(3) Perubahan Kata Baku

Setelah dilakukan tokenizing, akan dilakukan pengecekan setiap kata apakah kata tersebut merupakan kata yang tidak baku. Apabila terdeteksi sebagai kata tidak baku, maka akan dilakukan perubahan. Pengecekan dilakukan dengan membandingkan suatu kata dengan kata-kata tidak baku yang sudah disediakan sebelumnya di database. Apabila kata tersebut sama, makan akan diubah ke kata baku. Misalnya kata tidak baku ‘aq’ diubah menjadi ‘aku’, ‘adl’ diubah menjadi ‘adalah’, ‘blh’ diubah menjadi ‘boleh’, dan lain sebagainya.



Gambar 3. 5 Hasil dari proses perubahan kata baku

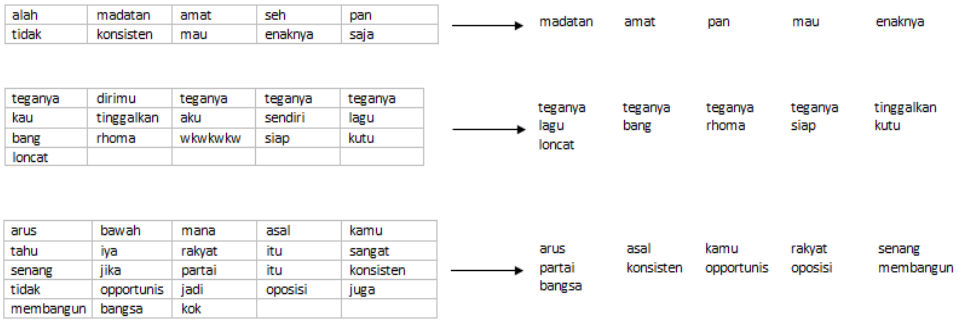
Gambar 3.5 merupakan hasil dari pemrosesan perubahan ke kata baku. Dari contoh kasus pertama ada 2 kata tidak baku yang diubah menjadi kata baku, yaitu kata ‘gak’ diubah menjadi kata ‘tidak’ dan kata ‘aja’ diubah menjadi kata ‘saja’.

Sedangkan untuk contoh kasus kedua tidak ditemukan adanya kata tidak baku, dan untuk kasus ketiga terdapat 3 kata baku yaitu, kata ‘tau’ diubah menjadi kata ‘tahu’, kata ‘ya’ diubah menjadi kata ‘iya’, dan kata ‘tdk’ diubah menjadi kata ‘tidak’.

(4) *Stopword removal*

Stopword removal digunakan untuk menghapus kata-kata yang sering muncul pada kalimat yang dianggap tidak terlalu mempengaruhi proses klasifikasi. Kata-kata yang biasanya dianggap sebagai *stopword* antara lain adalah kata penghubung (‘dan’, ‘yang’, ‘bahwa’, dan lain sebagainya), kata depan yaitu (‘di’, ‘ke’, ‘dari’, dan lain sebagainya), dan kata lain yang tidak memiliki pengaruh yang besar terhadap proses klasifikasi, misalnya ‘hampir’, ‘mampu’, ‘bisa’, dan lain sebagainya.

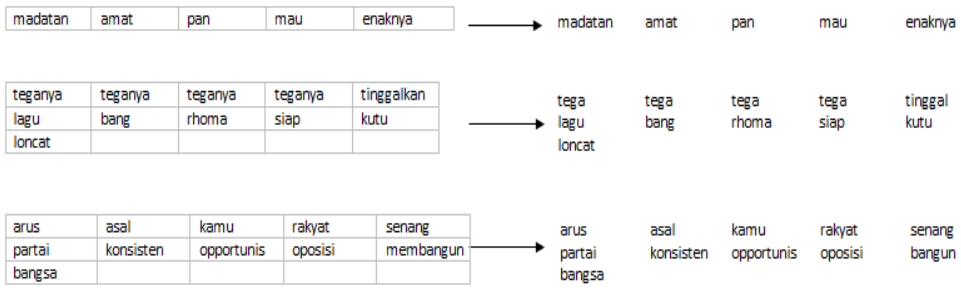
Untuk melakukan *stopword removal*, hasil kata yang sudah melalui proses tokenizing akan dilakukan pengecekan apakah kata-kata tersebut merupakan *stopword* atau bukan. Pengecekan kata akan dilakukan berdasarkan kamus kata *stopword* yang ada (daftar kata *stopword* terdapat di file stoplist.txt).



Gambar 3. 6 Hasil dari proses *stopword removal*

(5) Stemming

Stemming merupakan proses perubahan kata menjadi kata dasar. Misalnya kata ‘menolak’ memiliki kata dasar yaitu ‘tolak’ yang mendapat awalan ‘me’, kata ’menyatakan’ memiliki kata dasar ‘nyata’ yang mendapat awalan ‘me’ dan akhiran ‘kan’. Kemudian kata ‘memberi’, ‘diberi’, ‘memberikan’ adalah kata-kata yang memiliki kata dasar sama yaitu ‘beri’, sehingga akan memiliki keterkaikan nantinya antara satu kata dengan kata yang lainnya. Selain itu proses stemming dapat membuat tempat penyimpan lebih kecil.

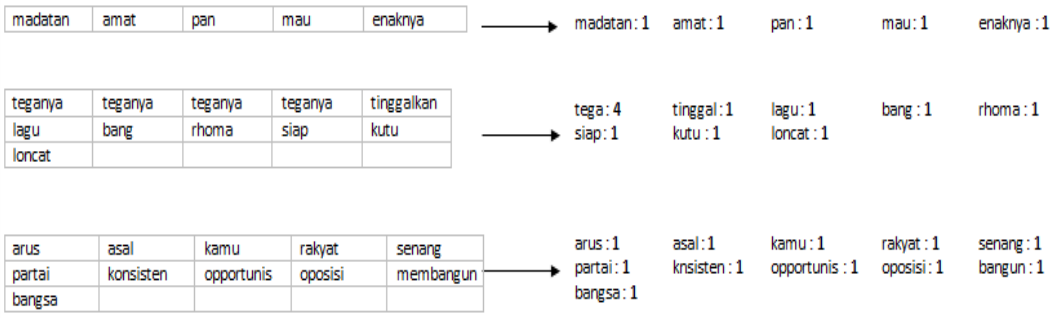


Gambar 3. 7 Hasil dari proses stemming

Pada gambar 3.7 menunjukkan hasil proses stemming setelah sebelumnya sudah dilakukan proses *stopword removal.*

b. Term Frequency (TF)

Term Frequency (TF) adalah frekuensi kemunculan suatu term atau kata yang ada pada suatu dokumen. Jadi pada setiap komentar akan dihitung satu per satu kata yang muncul dan seberapa banyak kata tersebut muncul. Misalnya pada gambar 3.7, pada contoh kasus pertama kata ‘bukti’ pernah muncul sebanyak 1 kali sendangkan kata ‘presiden’ muncul sebanyak 3 kali.



Gambar 3. 8 Perhitungan TF

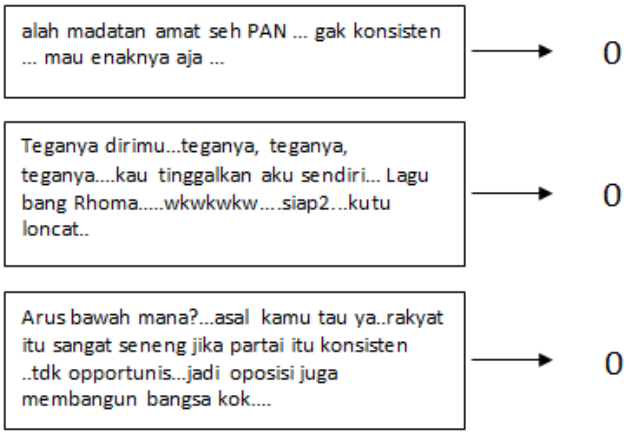
c. Seleksi Fitur dan Pembobotan

(1) Mendeteksi komentar yang berisi *link* aktif

Mendeteksi komentar satu per satu apakah memiliki *link* aktif atau tidak. Apabila memiliki *link* aktif maka akan diberi nilai 1 (true) dan jika tidak memiliki *link* aktif akan diberi nilai 0 (false).

Contoh bentuk dari link aktif antara lain, ‘facebook.com’, ‘www.viva.co.id’, ‘http://news.viva.co.id’, dan lain-lain.

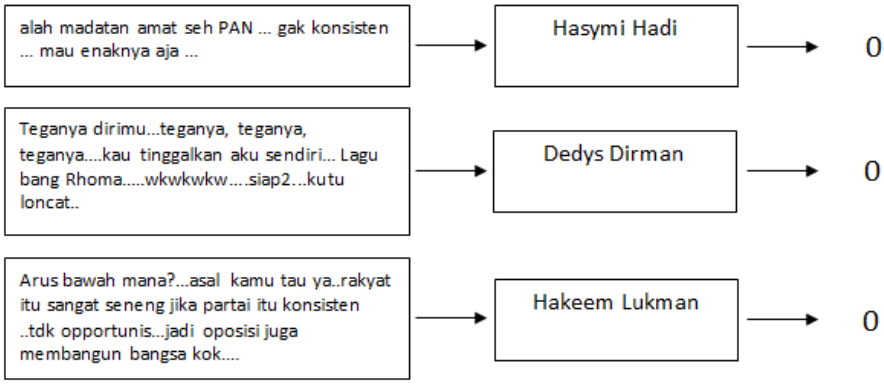
Berdasarkan 3 contoh kasus diatas (gambar 3.2), seteah di cek tidak terdapat adanya link. Jadi nilai fitur untuk ketiga contoh kasus diatas adalah 0 (false).



Gambar 3. 9 Pendekteksian link aktif

(2) Mendeteksi pengisian nama *author* komentar dengan anonim *author*

Penulisan nama author biasanya juga menentukan apakah komentar merupakan *spam* atau bukan. Untuk penentuan anonim *author* akan ditentukan berdasarkan nama *author* diisi atau tidak, berisi kata '*anonymous'*, 'anonimous' 'anonim', '*anonym',* dan 'no name'. Jika bernilai benar maka akan diberi nilai 1, dan apabila salah diberi nilai 0.



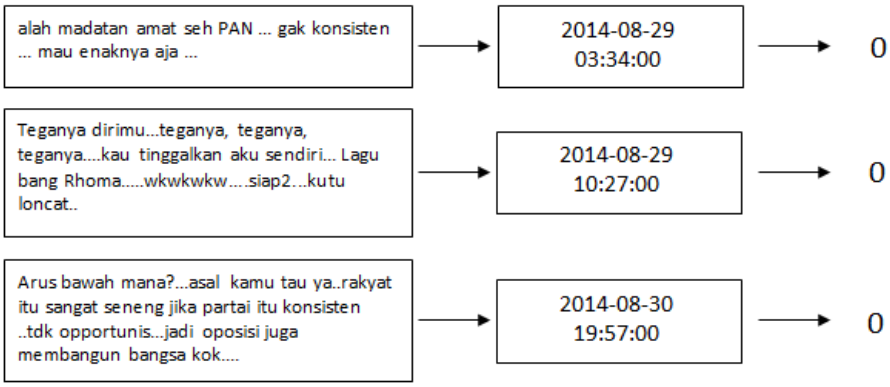
Gambar 3. 10 Pendekteksian anonim

(3) Mendeteksi seberapa besar perbedaan waktu komentar dan *posting*

Dalam proses penentuan komentar *spam* dan *non-spam*, interval perbedaan waktu antara *blog post* dan komentar juga dapat menentukan apakah suatu komentar itu merupakan *spam* atau *non-spam.* Semakin besar jarak antara waktu publikasi

komentar dan waktu publikasi *blog post* maka akan semakin memungkinkan jika komentar tersebut adalah komentar *spam*. Demikian pula sebaliknya, semakin kecil perbedaan jarak waktu publikasi komentar dan waktu publikasi *blog post* maka akan semakin kecil kemungkinannya jika komentar tersebut adalah komentar *spam.*

Untuk menentukan seberapa besar perbedaan waktu komentar dan *posting*, perlu ditentukan batasan waktu suatu komentar dianggap sebagai *spam*. Untuk penelitian kali ini, batasan yang akan ditentukan adalah 90 hari atau kurang lebih 3 bulan. Jika dianggap lebih dari 90 hari maka akan dianggap sebagai komentar *spam*.



Gambar 3. 11 Pendekteksian perbedaan waktu komentar dan *posting*

(4) Mendeteksi hubungan antara *blog post* dan komentar

Pertama-tama sebelum melakukan perhitungan frekuensi pada *blog post*, terlebih dahulu dilakukan preprosesing (casefolding, tokenizing, *stopword removal,* stemming). Setelah itu dicari TF untuk masing-masing kata. Setelah itu kita list semua kata yang ada pada *blog post* dan juga jumlah frekuensinya. Setelah mendapatkan list tersebut, tiap kata akan dicari di dalam komentar, apakah kata tersebut ada atau tidak, jika ada maka ada dikembalikan sesuai dengan jumlah kata yang ada di komentar, jika tidak ada maka akan dikembalikan angka 0.

Misal pada *blog post* terdapat 1 kata ‘kerja’ sedangkan pada contoh kasus pertama tidak terdapat kata ‘kerja’, jadi akan frekuensi kata yang dikembalikan adalah 0. $w_{kerja,\ blog}$ adalah 1 dan $w_{kerja,\ komentar}$ adalah 0, hasilnya 1x0 = 0. Misalkan pada *blog post* tedapat 3 kata ‘pan’ dan pada contoh kasus pertama juga terdapat 1

kata ‘pan’, jadi $w_{pan,\ blog}$ adalah 3 dan $w_{pan,\ komentar}$ adalah 1, hasilnya 3x1 = 3. Pada contoh kasus pertama, dari setiap list kata yang ada di dalam *blog post* hanya terdapat kata ‘pan’ saja. Jadi selain kata ‘pan’, pengembalikan frekuensinya adalah 0.

$$similarity\ kasus\ 1 = \frac{3}{\sqrt{1}\sqrt{234}} = \frac{3}{1 * 15.29705} = 0.19611$$

$$similarity\ kasus\ 2 = \frac{1}{\sqrt{1}\sqrt{234}} = \frac{1}{1\ * 15.29705} = 0.06537$$

$$similarity\ kasus\ 3 = \frac{6}{\sqrt{5}\sqrt{234}} = \frac{1}{2.23606 * 15.29705} = 0.17541$$

Dari perhitungan *similarity* pada *blog post* dan komentar pada contoh kasus pertama akan didapatkan hasil 0.19611, pada kasus kedua didapatkan hasil 0.06537, sedangkan pada kasus ketiga didapatkan hasil 0.17541. Semakin banyak kata yang pada komentar yang terdapat pada *blog post* akan menghasilkan *similarity* yang semakin besar.

(5) Mendeteksi ratio duplikasi kata

Dalam komentar *non-spam*, biasanya pengulangan kata yang sama jarang sekali terjadi. Jadi untuk mengatehui suatu komentar *spam* atau *non-spam* dapat dilakukan dengan mendeteksian ratio duplikasi kata. Pendeteksian ratio duplikasi kata dapat dianalisis berdasarkan pola perulangan kata. Untuk menentukan pola perulangan kata, akan dihitung secara matematika yaitu :

$$dupikasi\ kata = 1 - \frac{jumlah\ kata\ unik}{jumlah\ semua\ kata}$$

Ratio duplikasi kata dihitung dari jumlah kata unik yang ada pada komentar dibagi dengan jumlah semua kata yang ada di komentar. Pada contoh kasus 1, jumlah kata uniknya adalah 5 kata, sedangkan jumlah semua katanya adalah 5. Jadi hasil dari ratio duplikasi kata adalah 0.

$$dupikasi\ kata = 1 - \frac{5}{5} = 0$$

Pada contoh kasus 2, jumlah kata uniknya adalah 9 kata, sedangkan jumlah semua katanya adalah 12. Jadi hasil dari ratio duplikasi kata adalah 0,25.

$$dupikasi\ kata = 1 - \frac{9}{12} = 0{,}25$$

Sedangkan pada contoh kasus 3, jumlah kata uniknya adalah 11 kata, sedangkan jumlah semua katanya adalah 11. Jadi hasil dari ratio duplikasi kata adalah 0.

$$dupikasi\ kata = 1 - \frac{11}{11} = 0$$

(6) Mendeteksi ratio dari *stopwords.*

Hampir sama seperti ratio duplikasi data, untuk mencari ratio *stopwords* akan dilakukan dengan perhitungan matematika, yaitu sebagai berikut:

$$ratio\ stopwords = \frac{jumlah\ kata\ stopword}{jumlah\ kata}$$

Ratio *stopwords* adalah dimana jumlah kata *stopword* yang ada di komentar akan dibagi dengan banyaknya komentar yang ada. Pada contoh kasus 1, jumlah kalimat awal 10 kata, setelah di lakukan proses *stopword removal* maka hasilnya menjadi 5 kata, jadi kata *stopword* yang hilang adalah 5 kata. Setelah dilakukan perhitungan maka hasil dari ratio *stopword* adalah 0.5.

$$ratio\ stopwords = \frac{10}{5} = 0.5$$

Pada contoh kasus 2, jumlah kalimat awal 16 kata, setelah di lakukan proses *stopword removal* maka hasilnya menjadi 12 kata, jadi kata *stopword* yang hilang adalah 4 kata. Setelah dilakukan perhitungan maka hasil dari ratio *stopword* adalah 0.25.

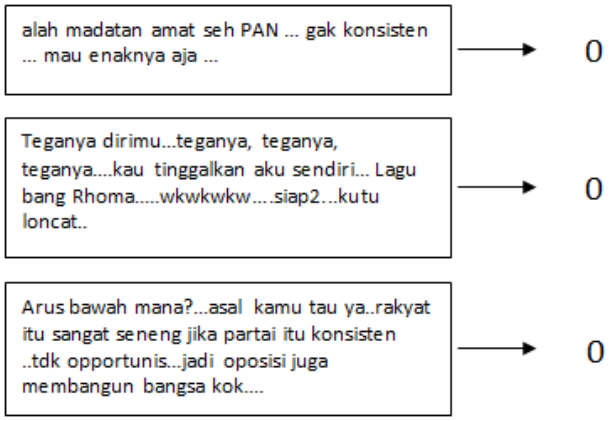
$$ratio\ stopwords = \frac{16}{12} = 0.25$$

Sedangkan pada contoh kasus 3, jumlah kalimat awal 26 kata, setelah di lakukan proses *stopword removal* maka hasilnya menjadi 19 kata, jadi kata *stopword* yang hilang adalah 7 kata. Setelah dilakukan perhitungan maka hasil dari ratio *stopword* adalah 0. 26923.

$$ratio\ stopwords = \frac{7}{26} = 0.26923$$

(7) Mendeteksi kalimat promosi atau ajakan

Mendeteksi komentar satu per satu apakah mengandung kalimat promosi atau tidak. Pendeteksian apakah suatu komentar mengandung kalimat promosi atau tidak, dilakukan dengan mengecek setiap kalimat di setiap komentar apakah memiliki kata ajakan seperti ‘mari’, ‘marilah’, ‘ayo’, ‘ayolah’, atau lainya. Apabila mengandung kalimat promosi maka akan diberi nilai 1 (true) dan jika tidak mengandung kalimat promosi akan diberi nilai 0 (false).



Gambar 3. 12 Pendekteksian kalimat promosi atau ajakan

d. SVM Format

Tahap selanjutnya setelah preprosesing, seleksi fitur, dan pembobotan setiap fitur, akan dilakukan perubahan format data pada data *train* (data latih) dan data *test*.

<label> <index1>:<value1> <index2>:<value2><index n>:<value v>

Masing-masing baris akan mewakili setiap komentar yang ada. Label merupakan integer yang mengindikasikan kelas dari sebuah data. Misal untuk komentar *spam* akan diwakili dengan label 1, sendangkan untuk komentar *non-spam* akan diwakili dengan label -1. Untuk pasangan <index1>:<value1> merupakan nilai dari suatu fitur. <index> menandakan sebuah integer yang dimulai dari 1, dan <value> merupakan nilai dari fitur tersebut yang merupakan angka real. Misalkan kita tentukan index1 adalah pengecekan *link* aktif, maka pada contoh kasus pertama hasilnya menjadi 1:0 (0 merupakan hasil dari pengecekan *link* aktif pada contoh kasus pertama).

-1 1:0 2:0 3:0 4:0.19611 5:0 6:0.5 7:0
1 1:0 2:0 3:0 4:0.06537 5:0.25 6:0.25 7:0
-1 1:0 2:0 3:0 4:0.1754 5:0 6:0.52173 7:0

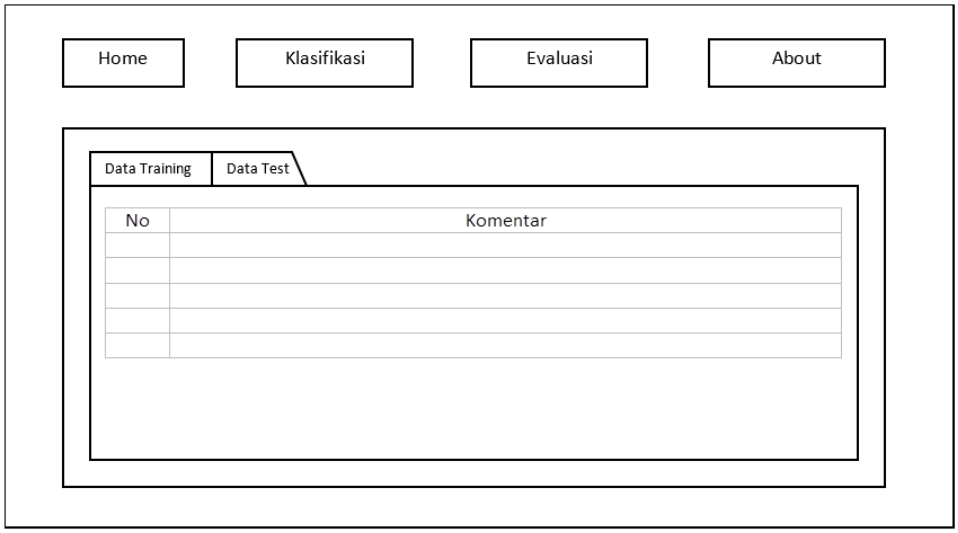
Data di atas adalah contoh dari format SVM berdasarkan hasil perhitungan fitur dari contoh kasus sebelumnya. Untuk beberapa fitur yang memiliki nilai 0, bisa tidak digunakan / dituliskan dalam format, karena hasilnya sama apabila dihilangkan dari format, seperti contoh dibawah ini.

-1 4:0.19611 6:0.5
1 4:0.06537 5:0.25 6:0.25
-1 4:0.1754 6:0.52173

3.3 Desain Antarmuka

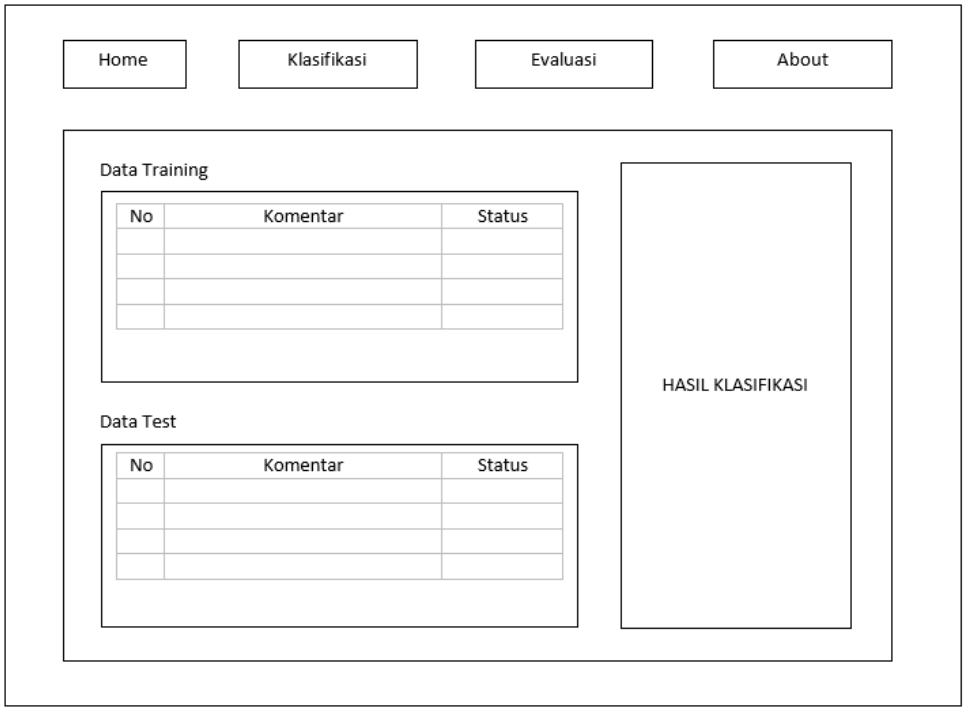
3.3.1 Tampilan Antarmuka Home

Gambar 3. 13 Tampilan antarmuka home – data *train*



Gambar 3. 14 Tampilan antarmuka home – data *test*

3.3.2 Tampilan Antarmuka Klasifikasi



Gambar 3. 15 Tampilan antarmuka klasifikasi – hasil klasifikasi